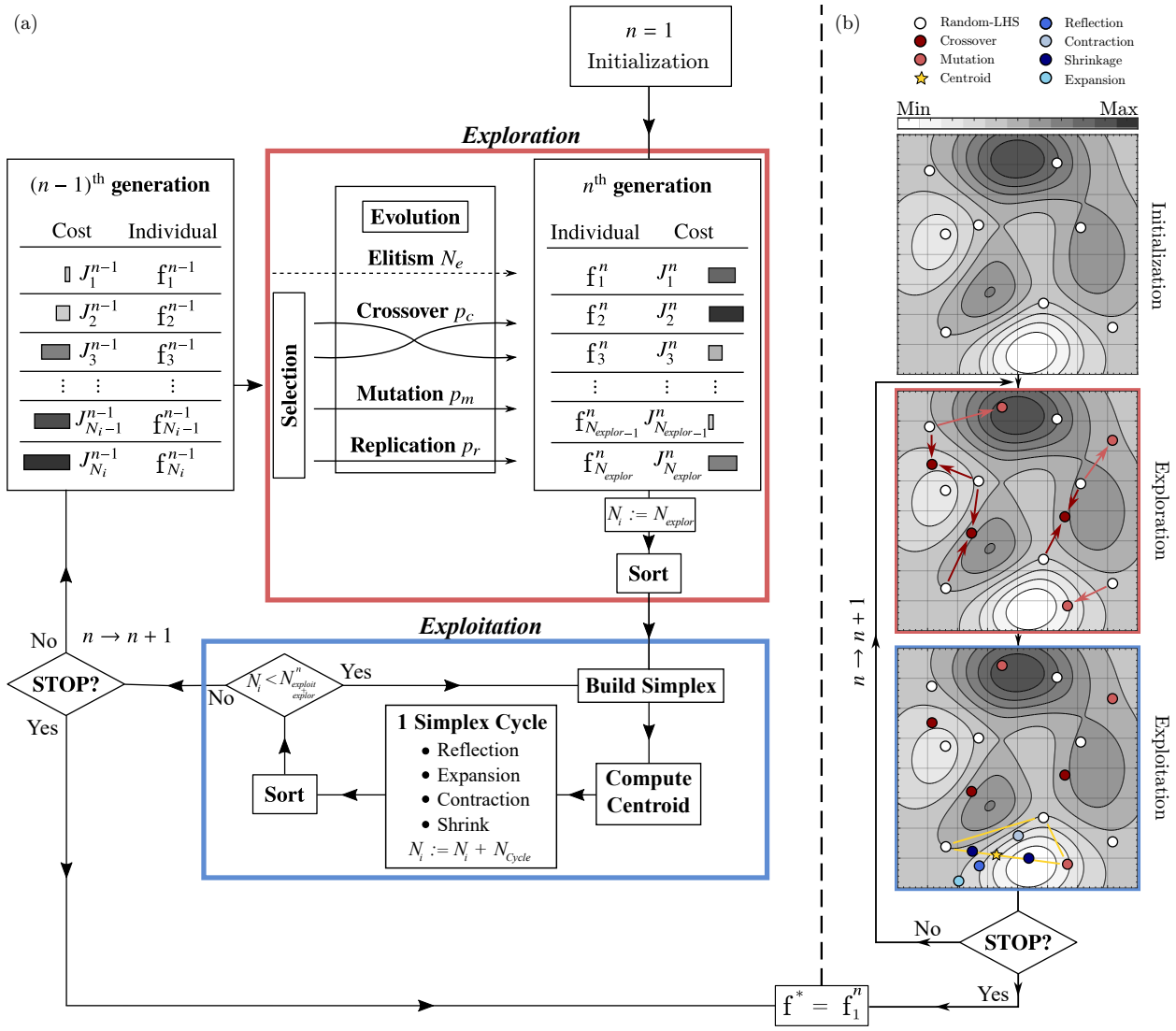
Graphical Abstract

Fast and robust parametric and functional learning with Hybrid Genetic Optimisation (HyGO)

Isaac Robledo, Yiqing Li, Guy Y. Cornejo Maceda, Rodrigo Castellanos



Highlights

Fast and robust parametric and functional learning with Hybrid Genetic Optimisation (HyGO)

Isaac Robledo, Yiqing Li, Guy Y. Cornejo Maceda, Rodrigo Castellanos

- Hybrid genetic framework HyGO combines global search and local refinement.
- Novel degeneration-proof DSM accelerates convergence in high-dimensional spaces.
- HyGO outperforms standard genetic algorithms on benchmark optimisation problems.
- Effective non-linear control laws discovered using HyGO with linear genetic programming.
- Application to Ahmed body achieves significant aerodynamic drag reduction (20%).

Fast and robust parametric and functional learning with Hybrid Genetic Optimisation (HyGO)

Isaac Robledo[a], Yiqing Li[b], Guy Y. Cornejo Maceda[a], Rodrigo Castellanos[a,*]

[a]*Department of Aerospace Engineering, Universidad Carlos III de Madrid Leganés, 28911, Madrid, Spain*
[b]*Department of Mechanical Engineering, University College London London, WC1E 6BT, London, United Kingdom*

Abstract

Evolutionary algorithms are widely used derivative-free global optimisation methods, yet they typically suffer from slow convergence rates, especially during late-stage learning. This work presents the Hybrid Genetic Optimisation framework (HyGO), which enhances evolutionary algorithms with local search algorithms for accelerated learning while maintaining robustness. The key enabler is a two-stage evolution process that balances exploration and exploitation: initially performing broad search via genetic evolution, followed by refined solution improvement using a local-search algorithm. HyGO is validated on (a) parametric optimisation, combining a standard genetic algorithm and a new degeneracy-proof Downhill Simplex Method (DSM), and (b) function optimisation, combining linear genetic programming and DSM. First, benchmarking against a suite of complex analytical functions demonstrates HyGO 's superior convergence speed and robustness compared to standard genetic optimisers. Second, HyGO is employed to stabilise a damped Landau oscillator, identifying effective control laws under nonlinear dynamics. Finally, the practical applicability of HyGO is demonstrated in a flow control problem for aerodynamic drag reduction on an Ahmed body using Reynolds-Averaged Navier-Stokes simulations. Results show that HyGO achieves consistent improvements in performance while providing physically interpretable solutions. Overall, HyGO emerges as a versatile hybrid optimisation framework suitable for a broad spectrum of engineering and scientific problems involving parametric and functional learning.

Keywords: Hybrid optimisation, Genetic Algorithms, Genetic programming, high-dimensional, Downhill Simplex Method, Flow control, Drag reduction, Ahmed body

1 Introduction

Global optimisation problems, particularly those characterised by strong non-linearity, high dimensionality, and the presence of multiple local minima, remain a central challenge across scientific and engineering disciplines. In engineering, the goal is often to determine the most efficient solution to a well-defined problem, constrained by design, operational, or physical limits. Modern tasks frequently involve high-dimensional design spaces, expensive and time-consuming evaluations, and intricate couplings between parameters, making them especially prone to entrapment in local optima and difficult to address with gradient-based methods. These factors not only lead to rugged and multimodal optimisation landscapes but also expose algorithms to the curse of dimensionality [1], where the volume of the design space grows exponentially with the number of variables, severely hampering convergence and efficiency. Representative examples include aerodynamic shape optimisation of vehicles or aircraft, where even small geometric variations can yield highly non-intuitive aerodynamic responses [2]; structural design under multi-physics constraints, such as in thermomechanical systems or energy absorbers [3]; and control of turbulence through active or passive devices, where system dynamics are governed by nonlinear interactions and time-delays between inputs and outputs.

*Corresponding author
Email addresses: isaac.robledo@alumnos.uc3m.es (Isaac Robledo), yiqing.li@ucl.ac.uk (Yiqing Li), gcornejo@ing.uc3m.es (Guy Y. Cornejo Maceda), rcastell@ing.uc3m.es (Rodrigo Castellanos)

To effectively address these challenges, optimisation algorithms must exhibit robustness to nonlinear, multimodal, and high-dimensional search spaces. This requirement limits the applicability of purely local (exploitation-driven) methods and demands strategies that balance exploration with local refinement.

To navigate the challenges posed by high-dimensional, non-linear, and multi-modal optimisation problems, a wide range of optimisation strategies has been developed. These include evolutionary algorithms, such as genetic algorithms and differential evolution; gradient-based methods, which rely on local derivatives and are particularly effective for smooth, convex problems; and reinforcement learning techniques, which frame optimisation as a sequential decision-making process with feedback from an environment. Each of these approaches offers distinct advantages and trade-offs with respect to convergence speed, scalability, sensitivity to local minima, and robustness to noise or non-differentiability. Among these, evolutionary algorithms stand out for their robustness and versatility in handling black-box functions, multi-objective formulations, and noisy evaluations Rocke [4].

Among the numerous evolutionary algorithms developed to address these interrogations, genetic optimisers have emerged as a versatile and robust class of meta-heuristics for exploring complex search spaces. As stochastic methods inspired by natural evolution, genetic optimisers work with a population of candidate solutions to iteratively evolve better solutions through selection, crossover, and mutation operations. Genetic optimisers have proven effective in finding near-global optima in diverse application areas, balancing solution space exploration with convergence toward promising regions [5]. However, they are known for significant shortcomings due to their inherent exploratory focus, which lacks local refinement capabilities, leading to inefficiency and suboptimal solutions.

Genetic optimisers are commonly categorised into two primary branches: genetic algorithms (GAs) and genetic programming (GP), each tailored to different types of solution representations. Genetic algorithms typically operate on parametrically encoded problems, where candidate solutions are described as fixed-length vectors of decision variables. This makes GAs particularly well-suited for engineering applications in which the optimisation objective is defined over a structured parameter space. Consequently, GAs have found widespread use in domains such as fluid flow control—e.g., the optimisation of plasma actuator parameters for flow reattachment behind a backward-facing step [6], or the enhancement of convective heat transfer in turbulent boundary layers via jet-based actuation [7]. Beyond flow applications, GAs have been employed in structural optimisation problems, including the design of 1D vibration-damped beams [8] and performance-driven architectural elements like building roofs [9]. They have also been explored in medical applications such as disease classification [10].

In contrast, genetic programming evolves variable-length symbolic expressions or computer programs instead of parameter vectors, making it highly effective for problems where the optimal solution is a function or decision-making policy. Although problems traditionally tackled with GP suffered from high computational demands and implementation complexity, recent advances in algorithmic design and parallel computing have significantly improved affordability. As a result, GP has gained traction in domains requiring interpretable, adaptive, and dynamic models—particularly in control systems and optimisation. In flow control, for instance, GP has been used to evolve control laws that maximise mixing in turbulent jets [11] and to develop strategies for stabilising complex flow regimes, such as wake control on a cylinder [12], and fluidic pinball [13], or drag reduction in Ahmed bodies [14, 15]. Its symbolic formulation capability has also proven valuable in solving ordinary and partial differential equations [16, 17], or in time series forecasting [18]. Outside engineering, GP has been applied to problems in finance—such as credit portfolio optimisation [19], and medicine, including medical data classification tasks [10], where the interpretability of symbolic rules is a key advantage. These diverse applications showcase GP's flexibility in producing time-dependent, adaptive, and interpretable control and decision-making strategies.

Over the years, the widespread adoption of genetic algorithms has spurred numerous methodological advancements aimed at improving their efficiency and reliability. These include improved selection mechanisms [20], adaptive crossover strategies [21, 22, 23], robust mutation operators [24, 25, 26, 27], and dynamic parameter tuning methods designed to adapt the search behaviour over time. Despite these improvements, both GAs and GP share a common limitation: their reliance on stochastic, population-based exploration often results in slow convergence and insufficient local refinement.

To address these shortcomings, hybrid methodologies have emerged as a powerful class of optimisation strategies that combine the global search capabilities of genetic techniques with local search mechanisms. These approaches aim to accelerate convergence toward near-global optima, reduce sensitivity to initial conditions, and prevent premature convergence to local minima. For instance, hybridising genetic optimisers with gradient-based methods have been shown to significantly improve convergence rates, particularly in smooth and differentiable landscapes [28]. Alterna-

tively, integrating local search algorithms [29] or stochastic refinement strategies such as Simulated Annealing [30] enhances robustness by enabling broader and more adaptive exploration of the solution space.

A particularly successful algorithm for local exploitation is the Downhill Simplex Method (DSM), valued for its simplicity, interpretability, robustness, and effectiveness in gradient-free scenarios. DSM has been widely used in hybrid frameworks with evolutionary algorithms [31], as well as with genetic algorithms [32], for tasks such as geometric and geoacoustic parameter estimation [33], chiller design optimisation [34], and, more recently, in combination with genetic programming for flow control applications [13]. Its continued popularity stems from its ability to complement global genetic strategies with efficient, gradient-free local optimisation, leading to improved convergence behaviour and solution quality in complex, high-dimensional problems.

By leveraging the complementary strengths of global and local search paradigms, such as combining genetic optimisation with the Downhill Simplex Method, hybrid optimisation frameworks offer a more balanced and effective approach for solving complex, high-dimensional engineering problems. These strategies form the foundation for advanced metaheuristics capable of addressing the increasing demands of modern applications in control, design, and optimisation.

Based on the need for robust hybrid genetic optimisers, we propose the Hybrid Genetic Optimiser (HyGO), building upon several hybrid mechanisms introduced by Cornejo Maceda et al. [13]. HyGO is designed as a flexible and extensible framework that integrates global and local search strategies, supporting seamless hybridisation with a wide range of local optimisers, such as BFGS and COBYLA, provided that proper parsing between the solution representation and the local method's input format is ensured. It includes robust uncertainty-handling mechanisms tailored for experimental optimisation, such as repeated evaluation of individuals and the exclusion of outliers based on user-defined thresholds. Additionally, HyGO implements a soft constraint system that regenerates invalid individuals rather than discarding them outright, preserving population diversity while respecting feasibility criteria. The framework also features built-in error recovery capabilities, including automatic checkpointing and secure fallback strategies that allow the optimisation to resume following evaluation failures. HyGO supports both parametric encodings and Linear Genetic Programming (LGP) formulations, enabling its application across a wide variety of problem types. Implemented entirely in Python, it offers a modular and customisable platform well-suited for integration into modern optimisation pipelines in both simulation-based and experimental contexts.

To validate the proposed hybrid optimisation framework, a series of test cases were conducted across diverse problem setups, each selected to highlight the strengths of both the GA and GP components, as well as the effectiveness of DSM-based local refinement. First, the GA was evaluated using standard analytical functions commonly employed to benchmark optimisation algorithms. These tests provided a controlled environment for assessing the stochastic performance of GA in terms of convergence speed and solution quality. Next, the GP component was applied to stabilise the damped Landau oscillator, a canonical test case that models the nonlinear dynamics of von Kármán vortex shedding behind a cylinder [35]. This test focused on the hybrid framework's capacity to improve GP performance through DSM enrichment, particularly in a non-trivial, time-dependent dynamic system. Finally, HyGO was employed for drag minimisation on an Ahmed body, a widely studied geometry in automotive aerodynamics, to evaluate the framework's applicability to complex, real-world problems. This case not only demonstrated the practical capabilities of HyGO in high-dimensional design spaces but also enabled direct comparison with the optimisation approach proposed by Li et al. [36], providing a broader context for assessing the effectiveness of the hybrid methodology.

The remainder of this paper is structured as follows. Section 2 introduces the proposed hybrid genetic optimisation framework (HyGO), detailing its architecture, genetic components, and local refinement strategy. Section 3 evaluates HyGO 's performance on standard analytical functions, comparing its convergence behaviour and robustness against classical optimisation techniques. In section 4, the framework is applied to the control of the damped Landau oscillator using LGP, serving as a benchmark for time-dependent control tasks. Section 5 presents a real-world application, optimising drag reduction on an Ahmed body using RANS simulations. Finally, section 6 concludes the paper and outlines directions for future research.

2 Hybrid Optimisation with HyGO

In this section, we introduce HyGO, our novel optimisation framework designed to facilitate the creation and deployment of hybrid genetic optimisers. This framework provides a structured environment for merging evolutionary

algorithms with various local enrichment methods. The methodology is illustrated with the merging of parametric and function optimisers with a local search algorithm. The optimisers considered are genetic algorithm and linear genetic programming for the parametric and function optimisation, respectively. Local search is carried out with a variant of the downhill simplex algorithm. The subsequent subsections will detail the architectural components of HyGO, discuss the critical balance between exploration and exploitation within hybrid optimisation, outline the roles of evolutionary algorithms for broad search, describe the integration of local refinement techniques, and finally, present the specific implementation of our hybrid methodology within the HyGO framework.

