A Framework for Multidisciplinary Design and Optimisation in Aeronautics

L. F.González [1], E. Whitney [1], K. Srinivas [1], K.C Wong [1]
[1] *School of Aerospace, Mechanical and Mechatronic Engineering,*
University of Sydney, NSW 2006, Australia
{gonzalez, eric,ragh, kc}@aeromech.uysd.edu.au
and
J. Périaux [2]
[2] *INRIA Sophia Antipolis, OPALE project associate and Pole Scientifique Dassault Aviation 78 Quai Marcel Dassault, 99214 Saint-Cloud, France*
jperiaux@noos.fr

Summary: This paper examines the initial development and application of a framework for Multidisciplinary Design and Optimisation (MDO) in aeronautics. Traditional deterministic optimisation techniques for MDO are effective when applied to specific problems and within a specified range. These methods are efficient to find optimal global solutions if the objective and constraints are differentiable. But if a broader application of the optimiser is desired, or when the complexity of the problem arises because they are multi-modal, involve approximation, are non-differentiable, or involve multiple objectives and physics, more robust and alternative numerical tools are required. Emerging techniques such as Evolutionary Algorithms (EAs) have shown to be robust as they require no derivatives or gradients of the objective function, have the capability of finding globally optimum solutions amongst many local optima, are easily executed in parallel, and can be adapted to arbitrary solver codes without major modifications. In this paper, the formulation and implementation of a framework for analysis and optimisation of multidisciplinary and multi-objective optimisation problems in aeronautics is described. The framework includes a Graphics User Interface (GUI) a robust EA optimiser, several design modules, and post-processing capabilities. The application of the method is then illustrated with application to a multi-objective wing design problem. Results indicate the practicality and robustness of the method in finding optimal solutions and trade-offs between the disciplinary analyses, and in producing a set of individuals represented in an optimal Pareto front.

Keywords: Multidisciplinary Design Optimisation (MDO), Evolutionary Design, Parallel Computing.

1. Introduction

Complex systems in engineering design and more demanding industrial requirements have pushed the need on increasing the development and of robust and fast numerical techniques to overcome difficulties associated with traditional deterministic optimisers. In aerospace engineering design and optimisation, the designer is usually presented with a problem that involves not only one objective but also numerous objectives and multi-physics. Hence a systematic approach, which is regarded as Multidisciplinary Design Optimisation (MDO) that accounts for the coupling between the variables and disciplines, is required. Problems in aeronautics usually involve a number of disciplines and objectives and where the search space can be multi-modal, non convex or discontinuous. Wing design is an example of a multi-objective MDO problem as there is a strong interaction between aerodynamics and structures.

There are different approaches for solving an MDO problem using traditional optimisation techniques [2, 3, 4, 7, 22, 36, 39]. These techniques are effective when applied to specific problems and within a specified range and efficient to find optimal global solutions if the objective and constraints are differentiable. But if a broader application of the optimiser is desired or when the complexity of the problem arises because they are multi-modal, involve approximation are non-differentiable, or involve multiple objectives and physics, more robust and alternative numerical tools are required.

One of the emerging optimisation techniques is Evolutionary Algorithms (EAs) [15, 20, 25]. EAs are based on Darwinian evolution, whereby populations of individuals evolve over a search space and adapt to the environment by the use of different mechanisms such as mutation, crossover and selection. An attractive feature of EAs is that they evaluate multiple populations of points and are capable of finding a number of solutions in a Pareto set [15, 25]. EAs have been successfully applied to different aeronautical design and CFD problems and there have been various efforts to explore the capabilities of EAs for aircraft, wing, aerofoil and rotor blade design [16, 17, 28, 29, 30, 44]. One drawback of EAs is that they are slow to converge as they require a large number of function evaluations and have poor performance with increasing number of variables. Hence the continuing challenge has been to develop robust and fast numerical techniques to overcome these challenges and facilitate the complex task of design and optimisation in aeronautics. In this paper, the strategy and implementation of a framework for the design and optimisation of aeronautical systems that uses a robust evolutionary technique, which is scalable to preliminary design studies with higher fidelity models for the solution is described. The fundamental idea with this framework is to simplify the task of integration to the user so he/she can focus on the problem itself.

The rest of the paper is organised as follows, section 2 summarises some requirements for a robust multi-objective multidisciplinary design optimisation framework, section 3 describes the optimisation method used in the framework and a general definition of multi-objective evolutionary algorithms, the formulation and implementation of the framework is presented in section 4, section 5 illustrates the application of the method to real world problems in aeronautics. Finally, section 6 provides summary and future directions for the research.

2. Requirements for a Multi-objective Multidisciplinary Design Optimisation Framework in Aeronautics.

Design and optimisation in aeronautics is a complex task as it involves non-linearities, multi-objective, and multidisciplinary considerations. In order to handle this complexity, it is desirable to develop a system, which facilitates integration of a series of design tools, graphical user interfaces (GUI), post-processing capabilities, among others to solve the problem. Such a system is termed a framework. This section focuses on the requirements, development and implementation of a framework using EAs in which different multidisciplinary and multi-objective problems in aeronautics can be analysed. The fundamental idea with this framework is to simplify the task of integration to the user so he/she can focus on the problem itself. The idea of this framework is a generic system that can be easily developed, maintained and extended. The basic requirements for a MDO framework can be subdivided in problem formulation and optimisation methods, problem execution, architectural design and information access [5, 6, 32, 33, 42].

Problem Formulation and Optimisation Methods.

The framework should allow:

1. ease of integration of robust optimisation methods;
2. the user to configure and reconfigure different MO and MDO formulations easily without low-level programming;
3. the user to incorporate legacy codes, which can be written in different programming languages, and proprietary software where no source code is available; and
4. integrating different disciplinary analysis with different optimisation methods and should provide schemes that involve sub-optimisation within each design module.

Problem Execution.

The framework should:

1. allow the execution and movement of data in an automated fashion;
2. be able to execute multiple processes in parallel and through heterogeneous computers; and
3. ensure that a batch mode be implemented.

Architectural Design.

The framework should:

1. be developed using object-oriented principles;
2. provide an easy to use and intuitive GUI;
3. be easily extended by developing new interfaces required to integrate new processes into the system;
4. not impose unreasonable overhead on the optimisation process;
5. handle large problem sizes; and
6. be based on acceptable standards.

Information Access.

The framework should:

1. provide facilities for database management;
2. provide capabilities to visualise intermediate and final result from the analysis or optimisation;
3. allow capabilities for monitoring and viewing the status of an execution and its system status; and
4. provide a mechanism for fault tolerance.

With these requirements in mind, the general scope for the framework could be identified. The framework developed in this research address these requirements to some extent. Fig. 1 shows a representation of different components to satisfy the requirements. The framework will have seven major constituents: A robust optimisation tool, a problem formulation capability within each analysis module, and some architectural design considerations such as a GUI, a Design of Experiments (DOE) module, some analysis modules, and capabilities for parallel computing and post-processing. In the following sections and subsection each of these constituents is detailed.

