Amortized Nonlinear Model Predictive Control

Francesco Pillitteri, Alberto Bemporad

***Abstract*— Nonlinear Model Predictive Control (MPC) requires solving a constrained *nonlinear program* (NLP) in real-time at every sampling instant, a computational bottleneck that often limits its deployment on resource-constrained hardware or at high sampling rates. For the broad class of input-affine nonlinear systems, we address this challenge and show that the MPC move can be approximately computed by solving a small *quadratic program* (QP) whose cost parameters depend on the current state and reference. We propose a single-network *residual-corrector* architecture: a state-dependent analytic baseline provides initial QP matrices, and the network learns only the corrections needed to match the full NLP solution. The network is trained on data generated by an NLP solver using a hybrid loss that combines supervised imitation and KKT-residual penalties; a differentiable interior-point QP layer guarantees that the control action satisfies the input constraints by construction. We validate the approach on a three-link planar robotic arm with Cartesian end-effector tracking, demonstrating orders-of-magnitude speedup over the NLP solver while maintaining comparable tracking performance.**

I. Introduction

Model Predictive Control (MPC) is widely recognized as a powerful approach to constrained optimal control: at each time step, the controller solves a finite-horizon optimization problem, applies the first element of the optimal input sequence, and repeats the process at the next step [1]. From the perspective of optimization theory, MPC is inherently a *parametric* program: the current state x_k (and any reference vector and measured disturbance) acts as the parameter, and the optimal first input $u_0^\star(x_k)$ is the corresponding solution map. For linear systems with quadratic costs and polyhedral constraints, the resulting program is a *convex QP* that can be solved in milliseconds or less by modern active-set [2], [3] or interior-point solvers [4]. An explicit piecewise-affine solution map can even be pre-computed offline via *multi-parametric programming* [5] to avoid online optimization altogether.

For nonlinear plant models, the problem is a nonlinear program (NLP), whose per-step solve time may be orders of magnitude larger, often rendering real-time implementation infeasible, and explicit nonlinear MPC is in general intractable. A classical workaround is to perform only a single Newton-type iteration per step, warm-starting from the previous solution shifted by one time step. This underpins *real-time iteration* (RTI) schemes [6], [7], which have proved effective in many applications, though the per-step cost can still be prohibitive at high sampling rates or on resource-constrained hardware, due to the computational complexity of constructing a QP problem and solving it with respect to the sequence of control moves in real-time.

Recent advances in machine learning offer an alternative route: to use function approximation to replace the online solver, in analogy to explicit MPC, or to substantially accelerate it by providing a good initial guess. This idea, also referred to as *amortized optimization* [8], has attracted growing interest for MPC due to the microsecond inference times it enables once the approximation is trained offline. A related approach to reduce the online computational burden in nonlinear MPC is to approximate the cost-to-go function (see, e.g., [9], [10]).

Due to the approximation, preserving *closed-loop guarantees* can be a challenge, and several strategies have been explored in the literature. For example, stability and feasibility can be recovered by augmenting the approximator with a safety filter [11]. Alternatively, learning a warm-start rather than the full solution preserves problem feasibility while reducing iteration counts [12]. Differentiable optimization layers embed a QP as a differentiable mapping inside the network, guaranteeing that the output satisfies the QP constraints by construction [13]. Our method belongs to this last family.

Contribution. We propose a method to learn a network whose outputs are the matrices of a *small QP*, whose unknowns correspond only to the first control move rather than the full control sequence. By using a differentiable QP layer during offline training, constraint enforcement is not a concern during training, as it is delegated to the solver adopted to solve the small QP online. After formalizing the MPC problem for general input-affine nonlinear systems and showing how the input-affine structure leads to a *parametric QP* that provides a structured approximation of the first control step, we propose a *residual-corrector QP architecture*, which learns how to correct a state-dependent QP baseline obtained analytically. Hard constraints are enforced by the QP solver, while state constraints are softened to prevent possible infeasibility. The network is trained offline on NLP solutions via a *hybrid KKT loss* combining supervised imitation and KKT-residual penalties. We validate the approach on a three-link planar robotic arm with Cartesian end-effector tracking, demonstrating orders-of-magnitude speedup over the NLP solver while maintaining comparable tracking performance and constraint satisfaction.

The paper is organized as follows. Section II formulates the MPC problem for input-affine systems. Section III

This work was funded by the European Union (ERC Advanced Research Grant COMPACT, No. 101141351). Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

The authors are with the IMT School for Advanced Studies, Lucca, Italy. Email: francesco.pillitteri@imtlucca.it

presents the proposed architecture and training procedure. Section IV reports simulation results on the robotic arm. Section V concludes the paper.

Notation. We denote by I_n the identity matrix of order n and by $\mathrm{diag}(v)$ the diagonal matrix formed from vector v. For a matrix $Q = Q^\top \succeq 0$, $\|a\|_Q^2 := a^\top Q\, a$ and for $Q \succ 0$ we denote by $L = \mathrm{chol}(Q)$ its lower-triangular Cholesky factor, $Q = LL^\top$. The operator $[\cdot]^+ = \max(\cdot, 0)$ and $[\cdot]^- = \max(-\cdot, 0)$ extract the positive and negative parts, respectively.

