

DGA: A DIFFERENTIAL RELAXATION OF GENETIC ALGORITHMS FOR RECEDING-HORIZON CONTROL OF NONLINEAR SYSTEMS

Ali Deeb

Higher School of Cyberphysical
Systems and Control
Peter the Great St. Petersburg
Polytechnic University
Saint Petersburg, Russian Federation
dib_a@spbstu.ru

Vladimir Khokhlovskiy

Higher School of Cyberphysical
Systems and Control
Peter the Great St. Petersburg
Polytechnic University
Saint Petersburg, Russian Federation
hohlovskij_vn@spbstu.ru

Viacheslav Shkodyrev

Higher School of Cyberphysical
Systems and Control
Peter the Great St. Petersburg
Polytechnic University
Saint Petersburg, Russian Federation
shkodyrev@spbstu.ru

Article history:

Received 25.02.2026, Accepted 19.07.2026

Abstract

The growing complexity of industrial cyber-physical systems demands control strategies capable of handling strong nonlinearities, tight actuator constraints, and competing objectives under real-time requirements. Although Genetic Algorithms (GAs) are widely used in this class of problems, their non-differentiable operators prevent integration with modern gradient-based and decision-aware learning frameworks.

This work proposes a Differentiable Genetic Algorithm (DGA) formulated as a smooth relaxation of a real-coded GA with simulated binary crossover and polynomial mutation, replacing hard components with distributional representations to enable gradient propagation through the evolutionary search process while preserving population-based exploration.

The DGA layer is embedded within a receding-horizon control framework and evaluated on a nonlinear boiler drum–turbine unit with actuator magnitude and rate constraints. Results show improved tracking accuracy, smoother control signals, and balanced constraint activity compared to classical and rejection-based GA-MPC variants. Computational benchmarking further demonstrates that the proposed distributional relaxation achieves comparable solution quality to EvoGrad while substantially reducing runtime. These results show that evolutionary optimization and differentiable learning can be unified within a coherent framework for constrained nonlinear control.

Key words

genetic algorithms, model predictive control, differentiable evaluation, process control, AI-driven control systems, boiler–turbine units.

1 Introduction

The need for effective control strategies for complex industrial systems continues to drive research toward increasingly sophisticated methodologies. The implementation of such strategies in real-time settings remains a significant research challenge. A large class of cyber-physical industrial systems fall under this description as they operate under a complex combination of physical constraints, safety margins, and competing performance objectives [Babich et al., 2025].

Boiler–turbine units, thermal power plants, and similar large-scale systems lie in this domain, as they exhibit strong nonlinearities, multivariable coupling, and regime-dependent dynamics while being required to track demand and satisfy safety constraints simultaneously [International Energy Agency, 2022, Kumar and Singh, 2022, Wang et al., 2018]. Related recent studies have also considered numerical optimization and learning-based optimal control for complex physical systems, including thermodynamic separation processes and PDE-constrained reservoir systems [Arguchintsev and Kopylov, 2024, Sirota and Ovsyannikov, 2025]. These properties make purely single-objective or unconstrained formulations inadequate for practical control.

State of the art works have investigated Model Predictive Control (MPC) as a promising solution in this field, as it provides a principled framework for handling constraints and multi-step decision making and is widely adopted in industrial practice [Ogata, 2020, Schwenzer et al., 2021]. However, for strongly nonlinear or simulation-based plants, real-time feasibility often relies on model simplifications such as local linearization, gain

scheduling, or piecewise and fuzzy representations [Wu et al., 2011, Wu et al., 2014, Ławryńczuk, 2017, Yang et al., 2022, Saini and Ohri, 2023]. While effective in many cases, these approximations introduce modelling error and limit expressiveness when operating conditions change. As a result, there has been sustained interest in evolutionary and learning-based optimization methods that can better cope with nonconvexity, uncertainty, and complex constraints [Slowik and Kwasnicka, 2020, Banzhaf et al., 2023, Lawrence et al., 2024, Deeb et al., 2024]. The main obstacle to their widespread adoption in closed-loop control remains computational cost and the difficulty of integrating them coherently with gradient-based learning pipelines.

A parallel and increasingly influential research direction is decision-aware learning, where downstream optimization or control objectives are embedded directly into the training loop of predictive models. Recent advances in differentiable optimization layers allow convex programs, fixed-point problems, and even implicit solvers to be treated as neural network layers with end-to-end backpropagation [Agrawal et al., 2019, Blondel et al., 2022, Bai et al., 2019, Amos et al., 2018, Rybalko and Fradkov, 2023]. These tools enable predictors to be trained not only for accuracy, but for the quality of the decisions they induce. However, existing mature toolchains largely rely on convexity, smoothness, or fixed-point structure. Many control problems of interest remain nonconvex, nonsmooth, or combinatorial, especially when hard constraints, discrete decisions, or complex simulators are involved.

Genetic Algorithms (GAs) and their multiobjective variants have long been used in such settings because of their robustness to nonconvexity, black-box objectives, and constraint-heavy formulations [Deb et al., 2002, Katoch et al., 2021]. In control and energy systems, GAs are routinely employed to optimize control sequences, tune MPC parameters, or approximate Pareto fronts in multiobjective formulations [Cavus and Allahham, 2024, Aguilar et al., 2024, Cordeiro-Costas et al., 2024]. Despite their success, these methods are typically deployed outside the gradient pipeline: predictive models are trained independently, and evolutionary optimization is applied downstream at inference time or in an outer loop. This separation creates a structural mismatch with decision-aware learning, preventing gradients from flowing through the optimization process and limiting the ability to train upstream models for downstream decision quality.

