DOI: 10.13009/EUCASS2017-204

Flight Control Laws Carefree Handling Clearance of a Highly Manoeuvrable Aircraft using Multi-Strategy Adaptive Global Optimization

R. Rodríguez Robles, M. Sabarís Boullosa* and F. Asensio Nieto**
**Airbus Defence and Space*
Paseo John Lennon, S/N, Getafe 28906, Spain

Abstract
This paper describes a global optimization algorithm developed for the carefree handling clearance of highly manoeuvrable aircraft. The algorithm; based on game theory; includes several players with different strategies that are automatically adapted at each game turn depending on its success. The cost function, which provides a mapping between optimization degrees of freedom and the non-linear simulator outputs, is presented and guidelines for its generation are detailed. The algorithm is applied to the Eurofighter Typhoon Flight Control Laws carefree handling clearance and the results and conclusions are presented as well as some indications for future development and further applications.

1. Introduction

Since the introduction of the first non-experimental digital fly-by-wire system in the F-8 Crusader [1], the exponential growth of the embedded control systems computing power has enabled the military aerospace industry to develop more efficient and agile aircraft. Generally, modern fighters are designed to be naturally unstable for the sake of manoeuvrability and agility, requiring the use of flight control laws (FCL) to artificially stabilize the aircraft dynamics and to provide diverse safety-critical flight envelope protections functions, along with automatic or semi-autonomous flight modes. To ensure the safe operation of the aircraft and the correct functioning of the FCL, a clearance assessment must be performed. The FCL clearance encompasses a set of activities on which the controller robustness against plant un-modelled dynamics and sensors measurement errors is assessed for every flight condition and pilot/operator commands that lie within the target operating envelopes, and for every aircraft system performance status, including all possible detected and un-detected failures that might occur during the life of the aircraft.

Many are the studies and investigations performed in the last decades addressing the FCL clearance from a global optimization problem perspective [2-3]. FCL clearance based on global optimization has demonstrated to be a very effective technique to increase the probabilities of finding hidden problems arising from parameters combinations and manoeuvres that are not covered in the framework of a typical grid-based FCL clearance. Derivative-free evolutionary algorithms like Differential Evolution (DE) [4] combined with local optimization algorithms provided very promising results with the ADMIRE model [5]. Nevertheless, after performing extensive benchmarking tests with a cost function that handles the flight test matched nonlinear simulator of the Eurofighter Typhoon, it was concluded that the extremely nonlinear aerodynamics and the FCL complexity of a real high performance fighter aircraft, make algorithms like DE, PSO [6] and ICA [7] to be not suitable for some of the FCL clearance problems.

This work presents a novel single-objective global optimization algorithm developed by Airbus Defence and Space S.A.U. with the aim to increase the probabilities of finding hidden and very local issues in the FCL of high performance combat aircraft. This algorithm, called Multi-strategy Adaptive Global Optimization (MAGO), solves the optimization problem using a cooperative sequential game between independent players, whose strategies are adapted in each game turn to maximize the probabilities of finding the global optimum of a cost function. With this approach, MAGO algorithm is able to iteratively converge to the best optimization strategy to solve a single-objective optimization problem. When applied to the FCL carefree handling clearance, MAGO algorithm has proved to be capable of finding the worst-case parameter combination that could make an aircraft depart from controlled flight or exceed any safety or clearance limit, with a higher confidence level than with other global optimization algorithms.

DOI: 10.13009/EUCASS2017-204

In this paper, we will briefly introduce the FCL clearance of a highly manoeuvrable aircraft, justifying the need to apply global optimization techniques to increase the probability of finding hidden FCL issues and unsafe violations of the flight envelope protection functions. Afterwards, the MAGO algorithm will be introduced describing its internal structure and its advantages compared to other cutting-edge global optimization algorithms like SPS-L-SHADE-EIG [8]. Lessons learnt and recommendations to consider when applying global optimization techniques in the FCL clearance will be presented. Maturity and performance of the developed optimization tool will be assessed and compared to other global optimization algorithms in a real FCL clearance case study. Finally, conclusions and future lines of work will be exposed.

2. Flight Control Laws Clearance

The FCL clearance can be defined as a model-based non-linear robustness assessment problem, with a large number of parameters defining the aircraft systems status and the operational conditions, with hundreds of “unknowns” and un-modelled dynamics that must be covered in the assessment work via tolerances. The main objective is to define the conditions and the flight envelope areas where the aircraft can be operated within the safety levels required by the airworthiness regulations.

FCL clearance activities can be categorized in three major groups: stability assessments, nonlinear offline simulations and manned simulations. The first group is composed by all the clearance tasks aimed at evaluating the robustness of the controller in terms of stability margins using linearized models, describing functions methods [9], and more generalized nonlinear stability assessment methodologies [10-11]. The other two major tasks rely mainly on complex nonlinear simulations models to assess the air vehicle flying and ground handling qualities, and to check the correct functioning and performance of the FCL flight envelope protections functions. In modern airliners and high performance military aircraft, these protection functions reduce the required pilot workload in complex scenarios with high-demanding tasks and provides Full Carefree Handling (FCH) capabilities to the aircraft, allowing the pilot to perform aggressive manoeuvres with fast variations on the stick, throttle and pedal inputs, without the need to observe and care for the aircraft structural and flight envelope limits.

The main objective of the offline nonlinear simulations assessments is to check the risk of departures from controlled flight for every condition within the aircraft flight envelope. To meet this goal, all the flight mechanics parameters that describe the aircraft dynamics and the embedded systems states are monitored in the nonlinear simulations to ensure that the exceedances of the flight protections limits remain bounded within the carefree limits, granting a low risk to encounter unrecoverable departures and to over-stress the airframe. The great amount of variables and parameters defining the aircraft systems status, flight conditions, model tolerances and pilot inputs, makes the FCL clearance a challenging task that requires a large number of super-computing resources to cope with the tight time-constraints of the FCL verification and validation process. Moreover, the complexity of the FCL clearance further increases in the particular case of FCL designed to provide FCH capabilities.

To solve this robustness assessment problem, it is customary in the aerospace industry to apply a deterministic grid-based approach, discretizing the problem space and simplifying the amount of combinations of cases to be analysed using engineering judgement and lessons learnt from previous FCL clearance assessments. Usually, this approach does not ensure to find all the clearance problems because worst results are not always obtained with the extreme combination of the inputs due to the high nonlinear FCL functions, and due to nonlinearities present in the physical models. The deterministic grid-based approach is suitable to provide an overview of the clearance problem, but it does not guarantee to find the worst combination of the inputs, because not all the inputs and tolerance combinations are explored and there is a discrete variation of the inputs, in spite of which it consumes a considerable amount of supercomputing resources. The solution to cope with all these drawbacks is to incorporate global optimization methodologies to the mainstream FCL clearance process in order to search quickly for any input combination in the whole continuous space of the optimization variables. The MAGO algorithm has been developed for such purpose, and has been added to the mainstream FCH clearance process performed by Airbus Defence and Space to increase the probability of successfully finding FCL issues, saving a considerable amount of computation time. Moreover, due to the efficiency and speed of the algorithm, it can be used at the first stage of the FCL clearance process in order to guide the definition of the grid-based process and the simplification of the inputs to be considered.

DOI: 10.13009/EUCASS2017-204

3 MAGO Algorithm

The MAGO algorithm is a population-based meta-heuristic single-objective global optimization algorithm developed by Airbus Defence and Space as a versatile optimization tool to be applied in the FCL clearance, with the main objective of detecting the conditions, tolerances combinations and pilot inputs that leads to the worst-case breach of the different safety and clearance requirements. As all population-based optimization algorithms, MAGO uses a population $X \in \mathbb{R}^{N \times D}$ of $N \in \mathbb{N}_{>0}$ individuals to search for the global minimum of a function $f: x \in \mathbb{R}^D \to \Delta \in \mathbb{R}$ in a hyper-cube defined by $lb \leq x \leq ub$, where $D \in \mathbb{N}_{>0}$ is the problem dimension, lb and ub are the lower and upper bounds of the optimization space respectively, and Δ is the cost. The population of candidate solutions evolves in each optimization generation g until reaching an exit criterion, usually the maximum number of function evaluations, a critical parameter due to the limited time and available supercomputing resources to perform a FCL clearance.