II. Problem Formulation

A. Input-Affine Nonlinear Models

We consider discrete-time input-affine nonlinear prediction models of the form

$$x_{k+1} = f(x_k) + G(x_k)\, u_k, \tag{1}$$

where $x_k \in \mathbb{R}^{n_x}$ is the state vector, $u_k \in \mathbb{R}^{n_u}$ the control input, and $f : \mathbb{R}^{n_x} \to \mathbb{R}^{n_x}$ and $G : \mathbb{R}^{n_x} \to \mathbb{R}^{n_x \times n_u}$ define the state-update equation. Affine systems (1) arise naturally in many engineering domains, such as in robotics; at the price of introducing an input delay, any general dynamics $x_{k+1} = F(x_k, u_k)$ can always be transformed into affine form, for example by defining the additional state $x_k^u = u_{k-1}$ and setting $x_{k+1} = F(x_k, x_k^u)$, $x_{k+1}^u = u_k$.

We assume that (1) is subject to constraints, namely *box constraints*

$$\begin{aligned} \mathcal{U} &= \{u \in \mathbb{R}^{n_u} : \underline{u} \le u \le \overline{u}\}, \\ \mathcal{X} &= \{x \in \mathbb{R}^{n_x} : \underline{x} \le x \le \overline{x}\}, \end{aligned} \tag{2}$$

and *affine-in-input* constraints

$$A_c(x_k)\, u_k \le b_c(x_k), \tag{3}$$

where $A_c(x_k) \in \mathbb{R}^{n_c \times n_u}$ and $b_c(x_k) \in \mathbb{R}^{n_c}$. Constraints of the form (3) capture, for example, joint-torque limits expressed via inverse dynamics ($\tau = M(q)u + h(q, \dot{q})$), as well as mixed state/input polytopic constraints that become affine in u_k for any given x_k.

B. Receding-Horizon Optimal Control Problem

At each time step k, MPC requires solving the following finite-horizon optimal control problem parameterized by the current state x_k and a task reference r_k:

$$\min_{\{x_t\},\{u_t\}} \; V_f(x_N;\, r_k) + \sum_{t=0}^{N-1} \ell(x_t, u_t;\, r_k) \tag{4a}$$

s.t.

$$x_{t+1} = f(x_t) + G(x_t)\, u_t, \quad t = 0, \dots, N-1, \tag{4b}$$

$$x_0 = x_k, \tag{4c}$$

$$\underline{u} \le u_t \le \overline{u}, \quad t = 0, \dots, N-1, \tag{4d}$$

$$\underline{x} \le x_t \le \overline{x}, \quad t = 1, \dots, N, \tag{4e}$$

$$A_c(x_t)\, u_t \le b_c(x_t), \quad t = 0, \dots, N-1. \tag{4f}$$

where, with a slight abuse of notation, t denotes the prediction time $k + t$, $\ell : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}_{\ge 0}$ is a stage cost and $V_f : \mathbb{R}^{n_x} \to \mathbb{R}_{\ge 0}$ is a terminal cost, both depending on the task reference r_k. MPC applies $u_k = u_0^\star$, i.e., only the first element of the optimal input sequence, and solves problem (4) again at the next step $k + 1$.

By collecting all the optimization variables into a single vector

$$w = \begin{bmatrix} x_0{}^\top & u_0{}^\top & x_1{}^\top & u_1{}^\top & \dots & x_{N-1}{}^\top & u_{N-1}{}^\top & x_N{}^\top \end{bmatrix}^\top \tag{5}$$

with $w \in \mathbb{R}^{n_w}$ and $n_w = (N+1)n_x + Nn_u$, Problem (4) can be written more compactly as the NLP

$$\begin{aligned} w^\star = \arg\min_{w} \; & J(w;\, r_k) \\ \text{s.t.} \quad & g_{\mathrm{eq}}(w;\, x_k) = 0, \\ & \underline{h} \le h(w;\, x_k) \le \overline{h}, \\ & \underline{w} \le w \le \overline{w}, \end{aligned} \tag{6}$$

where J is the total cost, g_{eq} collects all equality constraints (4b)–(4c), h collects general inequality constraints affine in the input, and $\underline{w}, \overline{w}$ encode the box constraints (2).

C. KKT Conditions of the NLP

The standard first-order KKT conditions for (6) require stationarity, primal/dual feasibility, and complementarity. Denoting the Lagrange multipliers for equality constraints, ranged inequalities, and variable bounds by μ, λ, and $z_L, z_U \ge 0$ (split into lower and upper components following the `IPOPT` convention [14]) respectively, stationarity reads

$$\nabla_w \mathcal{L}(w^\star, \mu^\star, \lambda^\star, z_L^\star, z_U^\star) = 0, \tag{7}$$

where $\mathcal{L} = J + \mu^\top g_{\mathrm{eq}} + \lambda^\top h + z_L{}^\top(\underline{w} - w) + z_U{}^\top(w - \overline{w})$. Primal/dual feasibility and complementarity conditions can be written in smooth form via the *Fischer–Burmeister* (FB) function [15]

$$\phi_{\mathrm{FB}}(a, b) = \sqrt{a^2 + b^2} - (a + b), \tag{8}$$

where $\phi_{\mathrm{FB}}(a, b) = 0 \iff a \ge 0,\ b \ge 0,\ ab = 0$.

D. Input-Affine Structure and Single-Step QP

The input-affine structure of (1) implies that the one-step state map is *affine* in u_0 for any fixed x_0:

$$x_1 \;=\; f_0 + G_0\, u_0, \tag{9}$$

where $f_0 \in \mathbb{R}^{n_x}$ is the drift under zero input and $G_0 \in \mathbb{R}^{n_x \times n_u}$ is the control-gain matrix, both evaluated at x_0.