2.1 Optimisation Framework

Let $J(\mathbf{f};\theta)$ represent the optimisation objective, or cost function, where $\mathbf{f} \in \mathscr{F}$ is a vector of optimisation variables within the feasible search space $\mathscr{F}$, and θ represents a set of fixed problem parameters defining the environmental or problem conditions, common to all evaluations. The optimisation problem aims to determine the optimal set of variables $\mathbf{f}^*$ such that:

$$\mathbf{f}^* = \arg\min_{\mathbf{f}\in\mathscr{F}} J(\mathbf{f};\theta) \tag{1}$$

Genetic optimisers address this problem by emulating natural evolution through *survival of the fittest* mechanisms. In these algorithms, each specific candidate solution within the search space is represented as an *individual*, and a collection of these individuals forms a *population*. The optimisation process is iterative, where each subsequent population $i+1$ is generated from the previous population i by applying evolutionary operators such as replication, crossover, and mutation. This process aims to iteratively improve the population of candidate solutions over successive generations, ultimately converging towards an optimal or near-optimal solution.

The optimisation process begins with an initial population of I individuals, generated using Monte Carlo Sampling (MCS) (alternatively, random or Latin hypercube sampling can be used) to ensure a diverse starting pool of solutions. Each individual is then evaluated based on the cost function, and the population is ranked according to performance:

$$J_i^1 \leq J_i^2 \leq \cdots \leq J_i^I \tag{2}$$

The evolution of the population towards better solutions is driven by four key operations: elitism, replication, mutation, and crossover. Elitism ensures that the top N_e individuals, based on performance, are directly passed to the next generation, preserving the best solutions found. For the remaining individuals, a tournament selection process is employed to choose parents for the subsequent evolutionary operations. In each tournament, N_t individuals are randomly selected from the current population and ranked based on their cost function values. The individual with the lowest cost is chosen as a parent with probability p_s. If this individual is not selected, the second-best is considered with the same probability, and so on, until a parent is selected. This method favors fitter individuals for reproduction while maintaining diversity within the population.

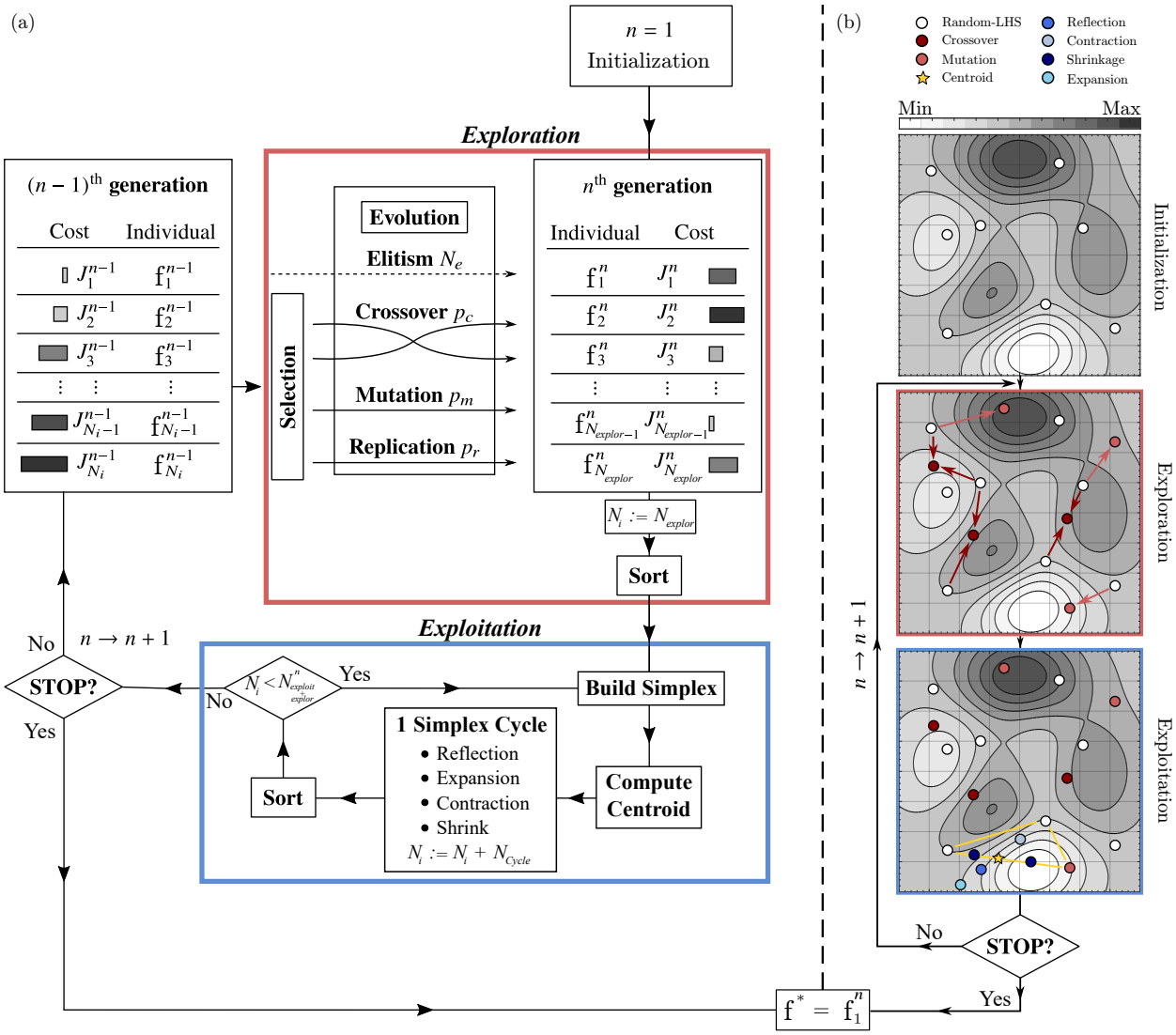


Figure 1: Hybrid Genetic Optimisation algorithm: exploration and exploitation scheme. (a) Schematic flowchart, highlighting its two-phase structure. In each generation n, exploration is performed by creating individuals $\mathbf{f}_i^n$ via genetic operations, which include elitism, crossover, mutation, and replication, selected through a tournament process; the resulting solutions are evaluated and sorted according to the cost function J_i^n. After exploration, exploitation is performed by applying DSM to refine the best candidates. The horizontal bars visually encode individual performance within the population, with brighter and shorter bars indicating lower (better) costs and darker, longer bars reflecting higher (poorer) solutions. (b) Conceptual map illustrating an example trajectory in a two-dimensional landscape, showing how the algorithm alternates between global exploration and local exploitation. Coloured points indicate the origin of each solution: random initialisation, genetic operations (crossover, mutation), and simplex-based moves (reflection, expansion, contraction, shrinkage, centroid). Light-shaded regions correspond to low (optimal) cost areas, while dark regions denote high (suboptimal) cost; the arrows and markers trace how exploration enables the population to sample broadly, while exploitation directs progress toward local minima.

Once parent selection is completed, genetic operators are applied probabilistically to generate the next generation of offspring, alongside the elitism individuals. Each selected parent (for replication or mutation) or pair of parents (for crossover) undergoes one of the following operations, based on predefined probabilities P_r, P_m, and P_c, where $P_r + P_c + P_m = 1$. Replication carries over the selected individuals to the next generation, excluding the elite, ensuring that promising solutions remain in the genetic pool. Mutation introduces random changes to a parent's genetic material to effectively explore the search space $\mathscr{B}$. Crossover combines the genetic material of two selected parents to produce one or more new offspring, enabling exploitation of promising solutions. The operation probabilities govern the balance between exploiting individuals from the previous generation and exploring new regions of the search space.

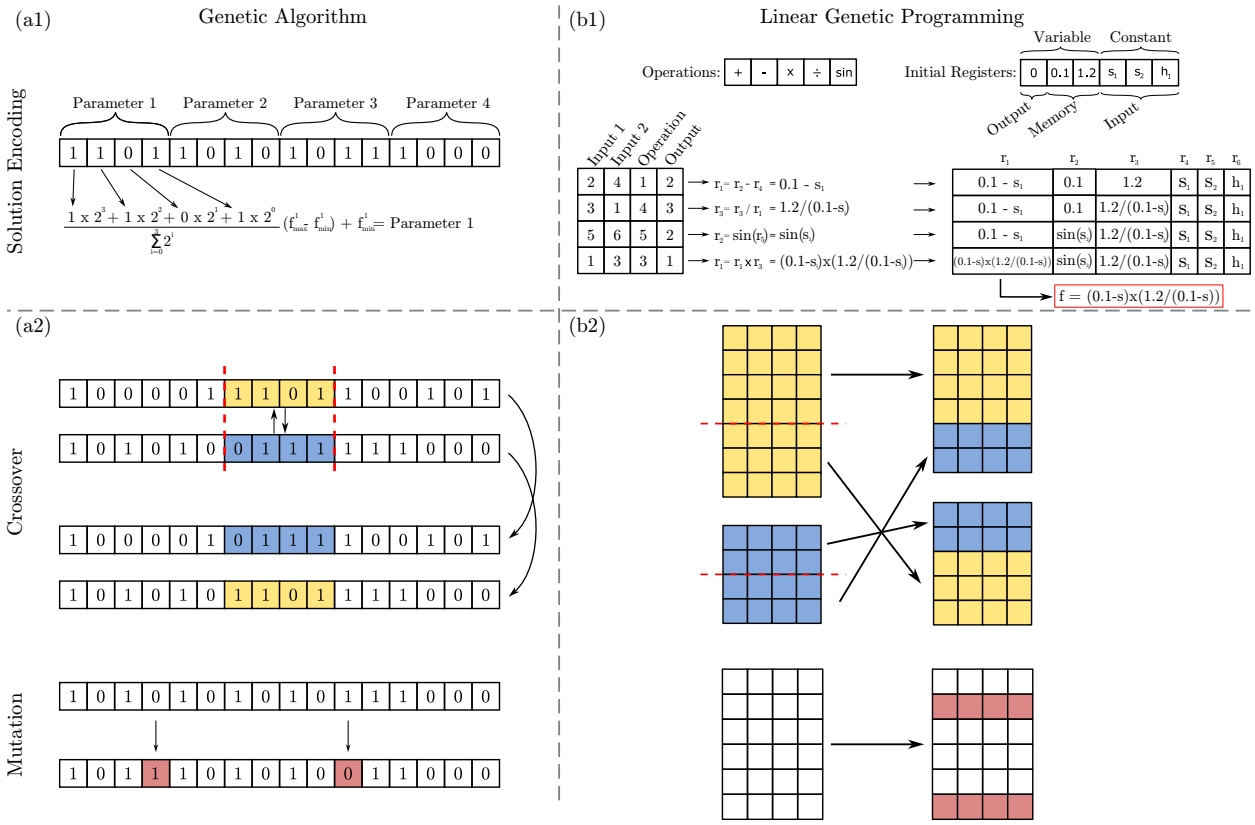


Figure 2: Solution encoding and genetic operations for Genetic Algorithm (GA) and Linear Genetic Programming (LGP). (a1) Binary-encoded chromosomes for GA represent sets of parameters. (b1) Encoding of LGP individuals as instruction matrices acting on program registers (variables, constants, memory, and inputs). (a2, b2) Examples of genetic operations: (a2) In GA, crossover swaps chromosome segments between parents (highlighted), and mutation randomly flips selected bits. (b2) In LGP, crossover exchanges blocks of instructions between programs, and mutation replaces random instructions, enabling structural diversity in candidate solutions.

The procedure allows iterating through generations until convergence or a maximum number of generations L or evaluations $M = I \times L$ is reached. The general optimisation procedure for genetic optimisers is included in Figure 1 bypassing the Exploitation section.

Within this general optimisation structure, genetic optimisers differentiate themselves through the genome representation used for individuals. The choice of encoding directly influences the algorithm's behaviour, allowing specific characteristics to be promoted, such as exploration, exploitation, or biasing the search to better match the nature of the problem. In the case of Genetic Algorithms, genomes typically consist of chromosomes that encode sets of fixed parameters, most commonly using binary encoding due to its simplicity, generality, and compatibility with standard operators such as crossover and mutation. This makes binary-encoded GAs especially effective for parametric optimisation tasks where candidate solutions can be naturally expressed as vectors of design variables.

By contrast, Genetic Programming represents individuals as computer programs rather than parameter vectors. This approach is particularly advantageous for problems requiring the evolution of decision-making rules or dynamic control strategies. A widely used and interpretable variant is Linear Genetic Programming, in which individuals are encoded as instruction matrices operating on a shared set of registers. This format allows for compact, symbolic representations of control laws that evolve over time or in response to sensor inputs, making LGP especially well-suited to problems such as real-time flow control.

2.1.1 Genetic Algorithm

In GAs, $\mathbf{f}$ is expressed as a vector of N actuation parameters $\mathbf{f} = [f_1, \ldots, f_N]^T$, within the domain Ω,

$$\mathbf{f} = [f_1, \ldots, f_N]^T \in \Omega \subset \mathbb{R}^N, \tag{3}$$

where each parameter is defined within a valid interval, $f_i \in [f_{i,\min}, f_{i,\max}]$, $i = 1, \ldots, N$, leading to a rectangular parametric domain Ω

$$\Omega = [f_{1,\min}, f_{1,\max}] \times \cdots \times [f_{N,\min}, f_{N,\max}]. \tag{4}$$

In binary-encoded GAs, each of the parameters b_i is represented by a string of n binary digits (bits), yielding a resolution of 2^n distinct values over the interval $[b_{i,\min}, b_{i,\max}]$. Typically, these values are distributed uniformly, and decoding the binary string yields a corresponding real-valued approximation of the parameter.

This encoding enables the straightforward application of genetic operators. The crossover operator partitions two parent chromosomes in N_S segments and interchanges them sequentially, or randomly, to generate one or two offspring. Meanwhile, the mutation operator introduces variation by flipping individual bits in a parent chromosome with a probability p_m, producing a new individual and promoting exploration of the search space. A visual summary of the encoding and the genetic operations is provided in Figure 2 for further clarification.

2.1.2 Linear Genetic Programming

In Genetic Programming, the actuation command $\mathbf{f}$ is expressed as an analytical function that depends on a set of numerical constants, external sensor inputs $\mathbf{s}$, and prescribed time-dependent functions $\mathbf{h}$.

$$\mathbf{f} = \mathbf{f}(\mathbf{h}, \mathbf{s}) \tag{5}$$

This formulation enables a continuous and adaptive response to the system state, making GP particularly well-suited for control applications [37]. Early implementations of GP employed tree-based representations, where control laws were encoded as hierarchical structures of operations, such as $+$, $-$, $\div$, $\times$, $max()$, $sin()$, and numerical constants. However, these tree-based models often suffered from uncontrolled growth and structural complexity, leading to the development of alternative representations.

Among these alternatives, Linear Genetic Programming (LGP) offers a more compact and interpretable format by encoding individuals as sequences of instructions, stored in matrices (see Figure 2 (b1)), that operate on a shared set of registers. Following the methodology introduced by Brameier and Banzhaf [38], each chromosome is expressed as an $N_{it} \times 4$ instruction matrix, which is interpreted sequentially from top to bottom.

This matrix consists of N_{it} instructions, each specifying how to combine elements from a set of registers containing time-dependent functions h_i, sensor readings s_i, or intermediate results. Each row in the matrix defines a single instruction composed of four fields: the first two columns specify the indices of the input arguments (if the operation is unitary, the second input is ignored), the third column denotes the index of the operation to be applied, and the fourth column indicates the destination register where the result will be stored. All individuals begin with the same initial set of input registers, and through sequential application of instructions, each evolves into a unique register configuration that defines the control output $\mathbf{f}$. The sequential nature of instruction execution is what defines the method as linear. The initial register set contains N_{out} output registers, which hold the resulting $\mathbf{f}$ vector for each individual, N_{mem} memory registers, used to store intermediate results and enable complex instruction interactions, and N_{in} input registers, which provide access to all sensor readings and time-dependent functions used in the optimisation. Registers can be either variable, allowing instructions to modify their contents, or constant, making their values accessible to every instruction without modification. Typically, sensors, time-dependent functions, and predefined constants are stored in constant registers, while the output register must always be variable. An example illustrating how a control law is constructed using this matrix-based representation is shown in Figure 2.

The application of genetic operations in LGP follows principles similar to those used in Genetic Algorithms, while preserving instruction-level coherence. The crossover operator exchanges subsets of instructions between two parent individuals, allowing for structural recombination without disrupting the logic of instruction sequences. Unlike binary-encoded GAs, LGP permits individuals to have varying chromosome lengths (up to a maximum ofN_{it}), introducing additional diversity into the population. The mutation operator modifies instructions with a probability p_m, replacing them with newly generated random instructions, which further enhances exploration of the solution space.