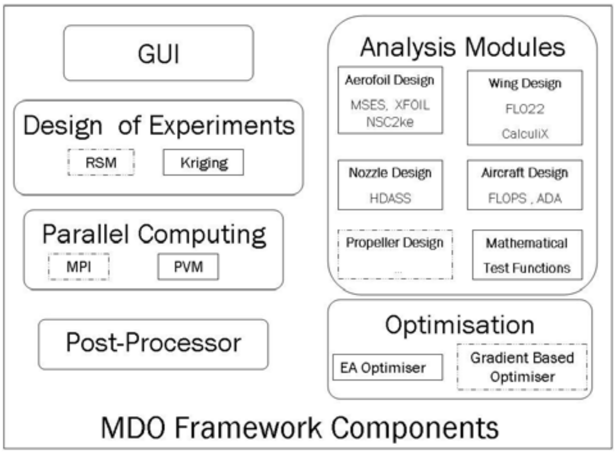

Fig.1: MDO Framework.

3. Optimisation Tools: Hierarchical Asynchronous Parallel Evolutionary Algorithms (HAPEA)

The first consideration is the incorporation of robust optimisation tools. In this research we use the Hierarchical Asynchronous Parallel Evolutionary Algorithm (HAPEA) approach developed by Whitney [43, 44] with some extensions for multidisciplinary and multi-objective analysis introduced since. The foundations of the algorithm lie upon traditional evolution strategies and incorporate the concepts of a multi-objective optimisation, hierarchical topology, asynchronous evaluation and parallel computing. A pseudo code of a canonical evolution strategy is illustrated in algorithm 1. A population ($\boldsymbol{\mu_o}$) is initialised and then evaluated. Then for a number of generations (***g***) and while a stopping condition (maximum number of function evaluation or target fitness value) is not met, offsprings (λ^{g+1}) go recursively through the process of recombination, mutation, evaluation and selection.

Initialise: $init(\mu_o)$

Evaluate: $f(\mu_o)$

g=0

while stopping condition not met,

Recombine: $\lambda_R^{g+1} = reco(\mu^g)$

Mutate: $\lambda_M^{g+1} = mut(\lambda_R^{g+1})$

Evaluate: $\lambda^{g+1} = f(\lambda_M^{g+1})$

Select: $\mu^{g+1} = sel(\mu + \lambda)$ (plus strategy) or,

$\mu^{g+1} = sel(\lambda)$ (comma strategy)

g=g+1

loop

Algorithm 1: Canonical Evolution Strategy

Extensions to the algorithm include an integrated or distributed MDO formulation as will be detailed in Section 4 and a mathematical test bench for multi-objective, goal programming and constrained optimisation problems and NASH equilibrium concept [27] for multi-objective problems as detailed in References [17, 18].

3.1 Multi-objective EAs (MOEAs)

Most real world problems involve conflicting objectives and there is no unique optimum, but a set of compromised individuals known as Pareto optimal solutions or non-dominated individuals. The Pareto Optimality principle is one where a solution to a multi-objective problem is considered Pareto optimal if there is no other solutions that satisfy better all the objectives simultaneously. Fig. 1 shows Pareto optimality for a two conflicting objectives problem. The objective of Pareto set is then to provide a set of Pareto optimal solutions that trade off the information among the conflicting objectives.

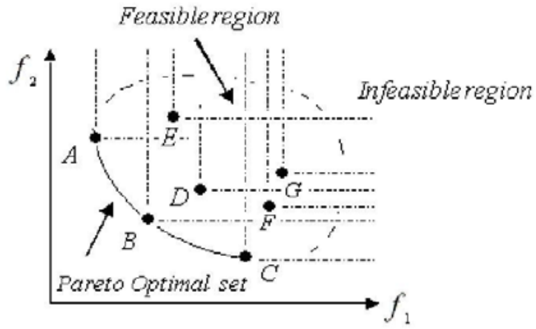

Fig. 2: Pareto Optimality

As EAs evaluate multiple populations of points, they are capable of finding a number of solutions in a Pareto set. Pareto selection ranks the population and selects the non-dominated individuals for the Pareto front.

There are some problems on applying EAs to multi-objective problems, Deb [10] for example describe and analyse problem features that might cause a multi-objective genetic algorithm to converge to the true Pareto front and define difficult test problems that serve as a guideline to evaluate the multi-objective features of an algorithm. By doing an analysis of these functions an algorithm can be tested for multi-modal multi-objective problems, deceptive multi-objective problems, multi objective problems having convex - non-convex and continuous optima fronts and non -uniformly represented Pareto optimal fronts. The HAPEA algorithm has been tested for some of these multi-objective test cases and has been proven to be robust and efficient to find the optimal Pareto front [17,43].

3.2 Hierarchical Topology

The HAPEA algorithm is designed to handle multiple fidelity models for the solution [34, 35]. Fig. 3a shows a representation of this formulation. The bottom layer can be entirely devoted to exploration, the intermediate layer is a compromise between exploitation and exploration and the top layer concentrates on refining solutions. To take full benefit of a hierarchical structure, the top layer uses a very precise model meaning a time--consuming solution. But at the same time, the sub-populations of the bottom layer need not yield a very precise result, as their main goal is to explore the search space. That means that they can make good use of simple models, with fast numerical solvers.

3.3 Parallel Computing and Asynchronous Evaluation

EAs are particularly adaptable to parallel computing, individuals can be sent to remote machines, evaluated and incorporated back into the optimisation process [8, 9, 40]. In this paper the optimisation was parallelised on a network of computers at The University of Sydney. The system has ten Intel Pentium-equivalent machines with performances varying between 1.0 and 2.4 GHz. The master computer carries on the optimisation process while the remote machines compute the solver code. The message-passing model used is the Parallel Virtual Machine (PVM)[15]. The parallel implementation requires modifications to the canonical ES [19, 25], which ordinarily evaluates entire populations simultaneously.

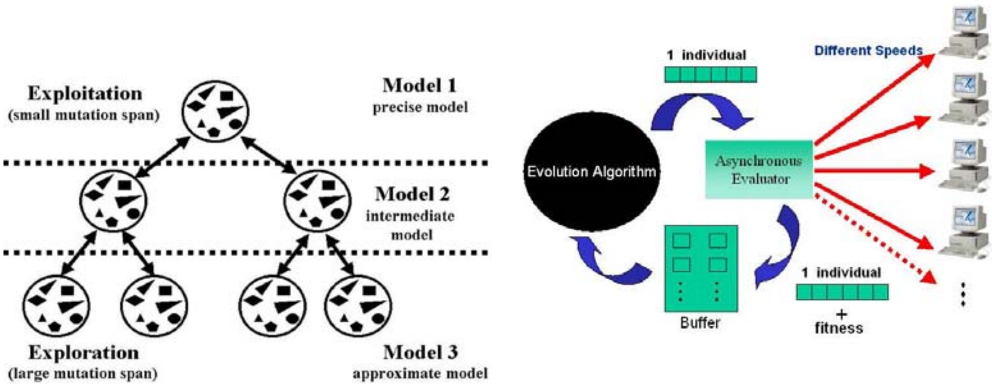

Fig. 2: a) Hierarchical Topology b) Parallel Computing and Asynchronous Evaluation

The distinctive method of an asynchronous approach is that it generates only one candidate solution at a time and only re--incorporates one individual at a time, rather than an entire population at every generation as is usual with traditional EAs [41]. Consequently solutions can be generated and returned out of order. This allows the implementation of an asynchronous fitness evaluation giving the method its name. Fig. 3b shows a schematic representation of this approach. HAPEA had been applied to different design problems including deceptive and multi modal Pareto solutions, viscous two-dimensional inverse and direct nozzle optimisation and multi-objective constrained aerofoil design problems. In all these cases the algorithm successfully converged to an optimal solution. Additional details on the algorithm can be found in [17,18,43,44].