Every constraint that is affine in u_0 for a fixed x_0 takes the form (3). We distinguish two roles. *Hard* constraints for input bounds $\underline{u} \le u_0 \le \overline{u}$ and any additional affine-in-input bound are encoded directly as

$$A_c(x_0)\, u_0 \le b_c(x_0). \tag{10}$$

Instead, we model the bounds $\underline{x} \le x_1 \le \overline{x}$ on the first predicted state as *soft* constraints

$$A_s(x_0)\, u_0 - s \le b_s(x_0), \quad A_s = \begin{bmatrix} G_0 \\ -G_0 \end{bmatrix}, \quad b_s = \begin{bmatrix} \overline{x} - f_0 \\ f_0 - \underline{x} \end{bmatrix}. \tag{11}$$

where $s \in \mathbb{R}^{n_s}_{\geq 0}$ is a vector of non-negative slack variables (with $n_s = 2n_x$), used to ensure the feasibility for all x_0 of the resulting *state-dependent QP* in (u_0, s):

$$\begin{aligned}(u_0^\star, s^\star) = \arg\min_{u_0,\, s\geq 0}\ & \tfrac{1}{2} {u_0}^\top H(x_0)\, u_0 + c(x_0, r)^\top u_0 \\ & + \rho\, 1^\top s + \tfrac{1}{2}\varepsilon_s \|s\|^2 \\ \text{s.t.}\quad & A_c(x_0)\, u_0 \leq b_c(x_0), \\ & A_s(x_0)\, u_0 - s \leq b_s(x_0),\end{aligned} \tag{12}$$

where $H(x_0) \succ 0$ and $c(x_0, r)$ are the parameters of the quadratic objective, which depend on the state and reference $r = r_k$, and $\rho > 0$ is the weight associated with the linear penalty on soft constraint violation and a small $\varepsilon_s > 0$ penalizes s quadratically, ensuring strong convexity jointly with $H(x_0) \succ 0$. The joint decision vector $z = [{u_0}^\top\ s^\top]^\top$ has dimension $n_z = n_u + n_s$, far smaller than $n_w = (N+1)n_x + Nn_u$ for typical horizons N.

Remark 2.1: The QP (12) guarantees by construction that the applied input $u_0^\star$ satisfies the hard constraints (10), while first-step state bounds hold only if $s = 0$ at optimality. Recursive feasibility, closed-loop stability, and full-horizon constraint satisfaction for (4) are not certified.

In this paper, we consider a stage cost of the form

$$\ell(x, u;\, r) \;=\; \|\varphi(x) - y_{\text{ref}}\|_Q^2 + \|u - u_{\text{ref}}\|_R^2\,, \tag{13}$$

where $\varphi : \mathbb{R}^{n_x} \to \mathbb{R}^{n_\varphi}$ is a (possibly nonlinear) output map, $Q \succ 0$, $R \succ 0$ are weight matrices, and $r = [{y_{\text{ref}}}^\top {u_{\text{ref}}}^\top]^\top$ is the task reference. By linearizing $\varphi(x_1)$ around x_0, we get the approximation

$$\varphi(x_1) \;\approx\; \underbrace{\varphi(x_0) + J_\varphi(x_0)\,(f_0 - x_0)}_{\varphi_{\text{drift}}} \;+\; \underbrace{J_\varphi(x_0)\, G_0}_{J_\varphi^u}\, u_0, \tag{14}$$

which is linear in u_0 for any fixed x_0, where $J_\varphi(x_0) \in \mathbb{R}^{n_\varphi \times n_x}$ is the Jacobian of φ at x_0. Substituting (14) into (13) and expanding yields the analytic baseline parameters:

$$\begin{aligned} H_{\text{an}} &= 2\big({J_\varphi^u}^\top Q\, J_\varphi^u + R\big), \\ c_{\text{an}} &= 2{J_\varphi^u}^\top Q\,(\varphi_{\text{drift}} - y_{\text{ref}}) - 2R\, u_{\text{ref}}. \end{aligned} \tag{15a}$$

When $\varphi(x) = x$, $J_\varphi = I$, $J_\varphi^u = G_0$, and $\varphi_{\text{drift}} = f_0$, so (15a) reduces to the familiar quadratic case:

$$H_{\text{an}} = 2(R + {G_0}^\top Q G_0),\ c_{\text{an}} = 2{G_0}^\top Q(f_0 - x_{\text{ref}}) - 2Ru_{\text{ref}}. \tag{15b}$$

In the sequel, we denote $L_{\text{an}} = \text{chol}(H_{\text{an}})$.

As $(H(x_0), c(x_0, r))$ in (12) must encode the *entire* N-step value function, not merely the one-step stage cost, while the analytic baseline (15) captures only the linearized one-step contribution, we need to learn the residual $(\Delta L, \Delta c)$ to account for the remaining $N-1$ horizon steps and, when φ is nonlinear, also for the linearization error. We describe next how to learn them via a neural network.

