OPTIMISATION OF A CONCEPTUAL AIRCRAFT MODEL USING A GENETIC ALGORITHM AND 3D COMPUTATIONAL FLUID DYNAMICS

Lok Chun Chan, *Mohammad Akrami, Akbar A. Javadi, Gavin Tabor, Mahdieh Dibaj,
Mohammad Hassan Khanjanpour
Department of Engineering, University of Exeter, Exeter, EX4 4QF, UK
*m.akrami@exeter.ac.uk

Summary

Aircraft design is fundamentally a multidisciplinary design activity which involves different models and tools for various aspects of the design. This paper uses a Multidisciplinary Design Optimisation (MDO) for design of a simplified commercial aircraft, aiming to optimise the objectives of cost, weight and drag. NSGA-II is used to optimise the weight and cost by changing the geometry to introduce lightweight airframe materials and composites with lower density. Reducing weight of the structure is one of the major ways to improve the performance of aircraft. Lighter, stronger material will allow a higher speed and greater range which may contribute to reducing operational costs. Drag reduction is also a major factor in aircraft design. Reduction of drag in an aircraft means that it can have a lower fuel consumption or travel at higher speed, both of which are beneficial to plane performance. A smart structural optimisation algorithm helps to optimise the cost, weight and drag, while drag is analysed based on CFD modelling results. The results are validated against some wind tunnel tests.

