
FAST AND ROBUST PARAMETRIC AND FUNCTIONAL LEARNING WITH HYBRID GENETIC OPTIMISATION (HYGO)

Isaac Robledo

Department of Aerospace Engineering
Universidad Carlos III de Madrid
Madrid, Spain
isaac.robledo@alumnos.uc3m.es

Yiqing Li

Department of Mechanical Engineering
University College London
London, United Kingdom
yiqing.li@ucl.ac.uk

Guy Y. Cornejo Macea

Department of Aerospace Engineering
Universidad Carlos III de Madrid
Madrid, Spain
gcornejo@ing.uc3m.es

Rodrigo Castellanos

Department of Aerospace Engineering
Universidad Carlos III de Madrid
Madrid, Spain
rcastell@ing.uc3m.es

ABSTRACT

The Hybrid Genetic Optimisation framework (HyGO) is introduced to meet the pressing need for efficient and unified optimisation frameworks that support both parametric and functional learning in complex engineering problems. Evolutionary algorithms are widely employed as derivative-free global optimisation methods but often suffer from slow convergence rates, especially during late-stage learning. HyGO integrates the global exploration capabilities of evolutionary algorithms with accelerated local search for robust solution refinement. The key enabler is a two-stage strategy that balances exploration and exploitation. For parametric problems, HyGO alternates between genetic algorithm and targeted improvement through a degradation-proof Dowhill Simplex Method (DSM). For function optimisation tasks, HyGO rotates between genetic programming and DSM. Validation is performed on (a) parametric optimisation benchmarks, where HyGO demonstrates faster and more robust convergence than standard genetic algorithms, and (b) function optimisation tasks, including control of a damped Landau oscillator. Practical relevance is showcased through aerodynamic drag reduction of an Ahmed body via Reynolds-Averaged Navier-Stokes simulations, achieving consistently interpretable results and reductions exceeding 20% by controlled jet injection in the back of the body for flow reattachment and separation bubble reduction. 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 optimisation · Downhill Simplex Method · Flow control · Drag reduction · Ahmed body

1 Introduction

Global optimisation problems, particularly those characterised by strong nonlinearity, 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 collinearity 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 (Arora et al., 1995), 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 (Li et al., 2022); structural design under multi-physics constraints, such as in thermomechanical systems or energy absorbers (Xie et al., 2023); and control of turbulence through active or passive devices, where system dynamics are governed by nonlinear interactions and time-delays between inputs and outputs (Brunton and Noack, 2015).

To address the challenges posed by high-dimensional, nonlinear, and multimodal global optimisation problems, where purely local (exploitation-driven) methods often fail, various optimisation strategies have been developed to balance broad exploration with focused refinement. These strategies include evolutionary algorithms, such as genetic algorithms and differential evolution, which excel in exploring complex search spaces without reliance on gradient information; gradient-based methods, effective primarily for smooth and convex landscapes; bayesian optimizers, providing uncertainty estimates; and reinforcement learning approaches that cast optimisation as sequential decision-making with feedback. Each approach presents trade-offs in convergence speed, robustness, scalability, and sensitivity to noise or local minima. Evolutionary algorithms, in particular, are distinguished by their robustness and flexibility in handling black-box, multi-objective, and noisy problems, making them highly suited to many engineering optimisation challenges (Rocke, 2000).

Within the family of evolutionary algorithms, genetic optimisers stand out as a versatile and robust meta-heuristic for exploring complex search spaces. As stochastic methods inspired by natural principles, genetic optimisers operate on populations of candidate solutions that evolve iteratively through selection, crossover, and mutation, effectively balancing exploration of the search space with convergence toward promising optima (Goldberg, 1989). Genetic optimisers commonly subdivide into two categories based on solution representation: genetic algorithms (GAs) and genetic programming (GP). GAs operate on fixed-length parametric vectors, making them particularly well-suited for engineering problems defined over structured parameter domains. Their applications range from fluid flow control, such as plasma actuator optimisation for flow reattachment behind a backwards-facing step (Benard et al., 2016) and heat transfer enhancement in wall-bounded turbulent flows (Castellanos et al., 2023), to structural engineering design, including the design of 1D vibration-damped beams (Wu et al., 2022) and performance-driven architectural elements like building roofs (Turrin et al., 2011). GAs also showcased in medical applications such as disease classification (Kumar et al., 2020).

Conversely, genetic programming evolves variable-length symbolic expressions or programs, which are effective when the optimisation target is a function or decision-making policy. While historically limited by computational cost and complexity, recent algorithmic innovations and parallelisation have expanded GP's accessibility and impact. GP has gained prominence in adaptive control and optimisation, notably in flow control scenarios such as turbulent jet mixing (Zhou et al., 2020), wake flow stabilisation (Castellanos et al., 2022), and drag reduction in complex aerodynamic configurations including Ahmed bodies (Li et al., 2017a,b). Its symbolic formulation capability has also proven valuable in solving ordinary and partial differential equations (Tsoulos and Lagaris, 2006; Sobester et al., 2008), or in time series forecasting (Wagner et al., 2007). Beyond engineering, GP's symbolic and interpretable nature has made it useful in finance (Wang et al., 2022) and medicine (Kumar et al., 2020), highlighting its versatility for producing adaptive, interpretable functional strategies.

Over the years, the widespread adoption of genetic algorithms has driven numerous methodological improvements aimed at enhancing their efficiency and reliability. These include advanced selection mechanisms (Holland, 1992), adaptive crossover techniques (Syswerda, 1989; Wright, 1991; Deb and Agrawal, 1995), robust mutation operators (Deb et al., 1996; Beyer, 2001; Goldberg and Lingle, 1985; Syswerda, 1991), and dynamic parameter tuning strategies designed to adapt search behaviour over time. Despite these advances, both GA and GP share a fundamental limitation: their reliance on stochastic, population-based exploration often results in slow convergence and limited local refinement. Their inherent exploratory nature tends to lead to inefficiency, with the risk of premature convergence to suboptimal solutions.

To mitigate these issues, recent developments have focused on hybridising genetic algorithms with local search techniques, effectively combining broad global exploration with targeted fine-tuning. Such hybrids aim to accelerate convergence toward near-global optima, reduce sensitivity to initial conditions, and prevent premature convergence to local minima. For example, hybridising genetic optimisers with gradient-based methods can significantly improve convergence rates, particularly in smooth and differentiable landscapes (Nocedal and Wright, 2006). Alternatively, integrating local search algorithms (Hoos and Stützle, 2004) or stochastic refinement strategies such as Simulated Annealing (Adler, 1993) 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 (Xu and Wang, 2017) and genetic algorithms (Yang and Douglas, 1998), for applications including geometric and geoacoustic parameter estimation (Musil et al., 1999), chiller design

optimisation (Maehara and Shimoda, 2013), and, more recently, in combination with genetic programming for flow control (Cornejo Maceda et al., 2021). Its sustained popularity stems from its ability to complement global genetic strategies with efficient, gradient-free local optimisation, improving 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 DSM, hybrid optimisation frameworks offer a more balanced and effective approach to solving complex, high-dimensional engineering problems. These strategies underpin advanced metaheuristics capable of addressing the increasing demands of modern applications in control, design, and optimisation. Building on these principles and motivated by the need for robust, flexible hybrid genetic optimisers, we propose the Hybrid Genetic Optimiser framework (HyGO), which extends the hybrid mechanisms introduced by Cornejo Maceda et al. (2021). HyGO is designed as a versatile, extensible framework that integrates global and local search strategies, supporting hybridisation with a broad range of local optimiser algorithms such as BFGS and COBYLA, provided suitable parsing between solution representations and local method inputs is ensured. However, this manuscript focuses specifically on hybridisation with DSM due to its widespread acceptance, prior successful use in hybrid frameworks, and to evaluate our proposed degeneracy-proof adaptation. This key innovation mitigates common degeneracy issues in high-dimensional simplices, such as collapse to lower-dimensional manifolds, by implementing corrective measures that preserve the simplex geometry throughout optimisation, thereby enhancing convergence robustness and efficiency. HyGO also incorporates robust uncertainty-handling tailored for experimental applications, including repeated individual evaluations and user-defined outlier exclusion. Moreover, HyGO employs a soft constraint system that regenerates invalid individuals rather than discarding them outright, preserving population diversity while maintaining feasibility. The framework also features built-in error recovery capabilities, such as automatic checkpointing and secure fallback strategies, enabling optimisation to resume following evaluation failures. Supporting both parametric encodings and Linear Genetic Programming (LGP) formulations, HyGO is applicable across a wide variety of problem types. Fully implemented in Python, it offers a modular, customisable platform well-suited for integration into contemporary optimisation pipelines in simulation-based and experimental contexts.

To validate the HyGO framework, a diverse suite of test cases was selected to highlight the strengths of both GA and GP components, as well as the DSM-based local refinement. Initially, the GA component was assessed on standard analytical benchmark functions, affording controlled evaluation of stochastic convergence speed and solution quality. Subsequently, the GP component was employed to stabilise the damped Landau oscillator, a canonical test problem modelling the nonlinear dynamics of von Kármán vortex shedding behind a cylinder (Luchtenburg et al., 2009). This case evaluated the hybrid approach’s capability to enhance LGP performance through DSM enrichment in a non-trivial, time-dependent dynamical system. Finally, HyGO was applied to aerodynamic drag minimisation on an Ahmed body, a widely studied geometry in automotive aerodynamics. This real-world application in a simulation-based environment using RANS simulations demonstrated the framework’s practicality in high-dimensional design spaces and enabled direct comparison against the optimisation approach proposed by Li et al. (2022), providing a comprehensive context for assessing hybrid methodology effectiveness.

The remainder of this manuscript is structured as follows. Section 2 introduces the proposed hybrid genetic optimisation framework, detailing its architecture, genetic components, and local refinement strategy. Section 3 evaluates HyGO’s performance on analytical benchmark 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 the drag reduction application on an Ahmed body. Finally, section 6 concludes the study and outlines directions for future research.

2 Hybrid Optimisation with HyGO

In this section, we introduce HyGO, our optimisation framework developed to facilitate the design and deployment of hybrid genetic optimisers. The framework offers a structured environment to combine evolutionary algorithms with a variety of local enrichment techniques. We illustrate the methodology by merging parametric and functional optimisers, specifically GA for parametric optimisation and LGP for functional optimisation, with a variant of the DSM for local search purposes. The following subsections detail HyGO’s architecture, discuss the critical balance between exploration and exploitation in hybrid optimisation, outline the role of evolutionary algorithms in global search, describe the integration of local refinement techniques, and present the specific implementation of our hybrid methodology within the HyGO framework.

2.1 Genetic Optimisation Framework

Let $J(\mathbf{f}; \theta)$ represent the optimisation objective, or cost function, where $\mathbf{f} \in \mathcal{F}$ is a vector of optimisation variables within the feasible search space \mathcal{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:

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

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 initial population of I individuals is generated using Monte Carlo Sampling (MCS) or alternative methods such as random or Latin Hypercube Sampling (LHS), to ensure diversity. Individuals are evaluated via the cost function and ranked as $J_i^1 \leq J_i^2 \leq \dots \leq J_i^I$. Population evolution relies on four key operations: elitism, replication, mutation, and crossover. Elitism preserves the top N_e individuals, directly passing them to the next generation. For selecting parents for evolutionary operators, tournament selection is employed: randomly selecting N_t individuals from the population, ranking them by cost, and selecting parents probabilistically according to their ranking and a selection probability p_s : 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 promotes fitter individuals while maintaining diversity.

Following parent selection, 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 (with $P_r + P_c + P_m = 1$). 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 \mathcal{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. 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 framework, the way individuals are represented through their genome primarily influences the algorithm's behaviour, affecting exploration, exploitation, and the ability to bias the search according to the problem's characteristics. In HyGO, binary encoding is employed for the GA component mainly for its pragmatic advantages, such as simplicity, computational efficiency, and robustness in implementation. For the functional optimisation component, linear GP is used, leveraging its symbolic and interpretable structure for evolving decision policies.