III. Amortized QP for Input-Affine NMPC

As shown in Fig. 1, to learn the residual corrections $(\Delta L, \Delta c)$, we propose a neural network ϕ_ψ. The network takes the concatenated input $\xi = [{x_0}^\top\ r^\top]^\top$ and returns $(\Delta L, \Delta c)$ to complement the analytic baseline $(L_{\text{an}}, c_{\text{an}})$. The corrected Cholesky factor $L_{\text{full}} = L_{\text{an}} + \Delta L$ defines a positive-definite cost matrix $H_u = L_{\text{full}} {L_{\text{full}}}^\top$, which together with $c_{\text{full}} = c_{\text{an}} + \Delta c$ and the (stacked) state-dependent constraint matrices $A(x_0), b(x_0)$ fully specify a small QP, to be solved online to generate the command input $u_0^\star$. The network is trained offline via a hybrid loss combining supervised imitation of NLP solutions, KKT stationarity, and Fischer–Burmeister complementarity residuals, as we will detail next.

A. QP-Parameter Network

The network ϕ_ψ outputs $n_u(n_u+1)/2$ lower-triangular Cholesky corrections ΔL and n_u linear cost corrections Δc, totalling $n_u(n_u+3)/2$ scalar outputs. The full linear cost is $c_{\text{full}} = c_{\text{an}} + \Delta c$, and the off-diagonal Cholesky entries are corrected additively as $[L_{\text{full}}]_{ij} = [L_{\text{an}}]_{ij} + [\Delta L]_{ij}$ for $i > j$. The diagonal is replaced by $[L_{\text{full}}]_{ii} = \text{softplus}([L_{\text{an}}]_{ii} + [\Delta L]_{ii}) + \varepsilon$, where $\text{softplus}(x) = \log(1 + e^x)$ and $\varepsilon > 0$, guaranteeing strict positivity of the diagonal and hence $H_u = L_{\text{full}} {L_{\text{full}}}^\top \succ 0$. The constraint matrices $A_c(x_0)$, $b_c(x_0)$, $A_s(x_0)$, $b_s(x_0)$ are computed analytically from the known dynamics and input-affine structure. During training, the QP is solved by the batched differentiable interior-point solver `QPAX` [16], producing u_0 satisfying (10) and slack s as in (11). Gradients of the QP solution are obtained by implicit differentiation of the solver's relaxed KKT system [13]. Since `QPAX` relaxes complementarity with a small barrier parameter, the gradients remain well defined even at active constraints.

B. Hybrid Loss Function

Let $\mathcal{D} = \{(x_0^{(i)}, r^{(i)}, w^{\star(i)}, \mu^{\star(i)}, \lambda^{\star(i)}, z_L^{\star(i)}, z_U^{\star(i)})\}_{i=1}^M$ be an offline dataset of M optimal NLP solutions and their dual variables. Given the network's predicted first-step control $\hat{u}_0^{(i)}$, we define the *mixed primal vector* $\hat{w}^{(i)}$ as the full optimal solution $w^{\star(i)}$ with the first control block and the resulting one-step-ahead state replaced by the network's prediction:

$$\hat{w}^{(i)} = w^{\star(i)}\Big|_{\substack{u_0 \leftarrow \hat{u}_0^{(i)}, \\ x_1 \leftarrow f_0^{(i)} + G_0^{(i)} \hat{u}_0^{(i)}}}. \tag{16}$$

The network is trained by minimizing the following hybrid loss:

$$\mathcal{L}_{\text{total}} = \alpha_{\text{sup}}\mathcal{L}_{\text{sup}} + \alpha_{\text{KKT}}\mathcal{L}_{\text{KKT}} + \alpha_{\text{FB}}\mathcal{L}_{\text{FB}} + \alpha_{\text{reg}}\mathcal{L}_{\text{reg}},$$

where $\alpha_{\text{KKT}}, \alpha_{\text{FB}}, \alpha_{\text{reg}} > 0$ are tunable weights, $\alpha_{\text{sup}} = 1$ without loss of generality, and the individual loss terms are defined as follows.

a) Supervised imitation loss:

$$\mathcal{L}_{\text{sup}} = \frac{1}{M}\sum_{i=1}^{M} \left\| \hat{u}_0^{(i)} - u_0^{\star(i)} \right\|^2 \tag{17}$$

penalizing the deviation from the optimal first control action ($\|\cdot\|$ denotes the Euclidean norm).

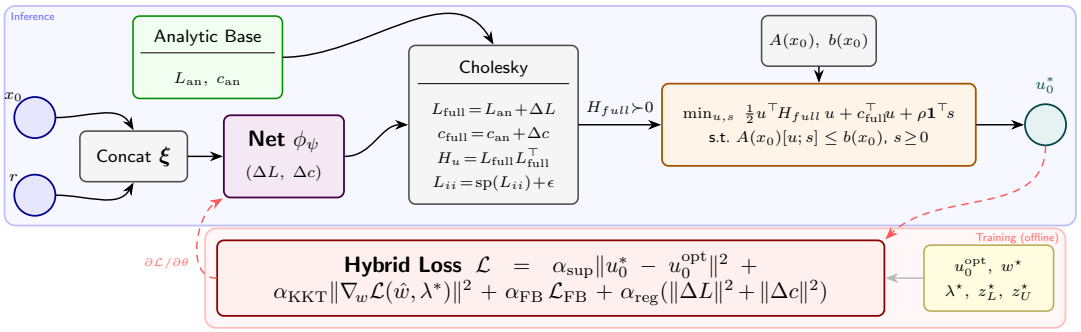

Fig. 1. Residual-corrector architecture for amortized input-affine MPC. An analytic baseline (L_{an}, c_{an}) is computed from the current state and reference; the network ϕ_ψ predicts residual corrections $(\Delta L, \Delta c)$. The Cholesky factor $L_{full} = L_{an} + \Delta L$ yields a guaranteed positive-definite Hessian $H_u \succ 0$ after applying softplus to the diagonal. The resulting QP is solved by a differentiable interior-point layer, producing $u_0^\star$ satisfying the QP hard constraints by construction. The network is trained offline on NLP solutions via a hybrid loss combining supervised imitation and KKT-residual penalties.