One of the most challenging problems for the end-user of any global optimization algorithms is the selection of the algorithm configuration parameters that maximizes the opportunities to find the global minimum of a cost function f. In black-box optimization problems, there is little or no previous information about the cost function topology, so the proper selection of the optimizer configuration parameters becomes a complex task, and usually requires multiple trial and error tests. To overcome this handicap, the optimization algorithm should be able to adapt itself throughout the optimization process by using information about the obtained improvements in the population cost as the algorithm progresses with the optimization. Finding the best optimization algorithm parameters is considered to be part of the algorithm design, and not part of its application. The aim is to have a robust well-performing algorithm with the minimum number of parameters to be selected by the end-user. This basic idea has been used in MAGO to elaborate a novel concept of smart adaptive behaviour using notions and theories from cooperative games.

3.1 Population Initialization

In all population-based optimization algorithms, the spatial distribution of the initial population X^0 has a meaningful effect on the performance of the optimization [12]. One of the simplest ways to improve the algorithm convergence to the global optimum is to uniform the distribution of the initial population. However, in many cases, the distribution of samples created with pseudo-random generators is not uniform in every dimension. In this point, quasi-random numbers are a good alternative as they share many similarities with pseudo-random numbers, but they are deterministically chosen based on low-discrepancy sequences. An advantage of these sequences is that they fill the D-dimensional optimization space more uniformly, and thus, an initial population created using these sequences will provide a more effective exploration of the optimization space.

Many are the studies that have shown low-discrepancy sequences like Sobol or Halton sequences lose efficiency filling the space when the dimension of the generated samples increases [13-14]. To overcome this handicap, MAGO uses a stratified sampling without replacement (SSWR) method with a low-discrepancy master sequence, in this case a Van der Corput sequence of base 2 [15-16]. As shown in Figure 1, the stratified sample without replacement of a Van der Corput sequence fits better the probability density function of a uniform distribution $U(0,1)$, with a lower spatial discrepancy than the samples obtained with a pseudo-random generator.

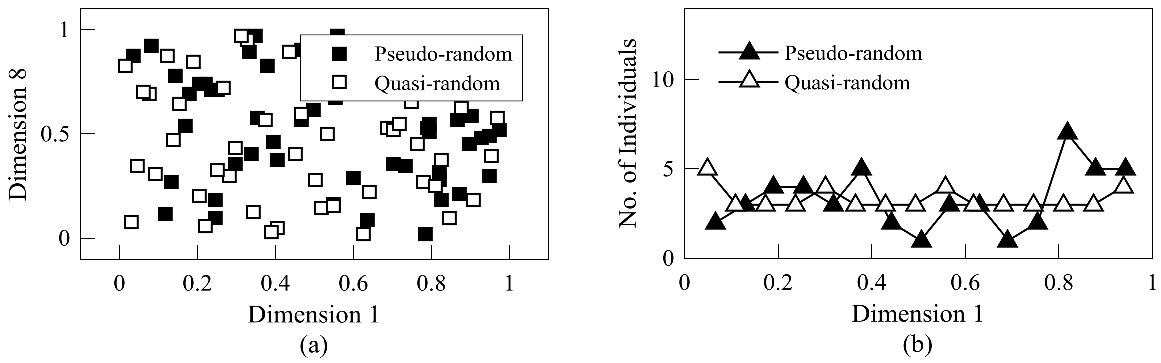

Figure 1: (a) Comparison of the 1st and 8th dimension of 50 individuals with $D = 10$ generated with a pseudo-random number generator and a SSWR of a Van der Corput sequence. (b) Density of individuals along the 1st dimension.

To generate N vectors of D quasi-random numbers, a Van der Corput master sequence $Q = \{q_1, q_2, \dots, q_r, \dots, q_D\} \in \mathbb{R}^D$ that follows a uniform probability density function $U(0,1)$ is computed. The q_r component of this sequence is

DOI: 10.13009/EUCASS2017-204

defined as $q_r = \sum_{i=0}^{\zeta} a_{\zeta-i}/2^{i+1}$ with $r = \sum_{i=0}^{\zeta} a_i \cdot 2^i$, where a_i is the coordinates of the integer number r expressed in base 2, and ζ is the lowest integer that makes $a_i = 0$, for $i > \zeta$, and it is equal to the integer part of $\ln(r)/\ln(2)$. Once the Q master sequence has been computed, MAGO generates the initial population X^0 mapping N random permutations of the elements q_r in Q to the optimization space.

3.2 The Game-based Adaptive Optimization Method

The optimization process in the MAGO algorithm is conceived as a communication-allowed sequential non-zero sum cooperative game with a finite set $P = \{1,2,\ldots,m\}$ of $m \leq \max\{1, \lfloor N/10 \rfloor\} \in \mathbb{N}_{>0}$ players, which are defined as independent entities that control the behaviour of subgroups of the population of candidate solutions. In each game turn g, each player k in P has a pure strategies set $S_k = \{a_{k,1}, a_{k,2}, \ldots, a_{k,n}\}$ with a finite number $n \in \mathbb{N}_{>0}$ of pure strategies $a: x_k^{g-1} \to h_k^g$ that can be applied by the player to modify a subgroup $x_k^{g-1} \in X^{g-1}$ obtaining an evolved subgroup h_k^g, where X^{g-1} is the population of candidate solutions at game turn $g-1$, fulfilling $X^{g-1} \equiv \bigcup_{k=1}^{m} x_k{}^{g-1}$. The size N_k of the population subgroup x_k^{g-1} controlled by player k is $N_k = \lfloor N/m \rfloor$, with the exception of x_m^{g-1}, which its size is computed as $N_m = N - (m-1)\lfloor N/m \rfloor$.

All pure strategies of player k in S_k have a selection probability defined by a vector $SSP_k^g \in \mathbb{R}^n$ defining a discrete probability density function (PDF) that changes accordingly to the payoff functions $F_k^g: S_1 \times S_2 \times \ldots \times S_m \to \mathbb{R}^+$. These payoffs functions, which define the dynamic optimization game to be played, depend on a pure strategy profile $\vec{s}^g$, which is an m-tuple association of pure strategies to players that contains the moves s_k^g of each player, such that $s_1^g \in S_1, s_2^g \in S_2, \ldots, s_k^g \in S_k, \ldots, s_m^g \in S_m$. It has to be noted that all payoff functions are symmetrical, as the game does not depend on which player chooses each pure strategy.

The optimization game method in MAGO starts with the creation of the initial population X^0, the evaluation of the cost function $f(X^0)$, and the initialization of the game payoff functions, setting $F_k^1 = 0$ for all possible pure strategies of player k. This will ensure all elements of SSP_k^1 to be equal to $1/m$ for every player k, and thus, all pure strategies will have the same probability to be selected at the first game turn $g = 1$.

Once the algorithm has been initialized, the optimization game is played sequentially by turns. In each game turn $g > 0$, the pure strategy profile vector $\vec{s}^g = (s_1^g, s_2^g, \ldots, s_m^g)$ is constructed as each player decides the pure strategy to play. To select this strategy, players need to know the probability vectors SSP_k^g. The game turn g starts with the first player computing the SSP_1^g vector, element by element. The l^{th} element of the SSP_1^g vector is calculated as the weighted sum of all payoff functions $F_k^g: S_1 \times S_2 \times \ldots \times S_m \to \mathbb{R}^+$ for every player strategy at the game turn g, assuming that the first player selects the $a_{1,l} \in S_1$ pure strategy. The weights for summing the payoffs are chosen so as to ensure the SSP_1^g vector properly defines a discrete PDF, this is, $\sum_{l=1}^{n} SSP_1^g(l) = 1$. Once the PDF defined by the SSP_1^g vector has been computed, a random sample $v_1 \in \{1,2,\ldots,n\}$ is generated and the first player determines the strategy profile s_1^g with pure strategy a_{1,v_1} as his selected move for the game turn g. In the next step, the SSP_2^g vector is generated, and once again, the l^{th} element of the SSP_2^g vector is then computed as the weighted sum of all payoff for every strategy of the non-strategy-assigned players, this is $k > 2$, assuming that the second player selects the pure strategy $a_{2,l} \in S_2$, and taking into account that the first player will be playing with the a_{1,v_1} pure strategy. Once more, a random sample $v_2 \in \{1,2,\ldots,n\}$ is generated from the PDF defined by the SSP_2^g vector, and the strategy profile s_2^g with the pure strategy a_{2,v_2} is selected by the second player as its move for the game turn g. This sequential procedure is repeated for every player turn until obtaining finally the complete pure strategy profile vector $\vec{s}^g = \left(s_1^g, s_2^g, \ldots, s_m^g\right)$. This pure strategy profile selection process is shown in Algorithm 1.