At the same time, recent work in differentiable evolutionary computation demonstrates that this separation is not fundamental. Differentiable genetic programming relaxes discrete program structures into continuous representations [Izzo and Biscani, 2020, Zeng et al., 2023], showing that evolutionary operators (such as selection, mutation, and crossover) can be reformulated in smooth or probabilistic forms, enabling end-to-end training while retaining evolutionary inductive biases. More-

over, EvoGrad [Citterio and Tangherloni, 2025] reformulates standard GA operators into differentiable operations. However, to date, there is no widely adopted, peer-reviewed framework that embeds a standard real-coded genetic algorithm with SBX crossover as a reusable, differentiable neural network layer comparable in scope to OptNet or DEQ-style layers.

This paper addresses that gap by introducing a *Differentiable Genetic Algorithm (DGA) layer* built as a distributional relaxation inspired by real-coded genetic algorithms with simulated binary crossover (SBX) and polynomial mutation. The proposed layer preserves key strengths of GAs—population-based exploration, robustness to nonconvexity, and natural handling of constraints—while exposing smooth Jacobians that enable gradient-based training of upstream models. In contrast to classical GA-in-the-loop pipelines, the Diff-GA layer operates as an integral component of the computational graph, allowing predictors to be trained directly for downstream decision performance without surrogate losses or hand-designed scalarizations.

From the physical perspective, the proposed framework is relevant to the dynamics and control of nonlinear systems. Boiler drum-turbine units are characterized by strongly coupled thermal, hydraulic, and mechanical processes whose transient behavior is constrained by physical limitations on actuators and admissible operating conditions. The results demonstrate that a differentiable distributional optimization mechanism can coordinate the evolution of coupled physical variables while reducing undesirable control oscillations and maintaining admissible actuator dynamics. In this sense, the work contributes to computational and applied physics, particularly to nonlinear thermodynamic systems, energy-conversion dynamics, and physics-constrained control.

The contributions of this work are as follows:

1. We propose a differentiable, distributional relaxation inspired by standard real-coded Genetic Algorithms with SBX crossover and polynomial mutation, formulated as a reusable optimization layer compatible with backpropagation.
2. Unlike differentiable genetic programming and recent evolutionary relaxations, the proposed DGA operates directly on real-coded control sequences and is tailored for receding-horizon optimization under hard actuator constraints.
3. In contrast to EvoGrad [Citterio and Tangherloni, 2025], which differentiates classical evolutionary operators through continuous relaxations, the proposed method adopts a distributional population representation that removes operator-level loops and enables fully vectorized updates, substantially reducing computational overhead.
4. We embed the DGA layer inside a constrained MPC framework and demonstrate improved closed-loop tracking performance on a nonlinear boiler drum-turbine unit compared to classical GA-MPC

and rejection-based GA variants.

The remainder of the paper is organized as follows. Section 2 reviews related work on differentiable optimization layers, evolutionary computation, and decision-aware learning. Section 3 formulates the constrained MPC problem and introduces the proposed Differentiable Genetic Algorithm layer. Section 4 presents closed-loop control results and computational benchmarks, including comparison with EvoGrad. Finally, Section 5 concludes the paper and outlines future research directions.

2 Related Work

2.1 Motivation and previous work

A major trend in recent years is to embed optimization problems directly as layers inside neural networks. For example, representative systems include differentiable convex optimization layers that expose differentials through solvers [Agrawal et al., 2019, Blondel et al., 2022]. Fixed-point layers such as Deep Equilibrium Models (DEQ) [Bai et al., 2019], and application-specific controllers like differentiable MPC [Amos et al., 2018] serve as well established differential solvers. Recent work pushes efficiency by decoupling forward optimization from backward sensitivity—e.g., reformulating the backward pass itself as a QP (BPQP) [Pan et al., 2024].

In parallel, Genetic Algorithms (GAs) remain competitive in nonconvex, discrete, and multi-criteria control and optimization problems in energy fields. Recent studies integrate GAs with predictive models to build optimization policies [Aguilar et al., 2024, Cordeiro-Costas et al., 2024], and with MPC to address nonlinearity/nonconvexity in microgrids [Cavus and Allahham, 2024]. For broader OR-style optimization, design, and scheduling, GA variants continue to evolve, covering a large set of applications and hybridizations [Londe et al., 2025]. Comprehensive surveys also track operator design and hybrid GA directions [Katoch et al., 2021]. These decision-layer uses of GA are typically outside the gradient pipeline, where predictions are trained with standard losses, and a GA runs downstream at inference or in an outer loop.

Another direction of using GAs with NNs is neuroevolution. Where EAs are being used to train networks. The most relevant line is differentiable genetic programming: dCGP provides analytic/automatic differentiation through Cartesian GP graphs [Izzo and Biscani, 2020], and newer differentiable GP (DGP) relaxes symbolic trees into continuous representations optimized by gradient methods before resampling [Zeng et al., 2023]. These show that (i) discrete evolutionary search spaces can be relaxed; (ii) selection/mutation/crossover can be made gradient-compatible with probabilistic parameterizations; and (iii) end-to-end training can benefit from

evolutionary inductive biases without breaking autodiff. However, we find no widely adopted, peer-reviewed work that embeds a standard real-coded GA with SBX crossover as a reusable, differentiable neural layer akin to OptNet/DEQ. This gap motivates our differentiable-GA layer built on distributional relaxation inspired by real-coded operators such as SBX and polynomial mutation, implemented through reparameterized sampling and expected-offspring updates.