b) KKT stationarity loss: To enforce (7), we penalize

$$\mathcal{L}_{KKT} = \frac{1}{M} \sum_{i=1}^{M} \left\| \nabla_w \mathcal{L}(\hat{w}^{(i)},\, \mu^{\star(i)},\, \lambda^{\star(i)},\, z_L^{\star(i)},\, z_U^{\star(i)}) \right\|^2 \tag{18}$$

at the optimal dual variables from the dataset.

c) Fischer–Burmeister complementarity loss: The violation of the complementarity conditions and inequality constraints are penalized via the FB function (8). For box constraints on w, we set $\mathcal{L}_{FB}^{box} = \frac{1}{M} \sum_{i=1}^{M} \Big(\left\| \phi_{FB}(\hat{w}^{(i)} - \underline{w},\, z_L^{\star(i)}) \right\|^2 + \left\| \phi_{FB}(\overline{w} - \hat{w}^{(i)},\, z_U^{\star(i)}) \right\|^2 \Big)$. For ranged inequality constraints $\underline{h_j} \le h_j(w) \le \overline{h_j}$ with signed multiplier $\lambda_j^\star$ (positive when the upper bound is active, negative when the lower bound is active), we penalize instead $\mathcal{L}_{FB}^{ineq} = \frac{1}{M} \sum_{i=1}^{M} \sum_j \Big(\left\| \phi_{FB}(\overline{h_j} - h_j(\hat{w}^{(i)}),\, [\lambda_j^{\star(i)}]^+) \right\|^2 + \left\| \phi_{FB}(h_j(\hat{w}^{(i)}) - \underline{h_j},\, [\lambda_j^{\star(i)}]^-) \right\|^2 \Big)$. The total FB loss is $\mathcal{L}_{FB} = \mathcal{L}_{FB}^{box} + \mathcal{L}_{FB}^{ineq}$. The optimal multipliers in $\mathcal{D}$ are treated again as fixed constants during training.

d) Regularization towards the analytic baseline: To prevent the learned corrections from deviating too far from the analytic baseline and regularize the learning problem, we include the ℓ_2-penalty $\mathcal{L}_{reg} = \frac{1}{M} \sum_{i=1}^{M} \big(\left\| \Delta L^{(i)} \right\|^2 + \left\| \Delta c^{(i)} \right\|^2 \big)$ on the residuals. The weight α_{reg} on $\mathcal{L}_{reg}$ is set to a small value.

Note that $\hat{w}^{(i)}$ in (16) satisfies the dynamic equations only at the first step when transitioning from x_0 to x_1; the dynamics at $t \ge 1$ may be violated when $\hat{u}_0 \ne u_0^\star$, as the subsequent states $x_2, \ldots, x_N$ are still taken from the optimal solution $w^\star$ and thus are consistent with $u_0^\star$ rather than $\hat{u}_0$. Therefore, the KKT and FB losses are evaluated at a dynamically infeasible point, accurate to first order in $\hat{u}_0 - u_0^\star$; they should thus be interpreted as informed regularization terms that shape the loss landscape near the NLP solution, rather than as rigorous optimality residuals for the original NLP.

IV. Case Study: Planar Robotic Arm

A. System Model

We test the proposed nonlinear MPC approximation method on a three-link planar robotic arm operating in the vertical plane under gravity, with link lengths $l = [1.0, 1.0, 0.5]$ m, masses $m = [1.0, 1.0, 0.5]$ kg, center-of-mass offsets $l_c = [0.5, 0.5, 0.25]$ m, and moments of inertia $I = [0.0833, 0.0833, 0.0104]$ kg m^2. The system is described by the Euler–Lagrange equations

$$M(q)\, \ddot{q} + C(q, \dot{q})\, \dot{q} + g(q) = \tau, \tag{19}$$

where $M(q) \in \mathbb{R}^{3\times3}$ is the inertia matrix, $C(q, \dot{q})$ the Coriolis matrix, $g(q)$ the gravity vector, and $\tau \in \mathbb{R}^3$ the joint torques.

Choosing the joint accelerations $u = \ddot{q} \in \mathbb{R}^3$ as the control input yields the continuous-time double-integrator prediction model $\frac{\mathrm{d}}{\mathrm{d}t} q = \dot{q}$, $\frac{\mathrm{d}}{\mathrm{d}t} \dot{q} = u$, with state $x = [q^\top \ \dot{q}^\top]^\top \in \mathbb{R}^6$; discretizing at T_s gives the input-affine discrete-time model (1). The applied torque $\tau_k = M(q_k) u_k + C(q_k, \dot{q}_k) \dot{q}_k + g(q_k)$ is computed by inverse dynamics at the start of each step and held constant over T_s, while the plant integrates the full dynamics (19) with RK4 at T_{sim}. The prediction model integrates $\ddot{q} = u$ with RK4 at T_s, which is exact for piecewise-constant inputs and coincides with the ZOH matrices (20) used in the QP.