When the pure strategy profile vector $\vec{s}^g$ has been already defined, each player k makes his move with the selected pure strategy a_{k,v_k} that will govern the rules of change of the population subgroup x_k^{g-1} towards a new evolved subpopulation h_k^g. Then, the cost function is evaluated for each new candidate solution in the evolved subpopulations h_k^g and only the improved individuals are selected to survive and substitute their fathers in x_k^{g-1}, this is:

$$x_{k,i}^g = \begin{cases} h_{k,i}^g, & if\ \ f\left(h_{k,i}^g\right) < f\left(x_{k,i}^{g-1}\right), \\ x_{k,i}^{g-1}, & otherwise, \end{cases} \qquad i = 1, \ldots, N_k \tag{1}$$

DOI: 10.13009/EUCASS2017-204

Where $x_{k,i}^{g}$ and $h_{k,i}^{g}$ are the i^{th} individual of the population subgroup x_{k}^{g} and the evolved subpopulation h_{k}^{g} respectively. Before finishing the game turn g, the payoff functions need to be updated on the basis of the results obtained with the pure strategy profile vector $\vec{s}^{g}$, assigning a higher payoff to pure strategies that lead to better results and a lower payoff to pure strategies that did not improve at all the optimization results. With this philosophy, the recursive law for updating the payoffs of the game is derived.

$$\left.\begin{array}{c} F_k^{g+1}\left(s_1^g, s_2^g, \dots, s_m^g\right) = F_k^g\left(s_1^g, s_2^g, \dots, s_m^g\right) + \frac{K_C}{N_k}\sum_{i=1}^{N_k}\left|IM_{k,i}^g\right| \\ IM_{k,i}^g = min\left\{0, f\left(x_{k,i}^g\right) - f\left(x_{k,i}^{g-1}\right)\right\} \\ F_k^{g+1}\left(s_1^{g\complement}, s_2^{g\complement}, \dots, s_m^{g\complement}\right) = F_k^g\left(s_1^{g\complement}, s_2^{g\complement}, \dots, s_m^{g\complement}\right) \end{array}\right\} \qquad \textbf{(2)}$$

In equation (3), $IM_{k,i}^{g}$ is the improvement in the cost of the i^{th} individual of the subgroup x_{k}^{g} controlled by player k with respect to its cost at the previous generation $g-1$, $s_k^{g\complement} = \left\{a \in S_k \middle| \, a \notin s_k^g\right\}$ represents all pure strategies for player k which are part of S_k, but different from the pure strategy s_k^g (the relative complement of s_k^g in S_k), and K_C is a parameter that controls the convergence speed to the best performing pure strategy profiles. Once all payoffs F_k^{g+1} for the $g+1$ game turn have been computed, they are divided by the sum of all payoffs for every pure strategy of every player to obtain a dimensionless payoff $F_k^{g+1} \in [0,1]$ for each player k.

$$F_k^{g+1} = F_k^{g+1} / \left\{\sum_{j_1=1}^{n}\sum_{j_2=1}^{n}\dots\sum_{j_m=1}^{n} F_k^g\left(a_{1,j_1}, a_{2,j_2}, \dots, a_{m,j_m}\right)\right\} \qquad \textbf{(3)}$$

With the aim to avoid playing a cooperative dynamic game where only a few strategies have a big probability to be played (as a consequence of good optimization performances of certain strategy profiles obtained repeatedly during the optimization and the deterioration of the selection probability of other pure strategies that may provide also good optimization performances but have no chance to be applied), the payoff functions are monitored and checked for proper payoff value properties. This is, if the dimensionless value of a payoff function exceeds 0.8 for a certain pure strategy profile vector, then, all payoff functions are reconfigured adding a constant reset value to the other payoff functions and each element of the payoff matrix is finally divided by the sum of all payoffs to maintain the range $[0, 1]$. This methodology is called dynamic game anti-stall monitoring. In addition, in order to avoid solutions stalls (the best solution found does not improve during a large number of game turns), once each G_M game turns, a percentage P_M of the population X^g is migrated randomly throughout the optimization space and the cost function is evaluated at these new points. In addition, for the sake of improving the success rate of MAGO algorithm, once each G_{Hyb} game turns, a random individual of the population X^g is selected as the initial point to perform a local constrained optimization with a method based on Sequential Quadratic Programming [17].

Just after the payoff checking step is finished, the game turn g ends and a new game turn $g+1$ starts. The optimization game previously described is played in turns until reaching the exit criterion imposed by the user, and MAGO algorithm outputs the best solution f_{best} found along with a history matrix containing all pairs of $\{f(x), x\}$ evaluated during the optimization run. The MAGO algorithm is shown in Algorithm 2.

Algorithm 1: PureStrategyProfileSelection

```
1   **for** $k = 1$ **to** $m$ **do** // Player's turn loop
2       **for** $l = 1$ **to** $n$ **do** // Player $k$ probability to select the pure strategy $a_{k,l}$ loop
            // Compute player $k$ probability to select the pure strategy $a_{k,l}$ adding the payoff function values for
            // possible pure strategies of the non-strategy-assigned players, from player $(k+1)$ to player $m$,
            // taking into account that players $1, 2, \dots, k-1$ // will be playing with strategies
            // $a_{1,v_1}, a_{2,v_2}, \dots, a_{k-1,v_{k-1}}$ respectively
3           $SSP_k^g(l) = \sum_{u=1}^{m}\sum_{p_1=1}^{n}\dots\sum_{p_{m-l}=1}^{n} F_u^g\left(a_{1,v_1}, a_{2,v_2}, \dots, a_{k-1,v_{k-1}}, a_{k,l}, a_{k+1,p_1}, \dots, a_{m,p_{m-k}}\right)$
4       **end do**
5       $SP_k^g = SSP_k^g / \sum_{p=1}^{n} SSP_k^g(p)$
6       Randomly select $v_k$ from the PDF defined by $SSP_k^g$
7       Update $s_k^g$ for player $k$ with the pure strategy $a_{k,v_k}$
8   **end do**
```

DOI: 10.13009/EUCASS2017-204

Algorithm 2: MAGO algorithm

```
Input: $f, D, N, m, S_1, S_2, \dots, S_m, lb, ub, exitCriterion, K_C, P_M, G_M, G_{Hyb}$
Output: $f_{best}, x_{best}$
Initialization:
Generate $X^0$ with a stratified sample without replacement of a Van der Corput master sequence
$g = 0$; Evaluate the function $f(X^0)$;
for $k = 1$ to $m$ do // Players loop
    $F_k^1(a_{1,j_1}, a_{2,j_2}, \dots, a_{m,j_m}) = 0; \quad \forall j_1 \in \{1, \dots, n\}, \forall j_2 \in \{1, \dots, n\}, \dots, \forall j_m \in \{1, \dots, n\}$
    Randomly assign a subpopulation $x_k^0 \in X^0$ of $N_k$ individuals to each player $k$
end do
while $exitCriterion$ is not met do
    $g = g + 1$; Apply Algorithm 1 and compute $\vec{s}^g = (s_1^g, \dots, s_m^g) = (a_{1,v_1}, \dots, a_{m,v_m})$
    for $k = 1$ to $m$ do // Player's turn loop
        Apply pure strategy $a_{k,v_k}$ to subpopulation $x_k^{g-1}$ and obtain the evolved subpopulation $h_k^g$
        Check $h_k^g$ fulfils the optimization space constraints $lb$ and $ub$
        for $i = 1$ to $N_k$ do // Subpopulation individuals loop
            Evaluate the cost function $f(h_{k,i}^g)$ with each individual of the evolved subpopulation
            if $f(h_{k,i}^g) < f(x_{k,i}^{g-1})$
                $x_{k,i}^g = h_{k,i}^g$
            else
                $x_{k,i}^g = x_{k,i}^{g-1}$
            end if
            $IM_{k,i}^g = \min\{0, f(x_{k,i}^g) - f(x_{k,i}^{g-1})\}$
        end do
    end do
    for $k = 1$ to $m$ do
        $F_k^{g+1}(s_1^g, s_2^g, \dots, s_m^g) = F_k^g(s_1^g, s_2^g, \dots, s_m^g) + \frac{K_C}{N_k}\sum_{r=1}^{N_k}|IM_{k,r}^g|$;
        $F_k^{g+1}(s_1^{g^C}, s_2^{g^C}, \dots, s_m^{g^C}) = F_k^g(s_1^{g^C}, s_2^{g^C}, \dots, s_m^{g^C})$
    end do
    if $remainder(g/G_{Hyb}) = 0$ then
        Select a random individual from $X^g$ and perform a constrained local optimization
    end if
    if $remainder(g/G_M) = 0$ then
        Migrate the worst $P_M/100 \cdot N$ individuals of $X^g$, check the optimization space constraints, and
        re-evaluate the cost function
    end if
    $F_k^{g+1} = F_k^{g+1} / \{\sum_{j_1=1}^{n}\sum_{j_2=1}^{n} \cdots \sum_{j_m=1}^{n} F_k^g(a_{1,j_1}, a_{2,j_2}, \dots, a_{m,j_m})\}$
    $f_{best} = \min f(X^g); \quad x_{best} = x_r^g$ such that $f_{best} = f(x_r^g)$;
    for $k = 1$ to $m$ do
        Randomly re-assign a subpopulation $x_k^g \in X^g$ of $N_k$ individuals to each player $k$
    end do
end
```