2.1.1 Genetic Algorithm Encoding

In GAs, the optimisation variables \mathbf{f} are expressed as a vector of N scalar design variables $\mathbf{f} = [f_1, \dots, f_N]^T \in \Omega \subset \mathbb{R}^N$, defined within the domain Ω , where each parameter f_i is defined within a valid interval, $f_i \in [f_{i,\min}, f_{i,\max}]$, $i = 1, \dots, N$, leading to a rectangular parametric domain

$$\Omega = [f_{1,\min}, f_{1,\max}] \times \dots \times [f_{N,\min}, f_{N,\max}]. \quad (2)$$

GA typically employ chromosomes encoding sets of fixed parameters, most commonly using binary encoding due to its simplicity, broad applicability, and compatibility with standard genetic operators such as crossover and mutation, leading to its implementation in HyGO. Furthermore, binary encoding benefits from efficient memory usage and straightforward implementation, making it particularly suitable for parametric optimisation tasks where candidate solutions are naturally represented as vectors of design variables. While alternative encoding schemes, such as real-valued or Gray code encodings, show potential to enhance precision and improve convergence speed, these alternatives typically require increased complexity in operator design and genotype-phenotype mapping, often without

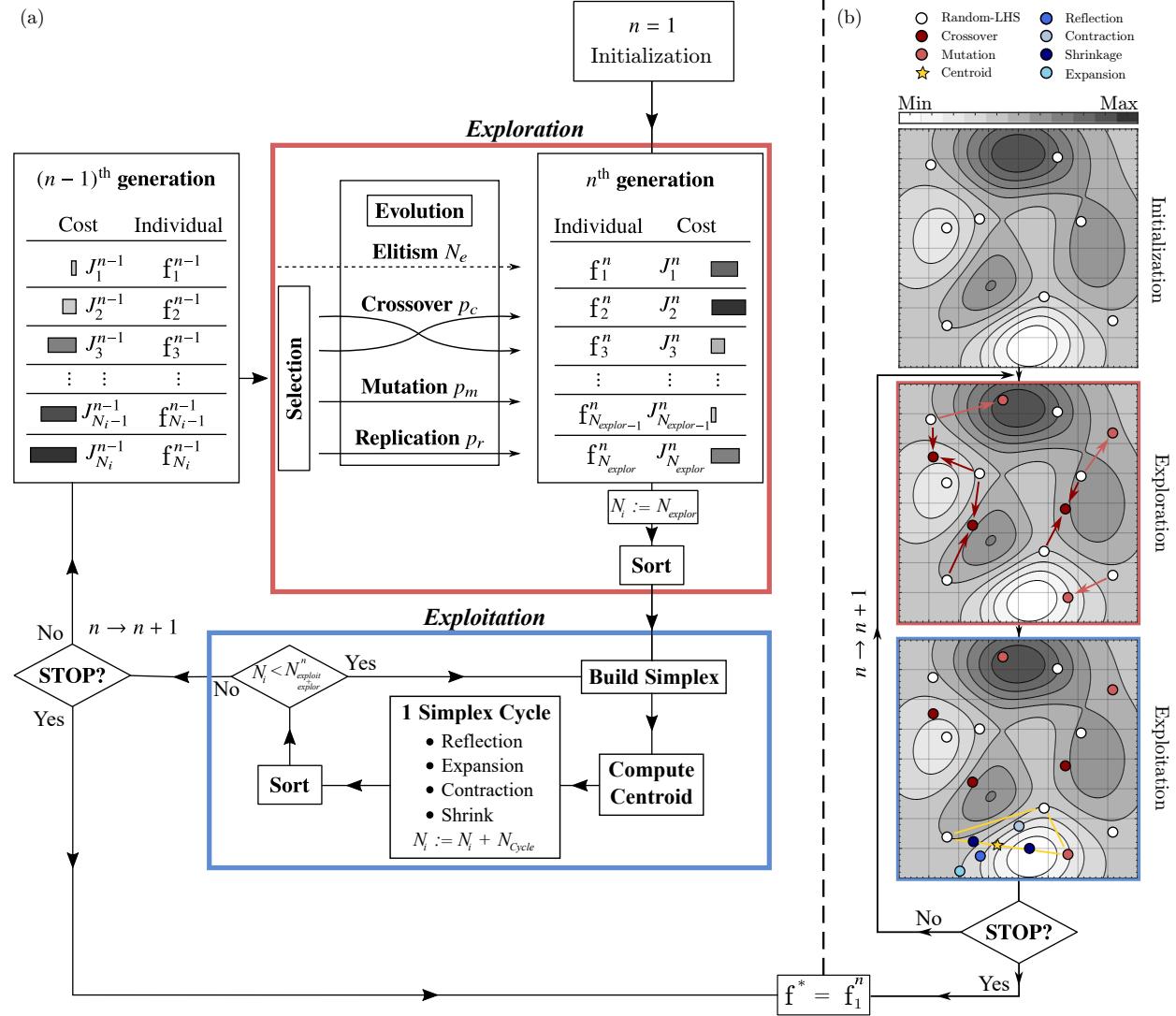


Figure 1: Hybrid Genetic Optimisation algorithm: exploration and exploitation scheme. (a) Schematic flowchart, highlighting its two-phase structure. In each generation n , exploration is performed by creating individuals f_i^n via genetic operations, which include elitism, crossover, mutation, and replication, 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 of an algorithmic sequence in a two-dimensional landscape, alternating between 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.

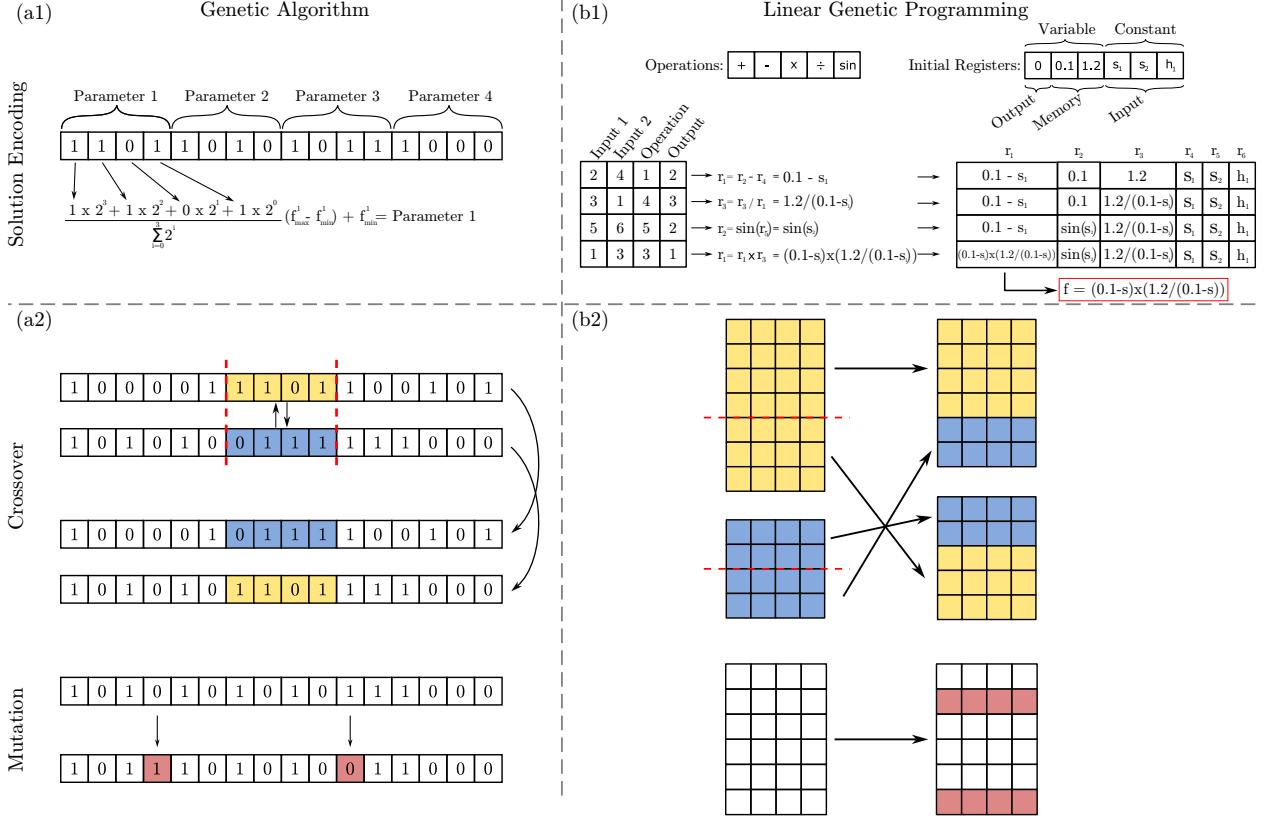


Figure 2: Solution encoding and genetic operations for Genetic Algorithm (GA) and Linear Genetic Programming (LGP). (a1) Binary-encoded chromosomes for GA represent sets of parameters. (b1) Encoding of LGP individuals as instruction matrices acting on program registers (variables, constants, memory, 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.

significant benefits in practical scenarios. For this reason, HyGO adopts binary encoding pragmatically, balancing robust performance with computational efficiency and ease of implementation.

In binary-encoded GAs as employed in HyGO, each of the optimisation variables f_i is represented by a string of n binary digits (bits), yielding a resolution of 2^n distinct values over the interval $[f_{i,\min}, f_{i,\max}]$. 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 into 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 clarity.

2.1.2 Linear Genetic Programming

In Genetic Programming, the design variable \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} , $\mathbf{f} = \mathbf{f}(\mathbf{h}, \mathbf{s})$. Unlike GAs, where individuals are fixed-length vectors, GP encodes individuals as computer programs, providing flexibility for evolving dynamic decision rules or control strategies Banzhaf (1993). Early implementations of GP employed tree-based representations, in which output functions were encoded as hierarchical structures of operations (such as +, -, ÷, ×, 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.

A widely used and interpretable variant is *linear* GP, in which individuals are encoded as instruction matrices operating on a shared set of registers. This format allows for compact, symbolic representations of functional laws that evolve

over time or in response to sensor inputs, making LGP especially well-suited to problems such as real-time control. As illustrated in Figure 2 (b1), and following the methodology introduced by Brameier and Banzhaf (2007), each chromosome is expressed as an $N_{it} \times 4$ instruction matrix, with each row specifying a single instruction that combines elements from a set of registers, which include sensor readings s_i , time-dependent functions h_i , intermediate variables, and outputs. Each instruction comprises four fields (columns): the first two columns specify the input arguments (if the operation is unary, the second input is ignored), the third column indicates the operator (e.g., addition, multiplication, sine), and the fourth column specifies the destination register where the result is stored.

All individuals share initial input registers containing: 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. Through sequential execution, the instruction set produces a program that dynamically maps system states to output functions \mathbf{f} . 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. The sequential nature of instruction execution is what defines the method as *linear*. An example illustrating how a functional output \mathbf{f} is constructed using this matrix-based representation is shown in Figure 2.

The evolutionary process in LGP follows principles similar to those used in binary-encoded GAs, 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 fixed-length binary encodings, LGP permits individuals to have varying chromosome lengths (up to a maximum of N_{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 where traditional gradient-based or local methods struggle to escape local minima or require smoothness and convexity in the objective function. Various practical scenarios highlight the necessity of balancing these two strategies. Large plateaus in the cost landscape require strong exploration, as the absence of gradient information provides little guidance for local descent. Conversely, narrow valleys or funnel-shaped regions benefit from exploitative steps that efficiently guide the search toward optima. Additionally, shallow gradients may allow convergence but at prohibitive computational cost, underscoring the need for exploration to “jump” toward steeper descent regions and accelerate optimisation.

The genotype-to-phenotype mapping in genetic optimisation amplifies sensitivity to small genetic changes, promoting effective exploration. Minor modifications in encoded individuals, such as bit flips in chromosomes or instruction changes in programs, can yield dramatically different solutions. This sensitivity is especially pronounced in control law design, where solutions close in actuation space may correspond to distant genetic representations. Consequently, GA and GP inherently favour exploration over exploitation, excelling at generating diverse candidate solutions but often requiring auxiliary mechanisms for effective local refinement.

Traditionally, balancing exploration and exploitation in genetic optimisation relies primarily on tuning the probabilities of replication, mutation, and crossover. Mutation introduces random variations, enhancing exploration, with the mutation probability P_m modulating exploratory intensity. Conversely, crossover combines genetic material from parents, promoting exploitation by steering the population toward local optima. Replication’s probability P_r influences the allocation of individuals to previously explored regions, typically reducing exploration without directly enhancing exploitation. The mutation rate within the genome controls offspring deviation from parents, enabling jumps across plateaus or local minima; excessive mutation, however, risks random search behaviour that can undermine population knowledge. Tournament selection further affects this balance; larger tournament sizes or lower selection probabilities increase exploration by allowing less-fit individuals reproductive chances, while smaller sizes intensify exploitation by favouring fitter individuals.

Recently, the focus has shifted from solely tuning hyperparameters toward hybrid algorithms combining genetic optimisers with complementary techniques to enhance performance. While genetic optimisers proficiently explore vast, complex search spaces, their capacity for efficient local exploitation is often constrained by the suboptimal nature of standard crossover operations. Though various crossover improvements exist (Syswerda, 1989; Deb and Agrawal, 1995), these alone rarely guarantee rapid, precise convergence. To overcome this, hybridisation with local search

Algorithm 1 HyGO pseudo-code

— *Initialisation* —

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

— *Exploitative stage* —

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

- 12: **while** not convergence **and** $g < N_g$ **do**

— *Explorative stage* —

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

— *Exploitative stage* —

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

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

methods has become the predominant strategy. Integrating genetic global search with precise, speedy local refinement enables algorithms to comprehensively explore solution spaces while effectively exploiting local optima. Such hybrid frameworks demonstrate superior convergence rates and solution quality across diverse complex optimisation problems, solidifying their status as powerful tools in engineering and scientific computation.

2.3 Hybridising 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 proposed in this work. Our approach synergistically combines the broad global search capabilities of binary-encoded GAs and LGP with the precise local refinement offered by local search techniques. The hybridisation is designed to retain the extensive exploratory strength of evolutionary algorithms while significantly enhancing local exploitation, leading to improved convergence speed and solution quality.