The affine one-step map (9) used to build the QP constraints is obtained by applying zero-order hold to the double integrator, giving the exact discrete matrices

$$G_0 = \begin{bmatrix} \frac{1}{2} T_s^2 I_{n_q} \\ T_s I_{n_q} \end{bmatrix}, \qquad f_0 = \begin{bmatrix} q + T_s \dot{q} \\ \dot{q} \end{bmatrix}. \tag{20}$$

B. MPC Problem Setup

We set up the MPC problem (4) with horizon $N = 15$ and the *Cartesian end-effector tracking* objective

$$\ell(x, u;\, p_{ref}) = \|p(q) - p_{ref}\|_{W_{ee}}^2 + \|\dot{q}\|_{W_{dq}}^2 + \|u\|_{W_u}^2, \tag{21}$$

where $p(q) \in \mathbb{R}^2$ is the Cartesian end-effector position (forward kinematics), $p_{ref} \in \mathbb{R}^2$ is the Cartesian target, and $W_{ee} = \mathrm{diag}(150, 150)$, $W_{dq} = 0.5\, I_3$, $W_u = 0.02\, I_3$. The

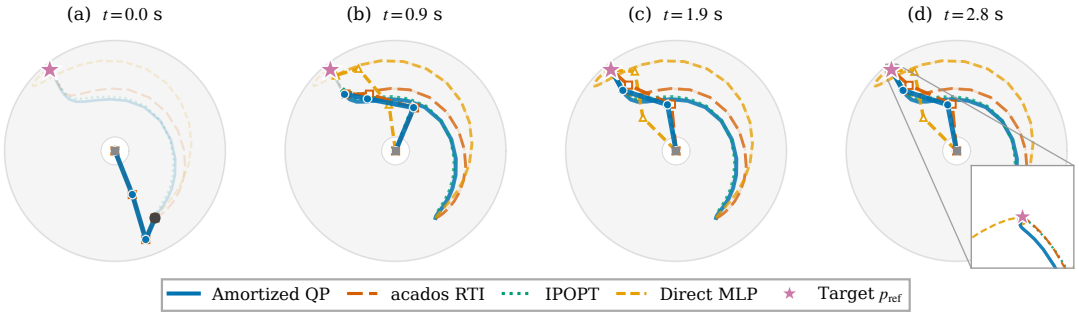

Fig. 2. End-effector trajectories in the Cartesian workspace, from a common initial configuration (filled circle) to the target p_{ref} (star); the last panel zooms on the target region. Only the amortized QP closely matches the IPOPT trajectory.

terminal cost uses $W_{ee,f} = 20\,W_{ee}$, $W_{dq,f} = 10\,W_{dq}$. Here $r = p_{\text{ref}}$.

The following constraints are enforced at every horizon step: *box constraints* on joint positions $|q_i| \le \pi$ rad, joint velocities $|\dot{q}_i| \le 5$ rad/s, and accelerations $|u_i| \le 20$ rad/s^2 for all $i = 1, 2, 3$; *affine-in-input constraints* (3), with joint torques expressed via inverse dynamics (19), $|\tau_i| \le 50$ N m for joints 1–2, and $|\tau_3| \le 20$ N m for the distal link.

The stage cost (21) is *nonlinear* in x (through the forward kinematics $p(q)$), so the closed-form parameters (15b) do not apply directly. We derive an analytic baseline by linearizing the end-effector position around the current configuration:

$$p(q_1) \approx \underbrace{p(q) + J_{ee}(q)\,T_s I_3 \dot{q}}_{p_{\text{drift}}} + \underbrace{J_{ee}(q)\,\tfrac{1}{2}T_s^2 I_3}_{J_{ee}^u}\,u,$$

where $J_{ee}(q) \in \mathbb{R}^{2\times 3}$ is the end-effector Jacobian and p_{drift} is the drift under zero control. Substituting into (21) and expanding yields

$$H_{\text{an}} = 2\big(J_{ee}^{u\top} W_{ee} J_{ee}^u + T_s^2 W_{dq} + W_u\big), \tag{22}$$

$$c_{\text{an}} = 2J_{ee}^{u\top} W_{ee}(p_{\text{drift}} - p_{\text{ref}}) + 2T_s W_{dq}\dot{q}. \tag{23}$$

The hard constraints (10) encode the torque bounds $\tau = M(q)u + n(q,\dot{q}) \in [\underline{\tau}, \overline{\tau}]$, with $n(q,\dot{q}) = C(q,\dot{q})\dot{q} + g(q)$, together with the acceleration bounds:

$$A_c(x) = \begin{bmatrix} M(q) \\ -M(q) \\ I_3 \\ -I_3 \end{bmatrix}, \quad b_c(x) = \begin{bmatrix} \overline{\tau} - n(q,\dot{q}) \\ n(q,\dot{q}) - \underline{\tau} \\ \overline{u} \\ -\underline{u} \end{bmatrix}. \tag{24}$$

The one-step-ahead state is exact under the ZOH model (20): $q_1 = q + T_s\dot{q} + \frac{1}{2}T_s^2 u$, $\dot{q}_1 = \dot{q} + T_s u$. Stacking the four upper/lower bound pairs gives the arm-specific instance of (11) using G_0 from (20):

$$A_s = \begin{bmatrix} \frac{1}{2}T_s^2 I_3 \\ -\frac{1}{2}T_s^2 I_3 \\ T_s I_3 \\ -T_s I_3 \end{bmatrix},\; b_s(x) = \begin{bmatrix} \overline{q} - (q + T_s\dot{q}) \\ (q + T_s\dot{q}) - \underline{q} \\ \overline{\dot{q}} - \dot{q} \\ \dot{q} - \underline{\dot{q}} \end{bmatrix}, \tag{25}$$

yielding $n_s = 4n_u = 12$ slack variables. The QP (12) therefore has $n_z = n_u + n_s = 15$ decision variables and 36 inequality rows (12 hard from (24), 12 soft from (25), and 12 non-negativity rows $-s \le 0$).