2.2 Why a Differentiable SBX-GA Layer

First, many decision problems driving the loss are nonconvex, nonsmooth, or combinatorial; convex layers may need surrogates or relaxations that distort the task loss. A differentiable GA layer preserves GA’s global, population-based exploration (diversity, multi-modality) while providing gradients for upstream predictors. Second, implicit/fixed-point tooling and efficient backward passes [Blondel et al., 2022, Pan et al., 2024] suggest practical templates for memory-efficient Jacobian-vector products through complex solvers. Third, continuous real-coded operators (e.g., SBX and polynomial mutation) admit reparameterizations that are differentiable w.r.t. parents and hyperparameters almost everywhere, enabling stable end-to-end training when paired with soft selection (expected-offspring) rather than hard tournament/rank picks. Finally, the control/energy literature’s persistent reliance on GA for real-world constraints [Cavus and Allahham, 2024, Aguilar et al., 2024, Cordeiro-Costas et al., 2024] indicates immediate application domains where a GA layer could unify prediction and downstream decision quality.

Our work sits at the intersection of optimization layers and evolutionary search: like OptNet/DEQ, it is a differentiable layer with implicit/expected gradients; like modern GA practice, it handles nonconvex, multiobjective, and constraint-heavy settings. Unlike prior GA-in-the-loop pipelines (decision-focused but non-differentiable), our layer exposes gradients through SBX and selection expectations, enabling decision-aware training without surrogate losses or non-differentiable control flow.

Recent work such as EvoGrad reformulates classical evolutionary operators into differentiable counterparts, enabling gradient-based optimization through selection, crossover, and mutation. While effective, these approaches retain operator-level transformations and per-individual evolutionary steps. In contrast, the proposed DGA adopts a distributional representation of the population, replacing explicit crossover and mutation operators with parameterized sampling and smooth refinement. This formulation eliminates discrete operator flow at the structural level and leads to a fully vectorized implementation, which significantly reduces computational overhead while preserving evolutionary exploration behavior.

Algorithm 1 Genetic Algorithm for MPC Inner Optimization

Require: Current state x_t , population size N_p , number of generations S

Ensure: Approximate optimal control sequence U_t^*

```

1: Initialize population  $\mathcal{P}_t^{(0)} \subset \mathcal{U}$ 
2: for  $s = 1$  to  $S$  do
3:   for each individual  $U_t^{(i)} \in \mathcal{P}_t^{(s-1)}$  do
4:     Evaluate cost  $J^{(i)} = J(U_t^{(i)}; x_t)$ 
5:   end for
6:   Select parent individuals based on fitness
7:   Apply crossover to generate offspring
8:   Apply mutation to offspring
9:   Project offspring onto admissible set  $\mathcal{U}$ 
10:  Form new population  $\mathcal{P}_t^{(s)}$ 
11: end for
12: return
```

$$U_t^* = \arg \min_{U_t^{(i)} \in \mathcal{P}_t^{(S)}} J(U_t^{(i)}; x_t)$$

3 Problem Formulation

Consider a general discrete-time plant:

$$\begin{aligned} x_{t+1} &= f(x_t, u_t) \\ y_t &= g(x_t, u_t) \end{aligned} \quad (1)$$

where:

$$x_t \in \mathbb{R}^{n_x}, \quad u_t \in \mathbb{R}^{n_u}, \quad y_t \in \mathbb{R}^{n_y}. \quad (2)$$

At time t , MPC optimizes a finite-horizon control sequence:

$$U_t \triangleq \{u_{t|t}, u_{t+1|t}, \dots, u_{t+H-1|t}\} \in \mathbb{R}^{Hn_u} \quad (3)$$

with predicted states:

$$\begin{aligned} x_{t|t} &= x_t \\ x_{t+k+1|t} &= f(x_{t+k|t}, u_{t+k|t}) \\ y_{t+k|t} &= g(x_{t+k|t}, u_{t+k|t}) \end{aligned} \quad (4)$$

For tracking control problem, we aim to minimize the cost function:

$$\min_{U_t} J(U_t; x_t) \quad (5)$$

where

$$\begin{aligned} J(U_t; x_t) &= \sum_{k=0}^{H-1} \gamma^k \ell(y_{t+k|t}, r_{t+k}, u_{t+k|t}) \\ &\quad + \ell_f(y_{t+H|t}, r_{t+H}) \end{aligned} \quad (6)$$

A common choice for the cost function is a weighted quadratic tracking with effort penalty:

$$\begin{aligned} \ell(y, r, u) &= (y - r)^\top Q (y - r) + u^\top R u, \\ \ell_f(y, r) &= (y - r)^\top Q_f (y - r), \end{aligned} \quad (7)$$

with:

$$Q \succeq 0, \quad Q_f \succeq 0, \quad R \succ 0, \quad \gamma \in (0, 1], \quad (8)$$

and:

$$\begin{aligned} u_{\min} &\leq u_{t+k|t} \leq u_{\max} \\ x_{\min} &\leq x_{t+k|t} \leq x_{\max}. \end{aligned} \quad (9)$$

After solving, MPC applies only the first input: $u_t = u_{t|t}^*$ and repeats at the next time step.

3.1 GA-Based MPC

For nonlinear, nonconvex, or simulator-based plants, the MPC inner optimization problem

$$\min_{U_t \in \mathcal{U}} J(U_t; x_t) \quad (10)$$

cannot generally be solved using analytical or convex optimization techniques. In such cases, a genetic algorithm (GA) is commonly employed to obtain an approximate solution.

At each time step t , the GA maintains a population of candidate control sequences

$$\mathcal{P}_t = \{U_t^{(i)}\}_{i=1}^{N_p}, \quad (11)$$

where each individual represents a feasible control trajectory over the prediction horizon. The cost of each candidate is evaluated through forward simulation of the system dynamics,

$$J^{(i)} = J(U_t^{(i)}; x_t), \quad (12)$$

and a fitness value is defined as a monotonic transformation of the cost $F^{(i)} = -J^{(i)}$.

