# Learning to optimize over linearly convergent algorithms: gotta characterize 'em all

**Andrea Martin**
KTH Royal Institute of Technology
andrmar@kth.se

**Ian R. Manchester**
The University of Sydney
ian.manchester@sydney.edu.au

**Luca Furieri**
University of Oxford
luca.furieri@eng.ox.ac.uk

## Abstract

In high-stakes engineering applications, optimizers must come with provable worst-case guarantees over a mathematically defined class of problems. Yet, designing for the worst case often sacrifices performance on the instances that arise in practice. While *learning to optimize* approaches hold promise in enhancing average-case optimizer performance on sets of target problems of interest, they often require conservative safe-guarding mechanisms in order to maintain convergent behavior. In this paper, we characterize the class of algorithms that achieve *linear convergence* for several classes of nonsmooth composite optimization problems. This is achieved by parametrizing all *augmentations* to a baseline linearly convergent algorithm – that is, all perturbations to its update rule – that preserve its certified rate on the entire class. Our results apply to augmenting algorithms such as gradient descent for nonconvex, gradient-dominated functions; Nesterov's accelerated method for strongly convex functions; and projected methods for optimization over polyhedral feasibility sets. We showcase effectiveness of the approach on solving optimization problems with tight iteration budgets in application to ill-conditioned systems of linear equations and model predictive control for linear systems.

## 1 Introduction

Guarantees of fast convergence are crucial whenever optimization must be executed under tight computational budgets, as in large-scale machine learning (ML) or real-time model predictive control (MPC). Worst-case linear convergence guarantees have been developed for iterative optimization algorithms over several classes of objective functions, whose structure – e.g., strong convexity, smoothness, or gradient dominance – can be exploited by first-order schemes. These include standard gradient descent and Nesterov's accelerated method [13] as well as optimal accelerated methods recently developed by the control community [9, 18, 20].

The learning-to-optimize (L2O) literature addresses the challenge of adopting user-defined performance metrics and uses ML to design algorithms tailored to those metrics. For instance, [2] proposes an algorithmic performance metric that balances convergence speed with solution precision and accordingly designs neural network (NN) update rules. Despite demonstrating performance that exceeds that of state-of-the-art classical algorithms, general NN update rules come with no convergence guarantees. This has motivated the study of safeguarding mechanisms [6], which, however, may introduce significant conservatism, or techniques that limit ML components to learning initializations [16] or to optimal parameter tuning [7] of classical algorithms.

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: ScaleOPT.

Another line of research seeks to use ML to design new convergent update rules, aiming to discover application-specific shortcuts unknown to classical update rules. This has been achieved by taking simple gradient descent as a baseline and enhancing it through learned perturbations from such gradient-based updates. The work [11] characterizes the class of all and only those perturbations that ensure asymptotic convergence to stationary points in nonconvex, unconstrained smooth optimization. The work [3] uses deep learning to train perturbations from gradient descent and saturates these updates with the norm of measured gradients, ensuring convergence for composite convex optimization. The numerical studies of [3, 11] empirically demonstrate that convergence rates superior to those of classical algorithms can be achieved through training on gradient descent perturbations.

Our main goal is to address a question that has remained open in the literature of learned optimization.

*Given any linearly convergent algorithm for solving a class of optimization problems, how can we improve its average-case performance over a class of problems of interest – without sacrificing its worst-case linear convergence rate over the entire class?*

We believe that a theoretical study of these trade-offs is important towards making learned optimization a standard and reliable component of linearly convergent algorithm design.

*Contributions:* Given any optimization algorithm that achieves linear convergence to a set of fixed points at a specified rate – henceforth the *baseline algorithm* – our main contributions are as follows.

- A characterization of the conditions on the baseline algorithm under which adding exponentially decaying perturbations preserves the same linear convergence rate, up to a higher-order polynomial term. These conditions identify fundamental trade-offs between the frequency of perturbations and their worst-case impact on the linear convergence rate.
- A completeness result for linearly convergent optimization: every update rule that converges linearly at a given rate can be written as the sum of the baseline algorithm and a suitably designed augmentation of its update rule that preserves linear convergence.
- We instantiate our results for augmenting state-of-the-art linearly convergent algorithms, notably 1) NAG for strongly convex and smooth optimization, and 2) projected gradient descent for convex optimization with polyhedral feasible sets. We report further applications to classes of nonconvex Polyak–Łojasiewicz (PL) functions in the supplementary material.

Numerical examples showcase the potential of learned linearly convergent optimization in augmenting the performance of classical algorithms under tight iteration budgets.

*Notation:* The set of all sequences $\mathbf{x} = (x_0, x_1, x_2, \ldots)$ where $x_t \in \mathbb{R}^n$ for all $t \in \mathbb{N}$ is denoted as $\ell^n$. For $\mathbf{x} \in \ell^n$, we denote by $z\mathbf{x} = (x_1, x_2, x_3, \ldots)$ the sequence shifted one time-step forward. For a function $g : \mathbb{R}^n \to \mathbb{R}^m$, we write $g(\mathbf{x}) = (g(x_0), g(x_1), \ldots) \in \ell^m$. For $m \in \mathbb{N}$, we use $\mathcal{P}_m(x)$ to denote the set of positive and monotonically non-decreasing polynomials $p(x)$ of degree at most $m$ in the variable $x$ with $p(1) \geq 1$. We define the set of fixed points of an operator $\pi$, assumed non-empty, as $\texttt{Fix}_\pi$. For $m \in \mathbb{N}$ and $\gamma \in (0, 1)$, we denote by $\ell_{exp}(m, \gamma)$ the class of signals $\mathbf{x} \in \ell^n$ for which there exists a polynomial $p_m(t) \in \mathcal{P}_m(t)$ such that $|x_t| \leq p_m(t)\gamma^t$. We write $\mathbb{I}_S$ for the indicator function of a set $S$ (0 if $x \in S$, $+\infty$ otherwise). The distance from $x$ to a set $S$ is defined as $\mathrm{dist}(x, S) = \inf_{y \in S} |x - y|$. For $u, v \in \mathbb{R}^n$, $u \leq v$ indicates the element-wise inequality.

## 2 Problem Formulation

Consider composite optimization problems of the form

$$\min_{x \in \mathbb{R}^d} \quad f(x) + g(x)\,, \tag{1}$$

where $x \in \mathbb{R}^d$ is the decision variable, $f : \mathbb{R}^d \to \mathbb{R}$ is proper and $\beta$-smooth, and $g : \mathbb{R}^d \to \mathbb{R} \cup \{+\infty\}$ is convex, proper, and lower semi-continuous, but potentially nonsmooth. We let $F(x) = f(x) + g(x)$ for brevity, and we assume that the set of optimizers $\mathcal{X}^\star = \mathrm{argmin}_{x \in \mathbb{R}^d}\ F(x)$ is non-empty. In particular, we note that (1) subsumes constrained optimization problems of the form

$$\min_{x \in \mathbb{R}^d} \quad f_0(x) \tag{2a}$$

$$\text{subject to} \quad f_i(x) \leq 0\,, \quad \forall i \in [1, M]\,, \tag{2b}$$

where $f_0 : \mathbb{R}^d \to \mathbb{R}$ is $\beta$-smooth, and each function $f_i : \mathbb{R}^d \to \mathbb{R}$ with $i \in [1, M]$ defines a non-empty convex feasibility set $\mathcal{X}_i \subseteq \mathbb{R}^d$. In fact, one can rewrite (2) as an instance of (1) letting $f(x) = f_0(x)$ and $g(x) = \max_{i\in[1,M]} \mathbb{I}_{\mathcal{X}_i}(x)$.

A standard method to solve problem (1) is to analytically construct iterations of the form:

$$\xi_{t+1} = \pi(F, \xi_t)\,,\ x_t = \phi(F, \xi_t)\,, \quad \xi_0 \in \mathbb{R}^n\,, \quad t \in \mathbb{N}\,, \tag{3}$$

where $\xi_t \in \mathbb{R}^n$ is the state variable, the decision $x_t \in \mathbb{R}^d$ is the output variable, and the operator $\pi$ is designed so that its set of fixed points $\texttt{Fix}_\pi$, that is, the points $\xi^\star$ such that $\pi(F, \xi^\star) = \xi^\star$ is related to $\mathcal{X}^\star$ through $\phi$; that is, a point $x^\star \in \mathcal{X}^\star$ can be reconstructed from a point $\xi^\star \in \texttt{Fix}_\pi$ as per $x^\star = \phi(F, \xi^\star)$.

A key metric for the performance of algorithms (3) when applied to a class of problems $F \in \mathcal{F}$ is how fast they converge to $\texttt{Fix}_\pi$. Classical optimization algorithms often come with convergence guarantees that hold for the worst-case instance of $F \in \mathcal{F}$. However, optimal control methods such as MPC require efficiently finding solutions to the instances of (1) that are encountered during deployment, where the objective $F$ is drawn from a specific distribution $\mathbb{D}_\mathcal{F}$ over the class $\mathcal{F}$. Motivated as such, in this work we investigate the following question.