2.2 Exploration vs Exploitation

Genetic optimisation is commonly classified as a global optimisation technique due to its inherent ability to combine *exploration*, the broad search of the solution space, with *exploitation*, the local refinement of promising solutions. This dual capability makes evolutionary algorithms particularly attractive for solving complex optimisation problems, especially in cases where traditional gradient-based or local methods struggle to escape local minima or require smoothness and convexity in the objective function. In practical search spaces, several scenarios highlight the importance of balancing these two strategies. Large plateaus in the cost landscape require strong exploratory behaviour, as the lack of gradient information provides little guidance for local descent. Conversely, narrow valleys or funnel-shaped regions benefit from exploitative steps that allow the algorithm to "slide" efficiently toward an optimum. Other areas may exhibit very shallow gradients, where convergence is technically possible but prohibitively slow, highlighting the need for exploration to "jump" closer to steeper descent regions and accelerate progress toward better solutions.

Genetic optimisation algorithms are specially well-suited for promoting exploration due to the high sensitivity of solutions stemming from the genotype-to-solution mapping. Small modifications in the encoded individual, such as a bit flip in a chromosome or a change in a program instruction, can lead to drastically different control outputs. This behaviour is especially significant in control law design, where solutions that are close in the space of actuation commands may correspond to vastly different genetic representations. As a result, Genetic Algorithms and Genetic Programming tend to be inherently more explorative than exploitative, excelling at discovering diverse candidate solutions but often requiring additional mechanisms for effective fine-tuning.

Genetic optimisation have traditionally addressed the challenge of balancing exploration and exploitation primarily by adjusting the probabilities of replication, mutation, and crossover operations. Mutation introduces random modifications that enable exploration of the search space, and increasing the mutation probability P_m directly enhances this exploratory behaviour. In contrast, crossover promotes exploitation by combining genetic material from parent individuals, effectively moving the population toward local minima already identified. Beyond these two straightforward mechanisms, other strategies exist, for example, increasing the replication probability P_r allocates more individuals to already explored regions, which reduces exploration without directly promoting exploitation. Additionally, the mutation probability within the genome, p_m, controls how drastically offspring differ from their parents, allowing jumps across plateaus or to new local minima. However, excessively high mutation rates can lead to a largely random search that neglects valuable information embedded in the current population, thereby undermining effective exploration. Another important factor influencing the exploration-exploitation balance is the tournament selection process. Here, a subset of individuals is randomly selected and ranked based on cost, and parents are chosen probabilistically in hierarchical order. Lower selection probabilities increase exploration by allowing individuals ranked lower in the tournament (i.e., worse-performing) a greater chance of selection. Conversely, larger tournament sizes promote exploitation by increasing the likelihood of selecting fitter individuals who have higher chances of reproduction.

However, over the past decades, the focus in genetic optimisation has progressively shifted from primarily tuning hyperparameters towards developing hybrid algorithms that combine genetic optimisers with complementary optimisation techniques to enhance overall performance. Genetic optimisers are highly effective at exploring vast and complex search spaces, however, their ability to efficiently exploit local minima is often limited by the inherently sub-optimal nature of standard crossover operation. Although numerous enhancements to crossover strategies have been proposed to mitigate this issue [21, 23], such improvements alone are frequently insufficient to guarantee rapid and precise convergence.

To address this challenge, hybridisation with local search algorithms has become the most prevalent and successful strategy. By integrating global search capabilities of GOs with the precision and speed of local refinement methods, hybrid algorithms can effectively balance exploration and exploitation. This synergy allows the optimiser to broadly investigate the solution space while simultaneously focusing in on local optima with greater efficiency. The resulting hybrid frameworks have demonstrated superior convergence rates and solution quality across a wide range of complex optimisation problems in engineering and scientific domains, making them a powerful tool in the optimisation community.

2.3 Hybridizing Genetic Optimisers with local search algorithms

Building on the foundational concepts of exploration and exploitation, and motivated by the limitations of genetic optimisers in efficiently refining local optima, this subsection introduces the hybrid optimisation methodology developed in this study. The proposed approach synergistically combines the global search strengths of binary-encoded

Algorithm 1 Hybrid Genetic Optimiser (HyGO) Pseudo-code

— Initialisation —

Generate N_{explor} individuals randomly or with LHS $\rightarrow \mathbf{f}_r^1, \;\; r = 1,2,\ldots,N_{explor}$
Evaluate the fitness of each individual in the population $\rightarrow J(\mathbf{f}_r^1;\theta), \;\; r = 1,2,\ldots,N_{explor}$.
Sort the population according to cost $\rightarrow \{J_1^1 < J_2^1 < \ldots < J_{N_{explor}}^1\}$.

— Exploitative stage —

$N_{ind}^1 = N_{explor}$
while $N_{ind}^1 < (N_{explor} + N_{exploit})$ **do**
    Generate individual with the local search method $\rightarrow \mathbf{f}_i^1$
    Evaluate the fitness of offspring $\rightarrow J(\mathbf{f}_i^1;\theta)$
    $N_{ind}^1 := N_{ind}^1 + N_{cycle}$
end while
Sort the population according to cost $\rightarrow \{J_1^1 < J_2^1 < \ldots < J_{N_{explor}+N_{exploit}}^1\}$.
$g = 2$

while not convergence **and** $g < N_g$ **do**

— Explorative stage —

    for $i = 1$ to N_{explor} **do**
        Perform tournament selection process to select parents
        Generate an individual by elitism/replication/crossover/mutation $\rightarrow \mathbf{f}_i^g$
    end for
    Evaluate the fitness of each individual in the population $\rightarrow J(\mathbf{f}_r^g;\theta), \;\; r = 1,2,\ldots,N_{explor}$.
    Sort the population according to cost $\rightarrow \{J_1^g < J_2^g < \ldots < J_{N_{explor}}^g\}$

— Exploitative stage —

    $N_{ind}^g = N_{explor}$
    while $N_{ind}^g < (N_{explor} + N_{exploit})$ **do**
        Generate individual with the local search method $\rightarrow \mathbf{f}_i^g$
        Evaluate the fitness of offspring $\rightarrow J(\mathbf{f}_i^g;\theta)$
        $N_{ind}^g := N_{ind}^g + N_{cycle}$
    end while
    Sort the population according to cost $\rightarrow \{J_1^g < J_2^g < \ldots < J_{N_{explor}+N_{exploit}}^g\}$.
    $g = g + 1$
end while

Return the best solution found $\rightarrow \mathbf{f}^* = \arg\min_{\mathbf{f}} J(\mathbf{f};\theta)$.

Genetic Algorithms and Linear Genetic Programming with local search techniques. This hybridisation is designed to maintain the broad exploratory capabilities of evolutionary algorithms while significantly enhancing their local exploitation power, resulting in improved convergence speed and solution quality.

To achieve this goal, the optimisation process in each generation consists of two distinct phases: an *explorative* phase, where genetic operations generate N_r new individuals from the existing population, and an *exploitative* phase, in which local search algorithms produce $N_{exploit}$ refined individuals. This structured alternation balances exploration and exploitation effectively within each generation. A detailed description of the algorithm is provided in Algorithm 1, and a visual representation of the two-phase optimisation is included in Figure 1.

A key challenge in maintaining the effectiveness of this iterative process is preserving genetic diversity throughout the evolutionary cycles, ensuring an effective exploration process to maintain the exploration-exploitation balance. The genetic operations described in subsection 2.1 are inherently stochastic, owing to the random selection of crossover points and mutation bit flips. As the optimisation progresses and genetic diversity diminishes, the likelihood of generating individuals identical to those from previous generations increases, which undermines the effectiveness of the evolutionary process. Repeated individuals not only waste valuable computational resources but also significantly

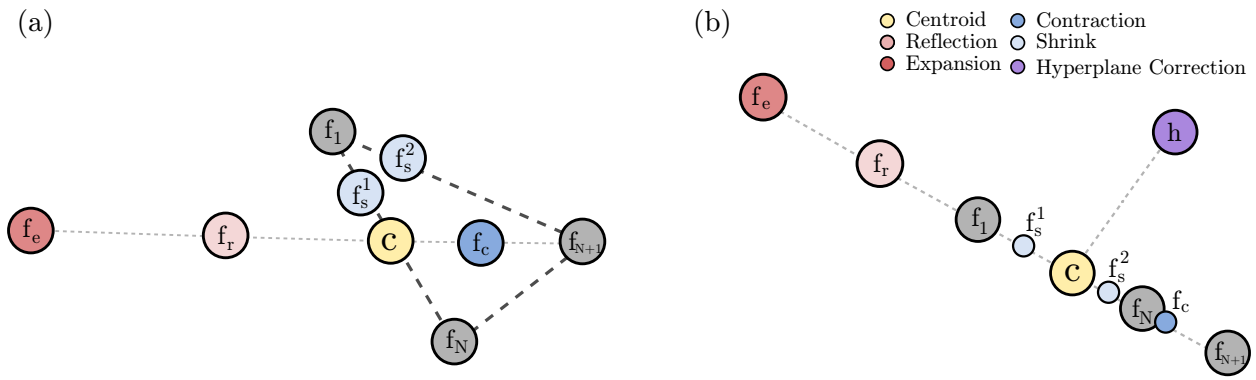


Figure 3: Hyperplane degeneracy into a 1-dimensional hyperplane in a 2D parametric domain. Image (a) displays all the operations and the topology under normal conditions. Image (b) shows a degenerate scenario where the Simplex collapsed into a line and lost the ability to move in the normal direction.

reduce the genetic diversity essential for robust search performance. To mitigate this, newly generated individuals from crossover or mutation undergo a validity check to determine if they already exist in the population. If an individual is found to be a duplicate, it is discarded and regenerated, with the number of regeneration attempts capped to prevent endless loops.

This validity-checking mechanism can be extended to enforce *soft constraints*, a longstanding challenge in genetic algorithms [39, 40, 41]. The term "soft" is used because strict constraint satisfaction cannot be guaranteed due to the limited number of regeneration attempts; if the constraint cannot be met, the individual may either be assigned a penalizing extreme cost or evaluated regardless. However, this issue typically arises only in later generations when genetic diversity is already low and convergence is near completion, minimizing its practical impact.

Downhill Simplex Method: A Robust Gradient-Free optimisation Technique

The Downhill Simplex Method (DSM), originally proposed by Nelder and Mead [42], is a widely used optimisation algorithm notable for its simplicity and robustness. Unlike gradient-based methods, DSM does not require explicit gradient information, which makes it especially well-suited for problems where derivatives are difficult or expensive to compute. This characteristic positions DSM as an effective local search algorithm within the HyGO framework, primarily to refine candidate solutions identified by global search heuristics.

DSM operates by constructing a simplex, a geometric figure composed of $N+1$ vertices in an N-dimensional parameter space, starting with an initial simplex spanning the domain Ω. The objective of each simplex iteration is to replace the vertex with the highest cost, $\mathbf{f}_h$, with a new, improved vertex $\mathbf{f}_{N+2}$. The iteration proceeds through the following steps:

1. **Ordering:** The vertices are sorted by their corresponding cost values $J_m = J(\mathbf{f}_m)$, such that $J_1 \leq J_2 \leq \cdots \leq J_{N+1}$.
2. **Centroid Calculation:** Compute the centroid $\mathbf{c}$ of the simplex, excluding the worst vertex $\mathbf{f}_{N+1}$:

$$\mathbf{c} = \frac{1}{N} \sum_{m=1}^{N} \mathbf{f}_m. \tag{6}$$

3. **Reflection:** Reflect the worst vertex $\mathbf{f}_{N+1}$ across the centroid $\mathbf{c}$ to generate a new vertex $\mathbf{f}_r$:

$$\mathbf{f}_r = \mathbf{c} + (\mathbf{c} - \mathbf{f}_{N+1}). \tag{7}$$

   If the cost of the reflected vertex $J_r = J(\mathbf{f}_r)$ falls within $J_1 \leq J_r \leq J_N$, replace $\mathbf{f}_{N+1}$ with $\mathbf{f}_r$ and proceed to the next iteration.
4. **Expansion:** If the reflection improves the cost beyond the best current value $J_r < J_1$, the simplex is expanded further in this direction:

$$\mathbf{f}_e = \mathbf{c} + 2(\mathbf{c} - \mathbf{f}_{N+1}). \tag{8}$$

The vertex with the lower cost between $\mathbf{f}_r$ and $\mathbf{f}_e$ replaces $\mathbf{f}_{N+1}$, and a new iteration begins.

5. **Contraction:** If the reflection does not improve the cost $J_r \geq J_N$, the algorithm contracts the simplex by moving the worst vertex halfway towards the centroid:

$$\mathbf{f}_c = \mathbf{c} + \frac{1}{2}(\mathbf{f}_{N+1} - \mathbf{c}). \tag{9}$$

   If $\mathbf{f}_c$ improves the cost, it replaces $\mathbf{f}_{N+1}$, and the next iteration starts.

6. **Shrinkage:** If neither reflection nor contraction improves the cost, the simplex is shrunk by reducing the distance of all vertices from the best vertex $\mathbf{f}_1$ by half:

$$\mathbf{f}_m \to \mathbf{f}_1 + \frac{1}{2}(\mathbf{f}_m - \mathbf{f}_1), \quad m = 2, \ldots, N+1. \tag{10}$$

   Shrinking is a last-resort operation, as it requires N additional function evaluations and is generally applied when the simplex has degenerated, meaning it may better follow the local gradients within a more confined region of the parameter space.
   The results of these operations are represented geometrically in Figure 3 (a).

The algorithm described and implemented in HyGO corresponds to the classical version of the Downhill Simplex Method, despite the existence of various alternative variants in the literature. While this classical DSM can be directly applied to parametric optimisation problems by mapping the solutions to the nearest binary-encoded points within the search domain, its application becomes more nuanced for matrix-encoded Linear Genetic Programming individuals. In such cases, the more sophisticated Subplex algorithm developed by Rowan [43] is employed. This approach, which has demonstrated success in prior work such as Cornejo Maceda et al. [13], introduces additional complexity and is beyond the detailed scope of this paper.

DSM is included in the exploitation phase of HyGO by performing a set of DSM iterations until generating $N_{exploit}$ individuals. However, the shrink operation of the simplex during the exploitation phase can produce more candidate individuals than the prescribed $N_{exploit}$. Truncating the number of individuals to enforce a fixed population size would disrupt the intended search dynamics and reduce algorithmic flexibility. Therefore, the size of $N_{exploit}$ can vary naturally. Like most optimisation algorithms, the DSM is sensitive to the initialisation of the simplex. A common practice is to start with one vertex at the center of the domain and N vertices distributed along each coordinate axis. However, in the methodology proposed here, the initial simplex within each population is constructed from the $N+1$ fittest individuals.

For objective functions $f \subset \mathbb{R}^m$ that exhibit directional gradients, losing control over the initial simplex configuration may cause it to degenerate into a lower-dimensional hyperplane $g \subset \mathbb{R}^n$ with $n < m$. In such cases, the DSM operations cannot recover the missing directions of movement, resulting in inefficient search behavior—particularly when the gradient path is non-linear, as in spiral-shaped functions.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{11}$$

To address this issue, HyGO checks whether the last N_f individuals generated by the simplex lie approximately on a hyperplane g. This is done by computing the coefficient of determination R^2 and comparing it to a user-defined threshold (see Equation 11). If the degeneracy condition is met, a new individual is generated in the direction $\vec{u}$ normal to the hyperplane g (randomly selecting between $\vec{u}$ and $-\vec{u}$), starting from the centroid and positioned at a distance determined by the grid resolution in the parametric space. Once this correction is applied, a new DSM cycle begins, re-building the simplex with the best $N+1$ individuals of the population to accommodate this correction individual if required.

While this corrective strategy may introduce additional individuals unnecessarily when the search path is strongly directional, increasing computational cost, the resulting adaptability has proven crucial for maintaining search effectiveness in many scenarios.

Table 1: Algorithm parameters for the analytical functions.

Name	Value	Description
		Common
D	5/25	Dimension
N_b	12	Number of bits for each parameter
N_G	50	Total number of generations
N_T	7/100	Tournament size
P_c	0.55	Crossover probability
P_m	0.45	Mutation probability
P_r	0	Elitism probability
		HyGO
N_{MC}	70	Monte Carlo initialisation
N_{explor}	70	Population size (exploration)
$N_{exploit}$	30	Simplex size (exploitation)
		GA
N_{MC}	100	Monte Carlo initialisation
N_{explor}	100	Population size (exploration)

3 Parametric optimisation on Analytical Functions

An envelope of traditionally complex analytical functions served as a baseline to test the convergence performance of the developed hybrid DSM-GA optimiser. Optimizing these analytical functions with and without the exploitation phase permitted an assessment of its effect on the optimisation. Furthermore, to represent a modern application, the comparison included another evolutionary optimiser, the CMA-ES algorithm. CMA-ES employs multivariate Gaussians distributions to generate populations as well enhancing convergence by exploiting the evolution path [44, 45].