Key Words: Optimisation, aircraft, NSGA-II, CFD

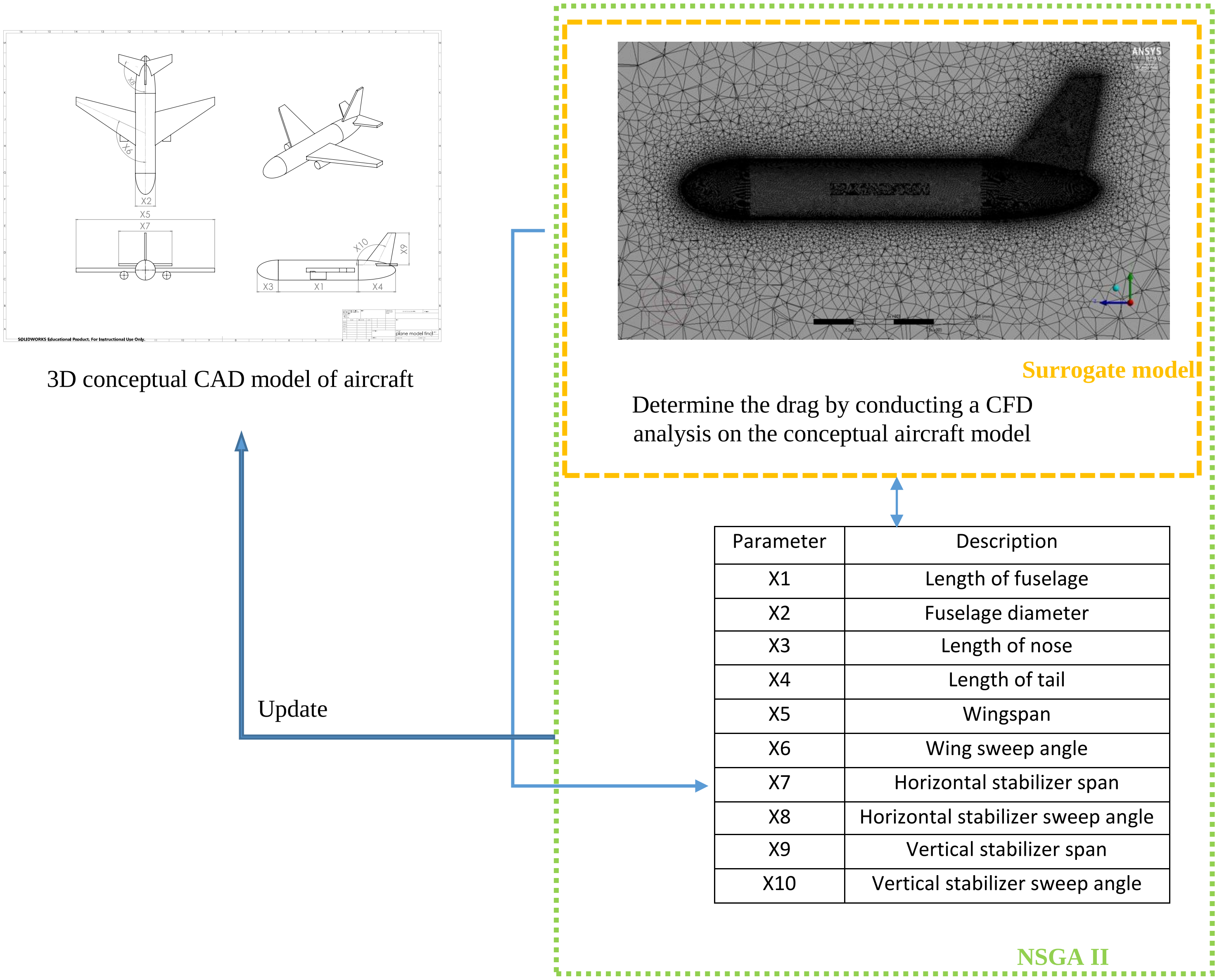
Introduction

The first commercial aircraft started back on 1st January 1914. It operated between St. Petersburg and Tampa, yet only lasted for four months. Since then, engineers and scientists start to design and investigate different methods to improve the performance of aircraft. The original aims are to increase the distance of travel and the capacity of the aircraft. Nowadays, commercial aircrafts are fairly developed and improved compared to the first commercial aircraft. Even though the aircrafts have massively improved compared to the earlier designs, but there are still areas to improve. Designers always try to come up with better designs and the aims have changed from achieving basic performance to improving the performance of flight such as reducing the fuel consumption. To achieve these aims, designers have applied various techniques and technologies to assist with their design works. For example, the use of computational simulation, wind tunnel experiment and mathematical models has helped to refine, improve and optimise the aircrafts in terms of their structure, material choice and geometry in order to achieve a “better” design. Aircraft design always contains more than one objective; e.g., reduction of fuel consumption, reduction of weight and increase in the payload; and involves various disciplines such as structure dynamics, fluid dynamics and materials engineering. In this paper, MDO is used to optimise a simplified model of a commercial aircraft in order to reduce weight, drag and cost, simultaneously.

Materials and Methods

During the conceptual design, ideas were formulated, rephrased, rejected or modified. Traditionally, aircraft design is separated into different areas of focus, such as structure, materials, dynamics and cost and there are aims and constraints in each area.

For example, the design of airframe focuses on structure and shape while the design of the wing will focus on lift and drag. All parts of an aircraft are closely related and interlinked with each other. Therefore, achieving one objective might have an adverse effect on the other design objective. For instance, replacing aluminium alloy by light-weight materials such as carbon fibre and other composites to achieve a weight reduction will have a trade-off of increased production cost. A complex design, such as aircraft, need to achieve different design objectives while there are different trade-offs. An MDO is a methodology to achieve the "best" solution by changing the design variables and maintaining different design constraints [1].


X2
X5
X7
X3
X1
X4
X10
X9
X6
X8
3D conceptual CAD model of aircraft
Surrogate model
Determine the drag by conducting a CFD analysis on the conceptual aircraft model
Update
NSGA II

Parameter	Description
X1	Length of fuselage
X2	Fuselage diameter
X3	Length of nose
X4	Length of tail
X5	Wingspan
X6	Wing sweep angle
X7	Horizontal stabilizer span
X8	Horizontal stabilizer sweep angle
X9	Vertical stabilizer span
X10	Vertical stabilizer sweep angle

Figure 1: Application of the NSGA-II algorithm and the assigned parameters in the model

Traditionally, design is a process focusing on one area at a time and the areas down the stream are heavily dependent on the previous results. As modern design has developed and become more complex, it has become very difficult to optimise multiple design objectives. MDO is a design process which evaluates all design variables simultaneously and considers the trade-off and interaction between different objectives and disciplines. MDO is a methodology for the design of complex engineering systems that are governed by mutually interacting physical phenomena and made up of distinct interacting subsystems in their design.