*Given a set of problem instances $F \sim \mathbb{D}_\mathcal{F}$ and a baseline algorithm $\pi$ to solve* (1)*, how can we improve its average-case performance over $\mathbb{D}_\mathcal{F}$, while retaining worst-case convergence guarantees over the entire class $\mathcal{F}$?*

In particular, this paper characterizes algorithms $\boldsymbol{\nu}$ that achieve *linear convergence* to $\texttt{Fix}_\pi$ for classes of functions $F \in \mathcal{F}$. We will showcase how to leverage such characterization for learning-based algorithm design in Section 4.

**Definition 1** *An algorithm $\xi_{t+1} = \nu_t(F, \xi_{t:0})$ is said to be linearly convergent to $\texttt{Fix}_\pi$ for $\mathcal{F}$ with rate $\gamma \in (0, 1)$ if there exists a polynomial $p(t) \in \mathcal{P}_m(t)$ such that*

$$\operatorname{dist}(\xi_t, \texttt{Fix}_\pi) \le p(t)\gamma^t \operatorname{dist}(\xi_0, \texttt{Fix}_\pi)\,,\ \forall \xi_0 \in \mathbb{R}^n\,,\ \forall F \in \mathcal{F}\,, \tag{4}$$

*at all times, where $\operatorname{dist}(\cdot,\cdot)$ is a distance function. In this case, we write that $\boldsymbol{\nu} \in \texttt{pExp}^\pi_\mathcal{F}(m, \gamma)$. When the focus is not on the polynomial order and only on the exponential convergence rate, we write $\boldsymbol{\nu} \in \widehat{\texttt{pExp}}^\pi_\mathcal{F}(\gamma)$. Additionally, if* (4) *holds with a polynomial such that $p(1) = 1$, we say that $\boldsymbol{\nu}$ is monotonically linearly convergent to $\texttt{Fix}_\pi$ since the distance to the set of fixed points shrinks monotonically with the iterations, and we write $\boldsymbol{\nu} \in \texttt{Exp}^\pi_\mathcal{F}(\gamma)$.*

To enhance the average-case performance of a baseline algorithm $\pi$ on problem instances $F \sim \mathbb{D}_\mathcal{F}$, we aim to design algorithm updates $v_t \in \mathbb{R}^n$ that do not jeopardize its convergence guarantees. Specifically, we will consider augmented update rules defined by the iterations:

$$\xi_{t+1} = \nu_t(F, \xi_{t:0}) = \pi(F, \xi_t) + v_t(F, \xi_{t:0})\,,\ \xi_0 \in \mathbb{R}^n\,, \tag{5}$$

and we will establish conditions ensuring that $\boldsymbol{\nu}$ is linearly convergent as per (4).

## 3 Main Results

In this section, we establish our main results on how introducing an augmentation term $v_t$ affects the worst-case linear convergence guarantees of a given baseline algorithm $\pi$. We first abstract away from the specific form of $\pi$ and the class of functions $\mathcal{F}$ it is designed to optimize; we only assume that $\pi$ is a linearly convergent fixed-point algorithm as per Definition 1. In Section 3.2, we present corollaries that reveal several classes of problems (1) and corresponding baseline algorithms $\pi$ that are compatible with our framework. To enhance readability, we defer all proofs to Appendix C of the supplementary material.

### 3.1 Characterizations of linearly convergent algorithms and their completeness under (5)

Consider any $\pi \in \widehat{\texttt{pExp}}^\pi_\mathcal{F}(\gamma)$ achieving linear convergence as per Definition 1. The property (4) implies that the signal $\operatorname{dist}(\xi_t, \texttt{Fix}_\pi)$ decays exponentially up to a polynomial factor $p(t)$, for any initial condition $\xi_0 \in \mathbb{R}^n$ and any objective $F \in \mathcal{F}$. We now characterize to what extent injecting exponentially decaying signals $v_t$ in the iterates of (5) can deteriorate the convergence guarantee of $\pi$.

**Theorem 1** *Consider the recursion* (5) *and assume that* $\pi \in \texttt{pExp}^{\pi}_{\mathcal{F}}(m, \gamma)$*. Choose any* $N \in \mathbb{N}$ *such that*

$$\rho = p(N)\gamma^N < 1\,, \tag{6}$$

*and any auxiliary signal* $\mathbf{w} \in \ell_{exp}(m, \rho)$*. For every* $t \in \mathbb{N}$ *construct the augmentation signal* $v_t$ *in* (5) *as follows:*

$$v_t = \begin{cases} w_{\frac{t+1}{N}-1} & \textit{if}\; t+1 \bmod N = 0\,, \\ 0 & \textit{otherwise}\,. \end{cases} \tag{7}$$

*Then, the iterates of* (5) *satisfy:*

$$\operatorname{dist}(\boldsymbol{\xi}, \texttt{Fix}_{\pi}) \in \ell_{exp}(m+1,\; \sqrt[N]{p(N)}\gamma)\,. \tag{8}$$

Theorem 1 establishes a trade-off between how often we inject an exponentially decaying perturbation – as measured by $N \in \mathbb{N}$ – and the degradation of the convergence rate. In particular, if $\pi \in \texttt{Exp}^{\pi}_{\mathcal{F}}(\gamma)$ then $p(1) = 1$ by definition and we observe that the asymptotic rate $\gamma$ does not change even with $N = 1$, as only the order of the polynomial factor in (4) is affected. This means that one can apply perturbation $\mathbf{v} \in \ell_{exp}(m, \gamma)$ at every time instant as per (5) without deteriorating the asymptotic convergence rate. For the general case where $\pi \in \texttt{pExp}^{\pi}_{\mathcal{F}}(m, \gamma)$, instead, the convergence rate increases at most to the value $\sqrt[N]{p(N)}\gamma \in (\gamma, 1)$. As expected, when $N$ tends to infinity, we recover the original rate of the baseline algorithm because $\lim_{N\to\infty} \sqrt[N]{p(N)} = 1$; this corresponds to the case $v_t = 0$ for all $t$.

It remains crucial to understand how large is the class of linearly convergent algorithms $\xi_t = \nu_t(F, \xi_{t:0})$ that can be achieved by perturbing a baseline algorithm $\pi$ with an exponentially decaying learned update $v_t$ as per (5). Our next result establishes conditions on the baseline algorithm $\pi$ that guarantee that *any* algorithm in $\texttt{pExp}^{\pi}_{\mathcal{F}}(m, \gamma)$ can be represented – provided that such target algorithm satisfies the following regularity condition.

**Definition 2** *Define the sequence of updates* $u_t = \xi_{t+1} - \xi_t$ *associated with a linearly convergent algorithm* $\boldsymbol{\nu} \in \texttt{pExp}^{\pi}_{\mathcal{F}}(m, \gamma)$*. We say that* $\boldsymbol{\nu}$ *is* regular *if the sequence of updates vanishes with the same exponential rate, that is,* $\mathbf{u} = z\boldsymbol{\xi} - \boldsymbol{\xi} \in \ell_{exp}(m, \gamma)$*.*

In other words, the definition above excludes pathological cases of linearly convergent algorithms that can cycle indefinitely among the points in $\texttt{Fix}_{\pi}$ even when $\operatorname{dist}(\xi_t, \texttt{Fix}_{\pi}) = 0$. We are ready to present our completeness result.

**Theorem 2** *Let* $\pi \in \texttt{Exp}^{\pi}_{\mathcal{F}}(\gamma)$ *be such that* $\pi(F, \xi)$ *is Lipschitz continuous in* $\xi$*. Consider the augmented algorithm* $\boldsymbol{\nu}$ *with iterates* $\xi_t$ *defined as per* (5)*, and any target algorithm* $\chi_{t+1} = \sigma_t(F, \chi_{t:0})$ *such that* $\boldsymbol{\sigma} \in \texttt{pExp}^{\pi}_{\mathcal{F}}(m, \gamma)$*. If* $\boldsymbol{\sigma}$ *is regular, there exists a sequence* $\mathbf{v}(F, \boldsymbol{\xi}) \in \ell_{exp}(m, \gamma)$ *such that the iterations of* $\boldsymbol{\nu}$ *initialized with* $\xi_0 = \chi_0$ *are equivalent to those of* $\boldsymbol{\sigma}$*. Additionally, the augmented algorithm* $\boldsymbol{\nu}$ *belongs to* $\widehat{\texttt{pExp}}^{\pi}_{\mathcal{F}}(\gamma)$ *for any* $\mathbf{v} \in \ell_{exp}(m, \gamma)$ *with* $m \in \mathbb{N}$*.*

A few remarks are in order. First, the completeness property of Theorem 2 is key in the context of automating the design of augmented algorithms, e.g. by learning from sampled problems, as it implies that (5) encompasses all linearly convergent algorithms in $\texttt{pExp}^{\pi}_{\mathcal{F}}(m, \gamma)$. Second, when we learn an augmentation term $\mathbf{v} \in \ell_{exp}(m, \gamma)$ by searching over the entire space of exponentially decaying updates, it is desirable that the baseline algorithm satisfies the stronger condition $\pi \in \texttt{Exp}^{\pi}_{\mathcal{F}}(\gamma)$. This is because, if $\pi \in \texttt{pExp}^{\pi}_{\mathcal{F}}(m, \gamma\,) \setminus \texttt{Exp}^{\pi}_{\mathcal{F}}(\gamma)$, Theorem 1 only guarantees linear convergence – with the worsened rate $\sqrt[N]{p(N)}\,\gamma$ – for those $\mathbf{v}$ chosen exactly as in (7). In other words, without $\pi \in \texttt{Exp}^{\pi}_{\mathcal{F}}(\gamma)$, most perturbations in $\ell_{exp}(m, \gamma)$ may not preserve linear convergence. Third, the assumption that $\pi(F, \xi)$ is Lipschitz continuous in $\xi$ is mild; we will show in the following that this condition is satisfied for baseline algorithms widely used for convex and composite optimization.