C. Data Generation and Training

Trajectories are generated under warm-started IPOPT [14] with $\epsilon_{\text{tol}} = 10^{-6}$ on all termination criteria, $T_s = 0.05$ s, and $T_{\text{sim}} = 0.005$ s. Each trajectory runs up to 80 steps from an independently drawn (x_0, p_{ref}); only trajectories ending with the end-effector within 0.04 m of p_{ref} and all joint speeds below 0.08 rad/s are retained. $(q, \dot{q}, p_{\text{ref}}) \in \mathbb{R}^8$ are drawn via Latin hypercube sampling: $q \sim \mathcal{U}([-\pi, \pi]^3)$, $\dot{q} \sim \mathcal{U}([-5, 5]^3)$, p_{ref} over the reachable annulus $\|p_{\text{ref}}\| \in [0.30, 2.35]$ m, yielding $M = 100{,}000$ samples.

The network ϕ_ψ is a three-hidden-layer MLP ($[128, 128, 64]$, ReLU), mapping $\xi \in \mathbb{R}^8$ to the 9-dimensional residual $(\Delta L, \Delta c)$ over the analytic baseline (22)–(23), initialized to zero so the QP recovers the analytic solution at startup. Training uses Adam [17] with cosine decay ($10^{-3} \to 10^{-5}$), 200 epochs, batch size 256, $(\alpha_{\text{sup}}, \alpha_{\text{KKT}}, \alpha_{\text{FB}}, \alpha_{\text{reg}}) = (1, 10^{-4}, 10^{-3}, 10^{-4})$, $\rho = 1000$, $\varepsilon = 10^{-3}$, $\varepsilon_s = 10^{-6}$, with QP gradients via implicit differentiation in JAX [18].

Remark 4.1: α_{KKT} and α_{FB} are kept small relative to α_{sup}: both losses are evaluated at the mixed primal $\hat{w}^{(i)}$, unreliable far from the NLP solution, and the KKT gradient (18) is numerically larger in scale.

D. Numerical Setup

All evaluations are run on an AMD Ryzen 7 PRO 5850U mobile CPU. We evaluate the trained policy in closed-loop against three baselines: via the `acados` SQP-RTI solver [19] performing one Gauss–Newton iteration per step; full-horizon `IPOPT` as an optimality reference; and a fully explicit approximate solution, `direct MLP`, that shares the same architecture and composite loss as the amortized policy but omits the differentiable QP layer, directly predicting $u_0^\star$ from ξ. `IPOPT` is interfaced through `cyipopt` with dense JAX-computed derivatives; its reported solve times are therefore conservative, and `acados` RTI is the performance-representative comparison. Each scenario draws q_0 uniformly in $[-\pi, \pi]^3$ at rest ($\dot{q}_0 = 0$) and a target p_{ref} from the reachable annulus $\|p_{\text{ref}}\| \in [0.30, 2.35]$ m; all controllers drive the nonlinear plant for 4 s (Figure 2). Because `IPOPT` solves a nonconvex NLP it may fail to converge to a global minimum; such scenarios are discarded and replaced so that `IPOPT` can serve as a local-optimality reference for the closed-loop MPC

TABLE I

CLOSED-LOOP RESULTS OVER 100 SCENARIOS ($T = 80$ STEPS); J_{cl} RATIOS RELATIVE TO IPOPT.

Metric	Amort.	RTI	IPOPT	MLP
Mean time (ms)	0.120	2.443	282.0	0.101
Max time (ms)	10.95	251.6	9699	0.321
Convergence	100/100	88/100	100/100†	73/100
Mean EE err. (m)	1.5e−3	4.2e−2	1.2e−3	4.0e−2
Max EE err. (m)	2.8e−2	1.163	3.4e−2	6.6e−1
Mean ratio	1.047	1.175	1.000	1.120
Median ratio	1.012	1.015	1.000	1.037
Max ratio	1.596	6.914	1.000	2.633
Max τ viol. (N m)	0	10^{-11}	10^{-7}	46.3
Max u viol. (rad/s^2)	0	10^{-15}	10^{-7}	20.8
Max $\dot{q}$ viol. (rad/s)	1.42	3.30	2.65	2.56
Slack active (%)	0.7	—	4.3	—

†Non-converging scenarios discarded by construction.

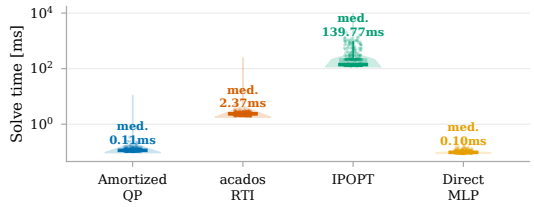

Fig. 3. Per-step solve time. The amortized policy (blue) is ≈20× faster than RTI (orange) and orders of magnitude faster than IPOPT (green). The direct MLP (yellow) is slightly faster, but this comes at the expense of constraint satisfaction and convergence reliability.

performance: $J_{\text{cl}} = \sum_{k=0}^{T-1} \ell(x_k, u_k) + \ell_T(x_T)$. We collect 100 valid scenarios under this protocol. In closed loop, all controllers soften the state bounds (4e) with the same ℓ_1 penalty $\rho = 1000$ used in (12), IPOPT and acados RTI via slack variables at every shooting node, since plant–model mismatch can otherwise render the NLP infeasible; input and torque bounds remain hard. The reported J_{cl} excludes slack penalty terms, so all controllers are compared on the same objective.