The population is iteratively evolved using selection, crossover, and mutation operators. Feasibility with respect to input and state constraints is enforced by projection onto the admissible set $\mathcal{U}$. The procedure is summarized in Algorithm 1.

The GA-based MPC solver is derivative-free and suitable for highly nonlinear or black-box systems. However, key operations such as selection, ranking, and population replacement are inherently discrete and non-differentiable, which prevents gradient propagation through the solver and limits its use in end-to-end learning frameworks.

3.2 DGA: Differentiable Relaxation of Genetic Algorithms for Control

To enable end-to-end learning while retaining the population-based structure of genetic algorithms, we introduce a *Differentiable Genetic Algorithm* (DGA). The key idea is to replace discrete selection and population updates with smooth, distribution-based operations that admit gradient propagation, while still allowing cost evaluation through a black-box simulator. While the distributional update resembles Estimation of Distribution Algorithms (EDA), the proposed DGA is embedded within a differentiable decision-aware learning framework and optimized through gradient-based updates of the initializer parameters, rather than relying solely on moment estimation from elite samples.

3.2.1 Distributional Representation of the Population Instead of maintaining an explicit population evolved by discrete operators, DGA represents the population implicitly through a parametric probability distribution over control sequences conditioned on the current state:

$$p_\theta(U | x_t). \quad (13)$$

In this work, we adopt a factorized Gaussian distribution whose mean is generated by a learnable initializer network:

$$p_\theta(U | x_t) = \mathcal{N}(U; \mu_\theta(x_t), \Sigma), \quad \Sigma = \sigma^2 I. \quad (14)$$

Moreover, a population of candidate sequences is obtained with the following reparameterization:

$$U^{(i)} = \mu_\theta(x_t) + \sigma \epsilon^{(i)}, \quad \epsilon^{(i)} \sim \mathcal{N}(0, I), \quad (15)$$

followed by projection onto the admissible input set:

$$\tilde{U}^{(i)} = \Pi_{\mathcal{U}}(U^{(i)}), \quad (16)$$

where $\Pi_{\mathcal{U}}$ denotes element-wise clipping to enforce hard constraints.

3.2.2 Soft Selection Classical GA selection relies on ranking and discrete sampling, which blocks gradient flow. In DGA, we replace this operation with a smooth weighting mechanism based on the evaluated costs:

$$w^{(i)} = \frac{\exp(-J^{(i)}/\tau)}{\sum_{j=1}^{N_p} \exp(-J^{(j)}/\tau)}, \quad (17)$$

where $\tau > 0$ controls the selection pressure. These weights define a differentiable surrogate of selection and allow the construction of a soft population representative,

$$\bar{U} = \sum_{i=1}^{N_p} w^{(i)} \tilde{U}^{(i)}. \quad (18)$$

3.2.3 Differentiable Population Refinement To emulate the effect of GAs' selection and recombination, DGA performs a smooth update of the distribution mean as:

$$\mu \leftarrow (1 - \alpha)\mu + \alpha \bar{U}, \quad (19)$$

where $\alpha \in (0, 1]$ is a refinement step size.

A new population is, then, regenerated around the updated mean:

$$\tilde{U}^{(i)} \leftarrow \Pi_{\mathcal{U}}(\mu + \sigma \xi^{(i)}), \quad \xi^{(i)} \sim \mathcal{N}(0, I), \quad (20)$$

and the procedure is repeated for a fixed number of DGA iterations S .

After S refinement steps, the optimized control sequence can be obtained either as the soft decision $\bar{U}$:

$$i^* = \arg \min_{i \in \{1, \dots, N_p\}} J^{(i)}, \quad U_t^* = \tilde{U}^{(i^*)}. \quad (21)$$

3.2.4 The trainable initializer Gradients with respect to the initializer parameters θ are computed through the sampling distribution. Using the score-function estimator:

$$\begin{aligned} \nabla_\theta \mathbb{E}_{U \sim p_\theta(\cdot | x_t)} [J(U; x_t)] &= \\ \mathbb{E}_{U \sim p_\theta(\cdot | x_t)} [J(U; x_t) \nabla_\theta \log p_\theta(U | x_t)]. \end{aligned} \quad (22)$$

With N_p samples and a baseline b , a practical estimator is

$$\nabla_\theta \mathcal{L}(\theta) = \frac{1}{N_p} \sum_{i=1}^{N_p} (J^{(i)} - b) \nabla_\theta \log p_\theta(\tilde{U}^{(i)} | x_t). \quad (23)$$

For the Gaussian parameterization, the log-probability decomposes as

$$\begin{aligned} \log p_\theta(\tilde{U}^{(i)} | x_t) &= \\ \sum_{k=0}^{H-1} \sum_{m=1}^{n_u} \log \mathcal{N}(\tilde{u}_{t+k|t,m}^{(i)}; \mu_{\theta,t+k|t,m}(x_t), \sigma^2). \end{aligned} \quad (24)$$

3.2.5 Rejection-Based Filtering To approximate discrete elitism and reduce the computational cost associated with expensive objective evaluations, the DGA framework incorporates a rejection-based filtering mechanism. This component is adopted from our prior work on rejection-based genetic algorithms (RB-GA) [Deeb et al., 2025], where it was shown to consistently improve convergence and tracking performance in receding-horizon control problems.

Within the DGA formulation, the rejection-based filter serves as a learned surrogate for hard elitist selection.

Its output is used to guide population refinement and fitness shaping without introducing non-differentiable operations into the core algorithm.