### 3.2 Learning over linearly convergent algorithms for convex composite optimization

Here, we establish how Theorem 1 and Theorem 2 enable the augmentation of existing linearly convergent methods for important classes of convex and composite optimization problems in the form (1). Further applications to classes of Polyak–Łojasiewicz (PL) nonconvex problems are reported as additional material in Appendix A. We first consider the case (1) where $g(x) = 0$ for all $x \in \mathbb{R}^d$, leaving us with the task of minimizing a $\beta$-smooth function $F(x) = f(x)$.

**Corollary 1** *Let $\mathcal{F}_{SC}^{\beta,\mu}$ denote the set of $\beta$-smooth and strongly convex functions and consider the NAG algorithm*

$$\pi(F, \xi_t) = \begin{bmatrix} 1+\alpha & -\alpha \\ 1 & 0 \end{bmatrix} \xi_t + \begin{bmatrix} -\eta \\ 0 \end{bmatrix} \nabla F\left(\begin{bmatrix} 1+\alpha & -\alpha \end{bmatrix} \xi_t\right), \tag{9}$$

*where $\xi_t = \begin{bmatrix} x_t^\top & x_{t-1}^\top \end{bmatrix}^\top$ and $\alpha \geq 0$ is the momentum coefficient. Let $\eta = \frac{1}{\beta}$ and $\alpha = \frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}$, where $\kappa = \frac{\beta}{\mu} \geq 1$ is the condition number. Choose any target rate degradation factor $\tau \in (1, \frac{1}{\gamma})$, where $\gamma = \sqrt{1 - \frac{1}{\sqrt{\kappa}}}$. Then, for any $N \in \mathbb{N}$ such that $p(N) < \tau^N$ and $\mathbf{v}$ constructed as per (7) using any $\mathbf{w} \in \ell_{exp}(m, \tau\gamma)$, the augmented algorithm $\boldsymbol{\nu}(F, \xi_{t:0})$ defined by $\xi_{t+1} = \pi(F, \xi_t) + v_t$ is such that $\boldsymbol{\nu} \in \widehat{\texttt{pExp}}^{\pi}_{\mathcal{F}_{SC}^{\beta,\mu}}(\tau\gamma)$.*

While Corollary 1 focuses on the case where NAG is used as the baseline algorithm $\pi$ in (5), we remark that the results extend analogously to any baseline algorithm $\pi \in \texttt{pExp}^{\pi}_{\mathcal{F}_{SC}^{\beta,\mu}}(m, \gamma)$ such as those with optimal convergence rates in [9, 19]. As also discussed after Theorem 1, we note that augmenting accelerated algorithms, which are not monotonic in general, involves a trade-off between keeping the worst-case degradation rate $\tau$ as small as possible and the frequency at which we can apply a learned update. Last, we remark that one can always impose a target $\tau \in (1, \frac{1}{\gamma})$. Indeed, a large enough $N \in \mathbb{N}$ such that $p(N) < \tau^N$ always exists since the exponential term dominates over the polynomial one.

Our next result focuses on the case where $g(x)$ represents the indicator function of a set of convex linear constraints (2b) to address constrained optimization problems of the form (2) with $f_0 \in \mathcal{F}_{SC}^{\beta,\mu}$. Specifically, we derive conditions on $\mathbf{v}$ to ensure not only linear convergence but also feasibility of all iterates of (5).

**Corollary 2** *Consider the constrained optimization problem (2) with $f_0 \in \mathcal{F}_{SC}^{\beta,\mu}$ and $f_i(x) = A_i x - b_i$ for all $i \in [1, M]$ and define the feasible set $\mathcal{X} = \{x \in \mathbb{R}^d : f_i(x) \leq 0,\ \forall i \in [1, M]\}$. Let $g(x) = \mathbb{I}_{\mathcal{X}}(x)$ and define $\mathcal{F}_{comp}$ as the set of all such functions $F(x) = f_0(x) + g(x)$. Let $\pi$ be the proximal gradient descent method performing the iterations*

$$x_{t+1} = \min_{x \in \mathbb{R}^d} \ g(x) + \frac{1}{2}|x - (x_t - \eta \nabla f(x_t))|^2 = \mathrm{prox}_g(x_t - \eta \nabla f(x_t)) = \mathrm{proj}_{\mathcal{X}}(x_t - \eta \nabla f(x_t)), \tag{10}$$

*where $\eta \in (0, \frac{1}{\beta}]$. Consider any regular algorithm $\chi_{t+1} = \sigma_t(F, \chi_{t:0})$ with feasible iterates $\chi_t \in \mathcal{X}$ and such that $\boldsymbol{\sigma} \in \texttt{pExp}^{\pi}_{\mathcal{F}_{comp}}(m, \gamma)$, with $\gamma = 1 - \eta\mu$. Then, there exists $\mathbf{v} \in \ell_{exp}(m, \gamma)$ such that*

$$A_i v_t \leq b_i - A_i \,\mathrm{proj}_{\mathcal{X}}(x_t - \eta \nabla f(x_t)), \tag{11}$$

*at all times, for all $i \in [1, M]$, and the augmented algorithm*

$$x_{t+1} = \nu_t(F, x_{t:0}) = \mathrm{proj}_{\mathcal{X}}(x_t - \eta \nabla f(x_t)) + v_t(F, x_{t:0}), \tag{12}$$

*is equivalent to $\boldsymbol{\sigma}$. Viceversa, for any $\mathbf{v} \in \ell_{exp}(m, \gamma)$ such that (11) holds at all times and for all $i \in [1, M]$, the algorithm (12) is such that the iterates $x_t \in \mathcal{X}$ and $\boldsymbol{\nu} \in \widehat{\texttt{pExp}}^{\pi}_{\mathcal{F}_{comp}}(\gamma)$.*

## 4 Numerical Experiments

We showcase the potential of learning algorithm augmentations in solving systems of linear equations the form $Ax = b$, where $A \succ 0$ and $b$ are sampled from a joint distribution $\mathcal{D}_{A,b}$. We report application to solving linear MPC with a tight iteration budget as additional material in Appendix B.

Although each instance of the problem admits the analytical solution $x^\star = A^{-1}b$, directly computing the matrix inverse becomes numerically unstable when the condition number $\kappa(A)$ of $A$ grows very large. To address this, we instead approximate $x^\star$ using iterative methods by solving the equivalent quadratic program:

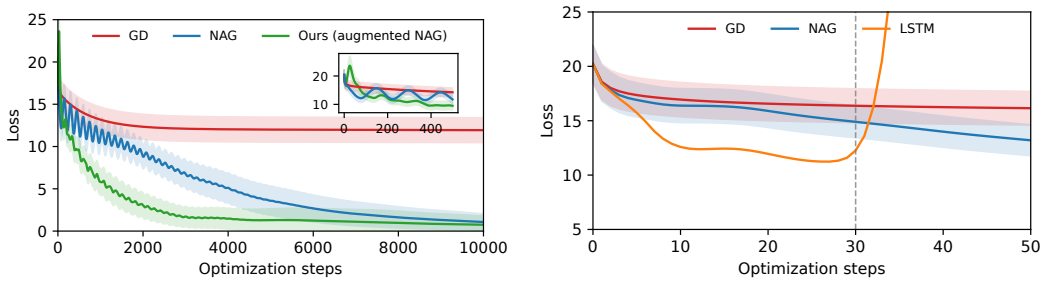
$$\min_{x \in \mathbb{R}^d} \ |Ax - b|^2 = \min_{x \in \mathbb{R}^d} \ x^\top A^\top A x - 2b^\top A x + b^\top b. \tag{13}$$

For our experiment, we assume that $A = \hat{A} + \delta_A$, where $\hat{A}$ corresponds to the matrix `bcsstk02` in the dataset [4] and is such that $\kappa(\hat{A}^\top \hat{A}) \approx 1.17 \times 10^8$, and $\delta_A$ is a matrix with entries drawn from a standard Gaussian random variable. Similarly, we assume that $b = \hat{b} + \delta_b$, where $\hat{b}$ is a vector with entries $\hat{b}_i = 0.5$ and $\delta_b$ is a vector with entries drawn from a zero-mean Gaussian random variable with standard deviation $0.2$.