4. Multi-objective Multidisciplinary Design Optimisation Problem Formulation and Execution

4.1 Overview.

A second consideration is how to incorporate different multi-objective and MDO formulations and different legacy codes within the framework. There are many strategies proposed for multi-objective and MDO and the development of these optimisation methods, architectures and decomposition methodologies has been an active field of research [2, 3, 4, and 36]. The framework developed in this research is applicable to an integrated analysis or distributed MDO analysis.

4.2 Integrated Multi-objective -Multidisciplinary Optimisation Formulation using Hierarchical Parallel Asynchronous Evolutionary Algorithms

In an integrated analysis, the set of design variables ***s*** are evaluated by solving a system of equations, guarantying interdisciplinary constraints and returning the objective functions to be manipulated by the optimiser. When integrated with a hierarchical evolutionary algorithm, this analysis takes the form as illustrated in algorithm 2.

Define design variables s parameters p , and constraints $g : Define(s, p, g_i, g_{ij})$

Define number of subpopulations (nodes) i , hierarchical levels and integrated analysis $k : Define\ (i, k)$

for all levels initialise subpopulations $(\mu_1^0, \mu_2^0, \mu_3^0, ..., \mu_n^0) : Analysis_k$

Layer 1: Uses Type 1 integrated analysis: $init(\mu_1^0) : Analysis_1$

Layer 2: Uses Type 2 integrated analysis: $init(\mu_2^0, \mu_3^0) : Analysis_2$

Layer 3: Uses Type 3 integrated analysis: $init(\mu_3^0, \mu_4^0, \mu_5^0, \mu_6^0) : Analysis_3$

loop
g=0
While stopping condition not met,

Recombine: $\lambda_R^{g+1} = reco(\mu^g)$

Mutate: $\lambda_M^{g+1} = mut(\lambda_R^{g+1})$

Evaluate candidate using specific integrated analysis type: $\lambda^{g+1} = f(\lambda_M^{g+1}) : Analysis_i$

Get output analysis a_i , parameters p , check constraints g and add

…penalty: $\lambda^{g+1} = f(\lambda_M^{g+1}) + penalty$

Select: $\mu^{g+1} = sel(\mu + \lambda)$ (plus strategy) or,

$\mu^{g+1} = sel(\lambda)$ (comma strategy)

If Multi-objective

Calculate Pareto fronts: Pareto $Pareto\ Front = Pareto(\lambda_M^{g+1})$

g=g+1
if epoch completed:

Start migration: $\mu_i^{g_k} = mig(\mu_i^{g_k} \to \mu_{i\pm1}^{g_k}) : Analysis_i$

Layer 1: Receive best solutions from layer 2 revaluate using Type 1 integrated analysis:
$(\mu_1^{g_k}, \mu_2^{g_k} \to \mu_0^{g_k}), (\mu_o^{g_k} = f\ (\mu_o^{g_k})) : Analysis_1$

Layer 2: Receive random solutions from layer 1 and best from layer 3 revaluate them using type 2 integrated analysis: $(\mu_0^{g_k} \to \mu_{1,2}^{g_k}, \mu_{3,45,6}^{g_k} \to \mu_{1,2}^{g_k}), (\mu_{1,2}^{g_k} = f\ (\mu_{1,2}^{g_k})) : Analysis_2$

Layer 3: Receive random solutions from layer 2 revaluates them using type 3 integrated analysis
$(\mu_{1,2}^{g_k} \to \mu_{3,45,6}^{g_k}), (\mu_{3,4,5,6}^{g_k} = f\ (\mu_{1,2}^{g_k})) : Analysis_3$

loop

Algorithm 2: Integrated MO-MDO formulation using EAs

This algorithm uses a hierarchical approach with three levels, on the bottom level a coarse type analysis to direct the exploration, at the top level more precise model that better describes the physics involved and at an intermediate level, a compromised balance between top and bottom layers is used. Initially the system will specify the design variables ***s***, constraints $\boldsymbol{g_i}$, $\boldsymbol{g_{ij}}$ and parameters, ***p***, then it will generate random sub-population of individuals $\boldsymbol{\mu_o}$ at each layer, then defines the number of subpopulations (nodes) ***i*** and number of hierarchical levels which for simplicity is equal to the number of analysis ***k***. Once these initial populations are generated the algorithm will go through an isolation phase where evolution occurs. During this evolution phase individuals go through the process of recombination, mutation and evaluation using integrated analysis ***k*** at the level to where they belong. The optimiser will take output analysis a_i and parameters $\boldsymbol{p}$ to guarantee satisfaction of constraints and compute the overall fitness function. If the problem is multi objective the algorithm will find the non-dominated individuals and will calculate the Pareto fronts. On a hierarchical topology with three levels, when an epoch is finished or the migration criteria is satisfied, the migration phase occurs: Layer 1 gets best solutions from Layer 2 and re-evaluates them using type of analysis one, Layer 2 gets random solutions from Layer 1 gets best solutions from Layer 3 and re-evaluates them using type of analysis two, Layer 3 gets random solutions from Layer 2 and re-evaluates them using type of analysis three. This process continues until a stopping condition is reached. These can be equal to a limited number of function evaluations, hours or a prescribed value on the fitness function.

4.3 Distributed Multi-objective -Multidisciplinary Optimisation Formulation using Hierarchical Parallel Asynchronous Evolutionary Algorithms

For simplicity, we limit our discussion to a two-discipline problem and assume that each discipline subsystem is based on a disciplinary analysis. Each disciplinary analysis takes as it inputs an individual member of population $\boldsymbol{\mu_i}$, which composed by as set of design variables $\boldsymbol{s}$ and input parameters $\boldsymbol{p}$ and produces a set of analysis outputs $\boldsymbol{a_i}$. The system level design variables set $\boldsymbol{s}$ includes local and shared variables between the disciplines. The input -output relations can be expressed as: $\boldsymbol{a_i=A_i(s,\ p)}$. One assumption is that the discipline specific analysis $\boldsymbol{a_i}$ are independently solvable. That is, given the system variables $\boldsymbol{s}$ and input parameters $\boldsymbol{p}$, each discipline analysis compute the solution $\boldsymbol{a_i=A_i(s,\ p)}$. The output of the analysis $\boldsymbol{a_i}$ includes satisfaction of local constraints $\boldsymbol{g_i}$ and data that is passed to the other discipline as parameters $\boldsymbol{p}$. If we consider an example for aero-structural wing design, given system variables $\boldsymbol{s}$ and parameters $\boldsymbol{p}$ for the geometry of the wing, the disciplinary analysis for aerodynamics computes the flow using $\boldsymbol{s}$ and $\boldsymbol{p}$ and produces an output solution $\boldsymbol{a_1}$. The input parameters $\boldsymbol{p}$, from structures to aerodynamics, include for example the wing geometry, while input parameters $\boldsymbol{p}$, from aerodynamics to structures, include aerodynamics loads. Similarly for structures given $\boldsymbol{p}$, and $\boldsymbol{s}$, the output analysis $\boldsymbol{a_2}$, which include structural responses, is computed. If the two discipline analysis is coupled with an optimisation problem, the formulation can be represented and expressed mathematically as

$$
\begin{gathered}
\min f(s,\ a_1(s,\ p),\ a_2(s,\ p)) \\
\textit{subject to:} \\
g_1(s,\ a_1(s,\ p)) \\
g_2(s,\ a_2(s,\ p)) \\
g_{12}(s,\ a_1(s,\ p),\ a_2(s,\ p))
\end{gathered}
$$

Where:
$\boldsymbol{g_1}$ and $\boldsymbol{g_2}$ are the disciplinary analysis design constraints, and $\boldsymbol{g_{12}}$ are the interdisciplinary constraints.