E. Results

Table I and Figure 3 show the obtained results in terms of closed-loop performance and solution time across scenarios.

The amortized policy converges on all 100 scenarios, with mean closed-loop cost within 4.7% of IPOPT; acados RTI diverges on 12 scenarios and sits 17.5% above on average (6.9× worst case). The direct MLP converges on only 73/100, sitting 12.0% above IPOPT with hard violations reaching 46.3 N m (torque) and 20.8 rad/s^2 (acceleration) — confirming the QP layer is essential for feasibility.

The amortized policy runs in a mean 0.120 ms, a 20× speedup over RTI and ≈2350× over IPOPT, including both network inference and the QP solve; its solve time is tightly concentrated (median 0.114 ms, 10.9 ms worst case). Hard torque/acceleration bounds are satisfied to numerical precision; soft joint-velocity bounds are mildly violated during transients, with smaller peak excursions than RTI (1.42 vs. 3.30 rad/s). Slack variables are active in only 0.7% of steps, confirming constraint softening plays a minor role in nominal operation.

V. CONCLUSION

We presented a learning-based amortized framework for MPC of input-affine nonlinear systems that only requires solving a QP online, whose matrices depend on the current state and reference signals. We showed that, by combining the differentiable QP layer with a hybrid loss mixing supervised imitation and KKT-residual penalties, we get superior performance and feasibility compared to merely fitting collected nonlinear MPC data, with considerable speedups over RTI and full-horizon NLP solvers at comparable tracking performance.

REFERENCES

[1] J. B. Rawlings, D. Q. Mayne, and M. Diehl, *Model Predictive Control: Theory, Computation, and Design*, 2nd ed. Nob Hill Publishing, 2017.

[2] H. J. Ferreau, C. Kirches, A. Potschka, H. G. Bock, and M. Diehl, "qpOASES: A parametric active-set algorithm for quadratic programming," *Mathematical Programming Computation*, vol. 6, no. 4, pp. 327–363, 2014.

[3] G. Cimini, A. Bemporad, and D. Bernardini, "ODYS QP Solver," ODYS S.r.l. (https://odys.it/qp), Sep. 2017.

[4] G. Frison and M. Diehl, "HPIPM: a high-performance quadratic programming framework for model predictive control," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 6563–6569, 2020.

[5] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos, "The explicit linear quadratic regulator for constrained systems," *Automatica*, vol. 38, no. 1, pp. 3–20, 2002.

[6] M. Diehl, H. G. Bock, and J. P. Schlöder, "A real-time iteration scheme for nonlinear optimization in optimal feedback control," *SIAM Journal on Control and Optimization*, vol. 43, no. 5, pp. 1714–1736, 2005.

[7] S. Gros, M. Zanon, R. Quirynen, A. Bemporad, and M. Diehl, "From linear to nonlinear MPC: Bridging the gap via the real-time iteration," *International Journal of Control*, vol. 93, no. 1, pp. 62–80, 2020.

[8] B. Amos, "Tutorial on amortized optimization," *Foundations and Trends in Machine Learning*, vol. 16, no. 5, pp. 592–732, 2023.

[9] S. Abdufattokhov, M. Zanon, and A. Bemporad, "Learning Lyapunov terminal costs from data for complexity reduction in nonlinear model predictive control," *Int. Journal of Robust and Nonlinear Control*, vol. 34, no. 13, pp. 8676–8691, 2024.

[10] M. Schaller, A. Bemporad, and S. Boyd, "Learning parametric convex functions," 2025, arXiv:2506.04183.

[11] K. P. Wabersich and M. N. Zeilinger, "A predictive safety filter for learning-based control of constrained nonlinear dynamical systems," *Automatica*, vol. 129, p. 109597, 2021.

[12] R. Sambharya, G. Hall, B. Amos, and B. Stellato, "Learning to warm-start fixed-point optimization algorithms," *J. Mach. Learn. Res.*, vol. 25, no. 1, Jan. 2024.

[13] B. Amos and J. Z. Kolter, "OptNet: Differentiable optimization as a layer in neural networks," in *Proc. International Conference on Machine Learning (ICML)*, 2017, pp. 136–145.

[14] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical programming*, vol. 106, no. 1, pp. 25–57, 2006.

[15] A. Fischer, "A special Newton-type optimization method," *Optimization*, vol. 24, no. 3–4, pp. 269–284, 1992.

[16] K. Tracy and Z. Manchester, "On the differentiability of the primal-dual interior-point method," 2024.

[17] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint 1412.6980*, 2014.

[18] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson *et al.*, "JAX: composable transformations of Python+NumPy programs," 2018.

[19] R. Verschueren, G. Frison, D. Kouzoupis, J. Frey *et al.*, "acados – a modular open-source framework for fast embedded optimal control," *Mathematical Programming Computation*, vol. 14, no. 1, pp. 147–183, 2022.