We first solve the optimization problem (13) using standard gradient descent (GD) and NAG methods, with step-size and momentum chosen according to the optimal tuning for quadratic functions given in [9, Proposition 1]. Using the latter method as a baseline optimizer, we then train a two-layer LSTM to learn an augmentation term $v_t$ improving the empirical performance (25) over a dataset $\mathcal{F}$train of $1024$ realizations of $A$ and $b$.[1] Specifically, we pick an algorithmic cost function defined as $\texttt{AlgoCost}(A, b, \mathbf{x}) = \sum t = 0^T |Ax_t - b|^2$ for $T = 10000$, and perform meta-optimization using Adam with a learning rate of $10^{-3}$ for $100$ epochs – full details on the training problem and how to enforce exponential decay of an LSTM-based $v_t$ are reported in AppendixB.

We construct a test dataset by sampling $256$ independent realizations of $A$ and $b$ and compare the average-case performance of our learned optimizer with standard methods in solving (4) in Figure 4. As expected, adding a momentum term enables NAG to converge faster than standard gradient descent. Still, both methods need many iterations to reach high-accuracy solutions. Remarkably, our linearly convergent L2O method learns to follow the *positive* gradient direction initially, thereby increasing the cost function rather than decreasing it. This behavior does not occur in classical optimizers and accelerates momentum accumulation in the early stages. As shown in Figure 4, this learned behavior improves transient performance without affecting the asymptotic convergence rate. We recall that our contribution concerns proving linear convergence of algorithm augmentations; we make no claim of universal performance improvement over classical algorithms.

**Ablation study:** We trained a 2-layer LSTM optimizer without enforcing an exponential decay of the augmentations $v_t$ in (5). Since the matrix $A^\top A$ is ill-conditioned, the training procedure becomes highly sensitive to the initial parameter choice $\theta_0$, resulting in diverging sequences of updates. As a result, the lack of formal guarantees makes training algorithms to perform a large number of optimization steps numerically challenging. Moreover, the figure on the right below shows that, without guarantees, learned optimizers may diverge after the training horizon (as marked by the gray dashed line) – even when training and test problems are drawn from the same distribution.




## 5 Conclusion

We have shown that many classical linearly convergent algorithms can be enhanced with an exponentially decaying augmentation term that improves average-case performance on a specified distribution of problems. Crucially, this can be achieved without sacrificing their provable worst-case linear rates, and specifically: (i) these augmentations can be injected at any frequency below a threshold, trading off average-case improvement against worst-case rate degradation in a quantifiable way; and (ii) every regular linearly convergent method admits exactly this form of decomposition. Our experiments showcase the potential of convergent-by-design learning augmentations to achieve significant speed-ups and better generalization over general-purpose solvers.

[1] We refer to https://github.com/andrea-martin/LinearlyConvergentL2O for the source code.

## Acknowledgments and Disclosure of Funding

This work was supported by Digital Futures and the Swiss National Science Foundation (SNSF) through the Ambizione grant PZ00P2_208951.

## References

[1] Shmuel Agmon. The relaxation method for linear inequalities. *Canadian Journal of Mathematics*, 6:382–392, 1954.

[2] Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando De Freitas. Learning to learn by gradient descent by gradient descent. *Advances in neural information processing systems*, 29, 2016.

[3] Sebastian Banert, Jevgenija Rudzusika, Ozan Öktem, and Jonas Adler. Accelerated forward-backward optimization using deep learning. *SIAM Journal on Optimization*, 34(2):1236–1263, 2024.

[4] Timothy A Davis and Yifan Hu. The university of florida sparse matrix collection. *ACM Transactions on Mathematical Software (TOMS)*, 38(1):1–25, 2011.

[5] Guillaume Garrigos and Robert M Gower. Handbook of convergence theorems for (stochastic) gradient methods. *arXiv preprint arXiv:2301.11235*, 2023.

[6] Howard Heaton, Xiaohan Chen, Zhangyang Wang, and Wotao Yin. Safeguarded learned convex optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 7848–7855, 2023.

[7] Jeffrey Ichnowski, Paras Jain, Bartolomeo Stellato, Goran Banjac, Michael Luo, Francesco Borrelli, Joseph E Gonzalez, Ion Stoica, and Ken Goldberg. Accelerating quadratic optimization with reinforcement learning. *Advances in Neural Information Processing Systems*, 34:21043–21055, 2021.

[8] Hamed Karimi, Julie Nutini, and Mark Schmidt. Linear convergence of gradient and proximal-gradient methods under the polyak-łojasiewicz condition. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 795–811. Springer, 2016.

[9] Laurent Lessard, Benjamin Recht, and Andrew Packard. Analysis and design of optimization algorithms via integral quadratic constraints. *SIAM Journal on Optimization*, 26(1):57–95, 2016.

[10] Feng-Yi Liao, Lijun Ding, and Yang Zheng. Error bounds, pl condition, and quadratic growth for weakly convex functions, and linear convergences of proximal point methods. In *6th Annual Learning for Dynamics & Control Conference*, pages 993–1005. PMLR, 2024.

[11] Andrea Martin and Luca Furieri. Learning to optimize with convergence guarantees using nonlinear system theory. *IEEE Control Systems Letters*, 8:1355–1360, 2024.

[12] Andrea Martin, Ian R Manchester, and Luca Furieri. Learning to optimize with guarantees: a complete characterization of linearly convergent algorithms. *arXiv preprint arXiv:2508.00775*, 2025.

[13] Yurii Evgen'evich Nesterov. A method of solving a convex programming problem with convergence rate $O\left(\frac{1}{k^2}\right)$. In *Doklady Akademii Nauk*, volume 269, pages 543–547. Russian Academy of Sciences, 1983.

[14] Antonio Orvieto, Samuel L Smith, Albert Gu, Anushan Fernando, Caglar Gulcehre, Razvan Pascanu, and Soham De. Resurrecting recurrent neural networks for long sequences. In *International Conference on Machine Learning*, pages 26670–26698. PMLR, 2023.

[15] Ernest K Ryu and Stephen Boyd. Primer on monotone operator methods. *Appl. comput. math*, 15(1):3–43, 2016.

[16] Rajiv Sambharya, Georgina Hall, Brandon Amos, and Bartolomeo Stellato. Learning to warm-start fixed-point optimization algorithms. *Journal of Machine Learning Research*, 25(166):1–46, 2024.

[17] Rajiv Sambharya and Bartolomeo Stellato. Data-driven performance guarantees for classical and learned optimizers. *arXiv preprint arXiv:2404.13831*, 2024.

[18] Carsten Scherer and Christian Ebenbauer. Convex synthesis of accelerated gradient algorithms. *SIAM Journal on Control and Optimization*, 59(6):4615–4645, 2021.

[19] Bryan Van Scoy, Randy A Freeman, and Kevin M Lynch. The fastest known globally convergent first-order method for minimizing strongly convex functions. *IEEE Control Systems Letters*, 2(1):49–54, 2017.

[20] Bryan Van Scoy and Laurent Lessard. The fastest known first-order method for minimizing twice continuously differentiable smooth strongly convex functions. *arXiv preprint arXiv:2506.01168*, 2025.

[21] Hui Zhang and Wotao Yin. Gradient methods for convex minimization: better rates under weaker conditions. *arXiv preprint arXiv:1303.4645*, 2013.

# Technical Appendices and Supplementary Material

## A Further applications to classes of composite optimization problems

Our first additional result focuses on classes of possibly nonconvex functions for which standard gradient descent achieves monotonic linear convergence.

**Corollary 3** *Let $\mathcal{F}_{RSI}^{\beta,\mu}$ be the class of $\beta$-smooth functions satisfying the restricted secant inequality (RSI) with constant $\mu > 0$, that is, those for which it holds*

$$\nabla F(x)^\top (x - x^\star) \geq \frac{\mu}{2} \operatorname{dist}(x, \mathcal{X}^\star)^2 \,, \; \forall x \in \mathbb{R}^d \,, \tag{14}$$

*for any $x^\star$ in $\operatorname{argmin}_{y \in \mathcal{X}^\star} \operatorname{dist}(x, y)^2$. Let $\pi$ be the gradient descent update rule $\pi(F, \xi_t) = \xi_t - \eta \nabla F(\xi_t)$ with $\eta = \frac{\mu}{\beta^2}$, and $\gamma = \sqrt{1 - \frac{\mu^2}{\beta^2}}$. Then, any regular algorithm $\boldsymbol{\sigma} \in \mathtt{pExp}^{\pi}_{\mathcal{F}_{RSI}^{\beta,\mu}}(m, \gamma)$ can be written as*

$$x_{t+1} = \nu_t(F, x_{t:0}) = x_t - \eta \nabla F(x_t) + v_t(F, x_{t:0}) \,, \tag{15}$$

*with $\mathbf{v} \in \ell_{exp}(m, \gamma)$. Vice-versa, for any $\mathbf{v} \in \ell_{exp}(m, \gamma)$, the algorithm (15) is such that $\boldsymbol{\nu} \in \widehat{\mathtt{pExp}}^{\pi}_{\mathcal{F}_{RSI}^{\beta,\mu}}(\gamma)$.*

The result of Corollary 3 enables learning over the class of *all* the linearly convergent regular algorithms in $\mathtt{pExp}^{\pi}_{\mathcal{F}_{RSI}^{\beta,\mu}}(m, \gamma)$, while ensuring that the augmented algorithm (15) never leaves the class $\widehat{\mathtt{pExp}}^{\pi}_{\mathcal{F}_{RSI}^{\beta,\mu}}(\gamma)$, irrespectively of how "badly" the enhancement term $\mathbf{v} \in \ell_{exp}(m, \gamma)$ may be chosen.