When integrated with a hierarchical evolution algorithm the distributed analysis takes the form illustrated in algorithm 3. Similar to section 3.2 the system uses a hierarchical approach with three levels. Initially the system will specify the design variables $\boldsymbol{s}$, constraints $\boldsymbol{g_i}$, $\boldsymbol{g_{ij}}$ and parameters $\boldsymbol{p}$, then it will generate random sub population of individuals $\boldsymbol{\mu_o}$ at each layer that are evaluated with each discipline type of analysis. Once individuals are evaluated they are returned to the system global optimiser where the interdisciplinary constraints are checked, the overall fitness function is evaluated and individuals are manipulated (recombination + mutation + selection). The process resumes in an isolation and migration phase in the same manner as described in the previous section but performing the corresponding type of disciplinary analysis at each level.

Subspace Optimisation: An alternative option for previous approach is the inclusion of a subspace optimisation using evolutionary algorithms. The main difference is that in this case individuals are optimised within each subspace EA optimiser. The approach considered was to define the lower and upper bounds in these sub-optimisations to be 10% of the value of the current design variable. Once individuals are optimised in this subspaces they are returned to the global level optimiser where the interdisciplinary constraints are checked, the overall fitness function is evaluated and individuals are manipulated (recombination + mutation + selection). The process continues in the same manner as described previously.

Define design variables s parameters p , and constraints $g : Define(s, p, g_i, g_{ij})$

Define number of subpopulations (nodes) i , hierarchical levels and integrated analysis $k : Define(i, k)$

for all levels initialise subpopulations $(\mu_1^0, \mu_2^0, \mu_3^0, ..., \mu_n^0): Analysis_k$

   Layer 1: Uses Type 1 analysis for each discipline: $init(\mu_1^0): Analysis_1$

   Layer 2: Uses Type 2 analysis for each discipline: $init(\mu_2^0, \mu_3^0): Analysis_2$

   Layer 3: Uses Type3 analysis for each discipline: $init(\mu_3^0, \mu_4^0, \mu_5^0, \mu_6^0): Analysis_3$

loop
g=0
while stopping condition not met,

     Recombine: $\lambda_R^{g+1} = reco(\mu^g)$

     Mutate: $\lambda_M^{g+1} = mut(\lambda_R^{g+1})$

     Evaluate candidate using specific analysis type: $\lambda^{g+1} = f(\lambda_M^{g+1}): Analysis_i$

     Get output analysis a_i , parameters p , check constraints g and add penalty: $\lambda^{g+1} = f(\lambda_M^{g+1}) + penalty$

     Select: $\mu^{g+1} = sel(\mu + \lambda)$ (plus strategy) or,

     $\mu^{g+1} = sel(\lambda)$ (comma strategy)

     If Multi-objective:

         Calculate Pareto fronts: Pareto $Pareto\ Front = Pareto(\lambda_M^{g+1})$

g=g+1
if epoch completed:

     Start migration: $\mu_i^{g_k} = mig(\mu_i^{g_k} \rightarrow \mu_{i\pm1}^{g_k}): Analysis_i$

     Layer 1: Receive best solutions from layer 2 revaluate using type 1 analysis for each discipline: $(\mu_1^{g_k}, \mu_2^{g_k} \rightarrow \mu_0^{g_k}), (\mu_o^{g_k} = f(\mu_o^{g_k})): Analysis_1$

     Layer 2: Receive random solutions from layer 1 and best from layer 3 revaluate them using type 2 analysis for each discipline: $(\mu_0^{g_k} \rightarrow \mu_{1,2}^{g_k}, \mu_{3,45,6}^{g_k} \rightarrow \mu_{1,2}^{g_k}), (\mu_{1,2}^{g_k} = f(\mu_{1,2}^{g_k})): Analysis_2$

     Layer 3: Receive random solutions from layer 2 revaluates them using type 3 analysis for each discipline $(\mu_{1,2}^{g_k} \rightarrow \mu_{3,45,6}^{g_k}), (\mu_{3,4,5,6}^{g_k} = f(\mu_{1,2}^{g_k})): Analysis_3$

Loop

Algorithm 3: Distributed MO-MDO formulation using EAs

4.5 Implementation of Different Legacy codes

To implement different legacy codes within the framework it can be noted that one of the benefits of EAs is that they require no derivatives of the objective function. The coupling of the algorithm with different analysis codes is by simple function calls and input and output data files. So far the implementation has been coupled with legacy codes in different programming languages C, C++, Fortran 90, and Fortran 77. The framework has been successfully coupled with the following aerodynamic and analysis software: *FLO22* [21], *FLOPS* [24], Aircraft Design and Analysis Software (ADA an in-house solver developed by the first author), *XFOIL* [12], *MSES* [13] and *CalculiX* [11].

4.6 Architectural Design and Information Access.

To satisfy the architectural design requirements the platform uses an object-oriented approach in C++. The benefits of using object-oriented software are the ease of implementation and extension of software in a modular fashion by the use of classes and methods. On an industrial and academic environment, the need for a user-friendly application is required, thus leading to the design of a simple GUI. There were many considerations and options for the GUI development, but knowledge in C++ and an object oriented principles was the main consideration, for this reason the Fast Toolkit (FLTk) library [38] was selected. This toolkit provides a friendly and easy to use environment for the implementation. The GUI is a simple and modular on its implementation and consists of five main modules as illustrated in Fig. 4.

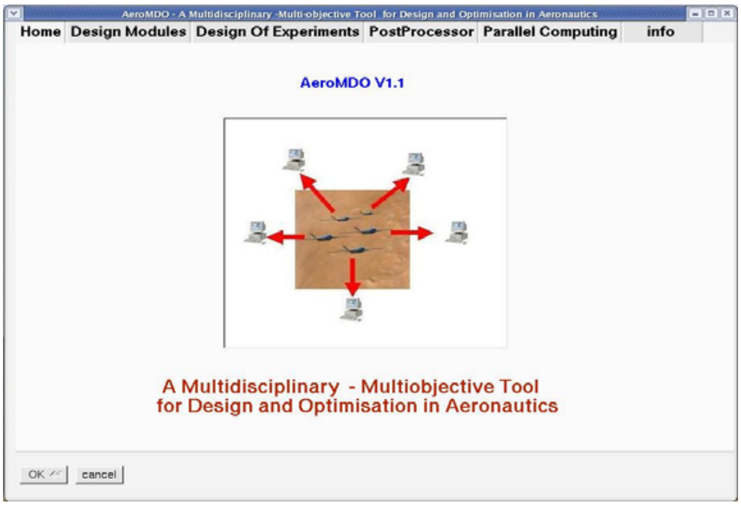

Fig. 4: GUI Sample

The GUI facilitates development, extension and modifications of modules in a rather simple manner. The user has to create only a few subroutines within the corresponding module. The GUI is configured so that the addition of design modules is simple and efficient and in such a way that the analysis codes are embedded within each design module. For example, within the aircraft design module, the user has the option of performing a single analysis or optimisation using two different analysis codes; *FLOPS* (Flight Optimisation system) and ADA. The main modules in this GUI are: Design and Analysis Module, Design of Experiments Module, Post Processing Module and Parallel Processing Module.