3.1 Preliminary optimisation

To illustrate the effects that gradient-enriching a genetic algorithm has on convergence, an optimisation on the 2D Rosenbrock function was performed. The results comprise five generations of 50 individuals, split into 30 generated by the explorative phase and 20 created by the exploitative phase. The initialisation consisted of 30 randomly generated individuals within the search space $x_i \in [-5,5]$, illustrated in Figure 4 (b) as the black dots in the first generation. Convergence is significantly expedited due to the exploitation of individuals in all generations, represented in different blue tones in Figure 4 (a), always present in the lower fraction of the cost distribution. In the first generation, the cost gains due to exploitation are significant due to the large gradients far from the characteristic valley of the Rosenbrock function. In later generations, improvements are much slower due to the shallow valley where the minimum is located. The vanishing gradients cause the simple steps to be subsequently smaller, lowering the exploitation capabilities. This effect is especially critical in the last generation, where the simplex individuals are slowly moving towards the global minima $x_i^* = [1,1]$ (as depicted by the red arrow in Figure 4 (b)) due to the small gradients present in the valley.

These results visually represent the strengths and characteristics of both the explorative and exploitative phases of the newly developed optimisation framework. However, the results are far from ideal in terms of optimisation performance. The results are subject to convergence improvements if some hyperparameters are accurately tuned. For example, generating the initial samples with Latin hypercube Sampling would prevent the first generation from moving towards a local minimum far from the global one, eliminating the slow evolution through the shallow valley. Furthermore, the selected mutation type was *at least one*, which sets the bit mutation probability such that most of the mutated individuals only have a single bit changed, modifying only one of the two parameters. This restriction forces the mutated individuals to move vertically or horizontally from the best individuals' cluster, as observed in generation

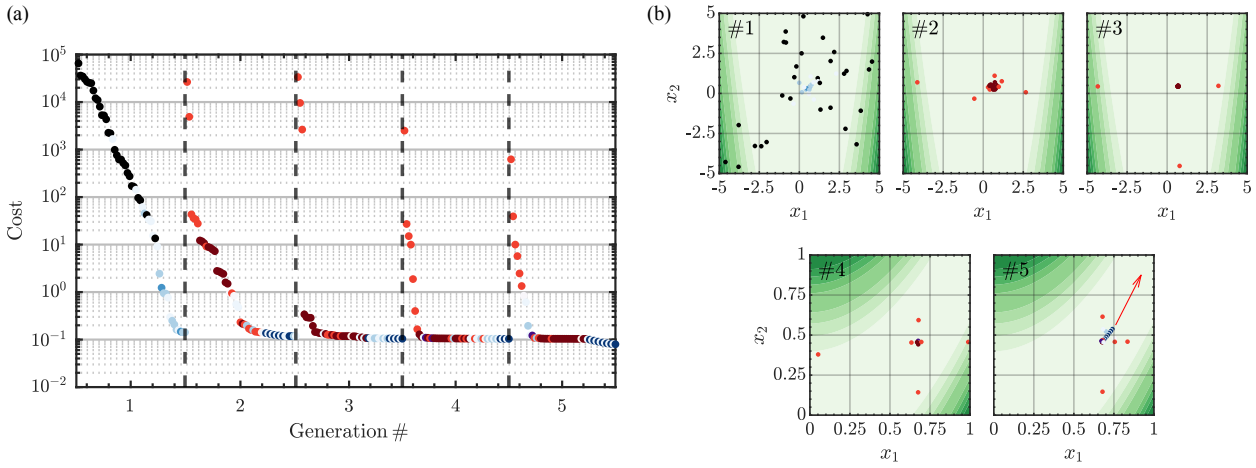


Figure 4: Example of HyGO performance on the optimisation of the Rosenbrock function in 2D: (a) cost evolution of the individuals through the generations. (b) Individuals' distribution in the parametric space in each of the five generations (#1 – 5). The arrow in generation #5 shows the tendency of the DSM solution towards the global minimum of the Rosenbrock function at (1,1). Individuals are coloured by the type of operation that created each individual (see Figure 1, black for random initialisation, red tones for the different genetic operations, and blue tones for the DSM operations).

4 in Figure 4 (b). Forcing a higher bit mutation probability might have enhanced the exploration capabilities, enabling diagonal mutation and not only relying on the exploitation phase.

3.2 Performance characteristics of HyGO

Assessing the performance characteristics of a non-deterministic optimisation method requires a stochastic analysis. A total of 15 analytical functions with distinct topological characteristics serve as benchmarks to try and understand the behavior of HyGO. To study the algorithm's performance in high-dimensional problems, some of these functions were optimised in low and high-dimensional scenarios, causing the total number of distinct scenarios to rise to 20. The comparison includes the classical GA, HyGO, and the CMA-ES algorithm (only included in high dimensions), limiting the number of evaluations to 5000. The analysis consisted of a k-fold study with 50 runs and randomised initial individuals to eliminate initialisation dependence. Table 3 contains the results of the analysis.

The DSM-enriched GA (HyGO) surpasses the classical GA optimisation in all performance metrics presented in Table 3. The convergence criteria employed for the analysis are strict since only a single combination of parameters serves as a global minimum. HyGO outperforms GA in half of the analytical functions, with a significant margin in most of them, reflected in the best cost average. Both algorithms reach the same convergence in 8 functions. However, 4 are high-dimensional cases where finding a specific individual with 25 parameters in a 2^{12} search space is almost impossible. In these high-dimensional scenarios, the relevant metric is the best cost average, where HyGO achieves a lower cost.

Only in the Ackley, Rastringuin, and Sphere functions in 2 dimensions GA outperforms HyGO in terms of convergence (The Sphere function achieves 100% convergence in both algorithms because of its geometric simplicity) and average best cost. Subsection 3.3 features an in-depth analysis of these functions to provide a more thorough analysis. Moreover, HyGO presents fewer evaluations and generations in most functions where convergence was commonly reached, with some exceptions, which include the previously mentioned cases. An interesting result is the Bukin N.6 function, which presents a complex topology featuring a valley filled with very similar local minima that complicate the optimisation. The number of evaluations metric must not be considered in the scenarios with 0 convergence since a lower number of evaluations is due to repeated individuals that could not be re-generated within the maximum number of attempts.

3.3 High-dimensionality performance

The Rosenbrock, Rastringuin, and Sphere functions serve to analyze the effect high dimensionality has on the algorithm's performance. These functions were selected due to the out-performance of GA in low dimensions and

Table 2: Analytical Functions and Their Parametric Limits

Function	Mathematical Formulation	Search Space
Ackley	$f(\mathbf{x}) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + e + 20$	$x_i \in [-5,5]$
Beale	$f(x_1,x_2) = (1.5 - x_1 + x_1x_2)^2 + (2.25 - x_1 + x_1x_2^2)^2 + (2.625 - x_1 + x_1x_2^3)^2$	$x_1,x_2 \in [-4.5,4.5]$
Booth	$f(x_1,x_2) = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2$	$x_1,x_2 \in [-10,10]$
Bukin N.6	$f(x_1,x_2) = 100\sqrt{\lvert x_2 - 0.01x_1^2\rvert} + 0.01\lvert x_1 + 10\rvert$	$x_1 \in [-15,-5]$, $x_2 \in [-3,3]$
Easom	$f(x_1,x_2) = -\cos(x_1)\cos(x_2)\exp\left(-((x_1-\pi)^2 + (x_2-\pi)^2)\right)$	$x_1,x_2 \in [-100,100]$
Eggholder	$f(x_1,x_2) = -(x_2+47)\sin\left(\sqrt{\lvert x_1/2 + (x_2+47)\rvert}\right) - x_1\sin\left(\sqrt{\lvert x_1 - (x_2+47)\rvert}\right)$	$x_1,x_2 \in [-512,512]$
Goldstein-Price	$f(x_1,x_2) = \left[1 + (x_1+x_2+1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)\right]$ $\times \left[30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)\right]$	$x_1,x_2 \in [-2,2]$
Himmelblau's	$f(x_1,x_2) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2$	$x_1,x_2 \in [-6,6]$
Holder Table	$f(x_1,x_2) = -\lvert\sin(x_1)\cos(x_2)\exp\left(\lvert 1 - \sqrt{x_1^2+x_2^2}/\pi\rvert\right)\rvert$	$x_1,x_2 \in [-10,10]$
Levi N.13	$f(x_1,x_2) = \sin^2(3\pi x_1) + (x_1-1)^2\left(1 + \sin^2(3\pi x_2)\right) + (x_2-1)^2\left(1 + \sin^2(2\pi x_2)\right)$	$x_1,x_2 \in [-10,10]$
Matyas	$f(x_1,x_2) = 0.26(x_1^2 + x_2^2) - 0.48x_1x_2$	$x_1,x_2 \in [-10,10]$
Sphere	$f(\mathbf{x}) = \sum_{i=1}^{n} x_i^2$	$x_i \in [0,2]$
Rastrigin	$f(\mathbf{x}) = 10n + \sum_{i=1}^{n}\left[x_i^2 - 10\cos(2\pi x_i)\right]$	$x_i \in [-5.12,5.12]$
Rosenbrock	$f(\mathbf{x}) = \sum_{i=1}^{n-1}\left[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right]$	$x_i \in [-5,5]$
Styblinski-Tang	$f(\mathbf{x}) = \frac{1}{2}\sum_{i=1}^{n}\left(x_i^4 - 16x_i^2 + 5x_i\right)$	$x_i \in [-5,5]$

their topological characteristics, enabling the mapping of exploration and exploitation-dominated functions. The Sphere function is gradient-controlled since it has a single global minimum and a clear direction of descent. The Rastringuin function has many global minima, causing the optimisation to be dominated by exploration since the prime aspect for convergence is to find the global minima among all the local minima. Lastly, the Rosenbrock requires a mix of exploration and exploitation because of the diminishing gradient near the global minima, causing a purely gradient-based algorithm to take a lot of iterations to converge. Therefore, a combination of rapidly moving close to the minima by efficiently exploring and good exploitation to move towards the minima is required. Figure 5 includes the results obtained in 2 and 25 dimensions.

The characteristics of each algorithm arise when carefully analyzing the results shown in Figure 5. GA is mainly exploratory; as such, it outperforms HyGO and CMA-ES in the $2D$ Rastringuin function since it can allocate more individuals to explore the parametric space. It is also the worst-performing algorithm in the purely exploitative Sphere function, where enhanced exploitation significantly expedites convergence. Interestingly, in the Sphere function, HyGO outperforms CMA-ES, meaning that the statistic-based gradient approximation is not as efficient as the geometric-based one when there is a clear direction of descent. When combining the requirements for exploration and exploitation in the Rosenbrock function, CMA-ES and HyGO perform similarly, with CMA-ES outperforming HyGO in terms of mean metric. This performance difference is due to the mentioned employment of the gradient in the Rosenbrock function near the minima, where the gradient is almost null. However, utilizing the evolution path of the CMA-ES algorithm to create new generations gives it an edge compared with a genetic algorithm.

The optimisation trends are maintained in the high dimensional scenarios, depicted in Figure 5. However, the significantly larger parametric space lowers the initialisation impact since all initialisations are equally bad due to the sparse initial individual distribution. The number of initial individuals was not adjusted because covering the parametric space with the same density as in the $2D$ scenarios would require too many individuals, defeating the purpose of the analysis.

In a purely gradient-based function (Sphere $25D$), HyGO and CMA-ES perform very similarly, being HyGO faster at the start (strengthening the claim that geometric-based gradient approximation is more efficient than CMA-ES) and reaching almost at the same time a cost value of $J \approx 10^{-2}$, which is already very low. However, the ability to save and maintain the evolution path of CMA-ES enables an acceleration in the latest optimisation stages, where the gradient vanishes. Interestingly, GA performs well in this gradient-controlled function. Nevertheless, even among EAs (as depicted), better algorithms exist when such a strong gradient is present.

When optimizing the purely exploratory Rastringuin function, the CMA-ES algorithm fails to converge. The large number of minima slows down the evolution of this algorithm. Moreover, due to the large number of parameters, the

Table 3: Comparative performance of GA and HyGO on the analytical benchmark functions from Table 2 (some in both 2D and 25D cases). For each function, the table presents the percentage of successful convergence, average number of generations, evaluations, and best achieved cost over 50 independent runs with randomised initialisation. Bold values indicate the superior algorithm for each metric. The bottom row summarises the frequency each method outperforms the other in each category, providing a global assessment of robustness and efficiency.

Name	Convergence %		Generations		Evaluations		Best Cost	
	GA	HyGO	GA	HyGO	GA	HyGO	GA	HyGO
Ackley-25D	0	0	50	50	4946.1	5000	7.401	**5.2588**
Ackley-2D	**100**	98	14.48	**5.72**	1290.3	**548.72**	**0.0020133**	0.053572
Beale	2	**86**	49.22	**14.08**	2689.9	**1372.7**	**0.029875**	0.10669
Booth	6	**100**	48.58	**2.82**	2542.4	**264.8**	0.12518	**2.0816e-07**
Bukin N.6	0	0	50	50	3706.8	4948.2	0.1236	**0.071234**
Easom	14	**96**	45.8	**11.12**	2640.1	**1077.9**	-0.65965	**-0.95999**
Eggholder	4	**18**	48.54	**41.88**	**2555.6**	4096.4	**-915.62**	-896.19
Goldstein-Price	100	100	17.8	**4.92**	1555.5	**468.78**	3	3
Himmelblaus	54	**100**	36.28	**3.62**	2117.6	**340.2**	0.0044025	**7.2368e-07**
Holder Table	54	**100**	34.02	**4.14**	2092.5	**390**	-19.207	**-19.209**
Levi N.13	12	**22**	45.76	**43.94**	**2512.7**	4243.2	**0.011477**	0.014706
Matyas	56	**100**	37.64	**3.42**	2286.6	**321.24**	0.00092671	**9.1272e-09**
Sphere-25D	0	0	50	50	4950.9	4909.6	0.13296	**0.021892**
Sphere-2D	100	100	13.62	**1.76**	1250.2	**151.02**	0	0
Rastrigin-25D	0	0	50	50	4735.5	4988.1	63.989	**31.346**
Rastrigin-2D	**98**	80	**16.36**	21.04	**1341.6**	2029.3	**0.024735**	0.26439
Rosenbrock-25D	0	0	50	50	4950.8	4917.7	10182	**193.29**
Rosenbrock-2D	2	**76**	49.52	**29.14**	**2710.7**	2863.4	0.20727	**0.12878**
Styblinski-Tang-25D	0	0	50	50	4950.7	5000	-932.05	**-936.58**
Styblinski-Tang-2D	72	**100**	30.86	**3.4**	1884.8	**317.94**	**-78.331**	-78.332
Total Wins	2	**10**	1	**13**	4	**10**	6	**12**

different minima are more challenging to exploit because finding the gradient direction is more complex. However, in this high-dimensional space, the local minima are *greater*, and finding the basin of the global minima is no longer enough for convergence. Thus, the ability to utilise the gradient information once the basin of the global minima is found gains importance, leading to HyGO excelling in the 25-dimension Rastringuin function. The challenge of approximating a high-dimensional gradient is also appreciated in the 25D Rosenbrock function, where the CMA-ES algorithm performs worse than GA in intermediate evaluation numbers. However, path exploitation enables the CMA-ES to achieve performance metrics similar to HyGO's in later stages where the geometry is very shallow. Combining a good exploration and a geometrically computed gradient enables HyGO to reach low-cost function values.

The algorithm's explorative and exploitative characteristics make it a strong contender in any optimisation scenario, especially in high dimensions, where this balance is crucial. In lower dimensions, where the differences between exploration and exploitation are more significant, HyGO still excels in performance, presenting similar results as GA in the Rastringuin function (explorative) and beating the CMA-ES in the Sphere function (exploitative).