Each optimisation generation is divided into two distinct phases. The *explorative* phase uses genetic operations to generate N_r new individuals from the current population, while the *exploitative* phase applies local search algorithms to produce $N_{exploit}$ refined individuals. This structured alternation effectively balances exploration and exploitation within every generation. A detailed description of the algorithm is provided in Algorithm 1, accompanied by a simplified flowchart in Figure 1.

A critical challenge in this iterative hybrid process is preserving genetic diversity throughout evolutionary cycles, which is essential for maintaining robust exploration. Genetic operations (as described in subsection 2.1) are inherently stochastic, involving random crossover points and mutation flips. As optimisation progresses and diversity diminishes, the probability of generating individuals identical to those already in the population increases, undermining search effectiveness. Such duplicate individuals waste computational resources and reduce the genetic variability vital for efficient search performance. To address this, newly generated offspring resulting from crossover or mutation undergo a validity check against the current population. If duplicates are detected, these individuals are discarded and regenerated, with regeneration attempts capped to prevent infinite loops.

This validity-checking mechanism extends naturally to enforce *soft constraints*, an enduring challenge in genetic optimisation (Michalewicz and Janikow, 1991; Homaifar et al., 1994; Tarkowski, 2022). The term “soft” reflects that strict constraint satisfaction cannot always be guaranteed due to limited regeneration attempts; when constraints cannot be met, individuals may receive penalising extreme fitness costs or be evaluated regardless. However, such situations typically arise only in late generations once genetic diversity is low and convergence is near completion, minimising their practical impact on optimisation robustness.

Downhill Simplex Method: A Robust Gradient-Free Optimisation Technique

The Downhill Simplex Method, originally proposed by Nelder and Mead (1965), is a widely used optimisation algorithm notable for its simplicity and robustness. Unlike gradient-based methods, DSM does not require explicit gradient information, making it especially well-suited to optimisation problems where derivatives are unavailable or costly 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 search domain Ω . Each iteration seeks to improve the simplex by replacing the vertex with the highest cost, \mathbf{f}_{N+1} , with a new, improved vertex \mathbf{f}_{N+2} . The iteration proceeds through the following steps and is represented geometrically in Figure 3(a):

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

$$\mathbf{c} = \frac{1}{N} \sum_{m=1}^N \mathbf{f}_m. \quad (3)$$

3. **Reflection:** The worst vertex \mathbf{f}_{N+1} is reflected about the centroid \mathbf{c} to generate a new vertex \mathbf{f}_r :

$$\mathbf{f}_r = \mathbf{c} + (\mathbf{c} - \mathbf{f}_{N+1}). \quad (4)$$

If the cost of the reflected vertex $J_r = J(\mathbf{f}_r)$ lies between J_1 and J_N ($J_1 \leq J_r \leq J_N$), replace \mathbf{f}_{N+1} with \mathbf{f}_r and proceed to the next iteration.

4. **Expansion:** If reflection yields a new best point $J_r < J_1$, the simplex is expanded further in this direction:

$$\mathbf{f}_e = \mathbf{c} + 2(\mathbf{c} - \mathbf{f}_{N+1}). \quad (5)$$

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 reflection fails to improve the cost $J_r \geq J_N$, the simplex contracts by moving the worst vertex halfway towards the centroid:

$$\mathbf{f}_c = \mathbf{c} + \frac{1}{2}(\mathbf{f}_{N+1} - \mathbf{c}). \quad (6)$$

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 reduces cost, the simplex is shrunk by moving each vertex halfway towards the best vertex \mathbf{f}_1 :

$$\mathbf{f}_m \rightarrow \mathbf{f}_1 + \frac{1}{2}(\mathbf{f}_m - \mathbf{f}_1), \quad m = 2, \dots, N+1. \quad (7)$$

Shrinking is a last-resort operation, as it requires N additional function evaluations, typically applied when the simplex has degenerated to regain local gradients within a more confined parameter region.

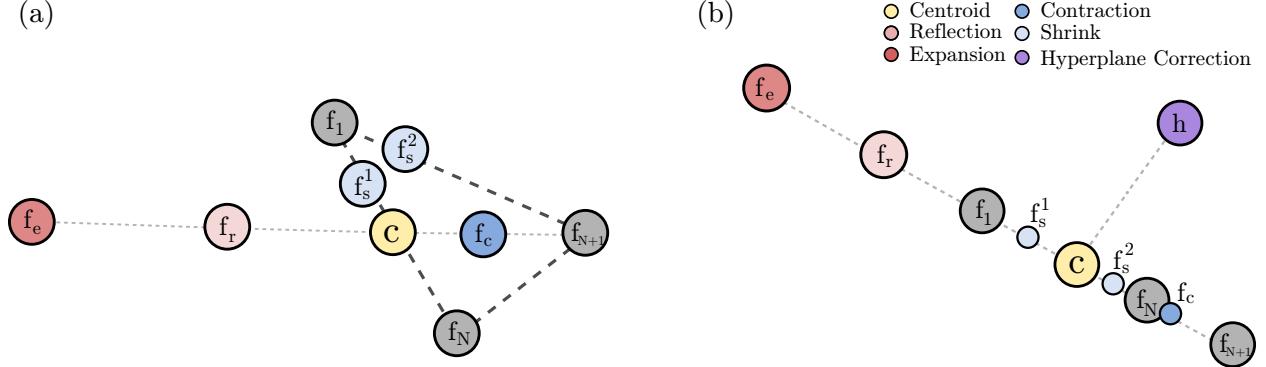


Figure 3: Hyperplane degeneracy into a 1-dimensional hyperplane in a 2D parametric domain. Image (a) displays all the operations and the topology under normal conditions. Image (b) shows a degenerate scenario where the Simplex collapsed into a line and lost the ability to move in the normal direction.

The DSM implementation in HyGO corresponds to this classical version. While alternative variants exist, the classical DSM is directly applicable to parametric optimisation problems by mapping solutions to their nearest binary-encoded points in Ω . For matrix-encoded LGP individuals, a more sophisticated Subplex algorithm (Rowan, 1990) is employed, as used in prior work (Cornejo Maceda et al., 2021), though its details are beyond this manuscript’s scope. Eventually, during the exploitation phase, HyGO executes a series of DSM iterations until generating $N_{exploit}$ individuals. However, the shrink operation may produce more candidates than the prescribed $N_{exploit}$, and truncating would disrupt the intended search dynamics and reduce algorithmic flexibility, so $N_{exploit}$ can naturally vary. Moreover, the initialisation of the simplex is key, directly affecting the effectiveness of DSM. Rather than starting with a canonical simplex with one vertex at the centre of the domain and N vertices distributed along each coordinate axis, HyGO initiates each simplex from the $N + 1$ fittest individuals in the current population.

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 $n < m$ (as illustrated in Figure 3b). In such cases, the DSM operations cannot recover the missing directions of movement, resulting in inefficient search behaviour, particularly when the gradient path is nonlinear, as in spiral-shaped functions. To detect degeneracy, HyGO computes the coefficient of determination R^2 of the last N_f simplex-generated individuals relative to a hyperplane. If R^2 exceeds a user-defined threshold, indicating near-planarity, a corrective step generates a new individual orthogonal to the hyperplane, placed at a distance determined by the grid resolution in the parametric space. This correction restarts the DSM cycle, rebuilding the simplex with the $N + 1$ best individuals to accommodate the new candidate. While this degeneracy correction may introduce extra individuals when the path is strongly directional, modestly increasing computational cost, it substantially enhances search robustness in complex landscapes.

3 Benchmarking Parametric Optimisation on Complex Analytical Functions

A suite of well-established analytical benchmark functions, known for their challenging optimisation landscapes, served as the baseline for evaluating the convergence performance of the proposed hybrid GA-DSM optimiser. Optimisations were conducted both with and without the exploitation phase, allowing assessment of the local refinement’s impact. To situate performance in a modern context, the approach was compared against the Covariance Matrix Adaptation Evolution Strategy (CMA-ES), a state-of-the-art evolutionary optimiser which utilises multivariate Gaussian distributions to generate populations and accelerates convergence by exploiting the evolution path (Hansen and Ostermeier, 2001; Hansen et al., 2003). The configuration parameters used for HyGO and GA optimisations are summarised in Table 1.

3.1 Preliminary benchmark: Rosenbrock 2D

To illustrate the convergence improvements obtained by incorporating DSM local search, an optimisation of the 2D Rosenbrock function was performed. This experiment comprised five generations of 50 individuals each, with 30 generated via the explorative phase and 20 produced by the DSM-driven exploitative phase. The initial population of 30 individuals was sampled uniformly at random within the bounds $x_i \in [-5, 5]$, depicted as black dots in the first generation in Figure 4.

Table 1: Algorithm configuration parameters for the analytical functions benchmark.

	Parameter	Value	Description
Common Parameters to both HyGO (GA-DSM) and GA	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)

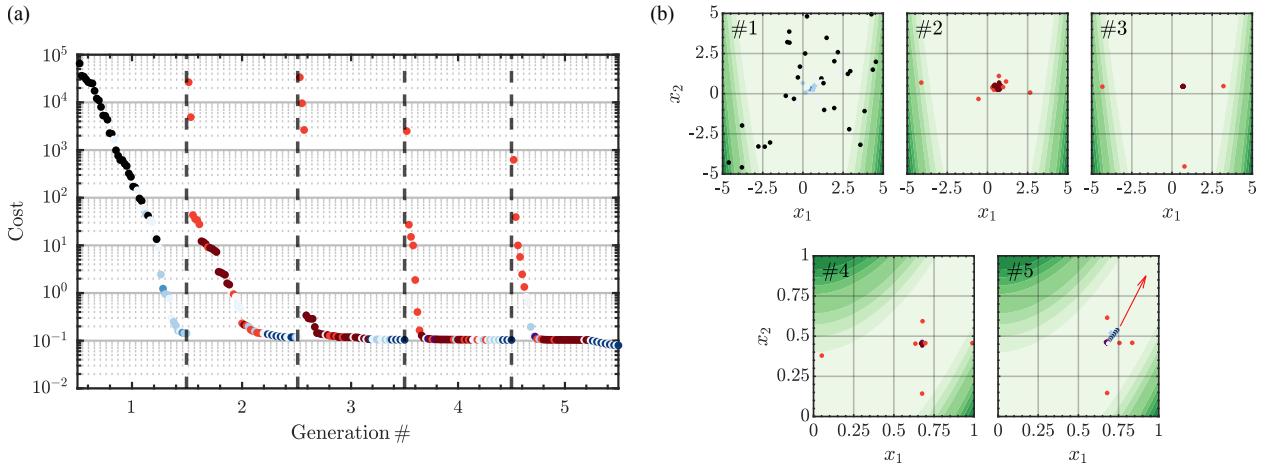


Figure 4: Example of HyGO performance on the optimisation of the Rosenbrock function in 2D: (a) cost evolution of the individuals through the generations. (b) Individuals’ distribution in the parametric space in each of the five generations (#1 – 5). The arrow in generation #5 shows the tendency of the 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).

In Figure 4, coloured dots represent the origin of each individual: red-toned dots correspond to explorative GA individuals, and blue-toned dots indicate exploitative DSM-refined individuals. The figure graphically demonstrates the acceleration in convergence enabled by the exploitation phase; exploitative individuals consistently occupy the lower cost region of the population distribution in every generation (panel (a)). The first generation particularly highlights these gains, as large gradients distant from the Rosenbrock valley allow the DSM to significantly reduce cost. As optimisation advances, improvements slow due to shallow gradients near the Rosenbrock valley minimum at $x_i^* = [1, 1]$ (marked by the red arrow in generation #5, panel (b)). The diminishing gradient magnitudes result in progressively smaller simplex steps, constraining the effectiveness of local refinement in late generations. The distribution plots in panel (b) additionally reveal how exploration and exploitation interplay spatially: explorative offspring are initially more dispersed, while exploitative individuals cluster tightly in promising regions, guiding convergence.