4.6.1 Design and Analysis Module

As illustrated in Fig. 5, the Design Module allows the user to conduct a single design and optimisation for different aeronautical applications and a series of mathematical test cases. So far, this module consists of five sub-modules for aerofoil, multi-element aerofoil, nozzle, wing, aircraft and mathematical test functions. Each of this will be described in detail in the following sub-sections. As developed the framework is flexible and provides for ease of implementation of other design modules such as those for propeller, cascade aerofoils or rotor blade design.

4.6.2 Development of Aeronautical Design Modules

Before using an analysis codes within the optimisation it is necessary to develop a design module interface. These comprise a series of files written in C++, which allow communication between the analysis codes, parallel processing architecture, the GUI and the optimiser. When designing the interfaces, a choice has to be made depending if the source code for the analysis tool was available or not. In the current implementations minimal modification to the source code was required, ideally it is desirable to operate only through the specific input/output files of the analysis tool. In all the implementations considered, a design template was used in conjunction with on or two additional files that contain the necessary linking subroutine allowing a rather fast implementation of the design modules. So far, there are subroutines for aircraft, nozzle, wing and full aircraft configuration. Each of these options allows the user to perform a single design analysis or a full optimisation. A general algorithm for the implementation of a new design module is represented in algorithm 3.

```
for ;;  //Infinite loop
   Receive information from optimizer: O:O → X, g_i, D // X Design variables, g_i
   constraints and D Aerodynamic Data
   Generate Geometry: G = f(X), G = (aerofoil, wing nozzle, aircraft shape)
   for i=0,n // N Number of objectives
      Evaluate: ℑ(G, g_i, D)// Analyse candidate geometry
      Check for convergence
      Calculate fitness: f_i = f(ℑ(G, g_i, D))
      Return the computed individual (Design variables + fitness vector to
      optimizer): X + f_i → O
   end loop
end
```

Algorithm 3: Design Modules Algorithm

Wing Design Modules: This module allows the user to conduct a single wing analysis or optimisation studies, these include single, multi-objective or multidisciplinary optimisation. Fig. 5 illustrates this module. Details on the analysis tools used within this module and applications on multi-objective and multidisciplinary wing design are presented in Section 5.

Aircraft Design and Optimisation: This module allows the user to analyse and optimise different problems in aircraft design. The user can select from two different analysis codes: An object oriented Aircraft Design and Analysis Software (ADA) developed by the first author or using the Flight Optimisation System (*FLOPS*) code [24] developed by Arnie McCullers at NASA Langley. ADA is a simple conceptual design and analysis software written using object oriented principles and is based on the formulation described in Reference [31]. *FLOPS,* a more robust solver, is a workstation-based code has capabilities for conceptual and preliminary design and evaluation of advanced design concepts. The sizing and synthesis analysis in *FLOPS* are multidisciplinary in nature. It has a numerous modules for noise, detailed takeoff, performance, structures, control, aerodynamics and other capabilities; it is used in some universities for MDO development as well as aerospace firms and government. It allows an integral analysis for the entire mission and the calculation of aircraft performance parameters such as range, endurance takeoff field length and landing field length. *FLOPS* has capabilities for drag estimation using empirical techniques and historical data, but it also allows input for externally generated aerodynamic data.

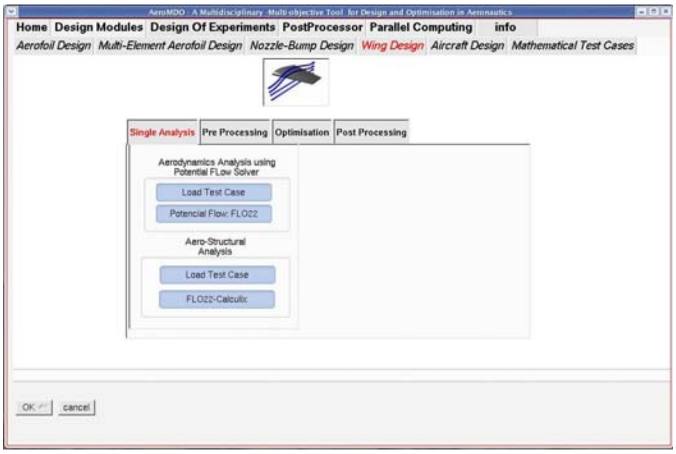

Fig. 5: Wing Design and Optimisation Module

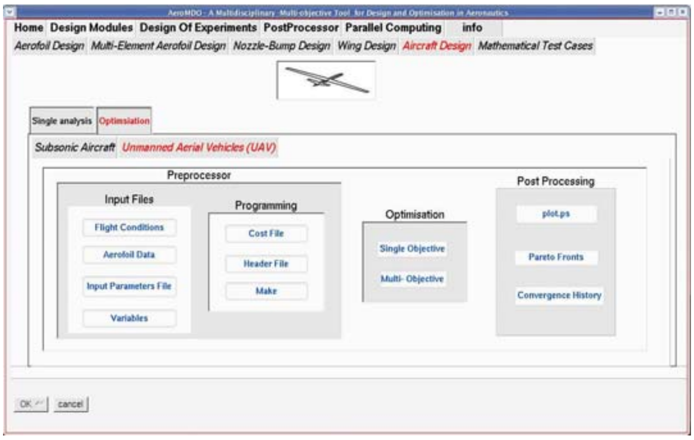

Fig. 6: Aircraft Design Module

Within the framework, different types of aircraft can be designed and optimised including subsonic, Unmanned Aerial Vehicles, transport or supersonic aircraft. Single or multi-objective optimisation studies can be performed; including comparison on multi-objective approaches such as Pareto optimality and Nash equilibrium approaches [18, 27]. Fig. 6 illustrates this module.

Aerofoil Design and Optimisation: This module allows the user to perform a single analysis or a full aerofoil optimisation routine, the user can choose from a combination of three different analysis codes: A panel method (XFOIL) an Euler + boundary layer (*MSES*) and a Navier-Stokes solver (*NSC2ke* [26])

Multi-element Aerofoil Design and Optimisation: Similar to the aerofoil design module this module allows the user to perform a single analysis or a full optimisation, the user can choose from an Euler or Navier-Stokes solution.

Nozzle-Bump Design and Optimisation: The Nozzle -Bump design module allows a single two-dimensional analysis or optimisation using the CUSP scheme developed by Srinivas [37].

Mathematical Test Functions: It is important to test the robustness and performance of an optimisation method before deciding on its application to real world problems. This module allows the user to design, and evaluate single, or multi-objective mathematical test functions. The current implementation includes mathematical test function for single or multiple objectives, constrained optimisation, DOE and goal programming problems.

4.6.3 Design of Experiments Module

In the implementation considered in this research, the optimiser uses an EA for the optimisation, but one of the drawbacks of EAs is that they suffer from slow convergence. By providing a DOE capability into the framework, the aim is to hybridise the desirable characteristics of EAs and surrogate models such as RSM, and obtain an efficient optimisation system. Within this context, the DOE samples a number of design candidates at which the analysis code (CFD) will run. The surrogate model is then constructed for the computationally expensive problem. In this option, the user can define and choose from different sampling and DOE strategies such as Latin hypercube, Response Surface Methods or DACE/Kriging. There is sufficient literature and software developed specifically for DOE, after a careful selection of software packages it was decided to implement the DACE tool box [23] which is robust and allows different options for sampling strategies and DOE. This software was ported to Octave (a mathematical package common in most UNIX installations) and integrated with the framework but if desired a different DOE method can be implemented.