3.3 Pure Strategies and the One-step Evolution Algorithms

Pure strategies a for each player k are defined by the user previous to run an optimization with the MAGO algorithm. These pure strategies are unique pairs $\{o_i, c_{i,j}\}$ of a one-step evolution algorithm $o_i \in O = \{o_1, o_2, \dots, o_{n_O}\}$ and its configuration parameters $c_{i,j} \in C_i = \{c_{i,1}, c_{i,2}, \dots, c_{i,n_{Ci}}\}$, being O a one-step evolution algorithm catalogue defined by the user, n_O the number of available one-step evolution algorithms, C_i the configuration parameters catalogue for each element i in O, and n_{Ci} the number of different configuration parameters for the i^{th} one-step evolution algorithm in O. A one-step evolution algorithm is defined as an external subroutine defining a function $o_i : X^{g-1} \times c_{i,j} \times g \rightarrow h_k^g$ that dictates the rules of change for a population subgroup x_k^{g-1} as a function of its distribution along the optimization space dimensions, a set of configuration parameters $c_{i,j}$ that affects the population

DOI: 10.13009/EUCASS2017-204

rules of change, and the game turn g (outputs of one-step evolution algorithms can depend also on the state of internal saved values).

In the current version of MAGO, the one-step evolution algorithms catalogue O can only be composed by two different algorithms, the population evolution rules of the DE algorithm, and the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [18] with slight modifications to increase its performances, this is, $o_1 = \text{DE}$, $o_2 = \text{CMA-ES}$ and $n_O = 2$. The evolution rules of these algorithms have been selected as the baseline one-step evolution algorithms for MAGO on the basis of the obtained results after performing extensive optimization efficiency assessments with a set of black-box cost functions.

Differential Evolution (DE)

DE population evolution rules are defined by different types of mutation and crossover operators. The mutation operators create a mutant individual $\mu_{k,i}^{g}$ from an initial individual $x_{k,i}^{g-1}$ using individuals picked from the population X^{g-1}, then the crossover operator recombines the mutant individual $\mu_{k,i}^{g}$ with $x_{k,i}^{g-1}$ to finally obtain $h_{k,i}^{g}$. Some of the most used mutation operators in the literature have been modified specifically for its application in MAGO. These ad-hoc operators (*) are defined as follows:

$$\text{DE/rand/1: } \mu_{k,i}^{g} = X_{r1}^{g-1} + M^{g} \cdot \left(X_{r2}^{g-1} - X_{r3}^{g-1}\right) \tag{4}$$

$$\text{DE/rand/2*: } \mu_{k,i}^{g} = X_{r1}^{g-1} + R_1 \cdot M^{g} \cdot \left(X_{r2}^{g-1} - X_{r3}^{g-1}\right) + (1 - R_1) \cdot M^{g} \cdot \left(X_{r4}^{g-1} - X_{r5}^{g-1}\right) \tag{5}$$

$$\text{DE/current-to-pbest/1*: } \mu_{k,i}^{g} = x_{k,i}^{g-1} + R_1 \cdot M^{g} \cdot \left(X_{pbest}^{g-1} - x_{k,i}^{g-1}\right) + (1 - R_1) \cdot M^{g} \cdot \left(X_{r1}^{g-1} - X_{r2}^{g-1}\right) \tag{6}$$

$$\text{DE/current-to-pbest/2*: } \left.\begin{array}{l} \mu_{k,i}^{g} = x_{k,i}^{g-1} + R_2 \cdot R_1 \cdot M^{g} \cdot \left(X_{pbest}^{g-1} - x_{k,i}^{g-1}\right) + \\ +R_2 \cdot (1 - R_1) \cdot M^{g} \cdot \left(X_{r1}^{g-1} - X_{r2}^{g-1}\right) \\ +(1 - R_2) \cdot M^{g} \cdot \left(X_{r3}^{g-1} - X_{r4}^{g-1}\right) \end{array}\right\} \tag{7}$$

Where X_{r1}^{g-1},X_{r2}^{g-1}, X_{r3}^{g-1}, X_{r4}^{g-1} and X_{r5}^{g-1} are non-repeated individuals randomly selected from the population X^{g-1} and different from $x_{k,i}^{g-1}$, $X_{pbest}^{g-1} \in X^{g-1}$ is an individual picked randomly from the best $p < N$ individuals of the population X^{g-1}, R_1 and R_2 are real numbers randomly selected from a continuous uniform distribution function $U(0,1)$, and M^{g} is a mutation scaling factor that can be constant (M-Const), and so equal to $MF \in [0.2,1]$ (a real parameter set by the user), or adaptive (M-Adapt), as originally proposed in [19]:

$$M^{g} = \begin{cases} 0.1 + rand_1 \cdot 0.9, & if\ rand_2 < \tau_1, \\ M^{g-1}, & otherwise, \end{cases} \tag{8}$$

The crossover operators for the DE have been also selected from the set of most used methods in the literature, the binomial (bin), and exponential (exp) crossover with crossover parameter $CR \in [0.2,1]$ defined by the user, are defined as:

$$\text{bin: } h_{k,i}^{g}(l) = \begin{cases} \mu_{k,i}^{g}(l), & if\ rand < CR, \\ x_{k,i}^{g-1}(l), & otherwise, \end{cases} \tag{9}$$

Algorithm 3: ExponentialCrossover

```
1   $h_{k,i}^{g} = x_{k,i}^{g-1}; l = irand(\{1, \dots, D\}); L = 1$
2   while $rand < CR\ or\ L < D$ do
3       $h_{k,i}^{g}(l) = \mu_{k,i}^{g}(l)$
4       $L = L + 1; l = l + 1;$
5   end
```

DOI: 10.13009/EUCASS2017-204

Covariance Matrix Adaptation Evolution Strategy (CMA-ES)

In contrast to the DE, the CMA-ES one-step evolution algorithm generates an evolved subpopulation h_k^g sampling N_k individuals form a multivariate normal distribution $\mathcal{N}(\theta^g, (\sigma^g)^2 \varsigma^g) \in \mathbb{R}^D$, where $\varsigma^g \in \mathbb{R}^{D \times D}$ is a covariance matrix that defines the dependencies between the components of the evolved subpopulation, $\theta^g \in \mathbb{R}^D$ is the mean value of the evolved subpopulation, and $\sigma^g \in \mathbb{R}$ is the mutation step-size. θ^g is computed as the weighted sum of the $\lfloor N/2 \rfloor$ best individuals of the population X^{g-1} as $\theta^g = \sum_{i=1}^{\lfloor N/2 \rfloor} w_i X_i^{g-1}$ (X_i^{g-1} are sorted in increasing cost function value), with $\sum_{i=1}^{\lfloor N/2 \rfloor} w_i = \mathbf{1}$**.** The covariance matrix ς^g is computed such that the likelihood of previously successful search steps is increased. This is achieved using the so called rank-one update, which is based on including the information of the search evolution vector into the update scheme of the covariance matrix. The mutation step size σ^g is updated using a search evolution path $p_\sigma^g \in \mathbb{R}^D$, which is constructed from the difference between consecutive subpopulation means θ^g and θ^{g-1}. Then, the covariance matrix ς^g is computed using its own evolution path vector p_ς^g:

$$\left.\begin{aligned}
p_\sigma^g &= (1-c_\sigma) p_\sigma^{g-1} + \sqrt{((1-(1-c_\sigma)^2)/(\textstyle\sum_{i=1}^{B} w_i^2))(\varsigma^{g-1})^{-1}} \left(\frac{\theta^g - \theta^{g-1}}{\sigma^{g-1}}\right) \\
\sigma^g &= \sigma^{g-1} \times \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\left\|p_\sigma^{g-1}\right\|}{E\|\mathcal{N}(0,I)\|} - 1\right)\right) \\
p_\varsigma^g &= (1-c_\varsigma) p_\varsigma^{g-1} + \varphi\left(\left\|p_\sigma^g\right\|, 1.5\sqrt{D}\right) \sqrt{\left(\left(1-\left(1-c_\varsigma\right)^2\right)/(\textstyle\sum_{i=1}^{B} w_i^2)\right)} \left(\frac{\theta^g - \theta^{g-1}}{\sigma^{g-1}}\right) \\
\varsigma^g &= (1-c_1-c_\mu) \varsigma^{g-1} + c_1 p_\varsigma^g {p_\varsigma^g}^T + c_\mu \textstyle\sum_{i=1}^{\lfloor N/2 \rfloor} \left(w_i \frac{\left(X_i^{g-1} - \theta^g\right)\left(X_i^{g-1} - \theta^g\right)^T}{\sigma^{g-1}} \right)
\end{aligned}\right\} \quad \mathbf{(10)}$$

Where $c_\sigma = \min(0.9, 3/D)$, $c_\varsigma = \min(0.9, 4/D)$, $c_\mu = \left(\sum_{i=1}^{\lfloor N/2 \rfloor} w_i^2\right)/D^2$, $E\|\mathcal{N}(0,I)\|$ is the expectation of a normal distribution with a variance $I_{D\times D}$ (identity matrix), d_σ is a parameter selected close to 1, and $\varphi\left(\left\|p_\sigma^g\right\|, 1.5\sqrt{D}\right)$ is an indicator function that evaluates to 1 if $\left\|p_\sigma^g\right\| \in [0, 1.5\sqrt{D}]$ and 0 otherwise. This one-step evolution algorithm is particularly useful for ill-conditioned cost functions and does not require a tedious parameter tuning for its application.

Individual Explorers

To enhance the diversity of the individuals created by the one-step evolution algorithms, the individual explorer concept has been introduced. An exploration initiator $X_{r1}^{g-1} \in X^{g-1}$ is defined as an individual randomly selected from the population who does not follow the evolution rules of the one-step evolution algorithms to generate a new candidate solution. Instead, the exploration initiator randomly seeds a finite exploration space with an explorer $\gamma_{k,u}^g$. The search space of each exploration initiator is a D-dimensional hypercube defined by two of their opposite corners. One corner of the hypercube is coincident with the exploration initiator X_{r1}^{g-1} and the other one is constructed by randomly picking up another individual from the population $X_{r2}^{g-1} \in X^{g-1}$. The i^{th} component of the explorer γ_u^g is computed as:

$$\gamma_{k,u}^g(i) = \min\left(X_{r1}^{g-1}(i), X_{r2}^{g-1}(i)\right) + rand \cdot \left(\max\left(X_{r1}^{g-1}(i), X_{r2}^{g-1}(i)\right) - \min\left(X_{r1}^{g-1}(i), X_{r2}^{g-1}(i)\right)\right) \quad \mathbf{(11)}$$

Player's Pure Strategy Sets Example

The pure strategy set S_k for player k in MAGO is a user-defined catalogue containing a finite number n of available combinations $a = \{o_i \in O, c_{i,j} \in C_i\}$ for that player. The number of pure strategies n is selected by the user and can be as large as the maximum number of combinations of available one-step evolution algorithms and their parameters configurations. To illustrate how these sets are created, an example of a pure strategy set for the 1st player or the game is shown in Table 1. This strategy set is defined as $S_1 = \left\{\{o_1, c_{1,1}\}, \{o_1, c_{1,2}\}, \{o_1, c_{1,3}\}, \{o_2, c_{2,1}\}, \{o_2, c_{2,2}\}\right\}$, with o_1=DE and o_2=CMA-ES, and $c_{1,1}$ to $c_{1,3}$ and $c_{2,1}$ to $c_{2,2}$ are different configuration parameters for the DE and CMA-ES one-step evolution algorithms.

DOI: 10.13009/EUCASS2017-204

Table 1: Example of pure strategy catalogue for player 1, with four different pure strategies

Pure Strategies	**One-step evolution algorithm**	**Configuration Parameters Sets**	**Mutation Operator**	**Crossover Operator**	**Explorers**
a_1	o_1=DE	$c_{1,1}$	DE/rand/1/M-const	Exp	3
a_2	o_1=DE	$c_{1,2}$	DE/rand/2*/M-adapt	Bin	1
a_3	o_1=DE	$c_{1,3}$	DE/ rand /2*/M-const	Exp	2
a_4	o_2=CMA-ES	$c_{2,1}$	N/A	N/A	2
a_5	o_2=CMA-ES	$c_{2,2}$	N/A	N/A	3

4 Nonlinear Simulator Parametric Cost Function for FCL clearance

To find the worst case conditions and possible departures in the nonlinear offline simulation assessments, the nonlinear simulation environment and the clearance requirements shall be transformed into an equivalent single-objective optimization problem defined by a cost function f. In the FCL clearance, the cost function $f: \mathcal{J} \in \mathbb{R}^D \times \vartheta \in \mathcal{C} \rightarrow \Delta \in \mathbb{R}$ is a surjective parametric black-box function that provides a mapping between the inputs $\mathcal{J}$ of a nonlinear simulator and a one-dimensional monitor variable Δ, which can be any elaboration of flight mechanic parameters, or a clearance criterion to be optimized during a manoeuvre. The additional input $\vartheta \in \mathcal{C}$ is a structure of numbers, booleans and strings belonging to the set $\mathcal{C}$ that contains all possible cost function settings for all the analysis performed in a FCL clearance. Using this additional input ϑ, the user can select the dimension of the optimization problem D, the type of monitor variable Δ (i.e. maximum angle of attack, sideslip angle, load factors, surfaces saturation time, etc.), which parameters will be selected to be part of the optimization space, and handle all failure and failure-free functionalities and configuration options of the non-linear simulator of the aircraft.

There are a few golden rules to follow when creating the cost function f to maximize the probabilities to find its global optimum, and thus, to facilitate the searching of the worst-case scenario for possible violations of the FCL flight envelope protection functions or the clearance safety requirements. These rules are:

a) The internal sub-functions used to elaborate the output Δ from time-varying signals in a nonlinear simulation shall be smooth and monotonous whenever possible in order to not introduce artificial non-linearities. Special attention shall be paid to dead zones or spiky conversion functions (i.e. when Δ is the position saturation time of a control surface during a manoeuvre, a smooth transition between non-control-saturated and control-saturated manoeuvres should be used to avoid dead-zones in Δ).

b) If the global optimizer is not fit for mixed-integer optimization problems, then the cost function shall convert internally all discrete variables to continuous. A continuous optimization variable must be defined which maps the real number set to the discrete number set of the original optimization variable. The mapping must be done in order to generate variations with physical sense.

c) Conventional approaches in the optimization algorithms to solve non-linear boundary constraints consist on solving the intersection point of the trajectory of the new individual with the bound defined by the nonlinear boundary function. Besides the time consuming effort made by the algorithm, this solution tends to increase the number of individuals located in the boundaries because of the intersections. Another solution consists on randomly generating a new individual and check if the boundary constraints are not violated, but this solution worsen the effectiveness of the algorithm because the "genetic" information of the former individuals is missed. To avoid this, the cost function f should internally transform the feasible optimization space defined by the linear and nonlinear constraints in $\mathcal{J}$ into a non-dimensional hyper-cube space.