The rapid development in the computational analysis in recent years has been largely influencing MDO. Many simulation results can now be linked and considered in the calculation process, which could be a significant aid, especially in aircraft design. Areas like wing lift and drag can be more easily considered during the design process. Apart from CFD or FEA, MDO can also involve diverse tools such as cost modelling and even experimental results. Non-dominated Sorting Genetic Algorithm (NSGA-II) is a modified version of NSGA which is a multi-objective optimisation model and was one of the first evolutionary algorithms [2, 3]. It generates a set of solutions, also known as Pareto-optimised solutions, which are superior to the rest of the solutions. There is no absolute better solution within this Pareto-optimised solutions and hence the choice of an optimal solution over the others requires problem knowledge and other problem factors. In this study, the ten initial parameters (Figure 1) were defined in order to optimise the three main objectives. These ten parameters will update in each iteration based on the geometrical parameters' ranges of the aircraft models to generate the optimal solution. Also, a surrogate model was constructed based on the initial results from the CFD model and the remaining simulations were analysed based on this model predictions to observe and choose the most promising ones until a "good design" can be achieved. By generating a surrogate model using CFD simulation which has been validated by the wind tunnel results (Figure 2) and adapted in the NSGA-II model, the range of optimal solutions are generated.

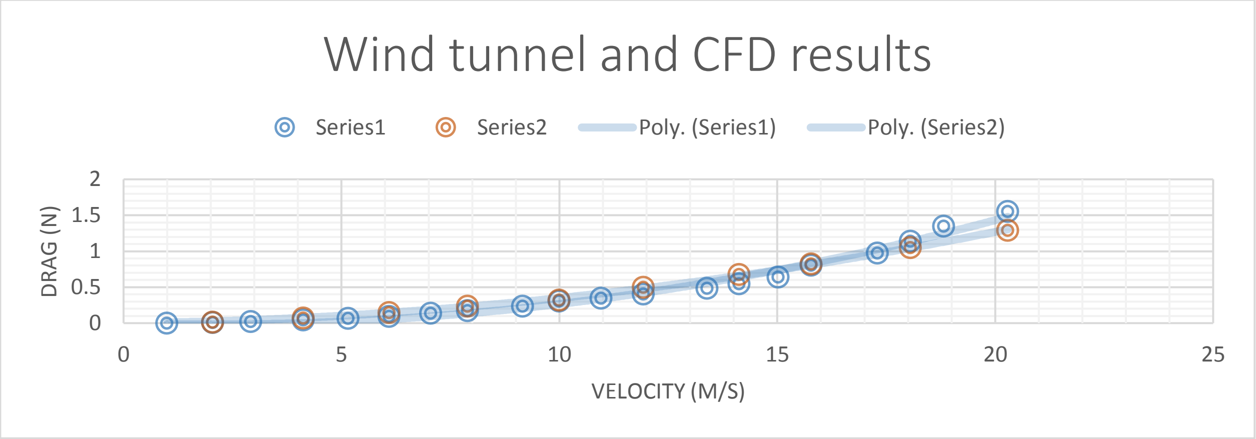


Figure 2: Validating the CFD model against the wind tunnel results based on the Drag force (N) and the Velocity (m/s)

Results and conclusion

By combining the surrogate optimisation model generated by multiple CFD simulations and NSGA II, multiobjective genetic algorithm, a Pareto front is generated which clearly shows a trade-off between different objectives. There is a significant trade-off between weight and cost, as weight decreases, the costs increases (Figure 3) [4, 5]. This trend clearly exists between the weight and Fuselage volume and also cost and the volume. The results help to find a range of solutions in order to optimise the the topology of the structure [6].

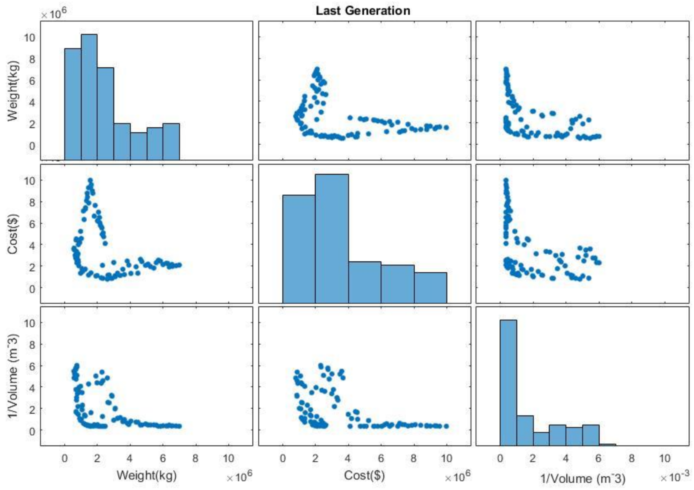


Figure 3: Optimisation results from the NSGA-II algorithm for the three main objectives of Cost, Weight and Fuselage volume