4.6.4 Parallel Computing Module

This module allows the users to dynamically create, add or delete nodes on the parallel implementation. Recent work on multi-objective parallel evolutionary algorithms has allowed significant performance and robustness gains in global and parallel optimisation [8]. In the implementation considered in this research, the parallel environment used is a cluster of PCs, wherein the master carries on the optimisation process while remote nodes compute the solver code. The message-passing model used is the Parallel Virtual Machine (PVM) [14]

4.6.5 Post Processing

Post processing capabilities are embedded within each module; this is due to the fact that different optimisation and analysis tools produce and might require different visualisation techniques. The approach considered was to use the full benefits of the visualisation characteristics embedded within each analysis software, and the use of the GNUplot (a graphics software common in most UNIX installations). Common to all design modules is of the fitness function and Pareto fronts for multi objective problems. Post-processing tools on each analysis module include a top view of the wing planforms and a general 3D view of the resulting aircraft configurations. Visualisation tools within each analysis software include the pressure coefficient distribution on the aerofoil using an Euler+BL solver (*MSES*), or pressure and Mach contours using Navier Stokes solver (NSC2ke). Examples of some of these tools are presented in the next section.

5. Applications

The methodology has been applied to several real world problems with different complexities including inverse and direct problems for aerofoil design, complex multi-element aerofoil design, parallel computation in aeronautics, and multidisciplinary and multi-objective wing and aircraft design [16, 17, 18, 44]. To illustrate these concepts, we consider a multidisciplinary, multi-objective optimisation of a swept forward wing design for an Unmanned Aerial Vehicle (UAV). The two objectives are minimisation of wave drag (C_D) and wing weight (W_{sc}). The cruise Mach number and altitude are 0.69 and 10000 *ft*. The wing area is set to 2.94 m^2 and the corresponding C_L is fixed at 0.19. For the solution we initially compute the pressure distribution over the wing using a potential flow solver to obtain the wing aerodynamics characteristics that include the span-wise pressure distribution, C_L and total drag coefficients C_D. The lift distribution is replaced by concentrated loads and the spar cap area is calculated to resist the bending moment. The weight is then approximated as the sum of the span-wise cap weight. The interaction between the aerodynamic pressure distribution and the structural deflections is ignored.

5.1 Design Variables and Constraints.

The wing geometry is represented by three aerofoil sections and nine variables for the wing planform. The aerofoil geometry is represented by the combination of a mean line and thickness distribution, which is very common concept in classical aerodynamics [1]. Both lines are represented by Bézier curves with leading and trailing edge points fixed at (0.0,0.0) and (1.0,0.0) respectively, and a variable number of intermediate control points whose *x*-positions are fixed in advance and whose *y*-heights form the problem unknowns. In this case we take six free control points on the mean line and ten free control points on the thickness distribution. In total fifty-nine design variables are used for the optimisation. Fig. 7 illustrates the main design variables and Table 1 indicates their upper and lower bounds. Constraints are imposed on minimum thickness ($t/c \geq 0.14$ root aerofoil, 0.12 intermediate aerofoil, 0.11 tip aerofoil) and position of maximum thickness. ($20\% \leq t/c \leq 55\%$). If any of these constraints is violated both fitness are linearly penalised to ensure an unbiased Pareto set.

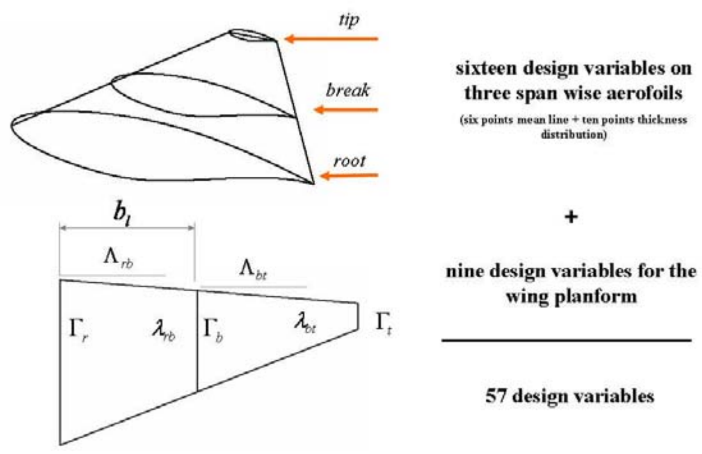

Fig. 7: Design variables for multidisciplinary wing design.

Table 1:Upper and lower bounds for multidisciplinary wing design variables.

Description	Lower Bound	Upper Bound
Wing Aspect Ratio [AR]	3.50	7.00
Break to root Taper [λ_{br}]	0.65	0.80
Break to tip Taper [λ_{bt}]	0.20	0.45
Wing 1/4 Chord inboard Sweep, deg [Λ_i]	10.00	20.00
Wing 1/4 Chord outboard Sweep, deg [Λ_o]	-20.00	0.00
Twist at Root, deg [Γ_r]	0.00	3.00
Twist at Break, deg [Γ_b]	-1.00	0.00
Twist at Tip, deg [Γ_t]	-1.00	0.00
Break Location, [b_l]	0.20	0.35

5.2 Fitness Functions

The two fitness functions to be optimised are defined as:

$$\min(f_1): \quad f_1 = c_{dwave} \tag{10}$$

$$\min(f_2): \quad f_2 = \sum W_{sc} \tag{11}$$

5.3 Aerodynamics and Weight Analysis

The aerodynamic characteristics of the wing configurations are evaluated using a three dimensional full potential wing analysis software (*FLO22)* This program uses sheared parabolic coordinates and accounts for wave drag [21]. *FLO22* was developed by Prof. Antony Jameson (NYU) and Prof. Dave Caughey (Cornell) for analysing inviscid, isentropic, transonic flow past 3-D swept wing configuration. Some details on the algorithm is that the free stream Mach number is restricted by the isentropic assumption and that weak shock waves are automatically located where ever they occur in the flow. Also the finite difference form of the full equation for the velocity potential is solved by the methods of relaxation, after the flow exterior to the aerofoil is mapped to the upper half plane. The mapping procedure allows exact satisfaction of the boundary conditions and use of supersonic free stream velocities. Details on the formulation and implementation can be found in [21].

The lift can be satisfied by performing an extra two function evaluations by varying the angle of attack at the wing root and assuming a linear variation of the lift coefficient. The lift distribution is replaced by concentrated loads. The wing weight is estimated from the wing spar cap area designed to resist the bending moment. The local stress has to be less than the ultimate tensile stress in this case for Aluminium Alloy 2024 -T6 $\leq \sigma_{ult}$.

5.4 Implementation

In this problem we use the wing design and analysis module, using two approaches, the first approach uses a traditional EA with a single population model and computational grid of 96 *x* 12 *x* 16. The second approach uses a hierarchical topology of CFD resolutions with the following settings:

Top Layer: A population size of 30, intermediate recombination used between two parents, and a mesh of 96 *x* 12 *x* 16.
Middle Layer: A population size of 30, discrete recombination used between two parents, and a mesh of 72 *x* 9 *x* 12.

Bottom Layer: A population size of 30, discrete recombination used between two parents, and a maximum of 48 *x* 6 *x* 8.

Using the parallel commuting module six machines were used in the calculation.