These results visually highlight the complementary strengths of the explorative and exploitative phases in the hybrid optimiser. Nonetheless, performance is subject to enhancement by hyperparameter tuning. For instance, initialising the population via Latin Hypercube Sampling would prevent premature convergence toward a local minimum far from the global optimum, mitigating slow progression down the Rosenbrock valley. Additionally, the current mutation operator employs a conservative *at least one* rule, which sets the bit mutation probability such that most mutated individuals differ by a single bit, altering only one parameter. This restriction limits mutation-induced exploration to vertical

Table 2: Analytical Benchmark Functions: Mathematical Definitions and Parametric Search Domains

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	$f(x_1, x_2) = (1.5 - x_1 + x_1 x_2)^2 + (2.25 - x_1 + x_1 x_2^2)^2 + (2.625 - x_1 + x_1 x_2^3)^2$	$x_1, x_2 \in [-4.5, 4.5]$
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{ x_2 - 0.01x_1^2 } + 0.01 x_1 + 10 $	$x_1 \in [-15, -5], x_2 \in [-3, 3]$
Easom	$f(x_1, x_2) = -\cos(x_1) \cos(x_2) \exp(-((x_1 - \pi)^2 + (x_2 - \pi)^2))$	$x_1, x_2 \in [-100, 100]$
Eggholder	$f(x_1, x_2) = -(x_2 + 47) \sin\left(\sqrt{ x_1/2 + (x_2 + 47) }\right) - x_1 \sin\left(\sqrt{ x_1 - (x_2 + 47) }\right)$	$x_1, x_2 \in [-512, 512]$
Goldstein-Price	$f(x_1, x_2) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1 x_2 + 3x_2^2)] \times [30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1 x_2 + 27x_2^2)]$	$x_1, x_2 \in [-2, 2]$
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) = - \sin(x_1) \cos(x_2) \exp(1 - \sqrt{x_1^2 + x_2^2}/\pi) $	$x_1, x_2 \in [-10, 10]$
Levi N.13	$f(x_1, x_2) = \sin^2(3\pi x_1) + (x_1 - 1)^2(1 + \sin^2(3\pi x_2)) + (x_2 - 1)^2(1 + \sin^2(2\pi x_2))$	$x_1, x_2 \in [-10, 10]$
Matyas	$f(x_1, x_2) = 0.26(x_1^2 + x_2^2) - 0.48x_1 x_2$	$x_1, x_2 \in [-10, 10]$
Sphere	$f(\mathbf{x}) = \sum_{i=1}^n x_i^2$	$x_i \in [0, 2]$
Rastrigin	$f(\mathbf{x}) = 10n + \sum_{i=1}^n [x_i^2 - 10 \cos(2\pi x_i)]$	$x_i \in [-5.12, 5.12]$
Rosenbrock	$f(\mathbf{x}) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	$x_i \in [-5, 5]$
Styblinski-Tang	$f(\mathbf{x}) = \frac{1}{2} \sum_{i=1}^n (x_i^4 - 16x_i^2 + 5x_i)$	$x_i \in [-5, 5]$

or horizontal movements in parameter space (see generation #4 in Figure 4(b)), potentially constraining robustness. Increasing mutation rates could enable diagonal moves and better complement the local search phase.

3.2 Performance Evaluation of HyGO as parametric optimiser

Evaluating the performance of a stochastic optimisation method necessitates rigorous k -fold statistical analysis. A set of 15 analytical benchmark functions featuring diverse and challenging topologies was used to assess HyGO’s behaviour. To probe scalability, some functions were optimised in low- and high-dimensional variants, resulting in 20 distinct scenarios. The study compares classical GA, HyGO, and CMA-ES (the latter only for high-dimensional cases), with a limit of 5000 function evaluations per run. Each scenario was repeated for 50 independent runs with randomised initialisations, ensuring robust statistical coverage and mitigating initialisation bias. Summary results are presented in Table 3, with function definitions and search domains listed in Table 2.

HyGO consistently outperforms the classical GA across measured metrics in Table 3. The convergence criterion is strict, demanding identification of the exact global minimum parameter set. HyGO achieves superior convergence on half of the benchmark functions, often by a substantial margin in the best objective costs. Equivalent convergence is observed on eight functions, although four are high-dimensional cases where the probability of exact convergence is extremely low due to the vast discretised search space (2^{12} points per variable in 25 dimensions). In these instances, mean best cost offers a more realistic metric of performance, and HyGO delivers notable improvements.

Exceptions include the Ackley, Rastrigin, and Sphere functions in two dimensions, where GA achieves higher convergence rates and better mean best cost; the Sphere function is trivial for both algorithms, given its geometric simplicity. Notably, in functions where convergence is achieved, HyGO generally requires fewer evaluations and generations than GA, highlighting the efficiency gained by DSM exploitation. Special consideration is required for functions with zero convergence rates, such as Bukin N.6, notable for its complex valley topology densely populated by similar local minima. Caution is advised when interpreting evaluation count metrics in zero-convergence scenarios due to potential re-sampling limits, i.e. repeated individuals that could not be re-generated within the maximum number of attempts.

Overall, the inclusion of DSM in the optimisation process substantially improves both solution quality and computational efficiency across a broad suite of analytical benchmarks. In terms of cost, HyGO delivers a superior result in 12 scenarios compared to 6 for GA, while also requiring fewer evaluations (10 versus 4), demonstrating the robustness and advantage of the hybrid strategy. These averages, derived from 50 independent runs per case, further underline the statistical advantage imparted by hybridisation, despite occasional outlier effects from especially favourable or unfavourable initialisations.

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	5.259
Ackley-2D	100	98	14.48	5.72	1290.3	548.72	0.002	0.054
Beale	2	86	49.22	14.08	2689.9	1372.7	0.030	0.107
Booth	6	100	48.58	2.82	2542.4	264.8	0.125	2.082e-07
Bukin N.6	0	0	50	50	3706.8	4948.2	0.124	0.071
Easom	14	96	45.8	11.12	2640.1	1077.9	-0.660	-0.960
Eggholder	4	18	48.54	41.88	2555.6	4096.4	-915.620	-896.190
Goldstein-Price	100	100	17.8	4.92	1555.5	468.78	3.000	3.000
Himmelblaus	54	100	36.28	3.62	2117.6	340.2	0.004	7.237e-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	0.011	0.015
Matyas	56	100	37.64	3.42	2286.6	321.24	9.267e-4	9.127e-09
Sphere-25D	0	0	50	50	4950.9	4909.6	0.133	0.0219
Sphere-2D	100	100	13.62	1.76	1250.2	151.02	0.000	0.000
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	0.025	0.264
Rosenbrock-25D	0	0	50	50	4950.8	4917.7	10182.000	193.290
Rosenbrock-2D	2	76	49.52	29.14	2710.7	2863.4	0.207	0.129
Styblinski-Tang-25D	0	0	50	50	4950.7	5000	-932.050	-936.580
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

3.3 High-dimensionality performance

The Rosenbrock, Rastrigin, and Sphere functions were selected to interrogate the influence of dimensionality on algorithmic performance. These functions, by virtue of their topological differences, provide archetypes for gradients (Sphere), multi-modality requiring broad exploration (Rastrigin), and landscapes mixing exploration and exploitation demands (Rosenbrock). Notably, GA outperforms or matches HyGO in certain low-dimensional scenarios, highlighting the importance of both landscape and dimensional scaling.

As visualised in Figure 5, each panel presents the evolution of the objective value J as a function of iteration for 50 independent runs per algorithm. The upper row shows 2D cases; the lower row, 25D. Solid and dashed lines correspond to mean and median performances, with lighter lines for individual trajectories, providing insight into both average trends and robustness.

For the 2D Rastrigin function (middle, top), GA’s strong exploratory focus enables it to find global minima most effectively, outperforming both HyGO and CMA-ES. In contrast, GA is least effective on the gradient-dominated Sphere function (top right), where exploitation is critical; here, HyGO converges fastest, outperforming both GA and CMA-ES. Notably, HyGO’s geometric gradient approximation proves more efficient at early stages, while CMA-ES’ statistical gradient estimation accelerates in late-stage fine-tuning. On the other hand, the Rosenbrock function (top left) demands a combination of tactics. In 2D, CMA-ES ultimately achieves the lowest mean cost by leveraging path exploitation, yet HyGO remains competitive, particularly considering its consistent convergence and robustness across runs.

The high-dimensional (25D) results, shown in the lower row, exhibit damped impact from initial sampling: all algorithms initialise from poorly informed, sparsely distributed populations. Here, in the high-dimensional Sphere, HyGO and CMA-ES are initially similar, 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, CMA-ES benefits from memory in later iterations, slightly outperforming HyGO. On the other hand, GA continues to lag due to its lack of exploitative power.

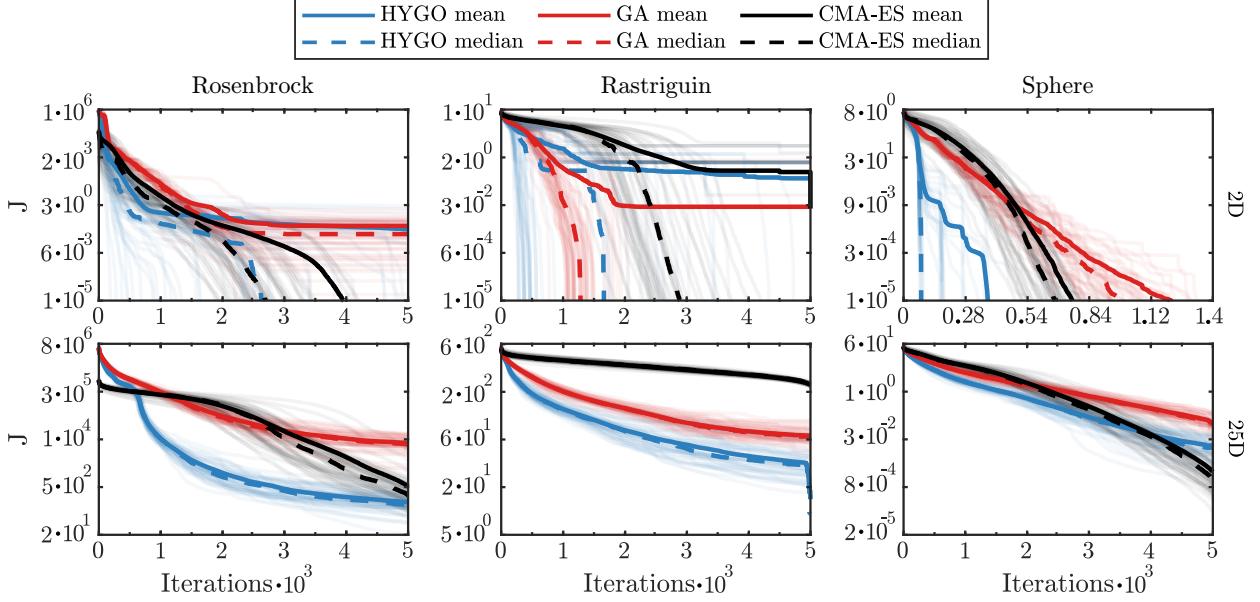


Figure 5: Convergence comparison of HyGO, GA, and CMA-ES algorithms on the Rosenbrock, Rastrigin, and Sphere benchmark functions in 2D (top row) and 25D (bottom row). Each panel shows the evolution of the cost function J as a function of the number of iterations across 50 independent runs with randomised 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

For the Rastrigin benchmark, both GA and CMA-ES struggle with exploration in the vast search space, and only HyGO can maintain an effective balance, lowering the objective more rapidly and to a greater degree through geometric exploitation once promising regions are found. The large number of minima slows down the evolution of CMA-ES. Moreover, due to the large number of parameters, the different minima are more challenging to exploit because finding the gradient direction is more complex 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-dimensional Rastrigin function.

In high-dimensional Rosenbrock, HyGO and CMA-ES again show strong late-stage convergence, with GA falling behind for most of the optimisation horizon. 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.

Overall, these results demonstrate that high-dimensional settings increase the relative value of effective local exploitation once regions of interest are identified, explaining the superior or comparable performance of hybrid strategies and CMA-ES versus GA alone. Notably, HyGO consistently delivers a robust trade-off between exploration and exploitation, yielding competitive or superior convergence profiles as dimensionality increases. In low-dimensional scenarios, algorithmic strengths depend more critically on landscape structure: GA excels in rugged, global landscapes, while HyGO and CMA-ES dominate where gradient exploitation is possible.

4 Functional learning benchmark: The Damped Landau Oscillator

The Landau oscillator serves as an analytical benchmark to validate the enhanced LGP optimiser. Its relative simplicity makes it an ideal testbed for evaluating LGP's ability to evolve effective control strategies and assessing the impact of Subplex local search enrichment on optimisation performance. Moreover, the Landau oscillator captures the oscillatory dynamics of von Kármán vortex shedding behind a cylinder (Luchtenburg et al., 2009), making it a relevant proxy for fluid flow control scenarios. The governing equations of the Landau oscillator are:

$$\begin{cases} \dot{a}_1 = (1 - a_1^2 - a_2^2) a_1 - a_2, \\ \dot{a}_2 = (1 - a_1^2 - a_2^2) a_2 + a_1 + b(a_1, a_2), \end{cases} \quad (8)$$

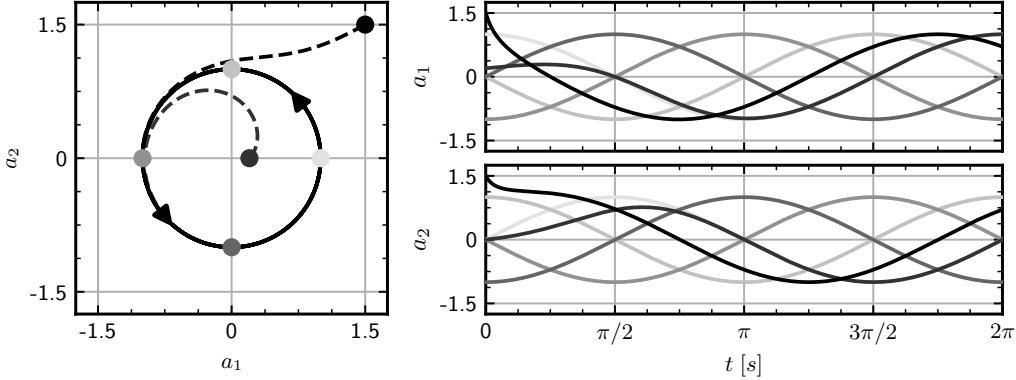


Figure 6: Phase space and time evolution of the undamped Landau oscillator for six representative initial conditions $(a_1, a_2)_{t=0} = [(1, 0), (-1, 0), (0, 2), (0, 1), (0, -1), (1.5, 1.5)]$. Left: Orbits in the (a_1, a_2) phase plane illustrating stable periodic motion on the unit circle. Right: Time 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.

where $b(a_1, a_2)$ represents the control input. Without control ($b = 0$), the system exhibits a stable periodic orbit on the unit circle, with solutions rotating counterclockwise indefinitely. Figure 6 illustrates six representative initial conditions rapidly entering the stable limit cycle.