4 Linear Genetic Programming: The Damped Landau Oscillator

The Landau oscillator is the analytical benchmark employed to validate the enriched Linear Genetic Programming optimiser. Its simplicity makes it an ideal candidate for testing the ability of LGP to yield successful control strategies while comparing the effect the introduction of the Subplex has on optimisation. Furthermore, this function represents the oscillatory motion of the von Kármán vortex shedding behind a cylinder [35], making it an ideal dummy problem. The pair of differential equations that govern the Landau oscillator are:

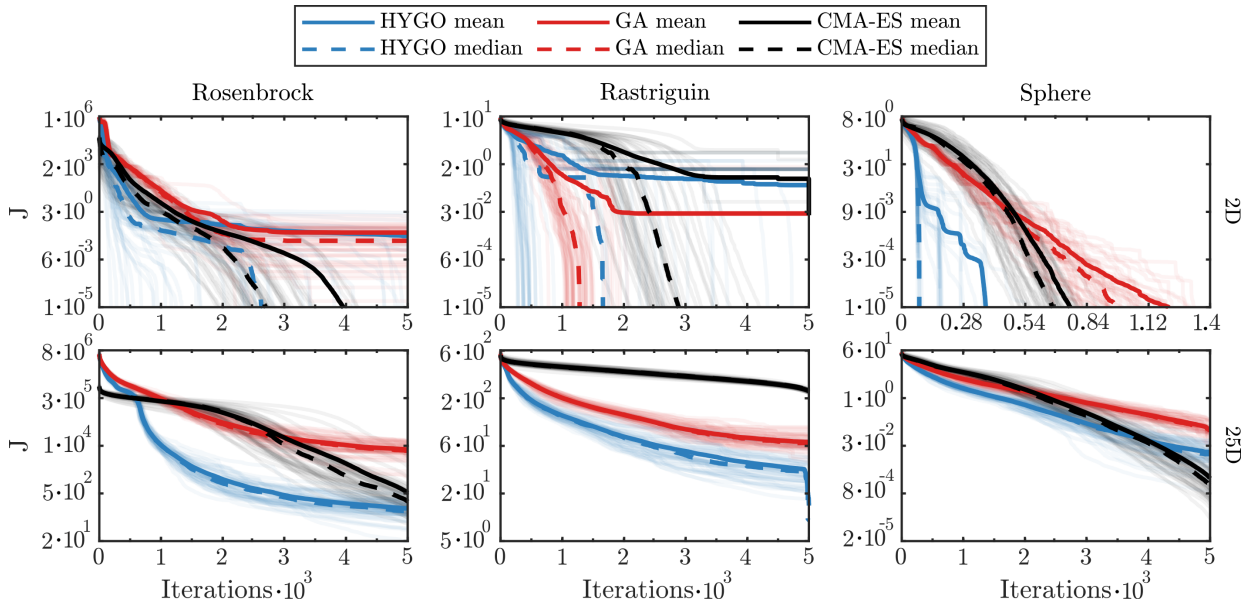


Figure 5: Convergence comparison of HyGO, GA, and CMA-ES algorithms on the Rosenbrock, Rastrigin, and Sphere benchmark functions in 2D (top row) and 25D (bottom row). Each panel shows the evolution of the cost function J as a function of the number of iterations across 50 independent runs with randomised initial conditions. Solid lines denote the mean run, dashed lines indicate the median, and lighter trajectories display individual runs. This visualisation highlights both the average performance and variability of each algorithm under different dimensionalities, illustrating convergence rates and robustness in complex landscapes

$$\begin{cases} \dot{a}_1 = \left(1 - a_1^2 - a_2^2\right) a_1 - a_2, \\ \dot{a}_2 = \left(1 - a_1^2 - a_2^2\right) a_2 + a_1 + b(a_1, a_2), \end{cases} \tag{12}$$

The Landau oscillator with no control ($b = 0$) presents a stable orbit in the circle of radius one where the solution endlessly rotates counter-clockwise. Figure 6 includes the results for six initial conditions, all rapidly entering the stable periodic orbit. The objective is to dampen the oscillator, stabilizing it at the coordinates $(a_1, a_2)_{t=t_f} = (0,0)$ as fast as possible with minimal control input.

The optimisation problem proposed is multi-objective. Since HyGO is a single-objective algorithm, the situation requires weighting the different objectives (also known as scalarisation). The stabilisation term J_a and the penalisation for the actuation J_b yield a combined cost function J as presented in Equation 13.

$$J = J_a + \gamma J_b \tag{13}$$

Equation 14 presents the expression of the individual cost terms. The stabilisation term includes the integral over the total integration time to favor actions that stabilise the oscillator earlier—due to the oscillatory behavior, squaring the coordinates allowed for avoiding convergence towards negative coordinates. The penalisation terms present the same structure to favour individuals with minimal control input. The angular frequency of the uncontrolled oscillator is 1 [13], thus T=2π. The total integration time taken is $20T$ to enable rapid evaluation.

$$J_a = \frac{1}{t_{end}} \int_0^{t_{end}} a_1^2 + a_2^2 \, dt \tag{14}$$

$$J_b = \frac{1}{t_{end}} \int_0^{t_{end}} b(a_1, a_2)^2 \, dt \tag{15}$$

The control only affects the second differential equation, leading to each individual only including a single control law, simplifying the optimisation. Both coordinates are the sensors of the control law to enable maximum adaptive

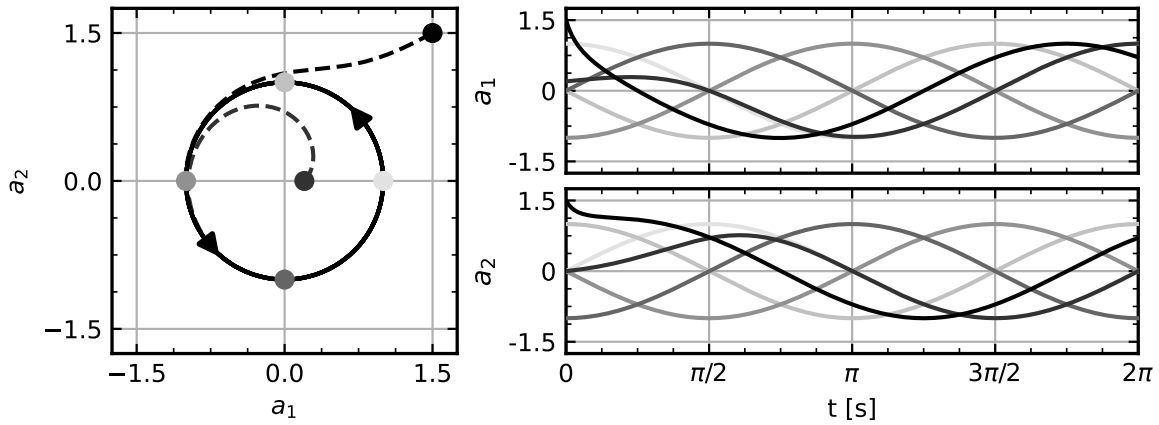


Figure 6: Phase space and time evolution of the undamped Landau oscillator for six representative initial conditions $(a_1,a_2)_{t=0} = [(1,0),(-1,0),(0.2,0),(0,1),(0,-1),(1.5,1.5)]$. Left: Orbits in the (a_1,a_2) phase plane illustrating stable periodic motion on the unit circle. Right: Time histories of a_1 (top) a_2 (bottom) for each initial state, showing sustained oscillatory dynamics. Each curve corresponds to one initial condition, highlighting solution periodicity and dependence on initialisation.

capabilities. In more complex problems, feeding the coordinates instantly may be insufficient. A possible approach to increase the information available is to set the coordinates' values at specific points in the past as artificial sensors. For example, in oscillatory problems with a precise period T, defining sensors $s_i = [a_i(\tau), a_i(\tau - T/4), a_i(\tau - T/2), a_i(\tau - 3T/4), a_i(\tau - T)]$ enables to encapsulate the information from the past cycle for the current actuation. However, the Landau oscillator is simple enough to not require such a complexity introduction since a larger number of sensors highly increases the *parametric space*, lowering convergence chances.

Performing two optimisations with and without the Subplex exploitation permits analyzing the algorithm capabilities and the effect the Subplex enrichment has on LGP. Furthermore, to improve the generalisation capabilities of the control laws, the final J of each individual is the average of its behavior across four initial conditions $(a_1,a_2)_{t=0} = [(1,0),(-1,0),(0,1),(0,-1)]$. The optimisations include a total of 10 generations of 100 individuals. In the hybrid optimisation, the exploration phase contained 80 individuals, and the exploitation included 20 individuals with the Subplex interpolation scheme. Initialisation consisted of randomly generated individuals with several instructions ranging between 5 and 35 after intron elimination. Thereafter, the minimum number of instructions was 2. An increased number of minimum instructions enabled initializing the individuals with complex control laws, allowing the algorithm to simplify them. After some trials, this proved the best approach since a low number of instructions leads to constant control laws (a single number, for example, $b = 0.54$). However, initializing the optimisation with more minimum instructions forces complex operations (including trigonometric functions, sensors…). The initial register included a 0 by default in the control output to avoid introducing a bias, two sensors representing the position at the given time instant, a randomly initialised memory register, and two constant registers, which also included random numbers in the range $[0,1]$.

There is no recommendation on the Subplex size to perform the exploitation since the problem effectively has infinite dimensions. After some testing, a good compromise found for the problem was 10 individuals, which enabled containing the fittest 10% of the population with enough variability of control laws. Furthermore, a subplex that is too big may lead to a large number of weighted terms in the output law, leading to more complex matrix reconstruction and lowering the interpolation effectivity. Nevertheless, this parameter highly depends on user experience and the targeted optimisation problem.

Figure 7 includes the overall cost function and each cost term's evolution. The optimisation is controlled by the stabilisation term J_a (due to the low value set to the weighting parameter γ) since the progress of both J and J_a are

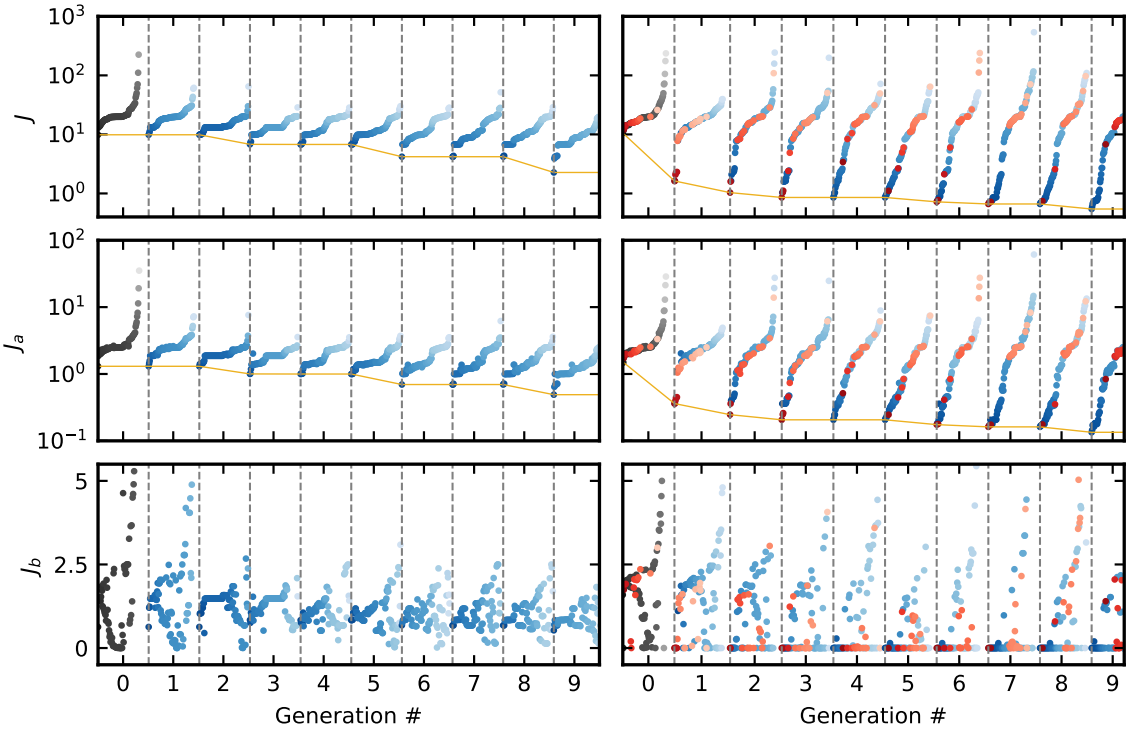


Figure 7: Evolution of the total cost J (top row), stabilisation cost J_a (middle), and the penalisation J_b (bottom) through the generations for a GP optimisation of the damped Landau oscillator, comparing cases without (left) and with (right) Subplex local search enrichment. Each dot represents an individual, coloured by origin: black for random initialisation, blue for genetic operations, and red for Subplex-generated individuals; darker dots indicate lower cost. The yellow line traces the best cost found per generation.

very similar. The enriched optimisation significantly outperforms the GA case. This difference is mainly caused by a group of Subplex-generated individuals in the second population, lowering the best cost by almost an order of magnitude. Once the Subplex introduced some expressions that yielded much lower values, the rest of the optimisation consisted of modifying these functions incrementally to improve the performance. However, in the GA optimisation, only incremental enhancements were produced, concluding that no mutated individuals significantly changed the first successful control laws. Therefore, most of the optimisation focused on refining these successful structures, similar to the hybrid optimisation after the Subplex found novel structures.

After careful analysis of the results, the improvement caused by this suplex-created group is related to the appearance of a large scaling number that multiplies the cost function. Although the value variable and constant initial registers were limited between $[0,1]$, summations and divisions allow combining scalar numbers beyond 1. This multiplication gave these individuals ample control authority, imposing significant changes in the first instants of the integration. Figure 8 includes the phase space for both optimal control laws, with their cost function expression, and the actuation for each initial condition. This figure illustrates the differences in control approaches, where the purely genetic optimisation focused on trigonometric functions, which cause many oscillations before finally reaching the optimum. However, the control authority of the hybrid solution enables it to move towards $(0,0)$ directly.

The results presented for the GA and LGP optimisations indicate the performance gains achieved by hybridizing a genetic optimiser with the Downhill Simplex algorithm. The exploitation significantly expedited the optimisation in both scenarios, yielding lower values in fewer iterations. This claim is reinforced by the stochastic approach used in GA optimisation and the generalisation capabilities imposed by different initial conditions in LGP.

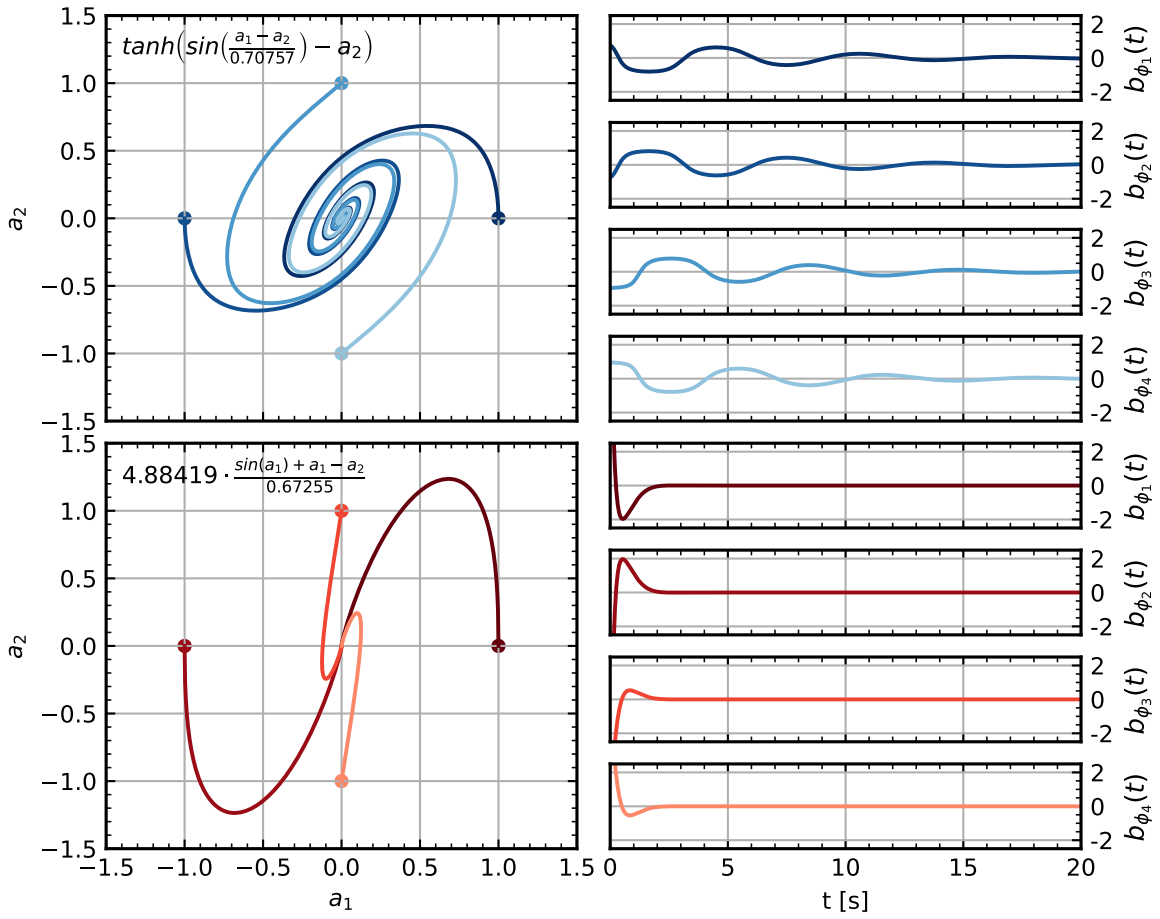


Figure 8: Comparison of optimal control laws discovered for the damped Landau oscillator via genetic programming, without (top) and with (bottom) Subplex enrichment. Left: Phase space trajectories show the system state (a_1, a_2) evolving from multiple initial conditions under the learned feedback. The control law expression for the best individual is depicted in each panel. Right: Corresponding control actuation $b_{\phi_i}(t)$ applied over time for each initial state, ϕ_i. Subplex-enriched optimisation yields faster and more direct stabilisation to the origin, demonstrated by tighter and less oscillatory trajectories and sparser control input.