5.5 Numerical Results

The algorithm was run five times for 2000 function evaluations and took in average six hours to compute. Fig. 8a shows convergence history for objective one and Fig. 8b shows the Pareto fronts obtained by using the two approaches. It can be seen how the optimisation technique gives a uniformly distributed front in both cases. By inspection we can see that the use of a multi-fidelity approach gives an overall lower front as compared to a single model approach. For illustration purposes a compromise design, Pareto member ten (PM10), taken from the middle of the Pareto set is taken for evaluation. Fig. 9a shows the root, break and tip aerofoils and Fig. 9b the wing geometry and Table 2 indicates the final design variables.

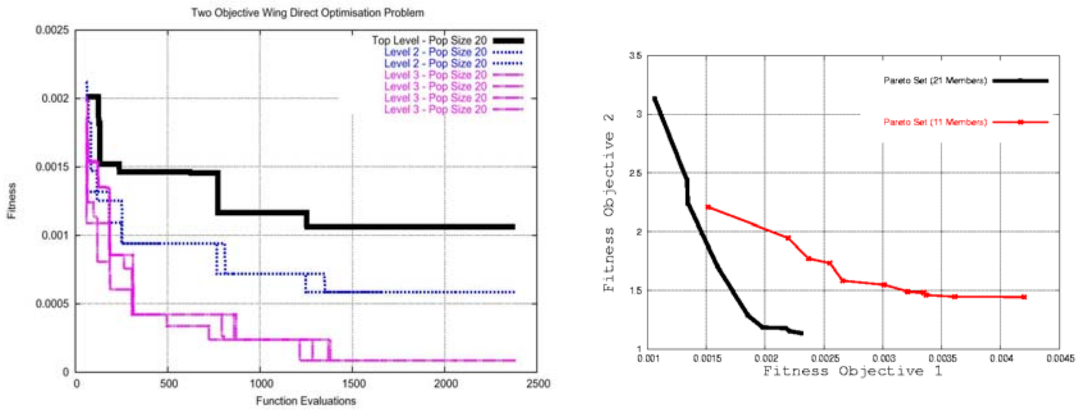

Fig. 8: a) Convergence history for objective one b) Pareto fronts after 2000 function evaluations.

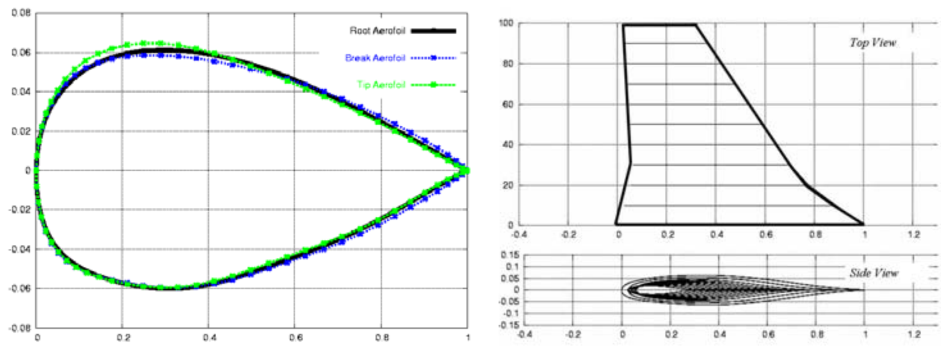

Fig. 9: a) Aerofoil sections (root- break and tip) on Pareto member ten.
b) Wing top and side view for Pareto member ten.

Table 2: Optimum design variables for UAV wing

Description	Pareto Member 10
Wing Aspect Ratio [AR]	3.5
Wing 1/4 Chord inboard Sweep, deg [Λ_i]	10.2
Wing 1/4 Chord outboard Sweep, deg [Λ_o]	-1.9
Lift to Drag Ratio [L/D]	146.62
Lift coefficient, CL	0.1970
Drag Coefficient, CD	0.0013+Cdo

These results show a computational gain on using a hierarchical topology of fidelity models as compared to a single model during the optimisation. The algorithm was capable of identifying the trade-off between the multi-physics involved and provides classical aerodynamic shapes as well as alternative configurations from which the designer can choose and proceed into more detailed phases of the design process.

6. Conclusions

This paper presented the requirements, formulation and implementation of a robust framework in which different aeronautical problems can be analysed. The paper gave a brief description of the different components of the framework. These included several algorithms and discussion on a graphical implementation of different modules for design, optimisation post-processing and parallel computing. Hence we have within the framework, a complete set of numerical tools for handling mathematical test functions and real world problems in aeronautics. The methodology was illustrated on its application of to a wing multi-objective and MDO problem showing the benefits of the method. The method was capable of identifying the trade-off between the multi-physics involved and provide classical aerodynamic shapes as well as alternative configurations from which the designer can choose. It was observed that there was a computational gain on using a hierarchical topology of fidelity models as compared to a single model during the optimisation.

Acknowledgments

The authors gratefully acknowledge Mourad Sefrioui, Dassault Aviation for fruitful discussions on Hierarchical EAs and his contribution to the optimisation procedure, and also Steve Armfield at the University of Sydney for allowing access to the cluster of computers. We would like to thank Arnie McCullers at NASA LaRC who kindly provided the *FLOPS* software, M. Drela at MIT for providing the *XFOIL* and *MSES* software and A. Jameson and Prof Shigeru Obayashi for the use of the *FLO22* software .

References

[1] I. H. Abott and A. E. Von Doenhoff. Theory of Wing Sections. Dover, 1980.

[2] N. M. Alexandrov and E. M. Lewis, Comparative Properties of Collaborative Optimization and Other Approaches to MDO, Proceedings of the First ASMO UK /