It is important to note that, unlike the rejection-based GA proposed in [Deeb et al., 2025], the present DGA does not rely on discrete offspring filtering as its core mechanism. Instead, rejection is incorporated as an auxiliary refinement component within an otherwise fully differentiable population update scheme.

The proposed relaxation can therefore be characterized by four complementary properties. First, differentiability is introduced through the probabilistic representation of the population and the gradient estimator associated with its trainable parameters. Second, stochastic exploration is retained through sampling from the parameterized population distribution. Third, the distributional formulation naturally supports vectorized population evaluation and refinement, avoiding explicit per-individual crossover and mutation loops. Finally, these properties make the optimization procedure compatible with gradient backpropagation and end-to-end decision-aware learning. Thus, the proposed approach does not merely replace individual genetic operators by smooth approximations, but reformulates the population evolution process itself as a differentiable stochastic optimization layer.

4 Experiments

4.1 Boiler Drum Turbine Unit

We consider the closed-loop control of a nonlinear boiler drum turbine unit described in discrete time by

$$x_{k+1} = f(x_k, u_k), \quad y_k = g(x_k, u_k), \quad (25)$$

where $x_k \in \mathbb{R}^3$, $u_k \in \mathbb{R}^3$, and $y_k \in \mathbb{R}^3$ denote the state, control input, and measured output, respectively. The sampling time is fixed to $T_s = 0.1$ s and each closed-loop episode lasts 300 s. We use the same nonlinear system described in [Deeb et al., 2026]. The initial conditions are

$$\begin{aligned} x_0 &= [137 \ 120 \ 299.09]^\top, \\ u_0 &= [0.563 \ 0.868 \ 0.743]^\top. \end{aligned} \quad (26)$$

The reference trajectory is piecewise constant:

$$r_k = [r_{1,k} \ r_{2,k} \ r_{3,k}]^\top, \quad (27)$$

with $r_{1,k} = 137.0$ and $r_{2,k} = 120.0$ for $t < 150$ s, followed by a step change at $t = 150$ s to $r_{1,k} = 148.94$ and $r_{2,k} = 140.7$, while $r_{3,k} = 0$ throughout the experiment. A receding-horizon MPC strategy is adopted, where the optimizer is triggered every 3 sampling steps. At each triggering instant, an open-loop control sequence $U_k \in \mathbb{R}^{H \times 3}$ is computed and the first three elements are applied, yielding an effective MPC update period

$$T_{\text{MPC}} = t_3 T_s = 0.3 \text{ s}. \quad (28)$$

All controllers operate under identical actuator constraints. Let $\hat{u}_k$ denote the candidate control generated by the optimizer and u_{k-1} the previously applied input. The implemented control is obtained by projection

$$u_k = \Pi_{\mathcal{U} \cap \Delta \mathcal{U}}(\hat{u}_k; u_{k-1}), \quad (29)$$

where the magnitude constraints are

$$u_{\min} = 0.01, \quad u_{\max} = 0.99, \quad (30)$$

and the rate constraints are defined component-wise as

$$\begin{aligned} u_{1,k} &\in [u_{1,k-1} - T_s \cdot 0.007, u_{1,k-1} + T_s \cdot 0.007], \\ u_{2,k} &\in [u_{2,k-1} - T_s \cdot 2.0, u_{2,k-1} + T_s \cdot 2.0], \\ u_{3,k} &\in [u_{3,k-1} - T_s \cdot 0.05, u_{3,k-1} + T_s \cdot 0.05]. \end{aligned} \quad (31)$$

Three closed-loop controllers are compared under identical simulation, reference, and constraint settings. The first is a baseline GA-MPC. The second controller is a rejection-based GA-MPC (RBGA), as in [Deeb et al., 2025]. The third controller is the proposed Differentiable GA-MPC (DGA-MPC), which replaces hard selection and rejection with a differentiable relaxation of the genetic operators. All GA-based controllers incorporate an identical LQR fallback mechanism: at each MPC update, if the instantaneous tracking error satisfies

$$\|y_k - r_k\|^2 < \varepsilon_{\text{track}}, \quad \varepsilon_{\text{track}} = 15, \quad (32)$$

a local linear model is used to compute an LQR control law

$$u_k = \mathcal{K}(y_k - r_k, u_{k-1}), \quad (33)$$

which is replicated over the prediction horizon. Otherwise, the corresponding genetic algorithm variant is employed to optimize the control sequence over the horizon using full population evaluations at each generation. In all cases, the final sequence U_k is applied under the same magnitude and rate constraints.

Performance is quantified in terms of tracking accuracy, control smoothness, and constraint activity. For each output channel $j \in \{1, 2, 3\}$, the Integral Absolute Error (IAE) and Integral Squared Error (ISE) are defined as

$$\text{IAE}_j = \sum_{k=0}^{N-1} |y_{j,k} - r_{j,k}| T_s, \quad (34)$$

$$\text{ISE}_j = \sum_{k=0}^{N-1} (y_{j,k} - r_{j,k})^2 T_s. \quad (35)$$

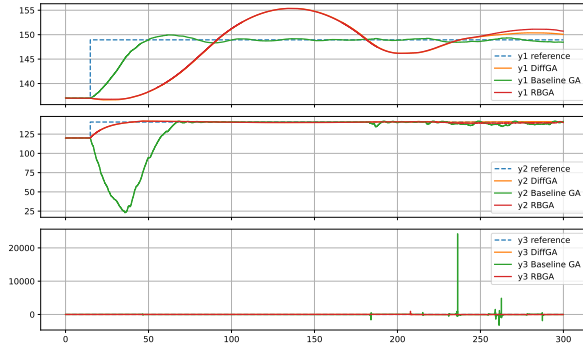


Figure 1: Closed-loop output trajectories for the three-output boiler system under identical constraints and sampling. The proposed DGA-MPC improves tracking and transient smoothness relative to baseline GA-MPC, while RBGA provides intermediate performance with reduced evaluation cost.