A few comments regarding the generality of the class of functions $\mathcal{F}_{RSI}^{\beta,\mu}$ are in order. First, $\mathcal{F}_{RSI}$ encompasses certain nonconvex functions, as highlighted in [21]. Second, it holds that $\mathcal{F}_{SC}^{\beta,\mu} \subset \mathcal{F}_{cPL}^{\beta,\mu} \subset \mathcal{F}_{RSI}^{\beta,\mu}$ , where $\mathcal{F}_{SC}^{\beta,\mu}$ is the set of $\beta$-smooth and strongly convex functions complying with

$$F(y) \geq F(x) + \nabla F(x)^\top (y - x) + \frac{\mu}{2} |y - x|^2 \,, \tag{16}$$

for some $\mu > 0$, and $\mathcal{F}_{cPL}^{\beta,\mu}$ is the set of all the $\beta$-smooth and convex functions that comply with the Polyak–Łojasiewicz (PL) inequality, that is, for some $\mu > 0$ it holds that

$$F(x) - \min_{x \in \mathbb{R}^d} F(x) \leq \frac{1}{2\mu} |\nabla F(x)|^2 \,. \tag{17}$$

**Remark 1** *It is well known that $\mathcal{F}_{RSI}^{\beta,\mu} \subseteq \mathcal{F}_{PL}^{\beta,\frac{\mu^2}{4\beta}}$, where $\mathcal{F}_{PL}^{\beta,\mu}$ is the set of all possibly nonconvex functions satisfying* (17)*, see [8]. For functions in $\mathcal{F}_{PL}^{\beta,\mu}$, the gradient descent rule $\pi(F,x) = -\frac{1}{\beta}\nabla F(x)$ achieves linear convergence in the function value as per $F(x_t) - F^\star \leq \left(1 - \frac{\mu}{\beta}\right)^t (F(x_0) - F^\star)$. However, $\pi$ induces a monotonically linearly convergent sequence of iterates only if the restricted secant inequality* (14) *also holds, see [10].*

Our second additional result focuses on the class $\mathcal{F}_{cPL}^{\infty,\mu}$ of potentially nonsmooth proper, lower semi-continuous, convex functions that comply with the following inequality

$$F(x) - \min_{x\in\mathbb{R}^d} F(x) \leq \frac{1}{2\mu}\operatorname{dist}(0, \partial F(x))^2 \,, \tag{18}$$

where $\partial F(x)$ is the convex subdifferential of $F$ at $x$, defined as $\partial F(x) = \left\{s \in \mathbb{R}^d : F(y) \geq F(x) + s^\top(y-x)\,,\ \forall y \in \mathbb{R}^d\right\}$. In particular, note that (18) corresponds to (17) when $F$ is differentiable.

**Corollary 4** *Consider the class of functions $F \in \mathcal{F}_{cPL}^{\infty,\mu}$. Let $\pi$ be the proximal point algorithm performing the iterations*

$$x_{t+1} = \operatorname{prox}_F^c(x_t) = \min_{x\in\mathbb{R}^d}\ F(x) + \frac{1}{2c}|x - x_t|^2 \,, \tag{19}$$

*where $c > 0$. Let $\gamma = \min\left\{\frac{1}{\sqrt{1+c\mu}}, \frac{1}{\sqrt{1+\frac{c^2}{\beta\mu}}}\right\} \in (0,1)$. Then, any regular algorithm $\boldsymbol{\sigma} \in \mathtt{pExp}^{\pi}_{\mathcal{F}_{cPL}^{\infty,\mu}}(m,\gamma)$ can be written as*

$$x_{t+1} = \nu_t(F, x_{t:0}) = \operatorname{prox}_F^c(x_t) + v_t(F, x_{t:0}) \,, \tag{20}$$

*with $\mathbf{v} \in \ell_{exp}(m,\gamma)$. Viceversa, for any $\mathbf{v} \in \ell_{exp}(m,\gamma)$, the algorithm* (20) *is such that $\boldsymbol{\nu} \in \widehat{\mathtt{pExp}}^{\pi}_{\mathcal{F}_{RSI}^{\infty,\mu}}(\gamma)$.*

The result of Corollary 4 holds for any objective $F \in \mathcal{F}_{cPL}^{\infty,\mu}$. In particular, $\mathcal{F}_{cPL}^{\infty,\mu}$ encompasses the class of optimization problems (1), where $f \in \mathcal{F}_{SC}^{\beta,\mu}$ and $g \in \mathcal{F}_{C}^{\infty}$, that is, $g(x)$ is nonsmooth and convex, for which ad-hoc algorithms have been developed to exploit the structure underlying these composite problems.

# B Implementation details for numerical experiments

To complement our theoretical results, in this section we illustrate how one can in practice leverage our developed characterization to design new update rules that yield improved average-case performance compared to a baseline linearly convergent algorithm while preserving the worst-case linear convergence guarantees we established earlier. We begin by formulating a meta-optimization problem over the innovation sequence $v_t(\xi_{t\,:\,0})$, where we aim at minimizing the expected cost incurred by the augmented algorithm under a given data distribution while enforcing worst-case linear convergence. Beyond the special case in which algorithmic costs are quadratic in the iterates, innovation sequences are linear functions and the objective functions $F \in \mathcal{F}$ are quadratic, the meta-optimization remains nonconvex. This motivates parameterizing $v_t(\xi_{t\,:\,0})$ as an exponentially decaying neural-network update and learning its parameters via empirical algorithmic cost minimization. We conclude by discussing the significant speed-ups achieved by learned innovations through examples drawn from linear regression and linear quadratic MPC.

## B.1 Augmenting average-case performance of a baseline algorithm through NN updates

Let $\mathbb{D}_{\mathcal{F}}$ denote a distribution of objective functions $F \in \mathcal{F}$ and let $\pi \in \mathtt{pExp}^{\pi}_{\mathcal{F}}(m,\gamma)$ be a known linearly convergent baseline algorithm for $\mathcal{F}$. The problem of designing an augmentation signal $v_t$ to

enhance the average-case performance of $\pi$ relative to $\mathbb{D}_{\mathcal{F}}$ is expressed as

$$\min_{v_0, v_1, \ldots} \quad \mathbb{E}_{F \sim \mathbb{D}_{\mathcal{F}}} \left[ \texttt{AlgoCost}(F, \boldsymbol{\xi}) \right] \tag{21a}$$

$$\text{subject to} \quad \xi_{t+1} = \xi_t + \pi(F, \xi_t) + v_t(\xi_{t:0}), \tag{21b}$$

$$\boldsymbol{\xi} \in \ell_{exp}(\overline{m}, \overline{\gamma}), \ \forall F \in \mathcal{F}\,, \tag{21c}$$

$$\xi_t \in \mathcal{X}, \ \forall t \in \mathbb{N}, \ \forall \xi_0 \in \mathcal{X}\,, \tag{21d}$$

where $\overline{m} \geq m$ and $\overline{\gamma} \in [\gamma, 1)$ in (21c) specify a target linear convergence rate over all functions $F \in \mathcal{F}$, $\mathcal{X}$ denotes the feasibility set, and $\texttt{AlgoCost}(\cdot)$ measures the cost incurred by the augmented algorithm (21b) in optimizing a function $F \in \mathcal{F}$ when starting from $\xi_0 \in \mathbb{R}^d$. We refer to [16, 2, 17] for commonly used algorithm performance metrics and their corresponding generalization bounds on how the resulting empirical performance relates to the expectation in (21a).