5 Control of shedding flows: Drag reduction on Ahmed body

The previous sections demonstrated the strengths of the proposed hybrid optimisation methodology in controlled benchmark scenarios, allowing for a detailed analysis of HyGO's exploration and exploitation dynamics. However, the analytical test functions used, while valuable for validation, do not reflect the complexity of real-world engineering applications and can often be addressed effectively with simpler optimisation techniques. To showcase the practical capabilities of HyGO, we now turn to a challenging computational fluid dynamics problem: the minimisation of aerodynamic drag on an Ahmed body. Moreover, by employing the same simulation setup as in Li et al. [36], a direct performance comparison with the Explorative Gradient Method (EGM) is facilitated, highlighting HyGO's competitiveness against state-of-the-art hybrid optimisers.

Table 4: Parameters for GA and HyGO

Hyper-Parameter	**GA**	**HyGO**	**HyGO-Stepped**
Population Size	100	11	50
Max Generations	10	100	20
Number of bits	8	8	8
Tournament Size	2	2	2
Tournament Probability	100%	100%	100%
Elitism Individuals	1	1	1
Crossover Points	1	1	1
Crossover Mix	True	True	True
Mutation Rate	0.2	0.075	0.075
Replication Probability	0	0	0
Crossover Probability	80%	55%	55%
Mutation Probability	20%	45%	45%
Simplex Offspring	N/A	11	11

Table 5: Simplex initialisation for the second half of HyGO-Stepped.

Index	U_1	U_2	U_3	U_4	U_5	θ_1	θ_2	θ_3	θ_4	θ_5
1	10.25	6.0	1.25	8.25	7.5	27.5	0.0	0.0	0.0	0.0
2	10.25	0.0	0.0	0.0	0.0	58.75	0.0	0.0	0.0	0.0
3	0.0	6.0	0.0	0.0	0.0	27.5	45.0	0.0	0.0	0.0
4	0.0	0.0	1.25	0.0	0.0	27.5	0.0	45.0	0.0	0.0
5	0.0	0.0	0.0	8.25	0.0	27.5	0.0	0.0	45.0	0.0
6	0.0	0.0	0.0	0.0	7.5	27.5	0.0	0.0	0.0	45.0
7	10.25	0.0	0.0	0.0	0.0	-3.75	0.0	0.0	0.0	0.0
8	0.0	6.0	0.0	0.0	0.0	27.5	-45.0	0.0	0.0	0.0
9	0.0	0.0	1.25	0.0	0.0	27.5	0.0	-45.0	0.0	0.0
10	0.0	0.0	0.0	8.25	0.0	27.5	0.0	0.0	-45.0	0.0
11	0.0	0.0	0.0	0.0	7.5	27.5	0.0	0.0	0.0	-45.0

5.1 Ahmed body configuration

The reference geometry employed in this study is a 1 : 3-scale Ahmed body, experimentally characterized by a slanted rear surface with an angle of $\alpha = 35^\circ$. The model dimensions are $L = 348$ mm in length, $W = 130$ mm in width, and $H = 96$ mm in height. The front edges are rounded with a radius of $0.344H$, and the body is supported by four cylindrical mounts of 10 mm diameter. A ground clearance of $0.177H$ is maintained. The Cartesian coordinate system (x, y, z) is defined such that the origin lies at the midpoint of the lower edge on the vertical rear base, aligned with the symmetry plane (see Figure 9). Here, x, y, and z represent the streamwise, spanwise, and wall-normal directions, respectively, with the corresponding velocity components denoted by u, v, and w. The incoming free-stream velocity is set to $U_\infty = 30\ \mathrm{m\,s^{-1}}$.

To enable flow control, five groups of steady slot actuators are installed along the perimeter of the rear window and the vertical base, as illustrated in Figure 9. Each slot has a uniform width of 2 mm. The top, middle, and bottom horizontal actuators span 109 mm, while the sidewise actuators on the upper and lower sections of the rear surface measure 71 mm and 48 mm, respectively. The actuation velocities $U_1, \ldots, U_5$ serve as independent control variables: U_1 corresponds to the upper edge of the rear window, U_3 to the middle edge, and U_5 to the bottom edge of the vertical base; U_2 and U_4 refer to the left and right side actuators.

In line with the experimental configuration described by Zhang et al. [46], the actuation angles are also adjustable, as shown in Figure 9. The objective of the optimisation is to minimise the aerodynamic drag, quantified by the drag coefficient $J = C_D$, to focus on the best possible actuation, disregarding input power or lift penalisation as in other studies. In the initial five-dimensional optimisation setup, the actuation velocities U_i, for $i = 1, \ldots, 5$, are bounded to not exceed twice the optimal single-actuator velocity (charactericed in Li et al. [36]), while the actuation angles θ_i are fixed at 0° to enforce purely streamwise blowing. The ten-dimensional optimisation later extends the parameter space by allowing angular variation, with $\theta_1 \in [-35^\circ, 90^\circ]$ and $\theta_{2,\ldots,5} \in [-90^\circ, 90^\circ]$.

To evaluate the performance of HyGO in this complex optimisation scenario, we compare it against three alternative strategies. First, results are benchmarked against the Explorative Gradient Method (EGM), as detailed in Li et al. [36], using the same simulation setup. Second, a traditional genetic algorithm without exploitation enrichment is included to isolate the hybridisation benefits. Finally, two configurations of DSM-enriched HyGO are considered: a direct ten-dimensional optimisation involving all control parameters from the outset, and a *stepped procedure*, where

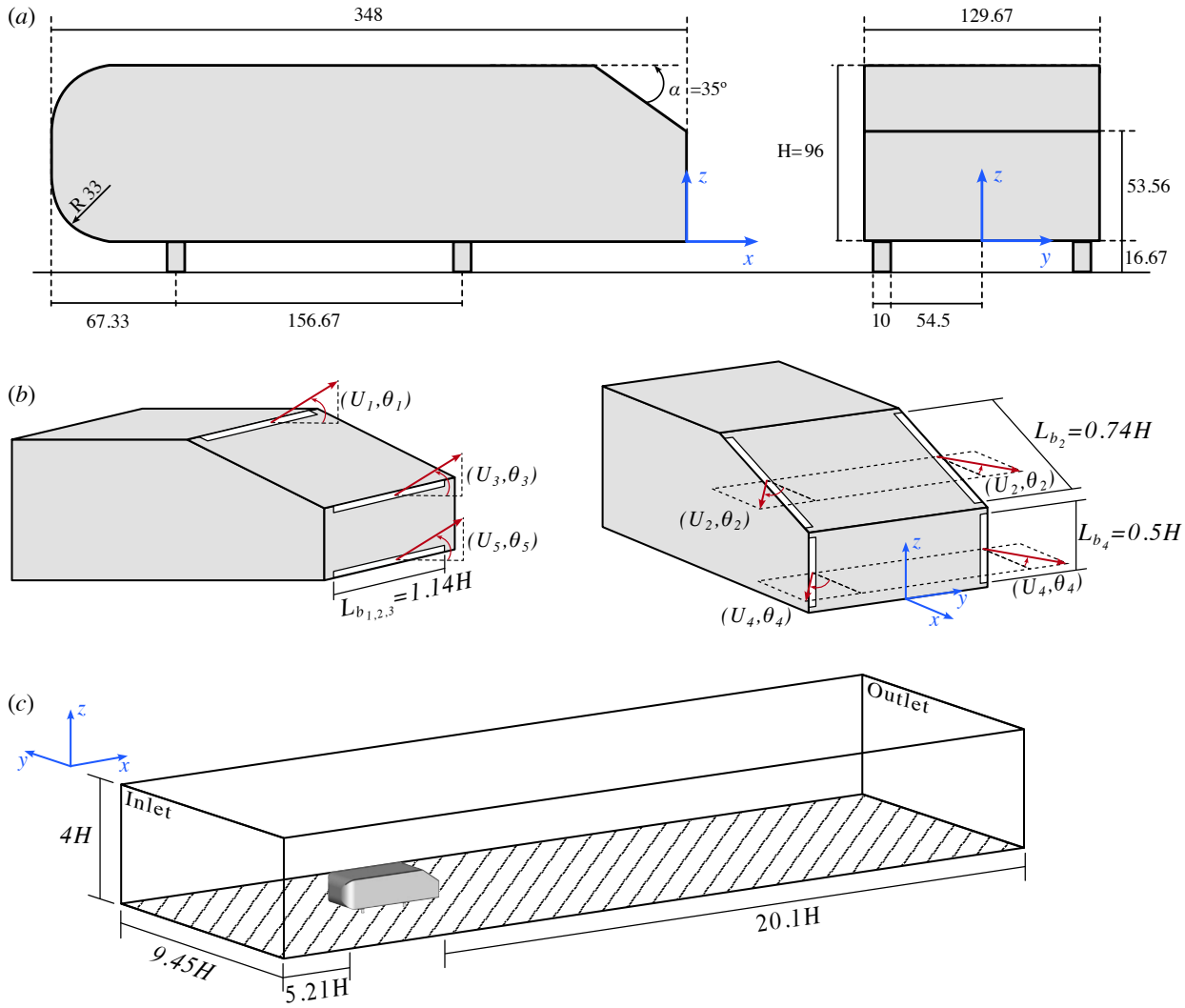


Figure 9: Ahmed body geometry, actuator layout, and computational domain for drag-reduction flow control studies. (a) Side and rear views of the Ahmed body model ($H = 96$), with relevant dimensions and coordinate axes (b) Deployment and blowing direction (arrows, defined by actuation speed U_i and angle θ_i) of the five groups of slot actuators on the rear window and base. Angles θ_i are positive when directed outward (right) or upward (left). Actuator spanwise lengths L_{b_i} are labelled relative to H. (c) Computational wind tunnel domain for RANS and LES simulations, showing Ahmed body placement. Figure adapted with permission from Li et al. [36].

only the actuation velocities U_i are optimised initially. Once convergence is achieved, the actuation angles θ_i are introduced to continue the optimisation in the full parameter space. For the ten-dimensional run (non-stepped HyGO), the initial population includes 11 individuals constructed using a simplex centered on the optimal solution obtained from the stepped procedure (see Table 5), while the other two optimisations (GA and HyGO-stepped) are initialised randomly. Table 4 includes the parameters for the genetic optimisation strategies.

5.2 RANS simulations

The numerical setup employed in this study is identical to that used in Li et al. [36]. The computational domain, constructed using Ansys ICEM CFD, consists of a rectangular wind tunnel (Figure 9 (c)) with dimensions $X_1 \leq x \leq X_2$, $0 \leq z \leq H_T$, and $|y| \leq W_T/2$, where $X_1 = -5.21H$, $X_2 = 20.17H$, $H_T = 4H$, and $W_T = 9.45H$, which exceed the minimum recommendations of Serre et al. [47] to ensure minimal boundary effects. An unstructured hexahedral mesh with approximately five million cells is adopted, offering a balance between accuracy and efficiency. The mesh resolution near walls and actuation slots ensures adequate capture of boundary layers and shear-layer dynamics, with wall distances of $\Delta x^+ = 20$, $\Delta y^+ = 3$, and $\Delta z^+ = 30$.

RANS simulations are performed using the realizable k–ε turbulence model in Fluent, with second-order spatial discretisation and semi-implicit pressure–velocity coupling. For the 35° Ahmed body, RANS provides reliable flow predictions, as validated in Li et al. [36], and is therefore considered suitable for actuator optimisation in this context.

To improve computational efficiency, we also adopt the acceleration strategies from Li et al. [36], including a 1-nearest-neighbor initialisation of velocity fields and quantisation of actuation parameters. These enhancements significantly reduce simulation time while maintaining solution accuracy.

5.3 Results and discussion

Figure 10 presents the optimisation results for the algorithms previously described. Each optimisation was allowed to run for the maximum number of generations specified in Table 4, unless early termination was triggered by a lack of improvement over four consecutive generations.

The standard GA was unable to further improve upon the best individual found in the initial, randomly generated population. As a result, the optimisation process stagnated and terminated early. Despite this limitation, the best solution obtained through GA achieved a 10.9% reduction in the drag coefficient relative to the baseline value $C_D^0 = 0.3134$, which is already a non-trivial improvement. Nevertheless, this falls short of the $\sim 20\%$ drag reductions reported in other studies [48]. One plausible explanation is that the relatively large population size enabled a successful solution to emerge early through random sampling, yet the complexity of the underlying flow dynamics limited the ability of simple genetic operations (mutation and crossover) to refine it further.

In contrast, the DSM-enriched optimisation (HyGO) exhibited a more conventional convergence profile. While its initial performance was similar to GA, the inclusion of DSM led to a notable improvement early in the process. From there, incremental enhancements in each generation resulted in a final drag reduction of 15.6%, comparable to

Table 6: Optimal control parameters of the different optimisation strategies, including the cost value and drag reduction.

Case	Drag (reduction)	Actuation parameters				
		Top	Upper	Middle	Lower	Bottom
Unforced	0.313 (0%)	—	—	—	—	—
GA	0.279 (10.88%)	31.5m/s	28m/s	19m/s	49m/s	16m/s
		−28.8°	−50.4°	−3.6°	−48.6°	57.6°
HyGO	0.265 (15.6%)	14.5m/s	17m/s	4m/s	19.5m/s	4.5m/s
		−27.5°	−43.2°	18°	−39.6°	25.2°
Li et al. [36]	0.258 (17.49%)	21.5m/s	23.81m/s	1.8m/s	25.2m/s	22.5m/s
		−27°	−42°	67°	−44°	22°
HyGO (stepped)	**0.249 (20.56%)**	34.5m/s	36m/s	8.5m/s	32.5m/s	21m/s
		−30°	−37.8°	−72°	−50.4°	48.6°

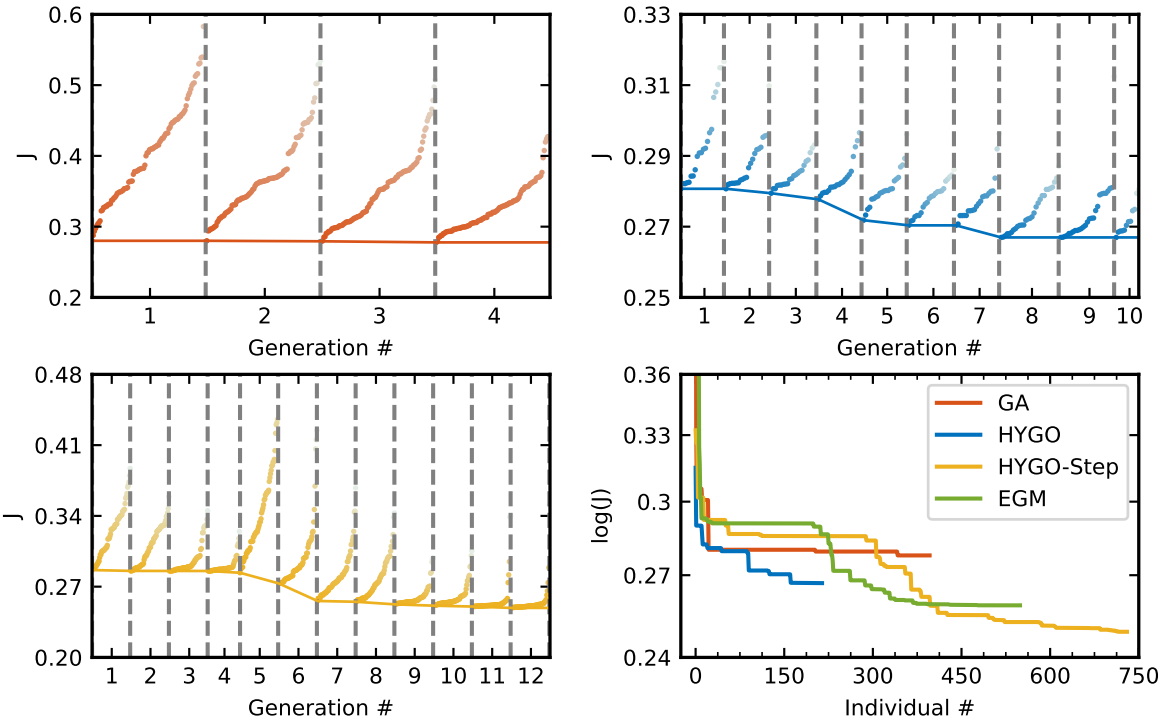


Figure 10: Evolution of the cost function J during the optimisation of drag reduction on the Ahmed body for four strategies. Each panel shows the cost across generations for a different method: GA (top left), HyGO (top right), and HyGO-Step (bottom left), while the bottom right plot compares the best cost evolution as a function of individual number for all approaches, including the EGM benchmark (green) from Li et al. [36]. Dots correspond to individual evaluations per generation; solid lines trace the minimum cost in each generation. This figure highlights the comparative convergence rates and final performance of the tested algorithms.

top-performing strategies reported in the literature. The exploitation phase played a key role throughout: in many generations, DSM-generated individuals outperformed those created via pure GA operations. However, this exploitation mechanism showed diminishing returns in later stages, as the algorithm increasingly converged around local optima.