DOI: 10.13009/EUCASS2017-204

4.1 Flight Envelope

Typical Mach number versus altitude flight envelope of a supersonic aircraft includes different limitations which lead to non-squared contours. The solution implemented in the cost function f consists on introducing a mapping function $\wp: \mathcal{J}_M \in [0,1] \times \mathcal{J}_H \in [0,1] \times \vartheta \rightarrow$ {Mach, Altitude} $\in FE$, where FE is the flight envelope portion where the optimization will be performed, where $\mathcal{J}_M$ and $\mathcal{J}_H$ are the input components of the cost function related to the Mach and altitude respectively. Function $\wp$ uses information contained in ϑ about the lower bounds M_{lb}, CAS_{lb}, and H_{lb}, and the upper bounds M_{ub}, CAS_{ub}, and H_{ub} of the flight envelope FE as shown in Figure 2. This means that the global optimization algorithm will be optimizing two independent dimensionless variables $\mathcal{J}_M$ and $\mathcal{J}_H$ which vary from 0 to 1, and the cost function f will transform using $\wp$ the dimensionless variables to a Mach and Altitude fulfilling the nonlinear constraints. This solution prevents the time consuming effort that requires solving nonlinear constraint problems for each individual.

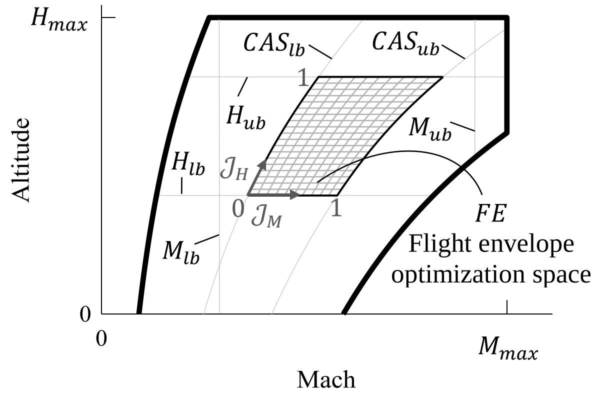

Figure 2. Mach vs. Altitude flight envelope FE mapped to a dimensionless squared contour

4.2 Simulation Models Tolerances and Configurations

A remarkable challenge of the FCL clearance process is the accuracy of the models involved in the clearance tasks. Because the flight dynamics models are not perfectly flight-matched and aircraft sensors are not perfect, it is required to introduce a level of tolerance/error in the relevant flight mechanics parameters in order to cover aerodynamic model uncertainties and sensor errors. Some of the most meaningful tolerances/errors considered in the FCL clearance process are the following:

1. **Aerodynamic Tolerances.** It consists in implementing a tolerance in a given aero-derivative coefficient. In order to avoid the simultaneous application of multiple aero-derivatives tolerances, which would produce extremely pessimistic scenarios in the FCL clearance, a single input neural network with a single neuron layer is defined to control all the aero-derivative coefficient tolerance factors with a single continuous optimization variable $\mathcal{J}_A \in [0, n_A]$, where n_A is the number of aerodynamic tolerance factors. Neuron activation functions are selected to be linear saw tooth functions such that only one tolerance factor is different from 0 for every $\mathcal{J}_A$ value.

2. **Aircraft Stores Configurations.** Fighter aircraft aerodynamics strongly depends on the external stores distribution. To perform a one-shot optimization over a set $A = \{T_1, \dots, T_{n_S}\}$ of n_S different aircraft store configurations, the cost function has a single continuous optimization variable denoted by $\mathcal{J}_S$ which maps the continuous space in the range $[0, n_S]$ to the elements in A, changing the applicable aerodynamic model and the mass-inertia-CG ranges for that specific aircraft configuration.

3. **Mass and Inertia Errors.** In complex FCL of a highly manoeuvrable aircraft it is customary to use mass properties estimations such as aircraft gross weight, CG position, or inertias for gain scheduling. The major contributor to the aircraft mass state estimation errors is the variation of the CG induced by fuel sloshing during aggressive manoeuvres or by fuel system failures. A dedicated analysis must be performed before starting the FCL clearance process in order to select the extreme mass states of the aircraft. These resulting discrete mass states are converted to a single optimization parameter $\mathcal{J}_W$ by assigning an index to each of them.

DOI: 10.13009/EUCASS2017-204

4. **IMU and Air Data Sensors Errors.** FCL clearance also covers the static errors of the aircraft sensors. The sensor model error must include the propagation of basic measure errors to all physical parameters depending on the relation between them. For instance, the airspeed errors must be computed with temperature and static pressure errors. As done for the aerodynamic tolerances, it is defined a single input neural network with a single neuron layer conversion function to transform a single optimization variable $\mathcal{J}_D$ into n_D different sensor error combinations.

The application rules to combine different model tolerances depend on the FCL clearance requirements but basically it consist in applying a given probability to each tolerance and to the different combinations between them in order to not consider an extremely pessimistic scenario. As the number of tolerated models increase, the maximum tolerance factor to apply to each model decreases.

4.3 Pilot Inputs Parameterization

On the basis on FCH definition, there are infinite combinations of stick, pedal and throttle inputs that must be covered in a FCL clearance, and consequently it is necessary to develop a mathematical formulation which can represent every input, or at least, the maximum number of inputs combinations. The cost function f includes different configuration options to define the pilot aggressiveness level regarding the stick, pedal and throttle inputs depending on the specific analysis required during a FCL clearance.

The “aggressive FCH manoeuvres” is the most generic option to cover any pilot stick input and it is only used during the initial stages of FCL carefree handling clearance to search for the worst-case combinations of pilot stick inputs. When this level is selected, the cost function discretizes both axis of the pilot stick into a polygonal function. Each point of the polygonal function adds two optimization variables to the optimization problem, the time value and the stick input value. With a maximum of 10 breakpoints per stick axis, it is possible to cover a wide variety of manoeuvres with a set of 40 optimization variables denoted by $\mathcal{J}t_{p_1}$, $\mathcal{J}p_1$, ..., $\mathcal{J}t_{p_{10}}$and $\mathcal{J}p_{10}$ for the pitch stick inputs, and $\mathcal{J}t_{r_1}$, $\mathcal{J}r_1$, ..., $\mathcal{J}t_{r_{10}}$and $\mathcal{J}r_{10}$ for the roll stick inputs as shown in Figure 3. Interpolation between breakpoints can be linear of through a B-spline curve.

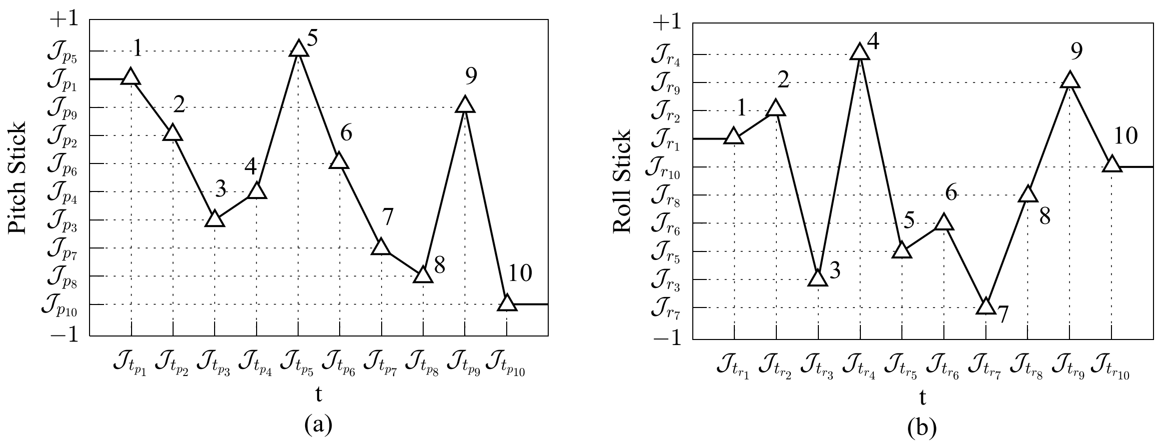

Figure 3. Optimization variables for “aggressive FCH manoeuvres” definition for the pitch (a) and roll (b) stick axis

5 Carefree Handling Clearance Case Study

MAGO algorithm capabilities to find departures and FCL clearance issues will be benchmarked with other two global optimization algorithms, the PSO and SPS-LSHADE-EIG algorithm. The global optimization problem will be set up to look for an old and well known carefree handling clearance issue (DEP1) detected at the very beginning of the Eurofighter Typhoon development program in the late 90’s. The DEP1 issue, which was related to aircraft departures from controlled flight, led to fixes in the FCL to solve the problem. Departures were located in a very narrow flight envelope zone ($\mathcal{J}_M = [0.5{,}0.65]$ and $\mathcal{J}_H = [0.75{,}0.9]$) and only arose under very specific pilot stick inputs performed with a specific timing. Due to the strongly deceptive nature of the worst-case condition, they remained undetected for some time until they were discovered late in the development process. As DEP1 issue was

DOI: 10.13009/EUCASS2017-204

associated only to an old development version of the FCL model, the cost function f was linked with the respective legacy code of the Eurofighter Typhoon nonlinear simulator.