In order to search over update functions $v_t(\xi_{t:0})$, and similar to the technique introduced in [11], it is convenient to decompose exponentially decaying augmentation terms $v_t(F, \xi_{t:0})$ as per

$$v_t(F, \xi_{t:0}) = M_t(F, \xi_0) D_t(F, \xi_{t:0})\,, \tag{22}$$

where $\mathbf{M}(F, \xi_0) \in \ell_{exp}(m, \gamma)$ must be an exponentially decaying magnitude term for any $\xi_0$, and $|D_t(F, \xi_{t:0})| \leq 1$ is an arbitrarily designed direction term. One can, for instance, employ a finite-dimensional parametrization of $v_t$ in (22) as per

$$v_t = \texttt{LRU}_t(\theta) \tanh(\texttt{LSTM}(\xi_t, \nabla F(\xi_t), F(\xi_t), \phi))\,, \tag{23}$$

where the $\texttt{LRU}_t(\theta)(v_{t:0})$ terms are generated by the linear recurrent unit (LRU) [14] defined as;

$$\begin{aligned} \zeta_{t+1} &= \Lambda\, \xi_t + \Gamma(\Lambda)\, B\, w_t, \\ \texttt{LRU}_t((w_{t:0}) &= \texttt{NN}(\mathbb{R}\text{e}(C\, \zeta_t) + D\, w_t, \psi) + F w_t\,, \end{aligned} \tag{24}$$

where $\mathbb{R}e$ denotes the real part operator and $\theta = (\Lambda, C, D, F, \psi)$, $w_0 = \xi_0$ and $w_t = 0$ for all $t \in [1, \infty)$. Exponential convergence of the sequence $v_t$ according to a rate $\overline{\gamma}$ is guaranteed by imposing that $\Lambda = \text{blkdiag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$ with $|\lambda_i| < \overline{\gamma}$ for every $i = 1, 2, \ldots, n$. The long short-term memory network (LSTM) architecture for the direction term in (23), with parameters $\phi$, is chosen due to its internal state that keeps memory of the sequence of its inputs, whereas the $\tanh$ clamps the magnitude of the direction term below 1 as per (22). When dealing with polyhedral constraints as per Corollary 2, one can use Agmon's iterative method [1] to enforce (11) after generating $v_t$ through (23).

In practice, letting $\mathcal{F}_{\text{train}} = \{F_1, F_2, \ldots, F_M\}$ be a set of example optimization problems drawn from $\mathbb{D}_{\mathcal{F}}$, one can approximate expectation (21a) as

$$\sum_{F \in \mathcal{F}_{\text{train}}} \texttt{AlgoCost}(F, \boldsymbol{\xi}(\theta, \phi))\,, \tag{25}$$

where the sequence $\boldsymbol{\xi}(\theta, \phi)$ is generated as per (21b) with $v_t(\xi_{t:0}, \theta, \phi)$ parametrized as per (22)-(23) and where linear convergence (21c) is guaranteed for any choice of $\theta$ and $\phi$. Automatic differentiation and NN optimizers can then be used in a standard way to improve over the empiric algorithm cost (25) over the parameters $(\theta, \phi)$.

**Remark 2** *Characterizing the gap between empiric algorithm cost and average cost is beyond the scope of the paper, as this aspect does not interact with the main results of Section 3. A corresponding analysis has recently been proposed in [17] when the algorithm cost is chosen as an indicator function on whether* `AlgoCost` *exceeds a certain threshold $\epsilon > 0$. We detail how these results apply to our framework in our technical report [12].*

### B.2 Augmenting projected gradient descent for MPC

In our second example, we consider a model predictive control setting where only a limited number of optimization steps can be executed in real-time. In particular, we study a discrete-time linear dynamical system described by the state space equation

$$x_{t+1} = A x_t + B u_t + w_t = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} x_t + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u_t + w_t\,, \tag{26}$$

where $x_t \in \mathbb{R}^2$ is the system state, $u_t \in \mathbb{R}$ is the control input, and $w_t \in \mathbb{R}^2$ represents a zero-mean Gaussian process noise term.

The goal at each time step is to solve the following finite-horizon linear quadratic control problem:

$$\min_{u_0,\dots,u_{T-1}} \sum_{t=0}^{T-1} x_t^\top Q x_t + u_t^\top R u_t + x_T^\top Q_T x_T \tag{27a}$$

$$\text{subject to } x_{t+1} = Ax_t + Bu_t\,, \tag{27b}$$

$$u_t \in \mathcal{U}_t\,, x_t \in \mathcal{X}\,, x_0 \in \mathbb{R}^2\,; \tag{27c}$$

for simplicity, we assume that the weighting matrices $Q$, $R$ and $Q_T$ are identity matrices of appropriate dimensions. We further set $\mathcal{U}_t = \{u : ||u||_\infty \leq 0.25\}$ and $\mathcal{X}_t = \mathbb{R}^2$ to account for actuation constraints yet sidestep recursive feasibility issues for ease of exposition, as our focus lies in efficiently solving the underlying quadratic program.

By introducing the stacked notation $\mathbf{u} = [u_0^\top \dots u_{T-1}^\top]^\top$ and $\mathbf{x} = [x_1^\top \dots x_T^\top]^\top$, (27) can be equivalently rewritten as:

$$\min_{\mathbf{u}} \ \mathbf{u}^\top(\mathbf{G}^\top\mathbf{Q}\mathbf{G} + \mathbf{R})\mathbf{u} + 2x_0^\top\mathbf{F}^\top\mathbf{Q}\mathbf{G}\mathbf{u} + x_0^\top\mathbf{F}^\top\mathbf{Q}\mathbf{F}x_0 \tag{28a}$$

$$\text{subject to } \mathbf{x} = \mathbf{F}x_0 + \mathbf{G}\mathbf{u}, \mathbf{u} \in \boldsymbol{\mathcal{U}}, x_0 \in \mathbb{R}^2\,, \tag{28b}$$

where $\mathbf{Q}$, $\mathbf{R}$, $\mathbf{F}$, and $\mathbf{G}$ are block matrices encoding the system dynamics and cost over a prediction horizon of length $T = 20$.

To solve the quadratic program above, we start from an initial guess $\mathbf{u}^{(0)}$ equal to zero and employ the projected gradient descent (PGD) method (10) with step size $\eta = \frac{1}{\lambda_{\max}(\mathbf{G}^\top\mathbf{Q}\mathbf{G}+\mathbf{R})} \approx 3.8 \cdot 10^{-5}$. Using this as our baseline optimization algorithm, we then learn an augmentation term $v_t$ parametrized as per (23) to minimize the total average cost (28a) over 100 optimization steps $\mathbf{u}^{(1)}, \dots, \mathbf{u}^{(100)}$ when each component of the initial state $x_0$ of the system (26) is drawn from a zero-mean Gaussian random variable with standard deviation of 0.5. Specifically, we perform meta-optimization using Adam with a learning rate of $5 \cdot 10^{-3}$ for 65 epochs. As highlighted in the left panel of Figure 1, while both algorithms converge in a small neighborhood of the solution of (28) after 100 iterations, our learned optimizer showcase improved transient performance. Numerically, we verify that this is because during training $v_t$ learns to promptly saturate the constraint $u_t \in \mathcal{U}_t$ when needed, leading to better performance in the testing phase. Remarkably, as shown in the right panel of Figure 1, we observe a similar trend even when we close the loop between the linear dynamical system (26) and our learned optimizer. In particular, despite the repeated solution of (28) in a receding horizon fashion induces a new distribution over the state space, using our learned optimizer – trained on initial conditions $x_0$ randomly drawn from a Gaussian distribution – enables a significant reduction of the closed-loop cost incurred over an horizon of length $N = 30$ when only a limited optimization steps can be performed in real-time.

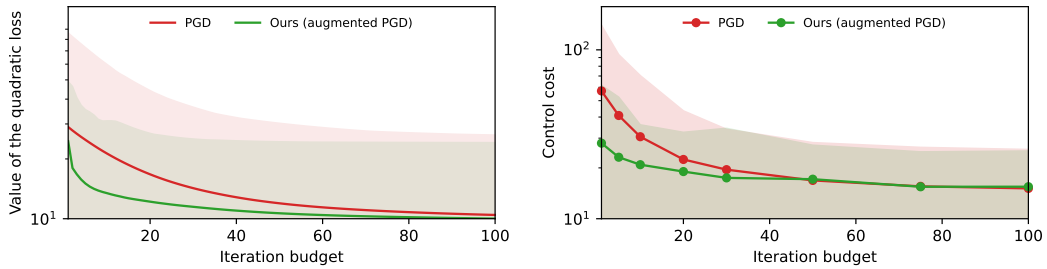



Figure 1: Comparison between the classical projected gradient descent method and its augmented version in solving (28) with $x_0$ drawn from a Gaussian distribution (on the left) and $x_0 = x_t$ in online MPC applications (on the right). Solid lines represent mean values, and shaded areas cover values up to the 90% percentile.