The optimisation goal is to stabilise the oscillator at the origin $(a_1, a_2)_{t=t_f} = (0, 0)$ as fast as possible with minimal control effort. The cost function J combines a stabilisation term J_a (the mean squared distance to the origin) and an actuation penalty J_b (the mean control energy), as defined in Equation 9. Formally, the proposed optimisation problem is multi-objective. Since HyGO is a single-objective algorithm, scalarisation is achieved by selecting an appropriate weight γ based on the following single-objective definition of the cost function:

$$J = J_a + \gamma J_b = \left(\frac{1}{t_{end}} \int_0^{t_{end}} a_1^2 + a_2^2 dt \right) + \gamma \left(\frac{1}{t_{end}} \int_0^{t_{end}} b(a_1, a_2)^2 dt \right) \quad (9)$$

with $t_{end} = 20T$, and $T = 2\pi$ corresponding to the oscillator's fundamental period given that the angular frequency of the uncontrolled oscillator is 1.

In this setup, the control input influences only the second equation, simplifying the optimisation to a single control law. Both state variables (a_1, a_2) serve as sensor inputs to the control law, ensuring adaptability to instantaneous system states. While more complex problems may require historical sensor values (e.g., delayed coordinates $a_i(\tau - kT/4)$ for $k = 0, \dots, 4$), as in Castellanos et al. (2022)), the Landau oscillator's simplicity allows effective optimisation without this complexity, avoiding unnecessary expansion of the solution space which can hamper convergence.

To assess the impact of Subplex local search enrichment, two optimisations were performed, with and without Subplex exploitation, using the average cost across four initial conditions $(a_1, a_2)_{t=0} = \{(1, 0), (-1, 0), (0, 1), (0, -1)\}$ for generalisation. The evolution of the optimisation process is illustrated in Figure 7. Each optimisation spanned 10 generations of 100 individuals, with the hybrid approach allocating 80 individuals to the explorative phase and 20 to Subplex-based exploitation. 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 initialising the individuals with complex control laws, allowing the algorithm to simplify them. After some analysis, this proved the best approach since a low number of instructions leads to constant control laws. However, initialising the optimisation with fewer instructions forces complex operations (including trigonometric functions, sensors...). Registers included a zero-initialised output register to avoid bias, two sensors for the current state, a randomly initialised memory register, and two constant registers holding random values in the range $[0, 1]$. Regarding the subplex, no definitive recommendation exists for Subplex size due to the problem's effective infinite dimensionality. Empirical testing indicated that selecting the top 10 individuals (10% of the population) balances fittest retention with control law variability. Larger Subplex subpopulations tend to increase output law complexity,

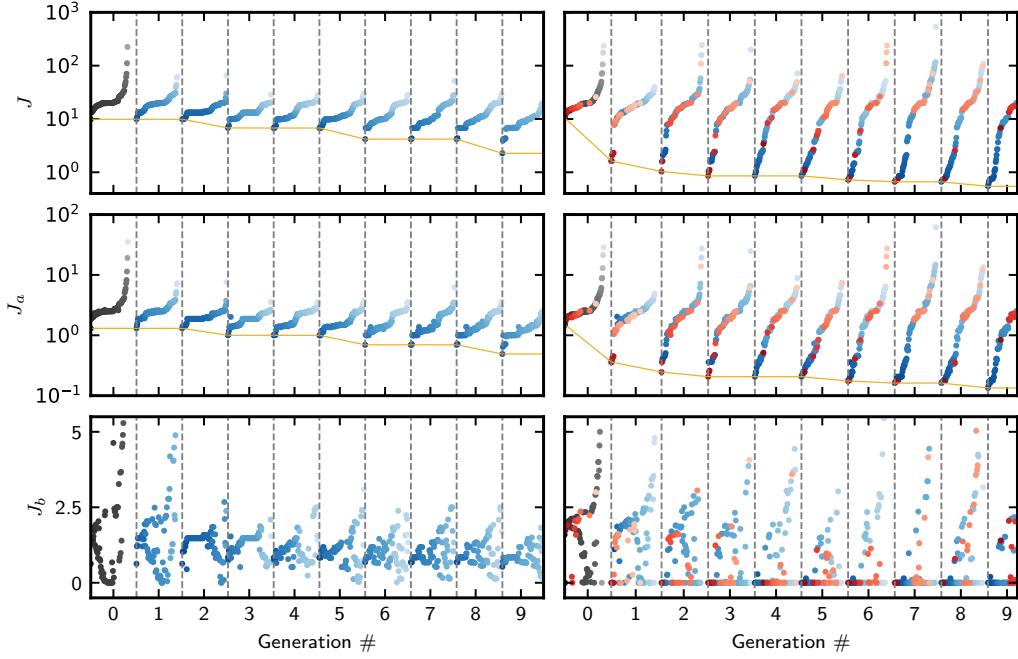


Figure 7: Evolution of the total cost J (top row), stabilisation cost J_a (middle), and the penalisation J_b (bottom) through the generations for a GP optimisation of the damped Landau oscillator, comparing cases without (left) and with (right) Subplex local search enrichment. Each dot represents an 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.

adversely affecting interpolation efficiency. Nevertheless, this parameter highly depends on user experience and the targeted optimisation problem.

Figure 7 depicts the evolution of total cost J , stabilisation term J_a , and control penalisation J_b . The optimisation is controlled by the stabilisation term J_a (due to the low value set to the weighting parameter γ), leading to a very similar progress of both J and J_a . Both algorithms demonstrate an ability to reduce J_a and J_b over successive generations, adapting control laws to trade off effectiveness (stabilisation) and efficiency (control sparseness). However, the trajectory of cost evolution is markedly different between approaches. Notably, HyGO’s Subplex enrichment results in rapid, early-stage reductions in both cost terms, evident as a step-change improvement early in the hybrid run, whereas pure LGP produces steady but incremental gains.

A critical qualitative distinction emerges in how the algorithms target the conflicting goals embedded in J_a and J_b . While both approaches reduce J_a by driving the system rapidly towards the origin, HyGO more effectively minimises J_b , leveraging Subplex-generated individuals that achieve stabilisation using sharp, short-duration bursts of control. This results in actuation that is highly efficient in time, since the control input is intense but localised, minimising its integral, and thus achieves lower overall cost per the objective definition.

Inspection of the best control laws in Figure 8 further underscores these strategic differences. LGP-only optimisation tends to favour solutions built from complex, oscillatory trigonometric expressions; these drive the system towards the origin but often at the price of extended actuation and oscillation before settling, as seen in the phase space and time histories. In contrast, Subplex-enriched HyGO rapidly identifies and refines control architectures characterised by larger, direct scaling factors, yielding nearly monotonic, efficient convergence to the desired state with sparser and more purposeful actuation. Analysis reveals that the improvement caused by this Subplex-created group stems from emergent large-scale multipliers generated via allowed arithmetic operations, enabling strong initial control inputs that rapidly reduce oscillations. Although the value variable and constant initial registers were limited between $[0, 1]$, summations and divisions allow combining scalar numbers beyond 1.

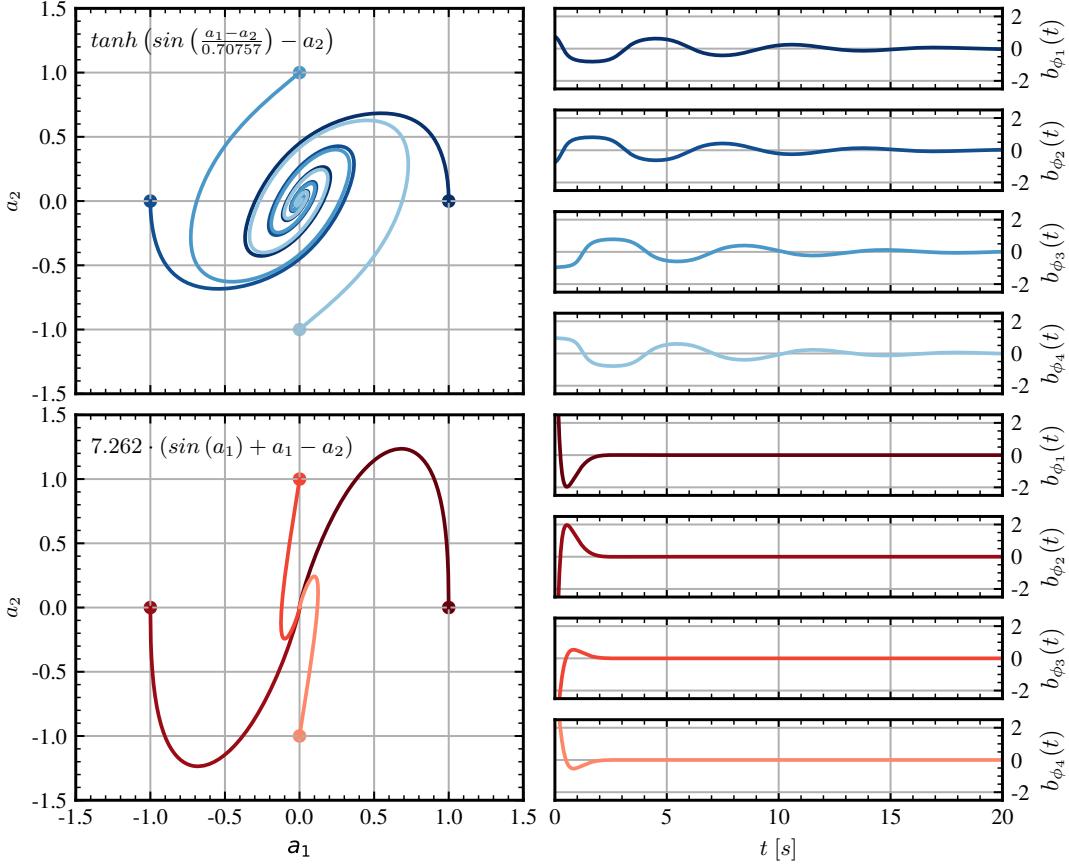


Figure 8: Comparison of optimal control laws discovered for the damped Landau oscillator via genetic programming, without (top) and with (bottom) Subplex enrichment. Left: Phase space trajectories show the system state (a_1, a_2) evolving from multiple initial conditions under the learned feedback. The 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.

Eventually, both optimisation strategies are capable of evolving control policies that balance stabilisation speed and energy efficiency, yet HyGO consistently finds superior solutions. Hybridisation via Subplex local search allows HyGO not only to escape the incrementalism characteristic of mutation/crossover searches but to discover control laws that target the time-localised intervention ideal for the Landau benchmark. These findings reinforce the value of integrating geometric, gradient-free exploitation into evolutionary control synthesis, especially when sparse, interpretable, and efficient functional solutions are sought.