Control smoothness is measured via the Total Variation (TV) for each actuator $m \in \{1, 2, 3\}$:

$$TV_m = \sum_{k=1}^{N-1} |u_{m,k} - u_{m,k-1}|. \quad (36)$$

Constraint activity is defined as the fraction of time steps at which either magnitude saturation or rate limiting is active.

Figure 1 shows the closed-loop output trajectories for all controllers. Under identical sampling and actuator constraints, the proposed DGA-MPC achieves improved tracking accuracy and smoother transients relative to the baseline GA-MPC, while RBGA provides intermediate performance with reduced evaluation cost. In particular, DGA-MPC maintains balanced tracking across all outputs without inducing excessive oscillatory behavior in the control inputs.

Table 1 reports the normalized performance metrics. The results confirm that the baseline controller, although competitive on y_1 , performs poorly on y_2 and y_3 , where it attains the worst normalized error values. This indicates that, under the nonlinear operating conditions of the boiler drum-turbine system, the baseline GA-MPC struggles to coordinate multi-variable dynamics effectively. The large tracking errors and elevated control variation, especially in u_2 , suggest that the optimization strategy may drive the system toward poorly conditioned or near-singular operating regions.

In contrast, DGA-MPC achieves consistently low normalized error across all outputs while maintaining smoother actuator behavior. RBGA improves upon the baseline and reduces computational burden, but does not match the overall balance and robustness exhibited by the proposed DGA-MPC. Overall, the normalized results demonstrate that DGA-MPC provides the most reliable and well-balanced closed-loop performance under strong nonlinearities.

4.2 Computational Benchmark Against EvoGrad

To evaluate the computational efficiency and optimization performance of the proposed DGA, we conducted a controlled benchmark against EvoGrad [Citterio and Tangherloni, 2025], a recently introduced differentiable evolutionary framework that reformulates classical evolutionary operators into differentiable counterparts.

All experiments were performed under a fixed evaluation budget of 20,000 objective function evaluations per run. Four standard continuous optimization problems were considered: Sphere, Rosenbrock, Ackley, and Griewank. These functions span convex unimodal structure (Sphere), narrow curved valleys (Rosenbrock), highly multimodal landscapes (Ackley), and oscillatory multimodal structure (Griewank). Each function was evaluated in dimensions $D = 30$ and $D = 50$. The search domain was bounded by

$$x_i \in [-100, 100], \quad i = 1, \dots, D, \quad (37)$$

with box constraints enforced via projection.

Both algorithms used identical population sizes and stopping criteria. The number of generations was determined by

$$G = \left\lfloor \frac{20,000}{\text{population size}} \right\rfloor, \quad (38)$$

ensuring equal evaluation exposure. For each problem and dimensionality, 10 independent runs were performed using different random seeds. Performance was assessed using the final best objective value and best-so-far convergence curves. Wall-clock runtime was measured using high-resolution timers with device synchronisation to ensure consistent timing.

Figure 2 illustrates the median best-so-far convergence curves across seeds. The results show that DGA reaches the same solution class as EvoGrad across all test functions and dimensions. Convergence trajectories are qualitatively similar, and the final objective values achieved within the fixed evaluation budget are nearly identical.

Table 2 reports the mean total runtime (seconds) averaged over 10 runs. Although solution quality remains comparable, a substantial difference in computational cost is observed in favour of DGA. DGA requires approximately 0.03–0.07 seconds per run, whereas EvoGrad requires approximately 3.4–3.6 seconds under identical conditions. This corresponds to a computational speed advantage ranging approximately between $50\times$ – $110\times$, while maintaining equivalent optimization quality.

The observed computational advantage is primarily attributable to the fully vectorised distribution update in DGA, which eliminates operator-level per-individual crossover and mutation loops. In contrast, EvoGrad retains operator-level transformations for selection,

Table 1: Normalized performance comparison of controllers (each metric divided by the worst value in its column; lower is better).

Controller	IAE _{y1}	ISE _{y1}	IAE _{y2}	ISE _{y2}	IAE _{y3}	ISE _{y3}	TV _{u1}	TV _{u2}	TV _{u3}	Constraint
DGA	0.94	0.99	0.07	0.01	0.20	0.00	0.95	0.07	0.82	0.96
Baseline	0.21	0.18	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.57
RBGA	1.00	1.00	0.10	0.01	0.21	0.00	0.95	0.43	0.68	1.00