ISSMO Conference on Engineering Design Optimization, July 8-9, La Jolla, California, July 8-9, 1999.
[3] N. M. Alexandrov and R. M. Lewis, Analytical and Computational Properties of Distributed Approaches to MDO, AIAA 2000-4718, September 2000.
[4] P. Bartholomew "The Role of MDO within Aerospace Design and Progress towards an MDO Capability, AIAA-98-4705, pp 2157-2165, 7th AIAA/USAF/NASA/ ISSMO Symposium on Multidisciplinary Analysis and Optimization, AIAA, St. Louis, Mo, 1998.
[5] J. A Bennett, M.E. Botkin, C. Koromilas, R. V. Lust, M.O. Neal, J.T. Wang and R.I. Zwiers, A Multidisciplinary Framework for Preliminary Vehicle Analysis and Design, Multidisciplinary Design Optimization: State of the Art, Alexandrov, N. and Hussaini, M. Y., editors, 1996.
[6] A. J. Booker, J. E. Dennis, Jr., P. D. Frank, D. B. Serafini, V. Torczon, and M. W. Trosset. A Rigorous Framework for Optimization of Expensive Functions by Surrogates. Structural Optimization, 17(1): 1--13, 1999.
[7] R. Braum and P. Gage and I. Kroo and I. Sobieski, "Implementation and Performance Issues in C.O, NASA-AIAA-96-4017, 1996.
[8] E. Cantu-Paz, Efficient and Accurate Parallel Genetic Algorithms. Kluwer Academic Pub, 2000.
[9] C.A. Coello Coello, D.A. Van Veldhuizen and G.B. Lamont, Evolutionary Algorithms for Solving Multi-Objective Problems, Kluwer Academic Publishers, New York, March 2002.
[10] K. Deb, Multi-Objective Optimization Using Evolutionary Algorithms, Wiley, 2003.
[11] G. Dhondt and K. Wittig, CalculiX: A Free Software Three-Dimensional Structural Finite Element Program, http://www.calculix.de.
[12] M. Drela. XFOIL 6.94 User Guide. MIT Aero Astro, 2001.
[13] M. Drela. A User's Guide to MSES V2.3," Feb. 1993.
[14] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek and V. Sunderam. PVM: Parallel Virtual Machine. A User's Guide and Tutorial for Networked Parallel Computing. Massachusetts Institute of Technology, 1994
[15] D. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley, 1989.
[16] L. González, E. Whitney and K. Srinivas and J. Périaux. "Multidisciplinary Aircraft Design and Optimisation Using a Robust Evolutionary Technique with Variable Fidelity Models" AIAA Paper 2004-4625, In CD Proceedings 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Aug. 30 - Sep. 1, 2004, Albany, NY.
[17] L.F Gonzalez, E.J. Whitney, J. Periaux, M. Sefrioui and K. Srinivas. "A Robust Evolutionary Technique for Inverse Aerodynamic Design", Design and Control of Aerospace Systems Using Tools from Nature. Proceedings of the 4th European Congress on Computational Methods in Applied Sciences and Engineering, Volume II, ECCOMAS 2004, Jyvaskyla, Finland, July 24-28, 2004 editor: P. Neittaanmaki and T. Rossi and S. Korotov and E. Onate and J. Periaux and D. Knorzer, University of Jyvaskyla, Jyvaskyla, 2004 pages: CD ISBN 951-39-1869-6.
[18] L. F. González, E. J. Whitney, K. Srinivas, K. C. Wong and J. Périaux "Multidisciplinary Aircraft Conceptual Design Optimisation Using a Hierarchical Asynchronous Parallel Evolutionary Algorithm (HAPEA)" In I.C. Parmee, editor, Proceedings of the Sixth International Conference on Adaptive Computing in Design and Manufacture (ACDM'2004), volume 6, Bristol, UK, April 2004. Springer-Verlag.
[19] N, Hansen, A. Ostermeier, Completely Derandomized Self-Adaptation in Evolution Strategies, In Evolutionary Computation, 9(2): 159—195, 2001.
[20] J.H. Holland. Adaption in Natural and Artificial Systems. University of Michigan Press, 1975.

[21] A. Jameson, D. Caughey, P. Newman and R. Davis. A Brief Description of the Jameson Caughey NYU Transonic Swept-Wing Computer Program *FLO22*. NASA Technical Memorandum, NASA TM X-73996, Dec. 1976.
[22] I. Kroo and S. Altus and R. Braun and P. Gage and I. Sobieski,"Multidisciplinary Optimization Methods for Aircraft Preliminary Design", AIAA 94-4325, Fifth AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, September 7-9, Panama City, Florida, 1994.
[23] S.N. Lophaven, H. B. Nielsen and J. Sondergaard, Aspects of the Matlab Toolbox DACE, 2002, 44, IMM-TR-2002-13
[24] A. McCullers. FLOPS User's Guide, Release 6.02, NASA Langley Research Center, March 2003.
[25] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs. Artificial Intelligence. Springer-Verlag, 1992.
[26] B. Mohammadi. Fluid Dynamics Computation with NSC2KE, User-Guide, Release 1.0. Technical Report RT-0164, INRIA, May 1994.
[27] J. F. Nash. Equilibrium Points in N-Person Games. In Nat. Acad. Sci, *USA*, pages 36:46-49, 1950.
[28] S. Obayashi. Multidisciplinary Design Optimization of Aircraft Wing Planform Based on Evolutionary Algorithms. In Proceedings of the 1998 IEEE International Conference on Systems, Man, and Cybernetics, La Jolla, California, IEEE, October 1998
[29] A. Oyama, M. -S. Liou, and S. Obayashi. Transonic Axial-Flow Blade Shape Optimization Using Evolutionary Algorithm and Three-Dimensional Navier-Stokes Solver, 9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Atlanta, Georgia, September, 2002.
[30] I. Parmee and A. H. Watson. Preliminary Airframe Design Using Co-Evolutionary Multiobjective Genetic Algorithms. In W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela and R. E. Smith, editors, Proceedings of the Genetic and Evolutionary Computation Conference, volume 2, pages 1657-1665, Orlando, Florida, USA, Morgan Kaufmann, July 1999.
[31] D. Raymer "Aircraft Design: A Conceptual Approach", American Institute of Aeronautics and Astronautics American Institute of Aeronautics and Astronautics, Third Edition, 1999
[32] A. O. Salas and J. C. Townsend, Framework Requirements for MDO Application Development, 7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, St. Louis, Missouri, AIAA 98-4740, September 2-4, 1998, pp. 11.
[33] M R. Sankar, I. Amitay, PM Mujumdar PM, K. Sudhakar, "MDO Framework Development - A Case Study With An Elementary Model Of Airborne Early Warning System Optimization", 9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Atlanta, Georgia, September 4-6, 2002
[34] M. Sefioui, J. Periaux, J.-G. Ganascia. Fast convergence thanks to diversity. Evolutionary Programming V. Proc. of the 5th Annual Conference on Evolutionary Programming. L.J.Fogel, P.J.Angeline and T.Back editors. MIT Press, 1996.
[35] M. Sefrioui and J. Périaux. A Hierarchical Genetic Algorithm Using Multiple Models for Optimization. In M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J.J. Merelo and H.-P. Schwefel, editors, Parallel Problem Solving from Nature, PPSN VI, pages 879-888, Springer, 2000.
[36] J. Sobieski, RT Haftka, Multidisciplinary Aerospace Design Optimization: Survey of Recent Developments, AIAA Paper No. 96-0711, 1996.
[37] K. Srinivas. Calculation of Cascade Flows by a Modified CUSP Scheme, Computational Fluid Dynamics Journal, 1999,2,pp 285—295.

[38] M. Sweet, C.P Earls and B. Spitzack. FLTK 1.1.6 Programming Manual, Revision 6. www.fltk.org.
[39] Z. Thomas and A. Green, "Multidisciplinary Design Optimization Techniques: Implications and Opportunities for Fluid Dynamics Research" AIAA Paper-1999-3798, Jun, 1999
[40] D. A. Van Veldhuizen, J.B. Zydallis and G. B. Lamont. Considerations in Engineering Parallel Multiobjective Evolutionary Algorithms, IEEE Transactions on Evolutionary Computation, Vol. 7, No. 2, pp. 144--173, April 2003.
[41] J. Wakunda J, A. Zell. Median-selection for parallel steady-state evolution strategies. In Marc Schoenauer, Kalyanmoy Deb, Günter Rudolph, Xin Yao, Evelyne Lutton, Juan Julian Merelo, and Hans-Paul Schwefel, editors, Parallel Problem Solving from Nature – PPSN VI, pages 405–414, Berlin, Springer, 2000.
[42] R P. Weston, J. C. Townsend, T. M. Eidson and R. L. Gates, A Distributed Computing Environment for Multidisciplinary Design**,** 5th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Panama City, FL, AIAA 94-4372, September 7-9, 1994.
[43] E. J. Whitney. A Modern Evolutionary Technique for Design and Optimisation in Aeronautics. PhD Thesis, The University of Sydney, 2003.
[44] E. Whitney, M. Sefrioui, K. Srinivas, J. Périaux: "Advances in Hierarchical, Parallel Evolutionary Algorithms for Aerodynamic Shape Optimisation", JSME (Japan Society of Mechanical Engineers) International Journal, Vol. 45, No. 1, 2002.