5 Control of shedding flows: Drag reduction on Ahmed body

The previous sections established the efficacy of the proposed hybrid optimisation methodology on controlled analytic benchmarks, enabling a detailed examination of HyGO’s exploration and exploitation capabilities. However, while such benchmarks are valuable for method validation, they do not capture the complexity typical of real-world engineering applications, where the parametric landscape is highly nonlinear, multidimensional, and physically constrained. To demonstrate the practical utility of HyGO, we now address a flow control benchmark: the minimisation of aerodynamic drag on a canonical Ahmed body. This flow control optimisation leverages the simulation setup of Li et al. (2022) for

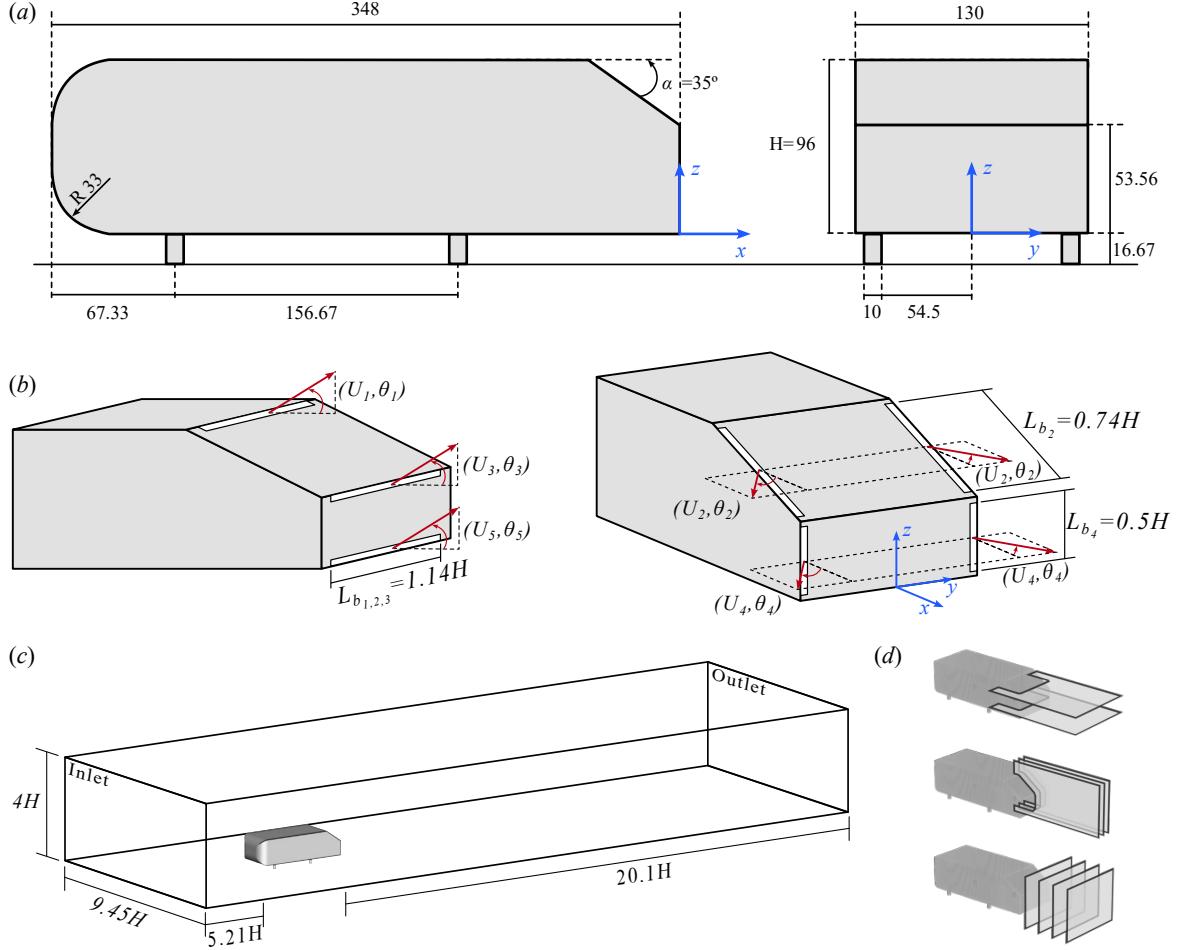


Figure 9: Ahmed body geometry, actuator layout, and computational domain for drag-reduction flow control studies. (a) Side and rear views of the Ahmed body model ($H = 96$), showing all relevant dimensions and axes. (b) Deployment and blowing direction (arrows, defined by actuation speed U_i and angle θ_i) of the five slot actuator groups 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 defined relative to H . (c) Computational domain for CFD simulation, locating the Ahmed body within its wind tunnel environment. (d) Schematics of visualisation planes and field of view for visualisation purposes in Figure 12, 13, and 14. Panels (a,b,c) are adapted with permission from Li et al. (2022)

direct comparison with the state-of-the-art Explorative Gradient Method (EGM), another leading gradient-free hybrid approach.

5.1 Ahmed body geometry and Simulation setup

The reference geometry employed in this study is a 1:3-scale Ahmed body, featuring a slanted rear surface at $\alpha = 35^\circ$. Key model dimensions are $L = 348$ mm (length), $W = 130$ mm (width), and $H = 96$ mm (height). Front edges feature a $0.344H$ rounding radius, and the body is supported by four cylindrical mounts of 10 mm diameter, and ground clearance of $0.177H$. A Cartesian coordinate system (x, y, z) with the origin at the midpoint of the rear vertical base's lower edge, aligned with the centreplane (Figure 9). Here, x , y , and z denote streamwise, spanwise, and wall-normal directions, and their velocity components are u , v , and w , respectively. The free-stream velocity is set to $U_\infty = 30 \text{ m s}^{-1}$.

Flow control is implemented using five groups of steady slot actuators placed along the rear slant and base perimeter, as depicted in Figure 9. All slots have a 2 mm. The top, middle, and bottom actuators span 109 mm; the upper and lower side actuators are 71 mm and 48 mm, respectively. The actuation velocities U_1, \dots, U_5 are independent optimisation variables: U_1 (upper window edge), U_3 (middle), and U_5 (bottom of vertical base); U_2 and U_4 (side

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 step 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

actuators). Following Zhang et al. (2018) and the same numbering as in velocities, the actuator angles θ_i are also optimisable, with $\theta_1 \in [-35^\circ, 90^\circ]$ and $\theta_{2,\dots,5} \in [-90^\circ, 90^\circ]$.

The objective is to minimise the drag coefficient $J = C_D$, that is, to maximise drag reduction, neglecting input power or lift penalties for clarity of comparison. First, a five-dimensional optimisation is conducted, where only the actuation velocities (each bounded not to exceed twice the optimal single-actuator value from Li et al. (2022)) are variable and angles are fixed at $\theta_i = 0^\circ$ (streamwise blowing). The search space is later expanded to ten parameters by including angular variation as described above.

The numerical and meshing setup matches that in Li et al. (2022). The simulation domain, constructed with Ansys ICEM CFD, forms 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 $W_T = 9.45H$, all exceeding recommended thresholds (Serre et al., 2013) to minimise boundary effects. The mesh consists of approximately five million hexahedral cells, providing a trade-off between accuracy and computational tractability. Near-wall and actuator slot resolution ($\Delta x^+ = 20$, $\Delta y^+ = 3$, and $\Delta z^+ = 30$) ensures reliable capture of boundary and shear-layer physics.

RANS simulations with the $k-\epsilon$ turbulence model are performed in Fluent, employing second-order spatial discretisation and semi-implicit pressure–velocity coupling. For this geometry (35° Ahmed body), RANS has been shown to yield trustworthy estimations (Li et al., 2022) and is thus appropriate for control actuator optimisation.

5.2 Optimisation results and data analysis

The performance of HyGO is systematically benchmarked against three alternative optimisation frameworks, all evaluated under an identical simulation protocol. The comparison encompasses: (i) the Explorative Gradient Method from Li et al. (2022), serving as a reference for local exploitation-driven search; (ii) a conventional Genetic Algorithm with no hybridisation or local search; and (iii) two configurations of DSM-augmented HyGO: a direct approach optimising all ten control variables simultaneously, and a *stepped* variant in which only the actuation velocities U_i are optimised initially, followed by a second phase where actuation angles θ_i are introduced. All strategies employ random initial populations to mitigate sampling bias. For the stepped run (HyGO-stepped), the transition to the full parameter space is seeded by an 11-individual simplex, constructed about the velocity-optimal solution as summarised in Table 5).

Key algorithmic and hyper-parameter choices are summarised in Table 4 for all genetic-based optimisers. A comprehensive summary of the optimisation histories is illustrated in Figure 10. In all cases, optimisations terminate either

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

Case	C_D (reduction)	Actuation parameters				
		Top	Upper	Middle	Lower	Bottom
Unforced	0.313 (0%)	—	—	—	—	—
GA	0.279 (10.88%)	31.5m/s −28.8°	28m/s −50.4°	19m/s −3.6°	49m/s −48.6°	16m/s 57.6°
HyGO	0.265 (15.6%)	14.5m/s −27.5°	17m/s −43.2°	4m/s 18°	19.5m/s −39.6°	4.5m/s 25.2°
Li et al. (2022)	0.258 (17.49%)	21.5m/s −27°	23.81m/s −42°	1.8m/s 67°	25.2m/s −44°	22.5m/s 22°
HyGO (stepped)	0.249 (20.56%)	34.5m/s −30°	36m/s −37.8°	8.5m/s −72°	32.5m/s −50.4°	21m/s 48.6°

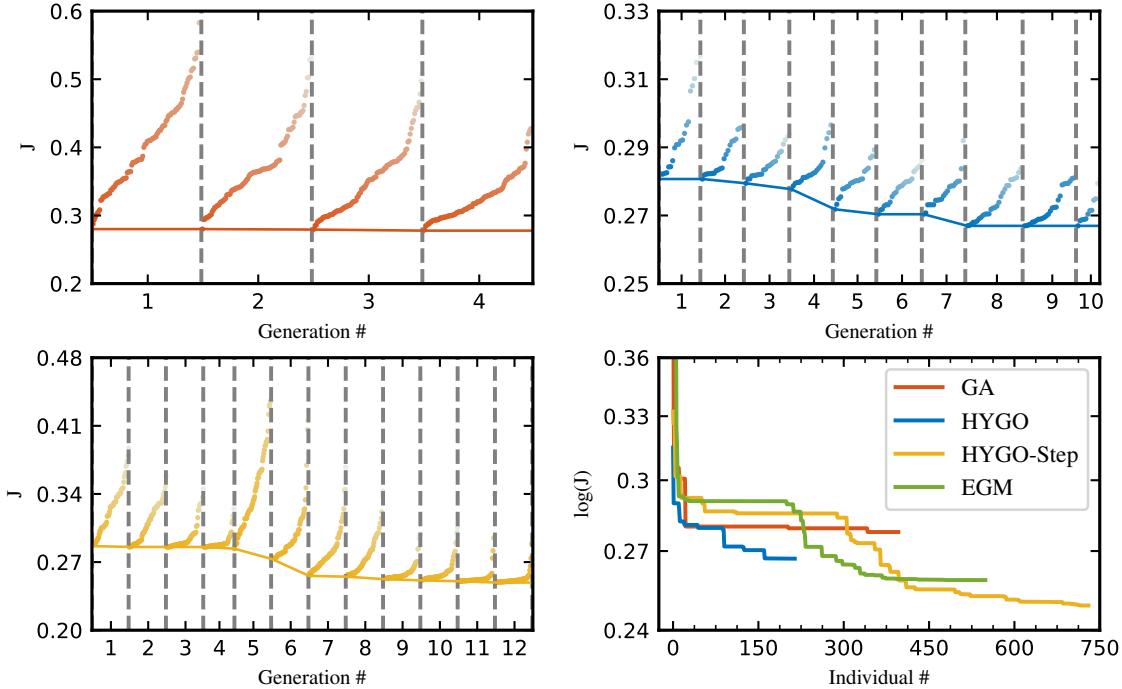


Figure 10: Evolution of the cost function J during the optimisation of drag reduction on the Ahmed body for four strategies. Each panel shows the cost across generations for a different method: GA (top left), HyGO (top right), and HyGO-Step (bottom left), while the bottom right plot compares the best cost evolution as a function of individual number for all approaches, including the EGM benchmark (green) from Li et al. (2022). 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.

upon exhausting the maximum generations (Table 4) or earlier if no improvement is observed across four successive generations. This approach provides a balanced assessment of convergence rates and stagnation risks for each algorithm.

The basic GA yielded an initial improvement, attaining a 10.9% reduction in drag compared to the reference value $C_D^0 = 0.3134$, yet failed to realise further gains beyond the best individual in its initial population. The evolutionary process stalled prematurely, evidenced by both early algorithmic termination and broad dispersion of individuals across the parameter space (Figure 10, orange). This result underscores the limitations of relying solely on stochastic genetic operations for high-dimensional, multimodal fluid-structure optimisation, where complex flow responses may easily trap the process in suboptimal configurations that are unreachable by simple crossover or mutation.

In contrast, HyGO exhibited a markedly different learning process (Figure 10, blue). Incorporation of the DSM enabled more consistent improvement across generations, with a final drag reduction of 15.6%. The synergistic use of DSM was particularly impactful in early and intermediate stages, with DSM-generated offspring regularly surpassing those from GA-only operations, consistent with literature identifying the benefits of local exploitation for complex design spaces. Nonetheless, as convergence progressed, the algorithm showed signs of diminishing diversity and local entrapment, as reflected in the clustering of individuals within a single region of parameter space in Figure 11, coloured in blue. This concentration is symptomatic of reduced genome variability, which can stymie further exploitation of alternative minima. The hybrid framework's exploration-exploitation interplay is most clearly highlighted during the first five generations. DSM was capable of rapidly descending to promising local optima, while the broader genetic search provided periodic discoveries of novel attraction basins, events visible as abrupt reductions in the best cost trajectory in Figure 10 blue line. Importantly, once the genetic operations identified a superior region, DSM was able to efficiently refine these new candidates, demonstrating the effectiveness of alternated exploration and exploitation mechanisms.