For this case study, the monitor variable Δ is a dimensionless measure of the aircraft departure risk. Values lower than -1 might indicate there are FCL clearance issues that need to be careful assesses to determine whether it is necessary to rise or not a flight limitation. The optimization problem dimension for "aggressive FCH manoeuvres" is $D = 47$, with $\mathcal{J} = [\mathcal{J}_M, \mathcal{J}_H, \mathcal{J}_A, \mathcal{J}_D, \mathcal{J}_{A/B}, \mathcal{J}_{t_{p_1}}, \mathcal{J}_{p_1}, \dots, \mathcal{J}_{t_{p_{10}}}, \mathcal{J}_{p_{10}}, \mathcal{J}_{t_{r_1}}, \mathcal{J}_{r_1}, \dots, \mathcal{J}_{t_{r_{10}}}, \mathcal{J}_{r_{10}}]$, being $\mathcal{J}_{A/B}$ an optimization variable for the airbrake position, $n_A = 15$, $n_D = 12$, $n_S = 32$ and $n_W = 26$. In order to obtain comparable results, the number of individuals is set to $N = 30$ and the optimization exit criterion is defined by the maximum number of function evaluations, in this case 15 000. To ensure the statistical representativeness of the benchmarking analysis, the number of independent runs for each optimization algorithms is selected to be 200. MAGO algorithm has been run with only one player with the pure strategy set S_1 shown in Table 2.

Table 2: Case study MAGO pure strategy set $\boldsymbol{S_1}$ for player 1, with eight different pure strategies

Pure Strategies	**One-step evolution algorithm**	**Mutation**	**Crossover**	**Explorers**
a_1	o_1=DE	DE/rand/1/M-const	Exp	2
a_2	o_1=DE	DE/rand/1/M-adapt	Exp	2
a_3	o_1=DE	DE/rand/2*/M-const	Exp	2
a_4	o_1=DE	DE/rand/1/M-const	Bin	2
a_5	o_1=DE	DE/rand/1/M-adapt	Bin	2
a_6	o_1=DE	DE/rand/2*/M-const	Bin	2
a_7	o_1=DE	DE/rand/2*/M-adapt	Bin	2
a_8	o_2=CMA-ES	N/A	N/A	2

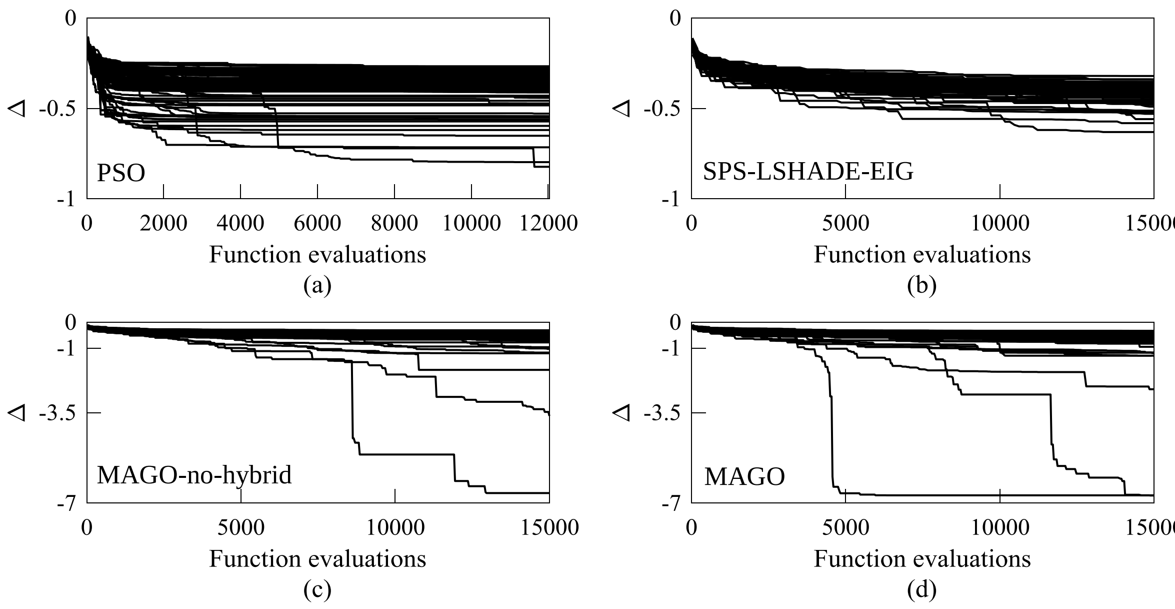

Figure 4. Optimization results with 200 independent runs for PSO (a), SPS-LSHADE-EIG (b), MAGO-no-hybrid (c) and MAGO (d).

DOI: 10.13009/EUCASS2017-204

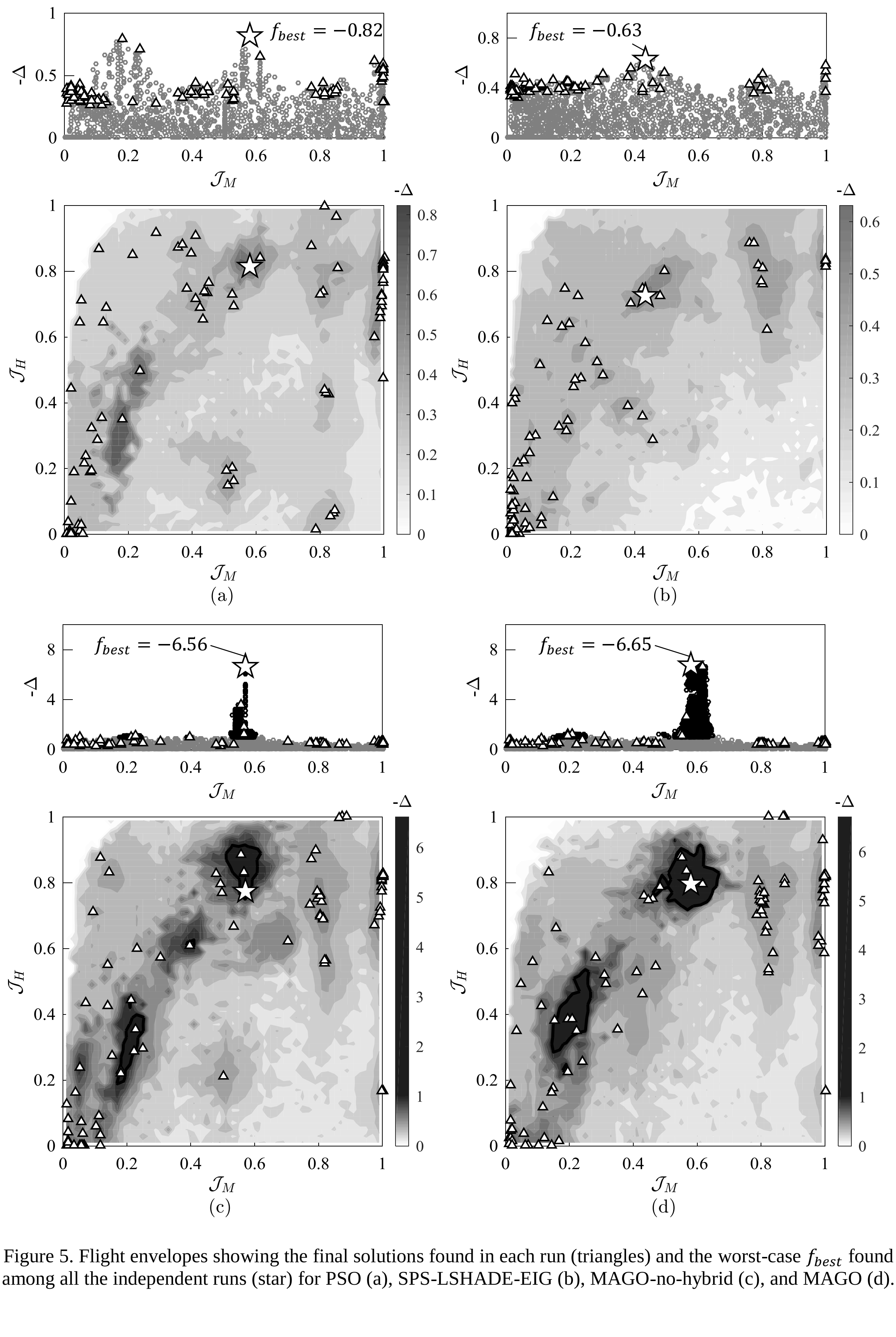

Figure 5. Flight envelopes showing the final solutions found in each run (triangles) and the worst-case f_{best} found among all the independent runs (star) for PSO (a), SPS-LSHADE-EIG (b), MAGO-no-hybrid (c), and MAGO (d).