The advantages of the hybrid approach become particularly evident when analysing the first five generations. In the initial generation, DSM effectively exploits a promising local minimum, resulting in a significant cost reduction. Although the following generations yield only marginal improvements, DSM continues to refine the identified solution. Notably, in the fifth generation, GA-generated individuals discover a new, superior solution located in a different region of the design space, as shown in Figure 11. DSM then exploits this newly identified region, further reducing the overall drag coefficient. This interplay between exploration and exploitation highlights the strength of the hybrid framework: GA enables a broad search and facilitates occasional jumps between basins of attraction, while DSM efficiently refines solutions within those basins.

Inspired by the approach in Li et al. [36], the stepped optimisation begins with a simplified five-dimensional problem involving only the actuation velocities. Reducing the dimensionality at this stage facilitates convergence and enhances global exploration by limiting the number of local minima, thereby increasing the likelihood of identifying an optimal region. Once the algorithm converges in this simplified domain, the resulting optimum is used to initialise a second optimisation phase, in which the actuation angles are introduced as additional degrees of freedom. This two-stage approach proves substantially more effective than the direct 10-dimensional optimisation, achieving a drag reduction of 20.5%, outperforming the best result obtained by EGM.

The low-dimensional manifold shown in Figure 11 provides a visual basis for comparing the optimisation strategies of the different algorithms. As previously discussed, the traditional GA fails to optimise this complex problem

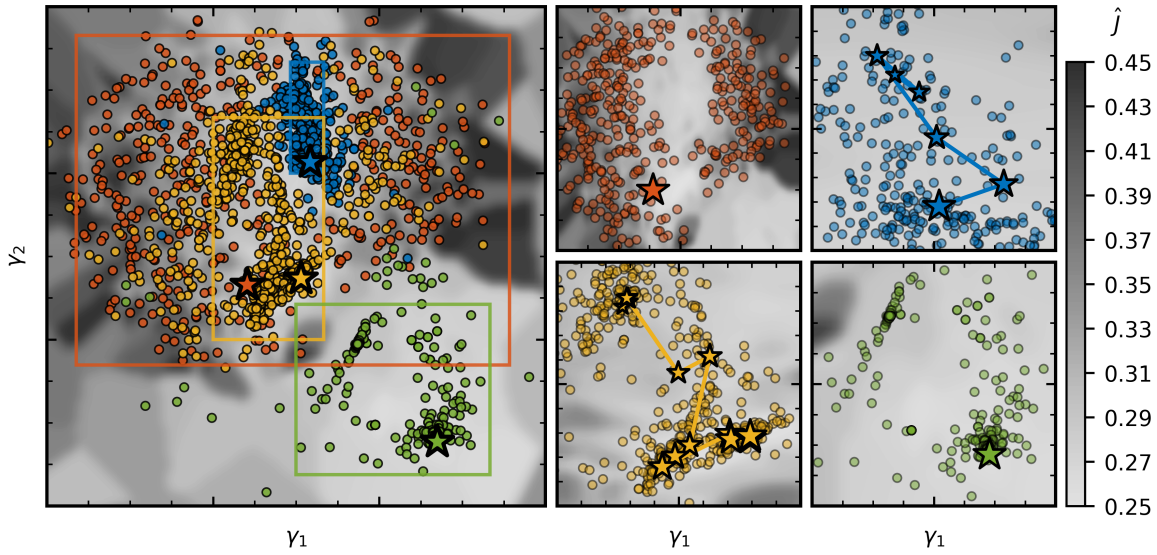


Figure 11: Proximity map of the optimisation parameters for all strategies, coloured by cost. The reduced coordinates are normalised. ● corresponds to GA, ● to HyGO, ● to HyGO-stepped, and ● to EGM. Squares on the general map indicate zoomed-in regions corresponding to each specific optimisation. The zoomed-in subplots illustrate the optimisation path: the best individual from each generation is marked with a star, with the star size increasing over generations (i.e., later generations are represented by larger stars).

due to its lack of exploitation capabilities, with its individuals scattered across the domain and no convergence toward the low-cost valley. In contrast, HyGO successfully identifies and exploits a local minimum but becomes trapped within it, reducing genome diversity as generations progress and causing individuals to cluster in a limited region of the search space. The stepped procedure, however, promotes broader exploration by initially reducing the problem's dimensionality. This behaviour is reflected in the wider spatial spread of the yellow points and the larger displacements of early-generation individuals, which even traverse multiple local minima. Building on this exploratory phase, the second stage of the algorithm effectively exploits the identified basin of attraction, ultimately achieving a substantially lower cost function value. Similarly, EGM converges around two distinct local minima, which it exploits efficiently. However, its exploration capability is more limited, as indicated by the relatively localised distribution of its individuals in the parameter space.

Table 6 lists the actuation parameters corresponding to the optimal solutions obtained by each algorithm. Notably, the more effective control strategies (HyGO-stepped and EGM) exhibit higher control than the HyGO optimum, as reflected by larger actuation velocity magnitudes, while maintaining similar actuation angles. Interestingly, the solution found by HyGO appears to be a refined version of the one identified by EGM: the actuation angles are similar, but the velocity magnitudes are lower. A key distinction, however, lies in the third actuator. While EGM directs this jet upwards, HyGO reverses its direction, possibly aiming to target the recirculation bubble more directly. This adjustment may reduce the bubble's size and contribute to the larger ΔC_D achieved. Such differences likely explain why the two optima occupy distinct basins in Figure 11.

To investigate the mechanisms responsible for the differences in drag performance across the optimal solutions, Figure 12 presents the pressure coefficient C_P distributions for the baseline and optimised cases. The plots include C_P values on the body's surface and in two horizontal planes located at $z/H = 0.179$ and $z/H = 0.502$. These planes correspond, respectively, to a horizontal slice through the midpoint of the vertical rear face and a region slightly above the junction between the slanted and vertical surfaces, effectively capturing key sections of the recirculation bubble.

All three optimised solutions reduce both the size and intensity of the recirculation bubble. Notably, the GA solution eliminates the bubble almost entirely, resulting in elevated pressure levels across the vertical rear surface.

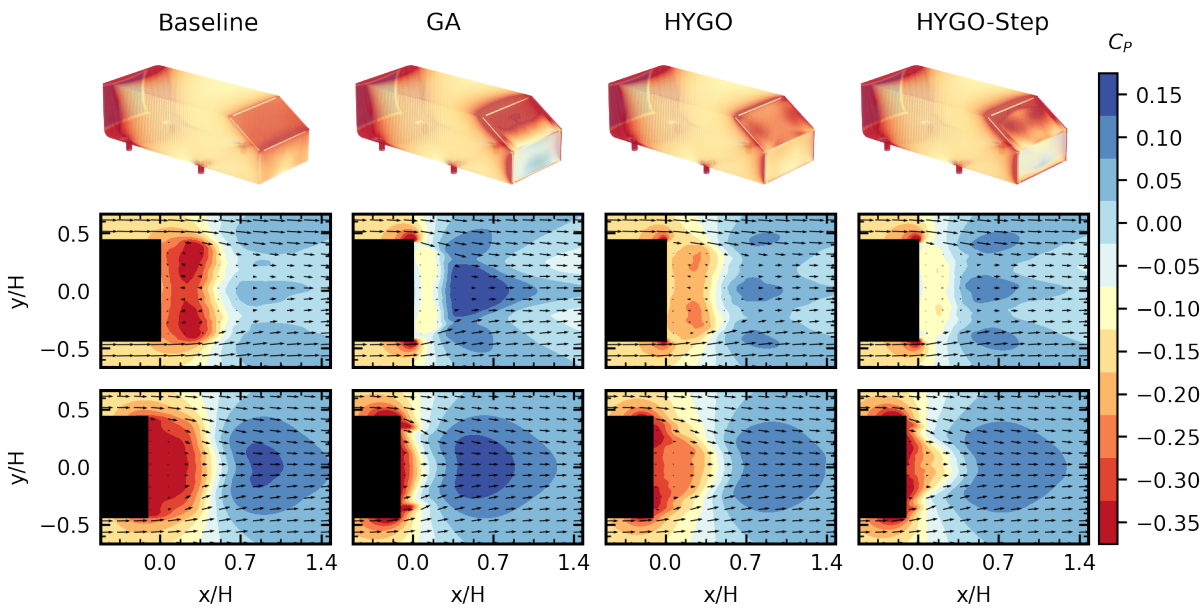


Figure 12: Top view for the spatial distribution of the pressure coefficient, C_p, on the Ahmed body. Columns, from left to right, represent the reference (baseline), the solution obtained with the genetic algorithm (GA), HyGO, and HyGO-stepped methodologies. The first row shows the full-surface C_p contours, while the second and third rows display $x-y$ planar sections at the non-dimensional heights $z/H = 0.179$ and 0.502, respectively. A common colour scale referenced to the free-stream static pressure is applied to every panel to enable quantitative comparison.

This reduces the pressure differential between the front and rear, thereby lowering drag. However, this benefit comes at the cost of generating strong under-pressure regions along the vertical edges of the rear surface and at the top of the slanted section. These effects are caused by intense vortex generation and boundary layer separation, as shown in Figure 13, ultimately leading to an overall increase in pressure drag.

The extreme bubble suppression of GA is associated with a large U_4 magnitude combined with a negative orientation angle θ_4 (directed inward). This is illustrated in Figure 13 by an almost ideal boat-tailing effect, where streamlines align with the slanted surface and recirculation is minimal, effectively mimicking a sharp triangular trailing edge. This severe separation occurs despite the compensatory efforts of actuators 1 and 2. Additionally, actuators 3 and 5 are oriented to reduce under-pressure on the rear surface, although their actuation magnitudes are considerably smaller. As a result, the total drag reduction achieved by GA remains moderate, despite the successful suppression of the recirculation bubble.

In contrast to the GA strategy, which eliminates the recirculation bubble, HyGO displaces the bubble away from the body (as seen in Figure 13) while reducing its intensity. The lower actuation velocities do not suppress the vortex outright but instead shift it downstream, allowing the first and second jets to focus more effectively on delaying boundary layer separation. As a result, pressure levels increase on both the slanted and vertical rear surfaces. This actuation pattern also generates significantly lower vertical vorticity near the body's edges, contributing to a smaller and less energetic recirculation zone. Interestingly, the pressure contours and velocity fields suggest that counter-rotating vortices drive separation at the lateral edges of the slanted surface, rather than by a dominant spanwise vortex at the top edge. HyGO's configuration localises the separation to a smaller region, preventing it from spreading across the full width of the body and thus reducing its contribution to drag.

The stepped optimisation combines the strengths of both the GA and full-dimensional HyGO strategies: it achieves partial elimination of the recirculation bubble, similar to GA, while also incorporating the separation control mechanisms observed in HyGO. The optimal solution exhibits minimal recirculation and under-pressure at the rear (though slightly higher than in GA), and confines separation primarily to the vicinity of the slanted surface. This configuration retains the beneficial low-edge vorticity, which helps prevent widespread separation and enables the highest overall drag reduction among all cases. The actuation parameters for the stepped solution also lie between those of GA

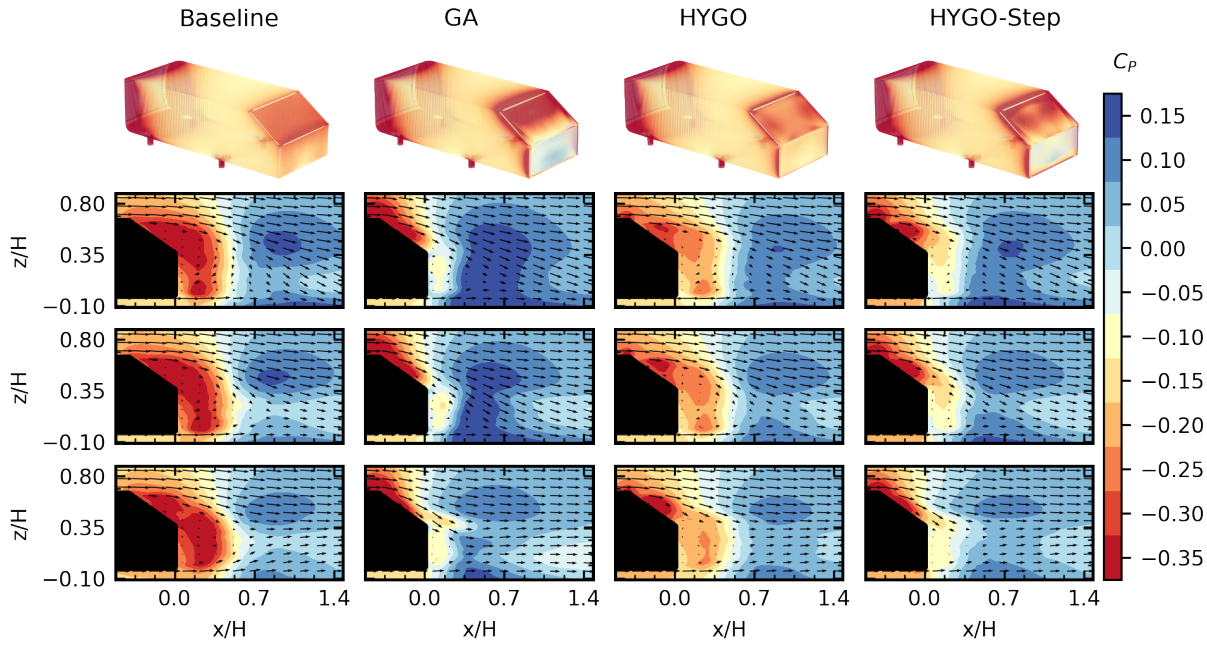


Figure 13: Side view for the spatial distribution of the pressure coefficient, C_p, on the Ahmed body. Columns, from left to right, represent the reference (baseline), the solution obtained with the genetic algorithm (GA), HyGO, and HyGO-stepped methodologies. The first row shows the full-surface C_p contours, while the second, third, and fourth rows present $x-z$ planes at the spanwise stations $y/H = 0$, 0.144, and 0.288, respectively. A common colour scale referenced to the free-stream static pressure is applied to every panel to enable quantitative comparison.

and HyGO. In particular, U_4 is larger than in HyGO to target the bubble more effectively, but smaller than in GA to avoid overly dominating the flow. This balance allows U_1 and U_2 to play a more active role in controlling separation, resulting in a more efficient and cooperative actuation strategy.

Figure 14 shows four cross-sectional planes at increasing streamwise distances from the rear of the body, visualising the evolution of the counter-rotating vortices generated by each actuation strategy. In all cases, the control mechanisms result in a noticeable modification in the size and strength of the primary vortex structures compared to the baseline.

The GA case, despite eliminating the recirculation bubble, produces more intense and long-lived vortices due to the aggressive separation induced by strong actuation. In contrast, the HyGO solution, which relies on more moderate jet velocities and refined control, generates weaker and more transient vortical structures, reducing their footprint and magnitude compared to the baseline. This leads to a smoother wake profile with less lateral momentum exchange, aligning with the observed lower drag values. The HyGO-stepped configuration further refines this behaviour, balancing effective bubble suppression with minimal vortex amplification, and achieving the most favourable compromise between wake structure control and drag reduction.

In addition to modifying the magnitude of the main vortices, the actuations introduce secondary vortex structures that rotate in the opposite direction, while also separating the main vertical structures. These smaller vortices, clearly visible between the principal vortices in both HyGO solutions, act to disrupt and separate the larger coherent structures. This disruption likely reduces the coherence and persistence of the primary vortices, contributing to drag reduction by promoting earlier wake recovery and pressure reattachment.