# C Technical proofs

## C.1 Proof of Theorem 1

We first prove the result by assuming that $\pi \in \texttt{Exp}^{\pi}_{\mathcal{F}}(\gamma)$. This is instrumental towards establishing the general result. Let $\delta_t = \operatorname{dist}(\xi_t, \texttt{Fix}_\pi)$ for compactness. By the algorithm definition (5) and the triangle inequality, we have that for every $F \in \mathcal{F}$

$$\begin{aligned}\delta_t &= \operatorname{dist}(\pi(F, \xi_{t-1}) + v_{t-1}, \texttt{Fix}_\pi) = \inf_{c \in \texttt{Fix}_\pi} \operatorname{dist}(\pi(F, \xi_{t-1}) + v_{t-1}, c) \\ &\leq \inf_{c \in \texttt{Fix}_\pi} \operatorname{dist}(\pi(F, \xi_{t-1}), c) + |v_{t-1}| = \operatorname{dist}(\pi(F, \xi_{t-1}), \texttt{Fix}_\pi) + |v_{t-1}| \,.\end{aligned}$$

Assuming that $\pi \in \texttt{Exp}^{\pi}_{\mathcal{F}}(\gamma)$, we have that (4) holds with $p(t) = 1$. It follows that $\delta_t \leq \gamma\delta_{t-1} + |v_{t-1}|$. Iterating this inequality, we deduce that

$$\delta_t \leq \gamma^t \delta_0 + \sum_{k=0}^{t-1} \gamma^k |v_{t-1-k}| \leq \gamma^t \delta_0 + \sum_{k=0}^{t-1} \gamma^k p_v(t-1-k)\gamma^{t-1-k} \leq \gamma^t \left( \delta_0 + \frac{1}{\gamma} \sum_{k=0}^{t-1} p_v(k) \right) ,$$

where we used the fact that $\mathbf{v} \in \ell_{exp}(m, \gamma)$. Let $q(t) = \sum_{k=0}^{t} p_v(k)$ and note that the right-hand side of the above can be written as $\gamma^t r(t)$ where $r(t) = \delta_0 + \frac{1}{\gamma} q(t-1)$. We study $q(t)$. By linearity of summation, $q(t)$ can be equivalently rewritten as

$$\sum_{k=0}^{t} \sum_{j=0}^{m} a_j k^j = a_m \sum_{k=0}^{t} k^m + \cdots + a_1 \sum_{k=0}^{t} k + a_0 \sum_{k=0}^{t} 1 \,,$$

where $a_j \in \mathbb{R}$ with $j \in \{0, \ldots, m\}$ is the $j$-th coefficient of the polynomial $p_v(\cdot)$. Faulhaber's formula implies that $q(t)$ is a polynomial of degree $m+1$ in the variable $t$ with coefficient $b_{m+1} = \frac{a_m}{m+1}$. Furthermore, $q(t)$ is positive and monotonically non-decreasing by construction, that is, $q(t) \in \mathcal{P}_{m+1}(t)$. Note that $q(t) \in \mathcal{P}_{m+1}(t)$ implies $r(t) \in \mathcal{P}_{m+1}(t)$. Hence, we conclude that $\delta_t \leq r(t)\gamma^t$ for all $t \in \mathbb{N}$, which proves the result for the case $\pi \in \texttt{Exp}^{\pi}_{\mathcal{F}}(\gamma)$.

We now turn our attention to the general case where $\pi$ is any linearly convergent algorithm in $\texttt{pExp}^{\pi}_{\mathcal{F}}(m, \gamma)$ certified by a polynomial $p(t) \in \mathcal{P}_m(t)$. We distinguish two cases. In the case $p(1)\gamma < 1$, we have that $\pi \in \texttt{pExp}^{\pi}_{\mathcal{F}}(m, \gamma)$ also lies in $\texttt{Exp}^{\pi}_{\mathcal{F}}(p(1)\gamma)$, and the analysis above applies with rate $p(1)\gamma$ in place of $\gamma$.

In the case $p(1)\gamma \geq 1$, we necessarily have than $N \geq 2$. For any $w_t \in \ell_{exp}(m, \gamma)$, consider the recursion

$$\zeta_{k+1} = \pi^N(F, \zeta_k) + w_t \,, \tag{29}$$

where $\pi^N(\cdot)$ denotes the repeated application of $\pi$ defined as $\pi^N(F, \zeta_k) = \pi(F, \pi^{N-1}(F, \zeta_k))$ with $\pi^1(F, \zeta_k) = \pi(F, \zeta_k)$.

We first observe that, if $\zeta_0 = \xi_0$ and $\mathbf{v}$ is constructed as per (7), then (29) is equivalent to (5) in the sense that $\zeta_k = \xi_{Nk}$ for every $k \in \mathbb{N}$. By construction, $\pi^N \in \texttt{Exp}^{\pi}_{\mathcal{F}}(\rho)$ and therefore complies with (4) with $p(t) = 1$ and $\gamma = \rho$. Hence, as proven above, it holds that

$$\operatorname{dist}(\boldsymbol{\zeta}, \texttt{Fix}_{\pi^N}) \in \ell_{exp}(m+1, \rho) \,.$$

We now argue that $\texttt{Fix}_{\pi^N} = \texttt{Fix}_\pi$. Clearly, $\texttt{Fix}_{\pi^N} \supseteq \texttt{Fix}_\pi$ since $\xi^\star = \pi(F, \xi^\star)$ for every $\xi^\star \in \texttt{Fix}_\pi$ and thus $\xi^\star = \pi^N(F, \xi^\star)$. To show that $\texttt{Fix}_{\pi^N} \subseteq \texttt{Fix}_\pi$, assume there exists $\zeta^\star \in \texttt{Fix}_{\pi^N}$ such that $\zeta^\star \notin \texttt{Fix}_\pi$. Since $\pi \in \texttt{pExp}^{\pi}_{\mathcal{F}}(m, \gamma)$, we have that $\lim_{t\to\infty} \operatorname{dist}(\pi^t(F, \zeta^\star), \texttt{Fix}_\pi) = 0$. At the same time, $\operatorname{dist}(\pi^{\tau N}(F, \zeta^\star), \texttt{Fix}_\pi) > 0$ for any $\tau \in \mathbb{N}$ because $\pi^{\tau N}(F, \zeta^\star) = \zeta^\star \notin \texttt{Fix}_\pi$. This is a contradiction, and thus $\texttt{Fix}_{\pi^N} = \texttt{Fix}_\pi$. We conclude that

$$\operatorname{dist}(\boldsymbol{\zeta}, \texttt{Fix}_\pi) \in \ell_{exp}(m+1, \rho) \,,$$

and therefore there exists a polynomial $q(k) \in \mathcal{P}_{m+1}(k)$ such that $\operatorname{dist}(\zeta_k, \texttt{Fix}_\pi) \leq q(k)\rho^k$ for all $k \in \mathbb{N}$.

Next, we note that, for any $s \in \{1, \ldots, N-1\}$

$$\begin{aligned}\operatorname{dist}(\xi_{Nk+s}, \texttt{Fix}_\pi) &= \operatorname{dist}(\pi^s(F, \xi_{Nk}), \texttt{Fix}_\pi) \\ &\leq p(s)\gamma^s \operatorname{dist}(\xi_{Nk}, \texttt{Fix}_\pi) \leq p(s)\gamma^s q(k)\rho^k \,,\end{aligned}$$

where we used the fact that $\operatorname{dist}(\xi_{Nk}, \texttt{Fix}_\pi) = \operatorname{dist}(\zeta_k, \texttt{Fix}_\pi)$ for any $k \in \mathbb{N}$. Letting $t = Nk + s$, and using the fact that $p(\cdot), q(\cdot) \in \mathcal{P}_{m+1}$, we obtain

$$\begin{aligned}\operatorname{dist}(\xi_t, \texttt{Fix}_\pi) &\leq p(N-1)\gamma q\left(\left\lfloor \frac{t-s}{N} \right\rfloor\right) \rho^{\lfloor \frac{t-s}{N} \rfloor} \\ &\leq p(N-1)\gamma q\left(\frac{t}{N}\right) \rho^{\lfloor \frac{t-N+1}{N} \rfloor} \leq \underbrace{\frac{p(N-1)\gamma}{\rho^{2-\frac{1}{N}}} q\left(\frac{t}{N}\right)}_{r(t) \in \mathcal{P}_{m+1}(t)} \left(\rho^{\frac{1}{N}}\right)^t .\end{aligned}$$

Since $\rho = p(N)\gamma^N$, we have that $\rho^{\frac{1}{N}} = \sqrt[N]{p(N)}\gamma$. This concludes the proof.