DOI: 10.13009/EUCASS2017-204

Both the PSO and SPS-LSHADE-EIG algorithms are pure single-objective global optimizers and do not use local optimization methods to enhance the exploitation and accelerate the searching of isolated global optimums. Therefore, to make a fair benchmarking, the MAGO algorithm will be run with two different settings, one that uses only global optimization methods denoted by MAGO-no-hybrid, and another that uses the hybrid version of the algorithm with the "fmincon" MATLAB local optimization function [17].

The evolution of the best solution found for each independent optimization run is shown in Figure 4 for each optimization algorithm. On the basis of the results obtained for this FCL clearance case study, it is concluded that both MAGO and MAGO-no-hybrid algorithms exhibit a higher optimization performance than the other global optimization algorithms, with a 10%-15% success rate, finding the clearance issue with Δ values lower than -1 in the region defined by $\mathcal{J}_M = [0.5{,}0.7]$ and $\mathcal{J}_H = [0.7{,}0.9]$ (see Figure 5). Although the PSO algorithm results exhibits a fast convergence to negative Δ values during the first 1 000 function evaluations, it rapidly gets stalled and solutions do not improve with further cost function valuations. Regarding the SPS-LSHADE-EIG algorithm, although it does not exhibit such a strong stall-prone characteristic as the PSO, it was not able to find the DEP1 FCL issue in any of the 200 independent runs. Moreover, as shown in Figure 5, only MAGO-no-hybrid and MAGO were able to find other old unobserved FCL issues in the area delimited by $\mathcal{J}_M = [0.15{,}0.25]$ and $\mathcal{J}_H = [0.2{,}0.5]$ (these additional deficiencies also disappeared with the FCL fix designed to solve the DEP1 clearance issue).

MAGO results shown in Figure 5 indicate that the hybrid optimization methodology enhance the exploration capabilities of MAGO algorithm, as the DEP1 issue spreads through a wider envelope area than that found by the MAGO-no-hybrid version. In addition, it is demonstrated that the local optimization also increases the convergence speed of the algorithm to the global wort-case. A good algorithm exploration capability is a property of great importance in the application of global optimization techniques in the FCL clearance mainstream assessments, as the global optimization method can be used with a dual purpose, to find the worst-case, and to highlight all the problematic area nearby affected by a common casuistry.

6 Conclusions

Application feasibility of global optimization techniques in the aerospace sector industry to solve complex robustness assessment problems like the FCL clearance of a high performance combat aircraft has been addressed in this work. The extreme complexity of the FCL of modern fighters and the growing need for a faster and more robust methodology to search for the worst-case condition in the FCL clearance assessments motivated the development of the in-house global optimization algorithm MAGO. As shown by the case study results, MAGO algorithm has exhibited excellent worst-case searching capabilities in complex black-box optimization problems, demonstrating an outstanding balance between exploration and exploitation of the search space.

Promising results have been obtained in the FCL carefree handling clearance assessment combining the current grid-based approach with the optimization-based approach. This hybrid methodology drastically increases the probabilities of finding hidden FCL issues in non-linear manoeuvres and conditions not covered by the framework of a typical FCL clearance task, and thus, it reduces the required time frame to perform the validation and verification process for new FCL developments. Nevertheless, worst-case results identified by means of optimization-based clearance methodologies shall be studied carefully; special attention must be paid to the control stick time-histories provided by the optimization. It must be noted that the use of global optimization techniques in highly non-linear and manoeuvrable aircraft with mature FCL, usually lead to find complex stick combinations that must be evaluated in the manned simulator in order to discard those manoeuvres that are not presentative from an operational point of view. It is especially important in the post-optimization assessment to decompose the manoeuvres found into a simpler one, and to isolate the manoeuvre segment which is critical for the issue onset to discover the root cause of the problems found.

7 Future Work

The novel game-based optimization methodology and the pure strategy set concept make MAGO to be a flexible algorithm with continuous growing capabilities. Currently, new disruptive one-step evolution algorithms are being developed under different research and development programs to further increase the optimization performances of the algorithm.

Further improvements in the FCL carefree handling clearance methodologies are currently being investigated. Main objective of one of these projects is to create a cost function that handles a combat mission simulator for a high

DOI: 10.13009/EUCASS2017-204

performance fighter aircraft. This parametric cost function will incorporate synthetic controllers that mimic the required pilot actions to accomplish different tasks, like high precision bombing or dogfight against enemy aircraft. In this case, the optimization variables will be no longer the pilot stick inputs, but the mission definition parameters, like the morphology of the enemy aircraft trajectory. With this new approach, the MAGO algorithm could be used to obtain more operational-representative worst-case pilot stick inputs.

References

[1] Tomayki, J. E. 2000. Computers take flight: a history of NASA's pioneering digital fly-by-wire project. The NASA history series. NASA.

[2] Skoogh, D. and Berefelt, F. 2010. Final report WP2.5. COFCLUO Technical Report D2.5.5. FOI

[3] Fielding, C. 2002. Advanced techniques for clearance of flight control laws. vol. 283. Berlin: Springer.

[4] Storn, R. and Kenneth, P. 1997. Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. *J. of Global Optimization* 11, 4 (December 1997), 341-359.

[5] Menon, P. P., Kim, J., Bates, D. G. and Postlethwaite, I. 2006. Clearance of Nonlinear Flight Control Laws Using Hybrid Evolutionary Optimization. *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 6: 689-699.

[6] Kennedy, J. and Eberhart, R. 1995. Particle Swarm Optimization. *Proceedings of the IEEE International Conference on Neural Networks*. 1942-1948.

[7] Atashpaz-Gargari, E. and Lucas, C. 2007. Imperialist Competitive Algorithm: An algorithm for optimization inspired by imperialistic competition. *IEEE Congress on Evolutionary Computation*. 7: 4661–4666.

[8] Guo, S. M., Tsai, J. S. H., Yang, C. C. and Hsu, P. H. 2015. A self-optimization approach for L-SHADE incorporated with eigenvector-based crossover and successful-parent-selecting framework on CEC 2015 benchmark set. In: *CEC*. 1003-1010.

[9] Krylov, N. M. and Bogoliubov, N. 1943. Introduction to Nonlinear Mechanics. Princeton Univ. Press.

[10] LaSalle, J. P. and Lefschetz, S. 1961. Stability by Liapunov's direct method: with applications. Mathematics in science and engineering. Academic Press

[11] Parks, P. C. 1992. A. M. Lyapunov's stability theory—100 years on. *IMA Journal of Mathematical Control & Information*. 9 (4): 275–303.

[12] Maaranen, H., Miettinen, K. and Mäkelä, M. M. 2004. A Quasi-Random Initial Population for Genetic Algorithms. *Computers and Mathematics with Applications*, vol. 47: 1885-1895.

[13] Morokov, W. J. and Caflish, R. E. 1994. Quasi-random sequences and their discrepancies. *SIAM J. Sci. Comput.*, 15(6):12511279

[14] Mascagni, M. and Chi, H. 2004. On the scrambled halton sequence. *Monte Carlo Methods Appl.*, 10(3):435–442

[15] van der Corput, J.G. 1935. Verteilungsfunktionen. I. Mitt. *Proc. Akad. Wet. Amsterdam*. 38: 813–821

[16] Pausinger, F. and A. Topuzoglu, Van der Corput sequences and permutation polynomials, Preprint.

[17] The MathWorks. 2000. Optimization Toolbox User's Guide, ver. 2.

[18] Hansen, N. and Ostermeier. A. 2001. Completely Derandomized Self-Adaptation in Evolution Strategies. *Evol. Comput.* 9, 2. 159-195.

[19] Brest, J., Greiner, S., Boskovic, B., Mernik, M. and Zumer, V. 2006. Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. *IEEE Trans. Evolutionary Computation*, 10(6): 646-657.