The stepped HyGO variant draws direct inspiration from the structured approach of Li et al. (2022), introducing an initial five-dimensional subproblem centred only on the actuation speeds. This dimensional reduction is not merely a heuristic for convergence acceleration, but is mathematically motivated: by initially decoupling less influential angular

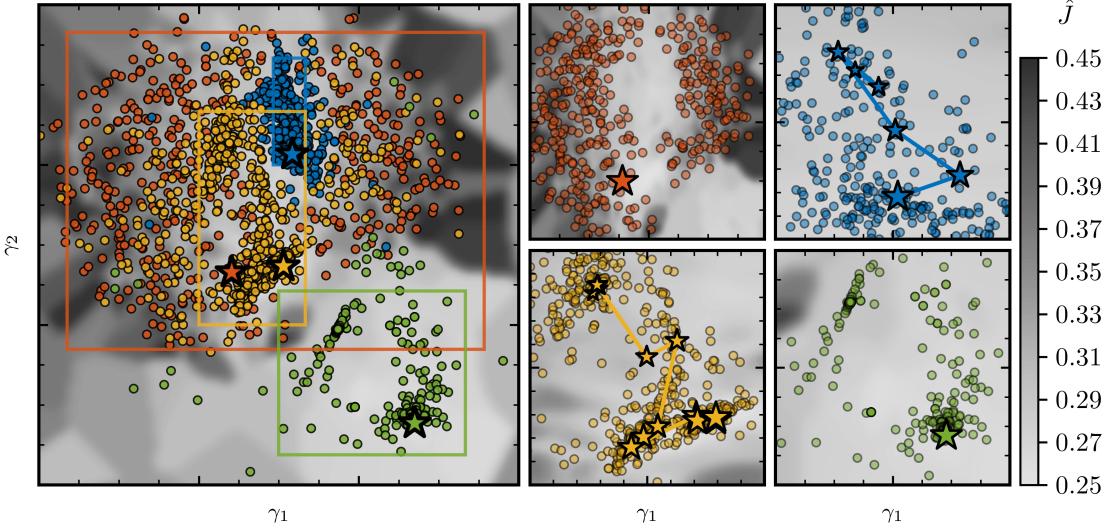


Figure 11: Proximity map of the optimisation parameters for all strategies, coloured by cost. The reduced coordinates are normalised. ● corresponds to GA, ● to HyGO, ● to HyGO-stepped, and ● to EGM. Squares on the general map indicate zoomed-in regions corresponding to each specific optimisation. The 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).

variables, the algorithm reduces the prevalence of spurious local minima linked to variable interactions. The resulting optimisation landscape in this stage has a gentler topology, increasing the probability of successfully locating regions leading to global optima. Following convergence in this subspace, the full parametric search is reinitiated with an affine simplex sampling (Table 5), restoring geometric diversity and maximising potential for effective local exploitation by DSM. The stepped strategy proved decisively more effective, attaining a 20.5% drag reduction, outperforming both the best direct ten-dimensional HyGO and the benchmarked EGM.

Comparative analysis of proximity maps in Figure 11 reveals fundamental differences in search behaviour. The GA's individuals scatter widely (orange), never converging to the most promising region. In contrast, HyGO's direct approach (blue) converges rapidly but at the cost of diversity, with limited ability to escape local traps. Conversely, HyGO-stepped (yellow) achieves a more thorough exploration phase, with early trajectories traversing multiple minima before settling into a lower-cost region. This two-stage protocol leverages both global structure and local refinement, as seen in the sequential contraction of individual spread and ultimate cost function minimisation. Similarly, EGM (green) converges around two distinct local minima, which it exploits efficiently; although its exploration capability is more limited, as indicated by the relatively localised distribution of its individuals in the parameter space.

Examining the control parameters at the identified optima listed in Table 6, the most successful strategies (HyGO-stepped and EGM) are found to employ greater actuation authority, in the form of higher velocity magnitudes, while maintaining similar actuation angles required for targeted flow manipulations. Interestingly, the solution found by HyGO-stepped 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-stepped reverses its direction, potentially offering more direct interfacing with the recirculation bubble. This subtlety may underpin the superior drag reduction achieved and explains the spatial separation between the distinct optima identified by each strategy in the solution proximity map in Figure 11.

In summary, these results elucidate not only the algorithmic advantage conferred by hybrid and staged global-local optimisation, but also provide insight into the physical mechanisms prioritised by algorithmic search: high-authority, tailored actuation strategies emerge as the dominant solution features for robust drag reduction in this geometry.

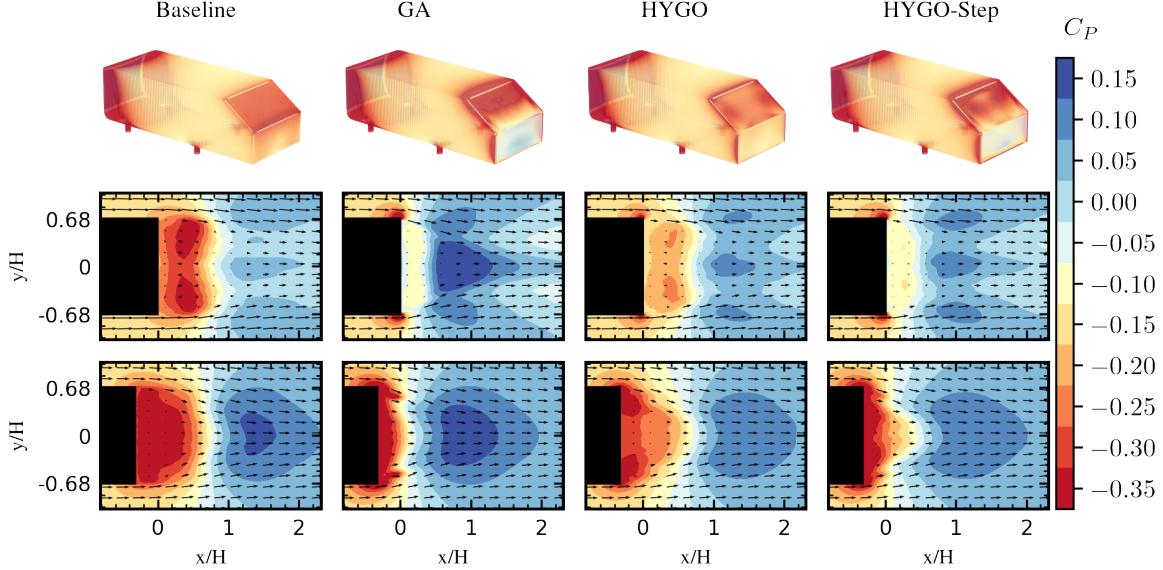


Figure 12: Top view for the surface distribution of the pressure coefficient, C_p , on the Ahmed body. Columns, from left to right, represent the reference (baseline), the solution obtained with the genetic algorithm (GA), HyGO, and HyGO-stepped methodologies. The first row shows the full-surface C_p contours, while the second and third rows display $x - y$ pressure fields at the non-dimensional heights $z/H = 0.28$ and 0.785 , respectively. A common colour scale referenced to the free-stream static pressure is applied to every panel to enable quantitative comparison.

5.3 Flow-physics interpretation

The optimisation exercises not only yield quantitative improvements in drag reduction but also manifest as pronounced alterations to the wake dynamics, as illustrated in Figures 12–14. These figures allow a direct assessment of the flow physics supporting each optimisation outcome, thus assessing the mechanisms via which the algorithms achieve, or fail to achieve, substantive control authority. This figure includes a three-dimensional representation of the pressure distribution (pressure coefficient C_p) on the body's surface, and in the planes highlighted in Figure 9 (d).

A comparative analysis of the pressure coefficient C_p distributions (Figure 12, top view, and Figure 13, side view) reveals that all optimised strategies reduce both the spatial extent and intensity of the recirculation bubble compared to the baseline. The most immediate effect of control, visible in every optimised case, is the forward shift of the pressure recovery region at the rear surface, and a partial or near-complete attenuation of the low-pressure zone characteristic of bluff-body wakes. Notably, the standard GA strongly suppresses the recirculation bubble, resulting in elevated rear-surface pressures and an apparent minimisation of the pressure differential across the body. Nevertheless, this outcome is associated with the creation of concentrated under-pressure regions, particularly along the vertical edges and near the junction of the slanted and vertical rear surfaces (Figure 13). These local deficits are indicative of strong, adverse pressure gradients and intense corner separations, suggesting the GA achieves recirculation suppression through energetically costly, non-cooperative actuation in which large jet velocities (notably U_4) are directed inwards. The resultant flow closely resembles a geometric boat-tailing, 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.

By contrast, the direct HyGO optimum pursues a more nuanced wake reorganisation, displacing but not eliminating the recirculation zone as shown in Figure 12 and Figure 13. Moderate actuation magnitudes allow the first and second jets to focus on stabilising the boundary layer and delaying its separation, resulting in expanded pressure recovery zones without the energetic penalties of the GA's solution. The angular distribution of the jets, with a reversed third actuator (compared to GA), produces a controlled deflection of the vortex core downstream, as highlighted by the diminished vorticity concentrations at the vertical edges (Figure 14). This leads to a recirculation structure of reduced area and strength, as well as a less severe pressure gradient across the wake. Interestingly, the pressure contours and velocity

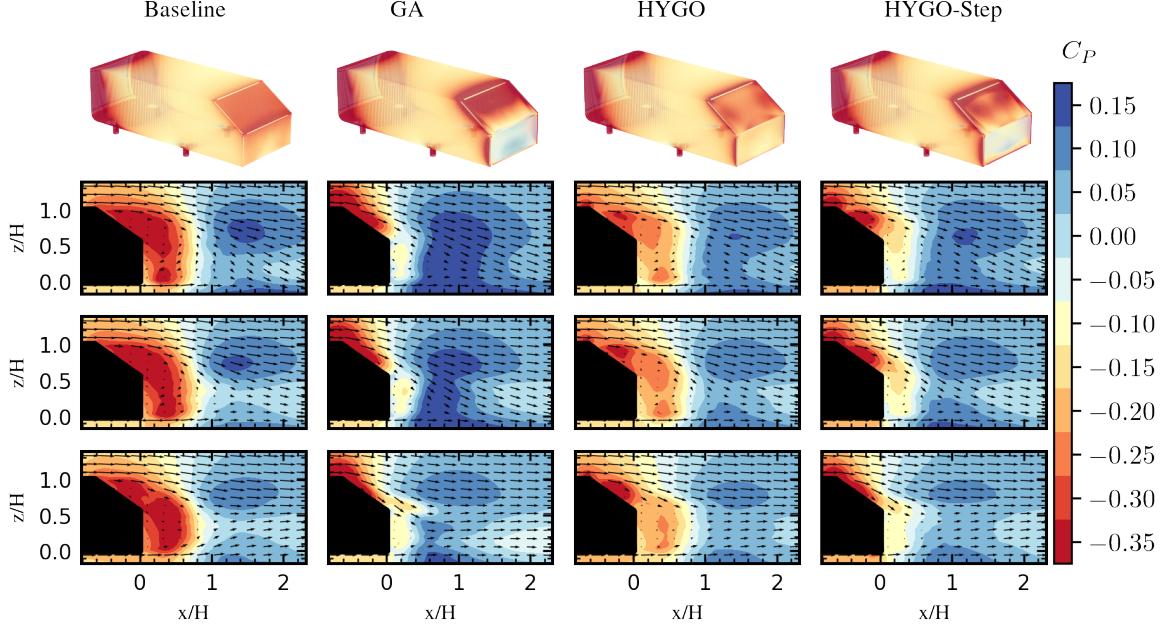


Figure 13: Side view for the surface distribution of the pressure coefficient, C_P , on the Ahmed body. Columns, from left to right, represent the reference (baseline), the solution obtained with the genetic algorithm (GA), HyGO, and HyGO-stepped methodologies. The first row shows the full-surface C_P contours, while the second, third, and fourth rows present $x-z$ pressure fields at the spanwise stations $y/H = 0, 0.225$, and 0.45 , respectively. A common colour scale referenced to the free-stream static pressure is applied to every panel to enable quantitative comparison.

fields suggest that counter-rotating vortices drive separation at the lateral edges of the slanted surface, a configuration that appears to limit the lateral spread of separation and thereby reduces overall drag contributions.

The stepped-HyGO approach synthesises the advantageous features of both preceding strategies. Figure 12 demonstrates that this configuration partially suppresses the recirculation bubble, similar to GA, while also incorporating the separation control mechanisms observed in HyGO. This results in 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. Importantly, the stepped solution confines separation predominantly to the near-wake of the slanted surface, avoiding the broad, persistent vortices of the GA strategy and the residual core retained in HyGO’s solution. The optimised actuation parameters (Table 4) reflect a compromise: U_4 is sufficiently large to exert decisive control over bubble location yet not so dominant as to induce spurious separations, thus facilitating a cooperative action amongst all jets, mainly allowing U_1 and U_2 to play a more active role in controlling separation. This cooperative mechanism is further corroborated by the streamwise vorticity fields in Figure 14, which show that in the stepped configuration, both main and secondary vortices are moderated, and their coherence is disrupted, promoting earlier wake recovery and enhanced pressure reattachment. Such disruption is likely aided by the appearance of secondary, oppositely signed vortex structures between the primary pairs, a feature more prominent in both HyGO solutions.