Based on the developed algorithm, the optimised ranges for the main 10 parameters are illustarted (Table 1) in order to minimise the aforementioned objectives.

Parameters	Initial range	Optimised range
X1	15 – 50	36.86 – 50
X2	3.76 – 8.5	6.648 – 8.5
X3	3.96 – 5	4.173 – 5
X4	7 – 15	8.115 – 13.473
X5	26 – 80	26 – 31.62
X6	90 – 125	105.03 – 123.23
X7	10 – 30	10 – 15.834
X8	115 – 140	119.694 – 127.67
X9	6 – 15	6 – 10.559
X10	115 – 140	115 – 139.07

References

1. Sobieszczanski-Sobieski, J., Multidisciplinary Design Optimization: An Emerging New Engineering Discipline, Advances in Structural Optimization (483-496), Kluwer Academic Publishers, the Netherlands, 1995

2. N. Srinivas, Kalyanmoy Deb, Multiobjective Optimization using Non-dominated sorting in Genetic Algorithms, 1994

3. Deb, K., Pratap, A., Agarwal, S. and Meyarivan, T. A fast and elitist multiobjective genetic algorithm: NSGA-II. Evolutionary Computation, IEEE Transactions on, 6, 2 2002)

4. Wang X, Hirsch C, Kang S, Lacor C. Multi-objective optimization of turbomachinery using improved NSGA-II and approximation model. Computer Methods in Applied Mechanics and Engineering. 2011;200(9-12):883-95.

5. Bellary SAI, Adhav R, Siddique MH, Chon B-H, Kenyery F, Samad A. Application of computational fluid dynamics and surrogate-coupled evolutionary computing to enhance centrifugal-pump performance. Engineering Applications of Computational Fluid Mechanics. 2016;10(1):171-81.