6 Conclusions

In this study, we introduced the Hybrid Genetic Optimisation (HyGO) framework, a novel hybrid algorithm that combines the global search capabilities of evolutionary algorithms with local search refinement strategies. This hy-

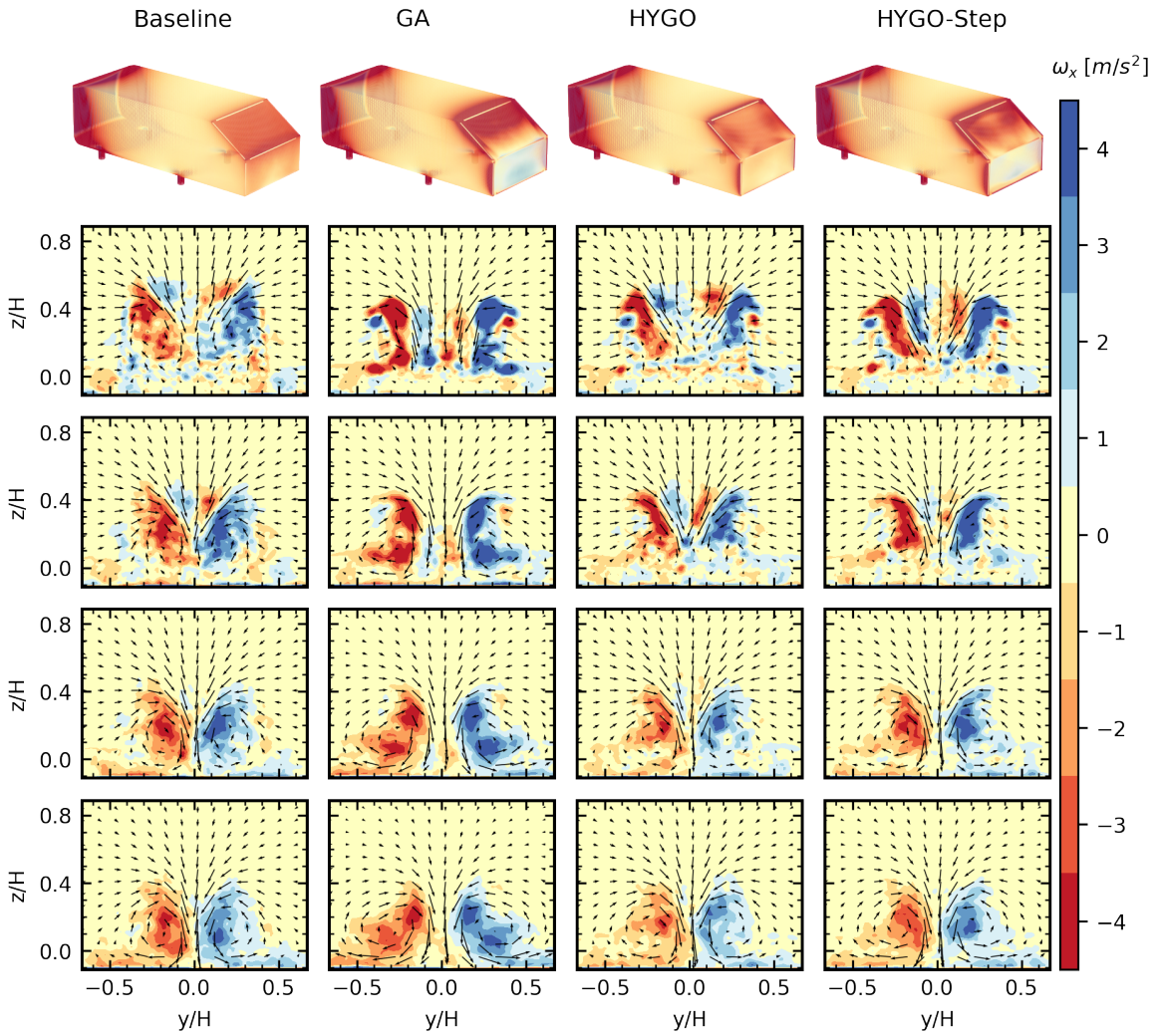


Figure 14: Streamwise vorticity maps for the fittest individuals of each optimisation approach compared to the baseline (no actuation) case at fixed streamwise distances, x/H, from the Ahmed body. The first row shows the pressure coefficient distribution on the surface of the Ahmed body. The second, third, fourth, and fifth rows display the streamwise vorticity maps at $x/H = 0.319$, 0.634, 0.959, and 1.279, respectively. A consistent colour scale is applied across all vorticity maps for clear comparison of the flow structures.

bridisation allows for an efficient balance between exploration and exploitation, making it suitable for complex optimisation problems characterised by high-dimensionality, non-linearity, and multimodal landscapes.

The performance of HyGO was first evaluated using a series of standard analytical benchmark functions, where it demonstrated superior convergence speed and robustness compared to traditional genetic optimisers. Specifically, HyGO outperformed classical genetic algorithms in terms of both convergence metrics and the number of evaluations required to reach optimal solutions. In particular, the integration of DSM significantly accelerated convergence, especially in high-dimensional and complex search spaces. However, in simpler, low-dimensional landscapes, DSM may occasionally slow down optimization. To address this, incorporating dynamically balanced exploration and exploitation schemes, such as Explorative Landscape Analysis, could enhance overall performance by adapting HyGO behavior to the specific characteristics of the optimization problem.

HyGO was also successfully applied to the control of the damped Landau oscillator, a canonical test case for vortex shedding dynamics. The results showed that the hybrid optimizer effectively identified control laws that stabilized the system, outperforming conventional genetic approaches. To further improve HyGO 's optimization capabilities, the data generated by LGP could be used to build a surrogate model, which would enhance both interpretability and convergence in the effectively infinite functional search space.

Finally, the method was tested on a real-world application: the aerodynamic drag reduction of an Ahmed body, where it achieved a notable reduction in drag coefficients compared to previous studies. The optimised actuation parameters provided physically interpretable solutions, which could be directly applied in experimental or design settings.

These results confirm that HyGO is a powerful optimization tool capable of handling complex, high-dimensional design spaces with robustness and efficiency. Its flexibility and effectiveness make it a valuable tool for solving a wide range of engineering and scientific challenges. Furthermore, replacing DSM with other local search algorithms, such as Particle Swarm Optimization (PSO) or Broyden-Fletcher-Goldfarb-Shanno (BFGS), could further enhance its adaptability, tailoring the optimization process to the specific needs of each problem.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Code and Data availability

The in-house HyGO (Hybrid Genetic Optimiser) toolbox, which is used for this study, is available on GitHub github.com/ipatazas/HYGO. Data will be made available upon request.

Supplementary materials

Supplementary material associated with this article can be found in the online version.

Acknowledgments

This activity is part of the project ACCREDITATION (Grant No TED2021-131453B-I00), funded by MCIN/AEI/10.13039/501100011033 and by the "European Union NextGenerationEU/PRTR". R.C. acknowledge funding by the Madrid Government (Comunidad de Madrid) under the Multiannual Agreement with Universidad Carlos III de Madrid in the line of 'Fostering Young Doctors Research' (SOLMETAI-CM-UC3M), and in the context of the V PRICIT (Regional Programme of Research and Technological Innovation). The authors thank Prof. Discetti, C. Sanmiguel-Vila and V. Duro for their useful comments and suggestions.

Declaration of generative AI and AI-assisted technologies in the writing process

During the preparation of this work, the authors used ChatGPT (OpenAI) and Grammarly to check grammar, enhance readability, and improve text clarity. After using these tools, the authors reviewed and edited the content as needed and take full responsibility for the content of the publication.

References

[1] J. S. Arora, O. A. Elwakeil, A. I. Chahande, C. C. Hsieh, Global optimization methods for engineering applications: A review, volume 9, 1995. doi:10.1007/BF01743964.

[2] J. Li, X. Du, J. R. Martins, Machine learning in aerodynamic shape optimization, Progress in Aerospace Sciences 134 (2022) 100849.

[3] N. Xie, Y. Zhang, X. Liu, R. Luo, Y. Liu, C. Ma, Thermal performance and structural optimization of a hybrid thermal management system based on mhpa/pcm/liquid cooling for lithium-ion battery, Applied Thermal Engineering 235 (2023) 121341.

[4] D. Rocke, Genetic algorithms + data structures = evolution programs by z. michalewicz, Journal of the American Statistical Association 95 (2000) 347–348. doi:10.2307/2669583.

[5] D. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, Addison Wesley series in artificial intelligence, Addison-Wesley, 1989.

[6] N. Benard, J. Pons-Prats, J. Periaux, G. Bugeda, P. Braud, J. Bonnet, E. Moreau, Turbulent separated shear flow control by surface plasma actuator: experimental optimization by genetic algorithm approach, Experiments in Fluids 57 (2016) 1–17.

[7] R. Castellanos, A. Ianiro, S. Discetti, Genetically-inspired convective heat transfer enhancement in a turbulent boundary layer, Applied Thermal Engineering 230 (2023) 120621. doi:10.1016/j.applthermaleng.2023.120621.

[8] K. Wu, H. Hu, L. Wang, Y. Gao, Parametric optimization of an aperiodic metastructure based on genetic algorithm, International Journal of Mechanical Sciences 214 (2022) 106878. doi:10.1016/j.ijmecsci.2021.106878.

[9] M. Turrin, P. von Buelow, R. Stouffs, Design explorations of performance driven geometry in architectural design using parametric modeling and genetic algorithms, Advanced Engineering Informatics 25 (2011) 656–675. doi:https://doi.org/10.1016/j.aei.2011.07.009, special Section: Advances and Challenges in Computing in Civil and Building Engineering.

[10] A. Kumar, N. Sinha, A. Bhardwaj, A novel fitness function in genetic programming for medical data classification, Journal of Biomedical Informatics 112 (2020) 103623.

[11] Y. Zhou, D. Fan, B. Zhang, R. Li, B. R. Noack, Artificial intelligence control of a turbulent jet, Journal of Fluid Mechanics 897 (2020) A27.

[12] R. Castellanos, G. Cornejo Maceda, I. De La Fuente, B. Noack, A. Ianiro, S. Discetti, Machine-learning flow control with few sensor feedback and measurement noise, Physics of Fluids 34 (2022).

[13] G. Y. Cornejo Maceda, Y. Li, F. Lusseyran, M. Morzyński, B. R. Noack, Stabilization of the fluidic pinball with gradient-enriched machine learning control, Journal of Fluid Mechanics 917 (2021) A42. doi:10.1017/jfm.2021.301.

[14] R. Li, B. R. Noack, L. Cordier, J. Borée, F. Harambat, Drag reduction of a car model by linear genetic programming control, Experiments in Fluids 58 (2017) 1–20.

[15] R. Li, B. R. Noack, L. Cordier, J. Borée, E. Kaiser, F. Harambat, Linear genetic programming control for strongly nonlinear dynamics with frequency crosstalk, arXiv preprint arXiv:1705.00367 (2017).

[16] I. Tsoulos, I. Lagaris, Solving differential equations with genetic programming, Genetic Programming and Evolvable Machines 7 (2006) 33–54. doi:10.1007/s10710-006-7009-y.

[17] A. Sobester, P. Nair, A. Keane, Genetic programming approaches for solving elliptic partial differential equations, IEEE Trans. Evolutionary Computation 12 (2008) 469–478. doi:10.1109/TEVC.2007.908467.

[18] N. Wagner, Z. Michalewicz, M. Khouja, R. McGregor, Time series forecasting for dynamic environments: The dyfor genetic program model, IEEE Trans. Evolutionary Computation 11 (2007) 433–452. doi:10.1109/TEVC.2006.882430.

[19] Z. Wang, X. Zhang, Z. Zhang, D. Sheng, Credit portfolio optimization: A multi-objective genetic algorithm approach, Borsa Istanbul Review 22 (2022) 69–76. URL: https://www.sciencedirect.com/science/article/pii/S2214845021000041. doi:https://doi.org/10.1016/j.bir.2021.01.004.

[20] J. H. Holland, Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence, The MIT Press, 1992. doi:10.7551/mitpress/1090.001.0001.

[21] G. Syswerda, Uniform crossover in genetic algorithms, in: ICGA, 1989, pp. 2–9.

[22] A. H. Wright, Genetic algorithms for real parameter optimization, in: G. J. Rawlins (Ed.), Foundations of Genetic Algorithms, volume 1, Elsevier, 1991, pp. 205–218. doi:10.1016/B978-0-08-050684-5.50016-1.

[23] K. Deb, R. B. Agrawal, Simulated binary crossover for continuous search space, Complex Syst. 9 (1995).

[24] K. Deb, M. Goyal, et al., A combined genetic adaptive search (geneas) for engineering design, Computer Science and informatics 26 (1996) 30–45.

[25] H. Beyer, The Theory of Evolution Strategies, Natural Computing Series, Springer, 2001.

[26] D. E. Goldberg, R. Lingle, Alleles, loci, and the traveling salesman problem, in: Proceedings of the first international conference on genetic algorithms and their applications, Psychology Press, 1985, pp. 154–159.

[27] G. Syswerda, Schedule optimization using genetic algorithms, Handbook of genetic algorithms (1991).

[28] J. Nocedal, S. Wright, Numerical Optimization, Springer Series in Operations Research and Financial Engineering, Springer New York, 2006.

[29] H. Hoos, T. Stützle, Stochastic Local Search: Foundations & Applications, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004.

[30] D. Adler, Genetic algorithms and simulated annealing: A marriage proposal, in: IEEE international conference on neural networks, IEEE, 1993, pp. 1104–1109.

[31] S. Xu, Y. Wang, Parameter estimation of photovoltaic modules using a hybrid flower pollination algorithm, Energy Conversion and Management 144 (2017) 53–68. doi:10.1016/j.enconman.2017.04.042.
[32] R. Yang, I. Douglas, Simple genetic algorithm with local tuning: Efficient global optimizing technique, Journal of Optimization Theory and Applications 98 (1998) 449–465.
[33] M. Musil, M. Wilmut, N. Chapman, A hybrid simplex genetic algorithm for estimating geoacoustic parameters using matched-field inversion, IEEE Journal of Oceanic Engineering 24 (1999) 358–369. doi:10.1109/48.775297.
[34] N. Maehara, Y. Shimoda, Application of the genetic algorithm and downhill simplex methods (nelder–mead methods) in the search for the optimum chiller configuration, Applied Thermal Engineering 61 (2013) 433–442. doi:10.1016/j.applthermaleng.2013.08.021.
[35] D. M. Luchtenburg, B. Günther, B. R. Noack, R. King, G. Tadmor, A generalized mean-field model of the natural and high-frequency actuated flow around a high-lift configuration, Journal of Fluid Mechanics 623 (2009) 283–316. doi:10.1017/S0022112008004965.
[36] Y. Li, W. Cui, Q. Jia, Q. Li, Z. Yang, M. Morzyński, B. R. Noack, Explorative gradient method for active drag reduction of the fluidic pinball and slanted ahmed body, Journal of Fluid Mechanics 932 (2022) A7. doi:10.1017/jfm.2021.974.
[37] W. Banzhaf, Genetic programming for pedestrians, in: Proceedings of the 5th International Conference on Genetic Algorithms, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993, p. 628.
[38] M. Brameier, W. Banzhaf, Linear Genetic Programming, Genetic and Evolutionary Computation, Springer US, 2007.
[39] Z. Michalewicz, C. Z. Janikow, Handling constraints in genetic algorithms., in: Icga, volume 398, 1991, pp. 151–157.
[40] A. Homaifar, C. X. Qi, S. H. Lai, Constrained optimization via genetic algorithms, Simulation 62 (1994) 242–253.
[41] T. Tarkowski, Genetic algorithm formulation and tuning with use of test functions, ArXiv abs/2210.03217 (2022).
[42] J. A. Nelder, R. Mead, A Simplex Method for Function Minimization, The Computer Journal 7 (1965) 308–313. doi:10.1093/comjnl/7.4.308.
[43] T. Rowan, The subplex method for unconstrained optimization, PhD thesis, PhD thesis, Ph. D. thesis, Department of Computer Sciences, Univ. of Texas (1990).
[44] N. Hansen, A. Ostermeier, Completely Derandomized Self-Adaptation in Evolution Strategies, Evolutionary Computation 9 (2001) 159–195. doi:10.1162/106365601750190398.
[45] N. Hansen, S. D. Müller, P. Koumoutsakos, Reducing the Time Complexity of the Derandomized Evolution Strategy with Covariance Matrix Adaptation (CMA-ES), Evolutionary Computation 11 (2003) 1–18. doi:10.1162/106365603321828970.
[46] B. F. Zhang, K. Liu, Y. Zhou, S. To, J. Y. Tu, Active drag reduction of a high-drag ahmed body based on steady blowing, Journal of Fluid Mechanics 856 (2018) 351–396. doi:10.1017/jfm.2018.703.
[47] E. Serre, M. Minguez, R. Pasquetti, E. Guilmineau, G. Deng, M. Kornhaas, M. Schäfer, J. Fröhlich, C. Hinterberger, W. Rodi, On simulating the turbulent flow around the ahmed body: A french–german collaborative evaluation of les and des, Computers & Fluids 78 (2013) 10–23. doi:10.1016/j.compfluid.2011.05.017, lES of turbulence aeroacoustics and combustion.
[48] P. Gillieron, A. Kourta, Aerodynamic drag control by pulsed jets on simplified car geometry, Experiments in Fluids 54 (2013) 1457. doi:10.1007/s00348-013-1457-y.