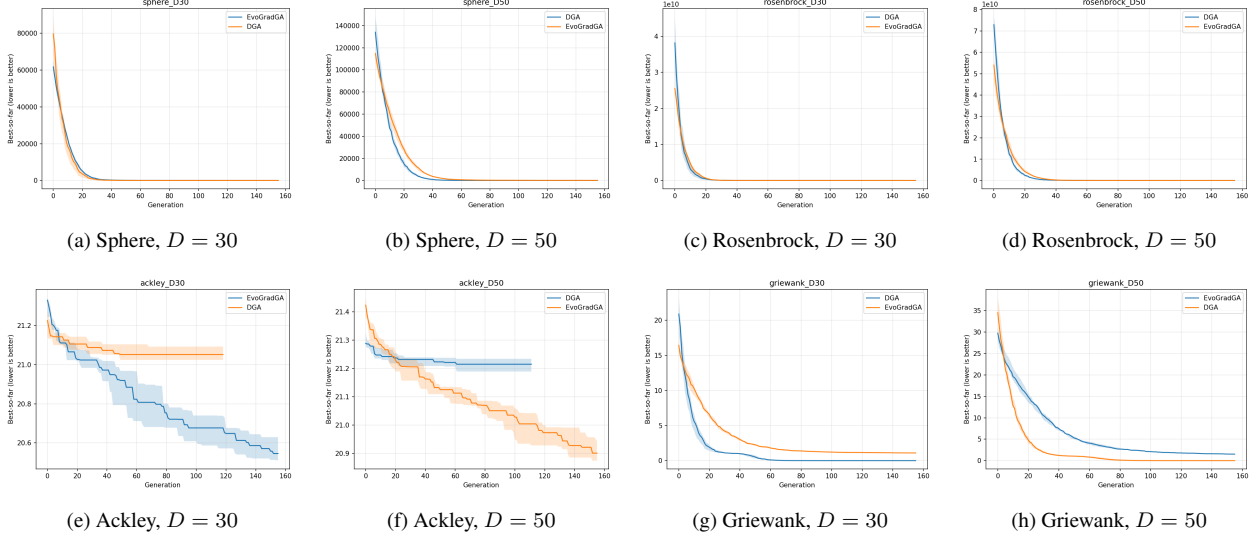


Figure 2: Median best-so-far objective value versus evaluations under a budget of 20,000 evaluations. DGA and EvoGrad exhibit comparable convergence behaviour across all benchmark functions and dimensions.

Table 2: Mean total runtime (seconds) under 20,000 objective evaluations.

Function	Dimension	DGA (s)	EvoGrad (s)
Sphere	30	0.0514	3.4523
Sphere	50	0.0549	3.4926
Rosenbrock	30	0.0584	3.4158
Rosenbrock	50	0.0635	3.5879
Ackley	30	0.0372	3.5158
Ackley	50	0.0304	3.4234
Griewank	30	0.0632	3.5490
Griewank	50	0.0662	3.5321

crossover, and mutation, introducing additional tensor operations and sampling overhead. The results demonstrate that distributional relaxation preserves optimization performance while significantly reducing computational cost, particularly in moderate-to-high dimensional settings.

5 Conclusion

This paper addressed the problem of integrating evolutionary optimization into differentiable, decision-aware control frameworks for nonlinear constrained systems. While Genetic Algorithms remain attractive for nonconvex and black-box control problems, their discrete operators traditionally prevent gradient propagation and limit their compatibility with modern learning pipelines.

We proposed a Differentiable relaxation of Genetic Algorithm (DGA) that reformulates a real-coded GA with simulated binary crossover and polynomial mutation into a smooth, distribution-based relaxation. By replacing hard selection and population updates with soft weighting and reparameterized sampling, the resulting layer preserves population-based exploration while enabling gradient-based training of upstream components.

When embedded into a receding-horizon MPC scheme for a nonlinear boiler drum–turbine unit, the proposed DGA-MPC demonstrated improved tracking performance, smoother actuator behavior, and more balanced constraint usage compared to baseline GA-MPC and rejection-based variants. The results indicate that classical GA strategies may drive the system toward poorly conditioned operating regions under strong non-

linearities, whereas the proposed distributional refinement maintains stability and coordination across outputs.

Furthermore, controlled benchmarks against EvoGrad on standard optimization problems showed that DGA achieves comparable convergence quality while significantly reducing computational overhead due to its fully vectorized distributional updates.

Overall, the proposed approach bridges evolutionary computation and differentiable optimization within a unified computational graph. From the standpoint of applied physics, the results provide additional insight into the control of strongly coupled nonlinear thermodynamic systems: improving the structure of the optimization process can simultaneously enhance reference tracking and produce smoother, more physically admissible transient actuation. This establishes a direct connection between differentiable evolutionary optimization and the controlled dynamics of energy-conversion systems. Future work will focus on extending the framework to multi-objective formulations, adaptive covariance structures, and large-scale industrial cyber-physical systems with partial observability and uncertainty.

Acknowledgment

The research is supported by the Ministry of Science and Higher Education of the Russian Federation under Agreement No. 075-15-2025-674 dated 08/20/2025.