6. Niemann, S., Kolesnikov, B., Lohse-Busch, H., Hühne, C., Querin, O.M., Toropov, V.V. and Liu, D., 2013. The use of topology optimisation in the conceptual design of next generation lattice composite aircraft fuselage structures. *The Aeronautical Journal*, *117*(1197), pp.1139-1154.

```
function [y, cons] = TP_CONSTR_objfun(x)
[Ane, Ate]= elliptical (x);
Abo=pi*x(1)*x(2); An=integral(Ane,-x(3),0);Ata=integral(Ate,-x(4),0);
Awis= ((x(5)/4)+1.5)*(x(5)/2);
Awint= (2*(((x(5)/2))/sind(x(6)))+1.5)*0.75;
Awi=Awis+2*Awint; Astas= ((x(7)/2.5)+1.5)*(x(7)/2);
Astat=(2*(((x(7)/2))/sind(x(8)))+1.5)*0.4;
Asta=Astas+2*Astat; Arus=(((x(9)/1.2)+1.5)*x(9))/2;
Arut=(2*(((x(9)/2))/sind(x(10)))+1.5)*0.3;
Aru=Arus+Arut; En= (1.5/2)^2*pi*3;
tbl = readtable('test2.xlsx','range','A1:K101','ReadVariableNames',false);
tbl.Properties.VariableNames =
{'X1','X2','X3','X4','X5','X6','X7','X8','X9','X10','drag'};
 gprMd =
fitrgp(tbl,'drag','KernelFunction','squaredexponential','FitMethod','fic','Predict
Method','exact','Standardize',1);
 C={x(1),x(2),x(3),x(4),x(5),x(6),x(7),x(8),x(9),x(10),[]};
T = cell2table(C);
T.Properties.VariableNames =
{'X1','X2','X3','X4','X5','X6','X7','X8','X9','X10','drag'};
y = [0,0,0];
cons = [0,0];
 y(1) = predict(gprMd,T);
y(2) =
((x(11)/(x(11)+x(12)+x(13)))*1750+(x(12)/(x(11)+x(12)+x(13)))*2800+(x(13)/(x(11)+x
(12)+x(13)))*7850)*0.5*(((Abo)+(abs(An))+(abs(Ata))+(2*Awi)+(2*Asta)+(2*Aru)+(2*En
)));
y(3) =
((x(11)/(x(11)+x(12)+x(13)))*1750*6.38+(x(12)/(x(11)+x(12)+x(13)))*2800*1.38+(x(13
)/(x(11)+x(12)+x(13)))*7850*0.3)*(0.5*((Abo)+(abs(An))+(abs(Ata))+(2*Awi)+(2*Asta)
+(2*Aru)+(2*En)));
c(1)= x(1)*(x(2)/2)^2*pi;
c(2)= x(1)*(x(2)/2)^2*pi;
%c(3)= y(3);
 if(c(1)<150)% new deisgn has a lower payload than the original Boeing 737 100
    cons(1) = abs(c(1)-150)*100000;
 if (c(2)>=150)
        cons(2)=150/((abs(c2)-150)*100000);
end
 %if (c(3)<=0)
%        cons(3)=0.001/((abs(c2)-150)*100000);
%end
end
clear All
 boundary
 options = nsgaopt();                    % create default options structure
options.popsize = 100;                  % populaion size
options.maxGen  = 100;                  % max generation
 options.numObj = 2;                     % number of objectives
options.numVar = 13;                    % number of design variables
options.numCons = 2;                    % number of constraints
options.lb = lb;                        % lower bound of x
options.ub = ub;                        % upper bound of x
options.objfun = @TP_CONSTR_objfun;     % objective function handle
options.plotInterval = 5;               % interval between two calls of
"plotnsga".
 result = nsga2(options);                % begin the optimization!
 %plotting
%optimisation_history
function varargout = plotnsga(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @plotnsga_OpeningFcn, ...
                   'gui_OutputFcn',  @plotnsga_OutputFcn, ...
                   'gui_LayoutFcn',  [], ...
'gui_Callback',   []);
if nargin && ischar(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end
 if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
function plotnsga_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;
guidata(hObject, handles);
 handles.bLoadFromFile = 0;
if( isempty(varargin) )
    error('PLOTNSGA:ParamError', 'Error: plotnsga should be called :
plotnsga(result, curGen) or plotnsga(''pops.txt'')');
 elseif(length(varargin) == 1)
    if( isstruct(varargin{1}) )
        handles.result  = varargin{1};
        handles.currentGen = 1;
            elseif( ischar(varargin{1}) )
        global oldresult;
        oldresult = loadpopfile(varargin{1});
        evalin('base', 'global oldresult');

        handles.bLoadFromFile   = 1;
        handles.strPopFile      = varargin{1};
        handles.result          = oldresult;
        handles.currentGen      = 1;
    end
    elseif(length(varargin) == 2)
    if( isstruct(varargin{1}) && isscalar(varargin{2}) )
        handles.result  = varargin{1};
        handles.currentGen = varargin{2};
    else
        error('PLOTNSGA:ParamError', ...
            'Error: plotnsga should be called : plotnsga(result, curGen) or
plotnsga(''pops.txt'')');
    end
end
popsize = size(handles.result.pops, 1);
strList = repmat({''}, [1,popsize]);
for i = 1:popsize
    strList{i} = sprintf('%d', i);
end
 curSel = handles.currentGen;   set(handles.listPop, 'string', strList);
set(handles.listPop, 'value', curSel);
 dispState(handles, curSel);
plotPopulation( handles, curSel );
guidata(hObject,handles);
 function varargout = plotnsga_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;
 function btnStop_Callback(hObject, eventdata, handles)
global STOP_NSGA;
STOP_NSGA = 1;
 function btnPause_Callback(hObject, eventdata, handles)
if isempty(getappdata(0,'gadsSolverState'))
    setappdata(0,'gadsSolverState','pause');
    set(hObject,'String','Continue');
        waitfor(hObject,'String');
else
    setappdata(0,'gadsSolverState','');
    set(hObject,'String','Pause');
end
```