### C.2 Proof of Theorem 2

Let $v_t(F, \xi_{t:0}) = -\pi(F, \chi_t) + \sigma_t(F, \chi_{t:0})$. We first show by induction that $\xi_t = \chi_t$ at all times, starting from the base case $\xi_0 = \chi_0$, which holds by construction. Assume now that $\xi_{t:0} = \chi_{t:0}$. We aim to prove that $\xi_{t+1} = \chi_{t+1}$. This holds because

$$\xi_{t+1} = \pi(F, \xi_t) - \pi(F, \chi_t) + \sigma_t(F, \chi_{t:0}) = \sigma_t(F, \chi_{t:0}) = \chi_{t+1} .$$

It remains to show that the sequence $v_t(F, \xi_{t:0}) = -\pi(F, \chi_t) + \sigma_t(\chi_{t:0})$ belongs to $\ell_{exp}(m, \gamma)$. To prove this, we rewrite $v_t$ as

$$v_t = -(\pi(F, \chi_t) - \chi_t) + \sigma_t(F, \chi_{t:0}) - \chi_t . \tag{30}$$

Since $\pi(F, \cdot)$ is Lipschitz continuous, letting $\chi_t^p$ be any element of $\operatorname{argmin}_{\chi \in \texttt{Fix}_\pi} |\chi - \chi_t|^2$, there exists a constant $L_\pi \in \mathbb{R}_+$ such that

$$\begin{aligned}|\pi(F, \chi_t) - \chi_t| &= |\pi(F, \chi_t) - \chi_t^p + \chi_t^p - \chi_t| = |\pi(F, \chi_t) - \pi(F, \chi_t^p) + \chi_t^p - \chi_t| \\ &\leq (L_\pi + 1)|\chi_t - \chi_t^p| = (L_\pi + 1) \operatorname{dist}(\chi_t, \texttt{Fix}_\pi) ,\end{aligned}$$

and hence $-(\pi(F, \boldsymbol{\chi}) - \boldsymbol{\chi}) \in \ell_{exp}(m, \gamma)$. We further have that $\boldsymbol{\sigma}(F, \boldsymbol{\chi}) - \boldsymbol{\chi} \in \ell_{exp}(m, \gamma)$ by the regularity assumption on $\boldsymbol{\sigma}$ as per Definition 2. Since the sum of signals in $\ell_{exp}(m, \gamma)$ belongs to $\ell_{exp}(m, \gamma)$, we conclude the proof by inspection of (30).

### C.3 Proof of Corollary 1

It is well known that the NAG algorithm (9) applied to the class $\mathcal{F}_{SC}^{\beta,\mu}$ with the parameters $\alpha$ and $\eta$ as above is such that $\pi \in \texttt{pExp}_{\mathcal{F}}^{\pi}(0, \gamma)$, see [13, 9]. Since $\sqrt[N]{p(N)} < \tau < \frac{1}{\gamma}$, we have that $p(N)\gamma^N < 1$ and Theorem 1 applies.

### C.4 Proof of Corollary 4

We begin by showing the baseline algorithm $\pi(F, x_t)$ in (19) is 1-Lipschitz continuous in $x_t$. Let $u = \pi(F, x)$ and $v = \pi(F, y)$. Since $u$ and $v$ are obtained by solving the minimization (19), it holds that $0 \in \partial F(u) + \frac{1}{c}(u - x)$ and $0 \in \partial F(v) + \frac{1}{c}(v - y)$ by the first-order optimality conditions. In other words, there exist subgradients $g_u \in \partial F(u)$ and $g_v \in \partial F(v)$ such that $x - u = cg_u$ and $y - v = cg_u$. Since $F$ is closed, convex, and proper, $\partial F$ is maximally monotone [15] and $(g_u - g_v)^\top (u - v) \geq 0$. Observe that

$$\begin{aligned}(g_u - g_v)^\top (u - v) &= \frac{1}{c}(x - u - (y - v))^\top (u - v) \\ &= \frac{1}{c}(x - y)^\top (u - v) - \frac{1}{c}|u - v|^2 \geq 0 .\end{aligned}$$

Therefore, we have $(x - y)^\top (u - v) \geq |u - v|^2$. On the other hand, by the Cauchy-Schwarz inequality, $|x - y||u - v| \geq (x - y)^\top (u - v)$. Combining the two last inequalities, we have $|u - v|^2 \leq |x - y||u - v|$, from which 1-Lipschitz continuity follows. Similarly to Corollary 3, the result then follows by combining our Theorem 2 with the linear convergence result of the proximal point method (19) when applied to functions $F \in \mathcal{F}_{cPL}^{\infty,\mu}$ from [10, Theorem 4.2] and the definitions of error bound and quadratic growth from [8].

### C.5 Proof of Corollary 2

The baseline algorithm $\pi$ defined in (10) is such that $\pi \in \texttt{Exp}^{\pi}_{\mathcal{F}_{\text{comp}}}(1-\eta\mu)$ as shown in [5, Theorem 11.5]. Now, consider any regular algorithm $\chi_{t+1} = \sigma_t(F, \chi_{t:0})$ with feasible iterates $\chi_t \in \mathcal{X}$ and such that $\boldsymbol{\sigma} \in \texttt{pExp}^{\pi}_{\mathcal{F}_{\text{comp}}}(m, \gamma)$, with $\gamma = 1 - \eta\mu$. Its iterates are equivalent to those of (12) by choosing

$$v_t = -\pi(F, \chi_t) + \sigma_t(F, \chi_{t:0}) \,, \quad x_0 = \chi_0 \,. \tag{31}$$

Next, we verify that $A_i v_t \leq b_i - A_i \operatorname{proj}_{\mathcal{X}}(x_t - \eta\nabla f(x_t))$ for all $i \in [1, M]$. We have

$$\begin{aligned} A_i v_t &= -A_i \pi(F, \chi_t) + A_i \sigma_t(F, \chi_{t:0}) \\ &= -A_i \operatorname{proj}_{\mathcal{X}}(\chi_t - \eta\nabla f(\chi_t)) + A_i \chi_{t+1} \,. \end{aligned} \tag{32}$$

By definition, $A_i \chi_{t+1} \leq b_i$ because $\chi_{t+1}$ lies in $\mathcal{X}$ and the claim follows by direct substitution in (32) since the sequence $\chi_t$ is equal to $x_t$. Next, we verify that $\pi(F, x)$ is Lipschitz in $x$. Since the projection onto an affine subspace is 1-Lipschitz, it holds that

$$\begin{aligned} |\pi(F, x) - \pi(F, y)| &\leq |x - y + \eta\nabla f(y) - \eta\nabla f(x)| \\ &\leq (1 + \eta\beta)|x - y| \,. \end{aligned}$$

Last, analogously to the proof of Theorem 2, it holds that $\mathbf{v} \in \ell_{exp}(m, 1-\eta\mu)$ because $\boldsymbol{\sigma}$ is regular. Viceversa, if $\mathbf{v} \in \ell_{exp}(m, \gamma)$ is such that at all times $A_i v_t \leq b_i - A_i \operatorname{proj}_{\mathcal{X}}(x_t - \eta\nabla f(x_t))$ for all $i \in [1, M]$, the iterates of (12) are feasible because

$$A_i x_{t+1} = A_i \operatorname{proj}_{\mathcal{X}}(x_t - \eta\nabla f(x_t)) + A_i v_t \leq b_i \,, \tag{33}$$

and $\boldsymbol{\nu} \in \widehat{\texttt{pExp}}^{\pi}_{\mathcal{F}_{\text{comp}}}(1-\eta\mu)$ by Theorem 1.

### C.6 Proof of Corollary 3

By Theorem 2.1 of [10], it holds that (4) holds for the gradient descent algorithm $\xi_{t+1} = \pi(F, \xi_t) = \xi_t - \eta\nabla F(\xi_t)$ with $\eta = \frac{\mu}{\beta^2}$ with $\gamma = \sqrt{1 - \frac{\mu^2}{\beta^2}} \in (0, 1)$. Further, we have that $\pi(F, \xi_t)$ is Lipschitz continuous since $|\pi(F, x) - \pi(F, y)| = |x - y - \eta\nabla F(x) + \eta\nabla F(y)| \leq (1 + \eta\beta)|x - y|$, where the last inequality follows from the $\beta$-smoothness of $F \in \mathcal{F}^{\beta,\mu}_{RSI}$. The result then follows by applying Theorem 2.

## NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Both abstract and introduction reflect the main contribution and scope by defining the notion of linear convergence as well as the classes of optimization problems these results apply to. Insights are provided on how these results are obtained.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   [Yes]

   Justification: The Theorem and Corollaries accurately define the classes of optimization problems the results apply to. The scope of the results is transparently presented throughout the paper. In terms of improved algorithm performance with respect to classical optimizer after training, it is made clear that such results strongly depend on the dataset of problems augmentations are trained for as well as the training method. While linear convergence is proven formally and therefore holds universally for the considered class of problems and algorithm augmentations, it is stated several times that results of improved performance are valid for the instances we have tested in simulation, with no claim of universal improvement.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: Our proofs are accurate and correct, to the best of our knowledge. All mathematical passages are fully explained and justified, with proper referencing where necessary.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We disclose all the information regarding our numerical experiments in a dedicated Appendix in the additional material.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We disclose our open-source code.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We report all these information in a dedicated Appendix in the additional material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: We report confidence standard deviation shown as an interval in Figure 1. In Figure 2 we report 90% percentiles.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: All numerical experiments were performed on a MacBook Pro with 2.3 GHz Intel Core i9 8 core and 32 GB 2667 MHz DDR4. The meta-training of our algorithms can be executed in less that one hour.

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: No sensitive domains or data are touched within the paper.

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA] .

    Justification: The paper is a theoretical grounded on mathematical definitions of the optimization problems involve. Specific applications of such optimization problems are not in the scope of the work.

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper poses no such risks.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: For our numerical experiments, we use data from the SuiteSparse Matrix Collection, which is properly referenced.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: The paper does not release new assets.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.