References

- Agrawal, A., Amos, B., Barratt, S., Boyd, S., Diamond, S., and Kolter, J. Z. (2019). Differentiable convex optimization layers. *Advances in neural information processing systems*, **32**.
- Aguilar, D., Quinones, J. J., Pineda, L. R., Ostanek, J., and Castillo, L. (2024). Optimal scheduling of renewable energy microgrids: A robust multi-objective approach with machine learning-based probabilistic forecasting. *Applied Energy*, **369**, pp. 123548.
- Amos, B., Jimenez, I., Sacks, J., Boots, B., and Kolter, J. Z. (2018). Differentiable mpc for end-to-end planning and control. *Advances in neural information processing systems*, **31**.
- Arguchintsev, A. and Kopylov, D. (2024). Numerical optimization of separation processes in a distillation column. *Cybernetics and Physics*, **13** (3), pp. 241–247.
- Babich, N., Marzel, E., Rybalko, A., Chen, O., Chulkin, V., and Fradkov, A. (2025). Outline of cybernetical neuroscience. *Cybernetics and Physics*, **14** (1), pp. 13–18.
- Bai, S., Kolter, J. Z., and Koltun, V. (2019). Deep equilibrium models. *Advances in neural information processing systems*, **32**.
- Banzhaf, W., Machado, P., and Zhang, M. (2023). Handbook of evolutionary machine learning. (*No Title*).
- Blondel, M., Berthet, Q., Cuturi, M., Frostig, R., Hoyer, S., Llinares-López, F., Pedregosa, F., and Vert, J.-P. (2022). Efficient and modular implicit differentiation. *Advances in neural information processing systems*, **35**, pp. 5230–5242.
- Cavus, M. and Allahham, A. (2024). Enhanced microgrid control through genetic predictive control: integrating genetic algorithms with model predictive control for improved non-linearity and non-convexity handling. *Energies*, **17** (17), pp. 4458.
- Citterio, B. F. and Tangherloni, A. (2025). Evograd: Metaheuristics in a differentiable wonderland. *arXiv preprint arXiv:2506.06320*.
- Cordeiro-Costas, M., Labandeira-Pérez, H., Villanueva, D., Pérez-Orozco, R., and Eguía-Oller, P. (2024). Nsga-ii based short-term building energy management using optimal lstm-mlp forecasts. *International Journal of Electrical Power & Energy Systems*, **159**, pp. 110070.
- Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, **6** (2), pp. 182–197.
- Deeb, A., Khokhlovskiy, V., and Shkodyrev, V. (2025). Enhancing evolutionary controllers with a rejection-based genetic algorithm. *Cybernetics and Physics*, **14** (4), pp. 325–333.
- Deeb, A., Khokhlovskiy, V., and Shkodyrev, V. (2026). Hierarchical multi-objective control of nonlinear systems with dynamical input constraints. In *Artificial Intelligence and Applications*, vol. 4, pp. 57–67.
- Deeb, A., Khokhlovskiy, V. N., and Shkodyrev, V. P. (2024). Model predictive control and genetic algorithms for optimization of continuous stirred tank reactors. In *Smart Electromechanical Systems: Mathematical and Software Engineering*, pp. 185–191. Springer.
- International Energy Agency (2022). *World Energy Outlook 2022*. IEA Publications, Paris, France.
- Izzo, D. and Biscani, F. (2020). dcgp: Differentiable cartesian genetic programming made easy. *Journal of Open Source Software*, **5** (51), pp. 2290.
- Katoch, S., Chauhan, S. S., and Kumar, V. (2021). A review on genetic algorithm: past, present, and future. *Multimedia tools and applications*, **80** (5), pp. 8091–8126.
- Kumar, S. and Singh, A. (2022). Dynamic modeling and control of thermal power units. *IEEE Transactions on Energy Conversion*, **37** (2), pp. 1234–1243.
- Lawrence, N. P., Damarla, S. K., Kim, J. W., Tulsyan, A., Amjad, F., Wang, K., Chachuat, B., Lee, J. M., Huang, B., and Gopaluni, R. B. (2024). Machine learning for industrial sensing and control: A survey and practical perspective. *Control Engineering Practice*, **145**, pp. 105841.
- Londe, M. A., Pessoa, L. S., Andrade, C. E., and Resende, M. G. (2025). Biased random-key genetic algorithms: A review. *European Journal of Operational Research*, **321** (1), pp. 1–22.
- Ogata, K. (2020). Modern control engineering.
- Pan, J., Ye, Z., Yang, X., Yang, X., Liu, W., Wang, L.,

- and Bian, J. (2024). Bpqp: A differentiable convex optimization framework for efficient end-to-end learning. *Advances in Neural Information Processing Systems*, **37**, pp. 77468–77493.
- Rybalko, A. and Fradkov, A. (2023). Identification of two-neuron fitzhugh–nagumo model based on the speed-gradient and filtering. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, **33**(8).
- Saini, N. and Ohri, J. (2023). Load frequency control in three-area single unit power system considering non-linearities effect. *Cybernetics and Physics*, **12**(1), pp. 60–69.
- Schwenzer, M., Ay, M., Bergs, T., and Abel, D. (2021). Review on model predictive control: An engineering perspective. *The International Journal of Advanced Manufacturing Technology*, **117**(5), pp. 1327–1349.
- Sirota, D. and Ovsyannikov, D. (2025). Pde-constrained optimal control of subsurface reservoir systems via neural operator surrogates. *Cybernetics and Physics*, **14**(3), pp. 285–291.
- Slowik, A. and Kwasnicka, H. (2020). Evolutionary algorithms and their applications to engineering problems. *Neural Computing and Applications*, **32**, pp. 12363–12379.
- Wang, C., Liu, M., Zhao, Y., Qiao, Y., Chong, D., and Yan, J. (2018). Dynamic modeling and operation optimization for the cold end system of thermal power plants during transient processes. *Energy*, **145**, pp. 734–746.
- Wu, X., Shen, J., Li, Y., and Lee, K. Y. (2011). Stable model predictive control based on ts fuzzy model with application to boiler-turbine coordinated system. In *2011 50th IEEE Conference on Decision and Control and European Control Conference*, IEEE, pp. 2981–2987.
- Wu, X., Shen, J., Li, Y., and Lee, K. Y. (2014). Hierarchical optimization of boiler–turbine unit using fuzzy stable model predictive control. *Control Engineering Practice*, **30**, pp. 112–123.
- Yang, C., Zhang, T., Zhang, Z., and Sun, L. (2022). Mld–mpc for ultra-supercritical circulating fluidized bed boiler unit using subspace identification. *Energies*, **15**(15), pp. 5476.
- Zeng, P., Song, X., Lensen, A., Ou, Y., Sun, Y., Zhang, M., and Lv, J. (2023). Differentiable genetic programming for high-dimensional symbolic regression. *arXiv preprint arXiv:2304.08915*.
- Ławryńczuk, M. (2017). Nonlinear predictive control of a boiler-turbine unit: A state-space approach with successive on-line model linearisation and quadratic optimisation. *ISA Transactions*, **67**, pp. 476–495.