The transition from brute-force bubble suppression (GA) to judicious wake restructuring (HyGO and, particularly, HyGO-Stepped) is thus made explicit in these comparative flow visualisations. While the GA achieves apparent bubble elimination, its approach incurs a penalty in the form of intense, spatially localised separations that limit its net drag reduction potential and produce more intense and long-lived vortices due to the aggressive separation induced by strong actuation (Figure 14). The direct HyGO solution, although not eradicating the bubble, uses lower actuation velocities and precision in jet orientation to yield a more uniform, energetically efficient wake. This leads to a smoother wake profile with less lateral momentum exchange, aligning with the observed lower drag values. Finally, the stepped strategy capitalises on both bubble mitigation and wake tailoring, achieving the largest observed drag reduction through a balance of spatially broad separation control and mitigated vortex amplification.

From the standpoint of optimisation, these results offer concrete evidence that the algorithmic decisions encoded by HyGO and its stepped variant (namely, the orchestration of actuation velocities and orientations, and the sequential refinement of parameter subspaces) result in more optimal cooperative control regimes. The consequential flow

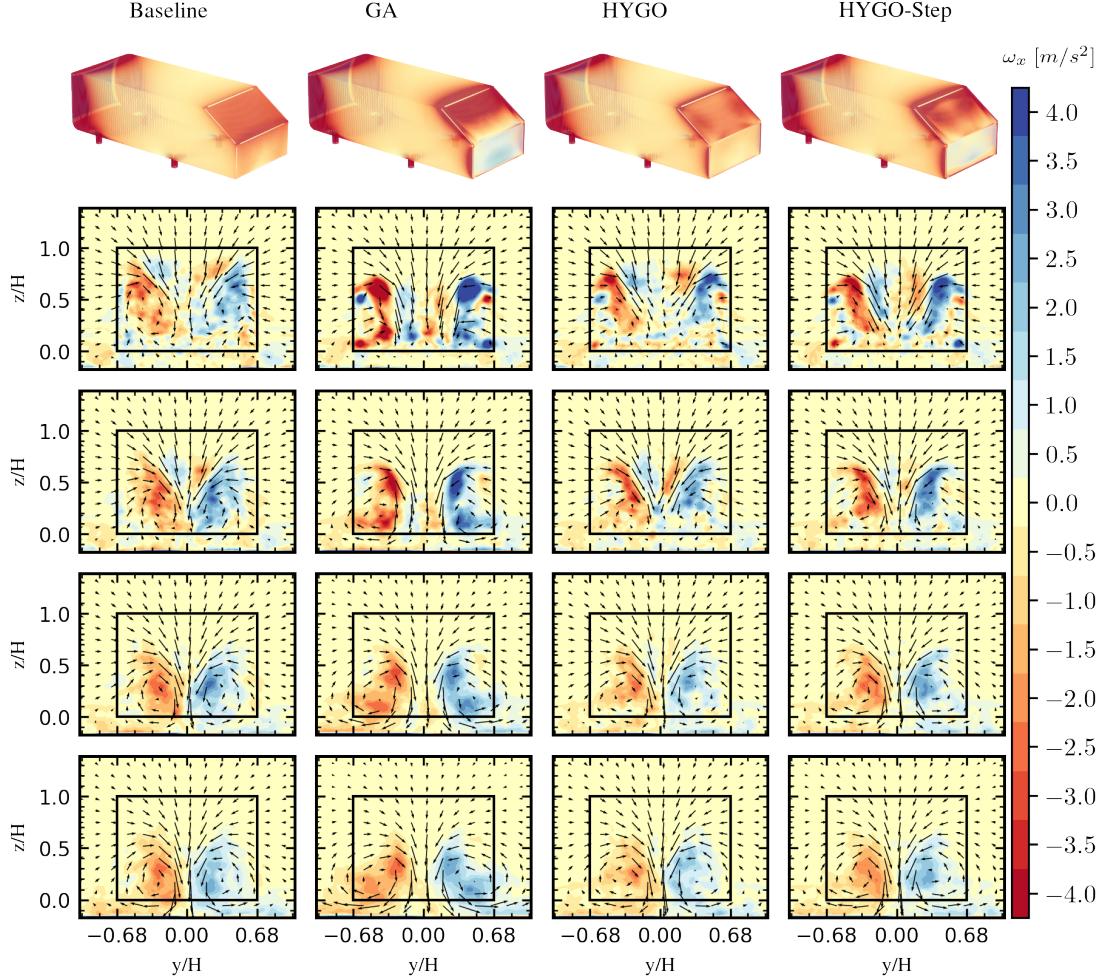


Figure 14: Streamwise vorticity maps for the fittest individuals of each optimisation approach compared to the baseline (no actuation) case at fixed streamwise distances, x/H , from the Ahmed body. The highlighted **black** rectangle represents the Ahmed body’s base profile. The first row shows the pressure coefficient distribution on the surface of the Ahmed body in the same scale as Figure 13. The following rows display the streamwise vorticity maps at $x/H = 0.5, 1.0, 1.5$, and 2.0 , respectively. A consistent colour scale is applied across all vorticity maps for clear comparison of the flow structures.

structures are physically distinct: improved pressure recovery, limited adverse gradients, and subdued, fragmented vortical patterns. The importance of deliberately staged actuation is thereby affirmed, with clear manifestations in both global cost reduction and flow-field topology. These insights clarify that algorithmic diversity management and staged exploitation do not merely accelerate convergence, but engender physically superior, more robust optimal states in complex fluid–structure control scenarios.

6 Conclusions

This work presents the Hybrid Genetic Optimisation framework, HyGO, as a robust, extensible solution for high-dimensional and complex global optimisation. By explicitly hybridising evolutionary algorithms with local refinement, specifically, a degeneracy-proof DSM, HyGO bridges global exploration with accelerated, reliable local exploitation within a unified, modular framework.

A cornerstone of the methodology is the custom degeneracy-proof extension to DSM. This innovation dynamically detects and corrects simplex degeneracies, ensuring that, even in high-dimensional and ill-conditioned search spaces, the simplex retains full rank and is able to efficiently navigate towards optima rather than becoming trapped in lower-

dimensional subspaces. This addresses a common and critical failure mode of conventional simplex approaches, greatly improving robustness and convergence, especially as dimensionality increases.

The proposed framework supports both binary-encoded GA and functional (LGP) representations. On a suite of analytical benchmark functions, ranging up to 25 dimensions and encapsulating both multimodal and ill-posed topologies, HyGO demonstrates consistent advantages in both convergence speed and solution quality versus classical GAs. These results were statistically robust across 50-fold repetitions and in direct comparison against state-of-the-art hybrid evolutionary algorithms, including CMA-ES, in high-dimensional settings. The exploitation component (DSM) imparts clear efficiency gains, particularly when the search landscape exhibits strong non-separability and premature convergence.

For functional learning, HyGO with LGP-DSM was demonstrated on the stabilisation of the nonlinear, time-dependent Landau oscillator, a canonical control benchmark. Here, the hybrid method not only accelerated discovery and refinement of stabilising control laws, but systematically yielded more efficient (sparser, energy-minimising) solutions, outperforming pure LGP in both speed and achievable cost. This result highlights the utility of the LGP-DSM hybrid for automated design of interpretable control policies in nonlinear dynamical systems.

The framework's efficacy was further established on an engineering task: simulation-based active flow control for drag reduction of an Ahmed body. In this computationally expensive, highly nonlinear and ten-dimensional optimisation problem, HyGO achieved robust and physically interpretable actuation strategies, realising over 20% mean drag reduction. The stepped-HyGO approach, optimising actuation velocities before full vector expansion to include angles with tailored simplex initialisation, proved critical for escaping local traps and achieving global performance. In direct, fair comparison (identical simulation setup and evaluation budget), HyGO decisively outperformed the EGM and classical GAs on this benchmark, validating its real-world competitiveness and broad applicability.

Overall, HyGO offers a computationally efficient, easily generalisable approach to high-dimensional global optimisation. The framework's degeneracy-proof local search, staged hybridisation protocols, and strong empirical validation on both canonical and engineering benchmarks position it as a valuable tool for a wide array of scientific and computational mechanics applications. Future work may further expand HyGO's reach by incorporating alternate local refinement schemes or adaptive exploration-exploitation scheduling, enabling even broader applicability across emerging automation and design challenges in computational engineering.

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 HyGO framework is openly available to the research and engineering community under the MIT license, including source code, documentation, and ready-to-use examples. HYGO can be installed directly from the Python Package Index (PyPI) at pypi.org/project/HYGO/ by executing `pip install HYGO`. In addition, the full development repository is hosted on GitHub at github.com/ipatazas/HYGO, including a suite of example scripts illustrating the usage of HyGO for both parametric benchmark functions and control law optimisation, including the stabilisation of the Landau oscillator.

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 line “Incentive for Research of Young Doctors” of the Pluriannual Agreement with the University Carlos III of Madrid (SOLMETAI-CM-UC3M), within the framework of the 6th Regional Plan for Scientific Research and Technological Innovation (VI PRICIT). The authors thank Prof. Discetti, A. Solera-Rico 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

- J. S. Arora, O. A. Elwakeil, A. I. Chahande, C. C. Hsieh, Global optimization methods for engineering applications: A review, volume 9, 1995.
- J. Li, X. Du, J. R. Martins, Machine learning in aerodynamic shape optimization, *Progress in Aerospace Sciences* 134 (2022) 100849.
- 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.
- S. L. Brunton, B. R. Noack, Closed-loop turbulence control: Progress and challenges, *Applied Mechanics Reviews* 67 (2015).
- D. Rocke, Genetic algorithms + data structures = evolution programs, *Journal of the American Statistical Association* 95 (2000) 347–348.
- D. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison Wesley series in artificial intelligence, Addison-Wesley, 1989.
- 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.
- R. Castellanos, A. Ianiro, S. Discetti, Genetically-inspired convective heat transfer enhancement in a turbulent boundary layer, *Applied Thermal Engineering* 230 (2023) 120621.
- 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.
- 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.
- A. Kumar, N. Sinha, A. Bhardwaj, A novel fitness function in genetic programming for medical data classification, *Journal of Biomedical Informatics* 112 (2020) 103623.
- 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.
- 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).
- 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 (2017a) 1–20.
- 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* (2017b).
- I. Tsoulos, I. Lagaris, Solving differential equations with genetic programming, *Genetic Programming and Evolvable Machines* 7 (2006) 33–54.
- A. Sobester, P. Nair, A. Keane, Genetic programming approaches for solving elliptic partial differential equations, *IEEE Trans. Evolutionary Computation* 12 (2008) 469–478.
- 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.
- Z. Wang, X. Zhang, Z. Zhang, D. Sheng, Credit portfolio optimization: A multi-objective genetic algorithm approach, *Borsa Istanbul Review* 22 (2022) 69–76.
- J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*, The MIT Press, 1992.
- G. Syswerda, Uniform crossover in genetic algorithms, in: *ICGA*, 1989.

- 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.
- K. Deb, R. B. Agrawal, Simulated binary crossover for continuous search space, *Complex Syst.* 9 (1995).
- K. Deb, M. Goyal, et al., A combined genetic adaptive search (geneas) for engineering design, *Computer Science and informatics* 26 (1996) 30–45.
- H. Beyer, *The Theory of Evolution Strategies*, Natural Computing Series, Springer, 2001.
- 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.
- G. Syswerda, Schedule optimization using genetic algorithms, *Handbook of genetic algorithms* (1991).
- J. Nocedal, S. Wright, *Numerical Optimization*, Springer Series in Operations Research and Financial Engineering, Springer New York, 2006.
- H. Hoos, T. Stützle, *Stochastic Local Search: Foundations & Applications*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004.
- D. Adler, Genetic algorithms and simulated annealing: A marriage proposal, in: *IEEE international conference on neural networks*, IEEE, 1993, pp. 1104–1109.
- S. Xu, Y. Wang, Parameter estimation of photovoltaic modules using a hybrid flower pollination algorithm, *Energy Conversion and Management* 144 (2017) 53–68.
- R. Yang, I. Douglas, Simple genetic algorithm with local tuning: Efficient global optimizing technique, *Journal of Optimization Theory and Applications* 98 (1998) 449–465.
- 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.
- 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.
- 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.
- 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.
- 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.
- 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.
- M. Brameier, W. Banzhaf, *Linear Genetic Programming*, Genetic and Evolutionary Computation, Springer US, 2007.
- Z. Michalewicz, C. Z. Janikow, Handling constraints in genetic algorithms., in: *Icga*, volume 398, 1991, pp. 151–157.
- A. Homaifar, C. X. Qi, S. H. Lai, Constrained optimization via genetic algorithms, *Simulation* 62 (1994) 242–253.
- T. Tarkowski, Genetic algorithm formulation and tuning with use of test functions, ArXiv abs/2210.03217 (2022).
- J. A. Nelder, R. Mead, A Simplex Method for Function Minimization, *The Computer Journal* 7 (1965) 308–313.
- T. Rowan, The subplex method for unconstrained optimization, PhD thesis, Department of Computer Sciences, Univ. of Texas (1990).
- N. Hansen, A. Ostermeier, Completely Derandomized Self-Adaptation in Evolution Strategies, *Evolutionary Computation* 9 (2001) 159–195.
- 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.
- 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.
- E. Serre, M. Minguez, R. Pasquetti, E. Guilmeneau, 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. LES of turbulence aeroacoustics and combustion.