Neuro-Adaptive Control

Myeongseok Ryu [*] and Naol Samuel Erega [†]

Complied on August 14, 2026

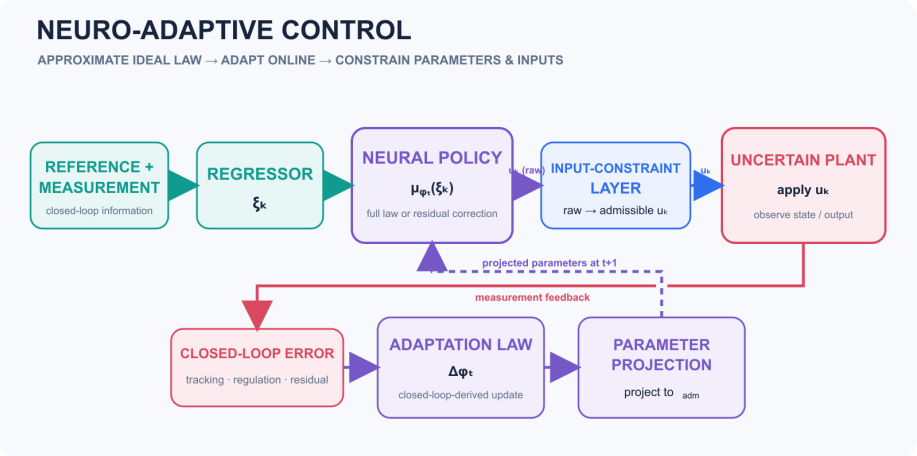

Figure 1: An uncertain ideal control law is represented by a neural policy whose parameters are adapted online under closed-loop and constraint conditions.

Physical interaction is indexed by k, whereas neural-policy updates are indexed by t; the two clocks need not advance at the same rate. Here $t(k)$ is the latest neural-policy update available at physical step k, and $k(t)$ is the latest physical sample available at learner update t. ϕ_t denotes the policy parameter available before update t, and ϕ_{t+1} denotes the updated parameter.

1 Problem definition

Online Learning-Based Optimal Control defines the desired policy through Bellman optimality when the decision problem and required information are available. Neuro-Adaptive Control addresses a related but distinct situation: system uncertainty prevents direct implementation of the ideal feedback law required for a declared tracking, regulation, or stabilization objective.

[*]Myeongseok Ryu is Ph.D. student at KAIST, dding_98@kaist.ac.kr
[†]Naol Samuel Erega is M.S. student at KAIST, samuelnaol7@kaist.ac.kr

Consider

$$x_{k+1} = f(x_k, u_k, w_k), \qquad y_k = h(x_k) + v_k, \tag{1}$$

where x_k, u_k, and y_k are the state, control input, and measurement; f and h are the state-transition and measurement maps; w_k represents process uncertainty or a disturbance; and v_k is measurement noise. When x_k is not measured, $\widehat{x}_k$ denotes its estimate. The dynamics may contain unknown nonlinearities or uncertain parameters. The unavailable target is first written as the ideal feedback law

$$u_k^{\text{ideal}} = \mu_{\text{ideal}}(x_k). \tag{2}$$

Neuro-Adaptive Control then **parameterizes and approximates** this law with a selected neural function class:

$$\mu_{\text{ideal}}(x) = \mu_{\phi^\star}(x) + \varepsilon_{\text{app}}(x). \tag{3}$$

The implemented policy and its online adaptation are

$$\begin{aligned} u_k &= \mu_{\phi_{t(k)}}(\widehat{x}_k), \\ \phi_t &\xrightarrow{\text{online adaptation}} \phi_{t+1}. \end{aligned} \tag{4}$$

Here $\phi^\star$ is an ideal parameter in the chosen neural class, ε_{app} is its approximation error, and ϕ_t is the deployed parameter at learner update t. The law μ_{ideal} is Bellman-optimal only when that connection is explicitly established.

If the controller uses a recurrent or other sequence-processing network, replace $\widehat{x}_k$ by a declared history-dependent information vector ξ_k. The network may approximate the complete feedback law or a residual correction $u_k = u_{\text{nom},k} + \mu_{\phi_{t(k)}}(\xi_k)$.

Direct approximation of μ_{ideal} folds the control effect of uncertain dynamics into the policy. It does not automatically identify a reusable model $\widehat{f}$.

Within the overview taxonomy, this Theme is placed in the joint control–identification lane because closed-loop information about uncertainty directly changes the controller. This placement does not mean that a reusable dynamics model is identified.

2 Why this problem matters

The exact stabilizing or performance-improving feedback often depends on nonlinearities, disturbances, and parameters that are not available to a nominal controller. A neural function class can represent richer compensation than a fixed low-dimensional adaptive law and can update during operation.

Direct-law approximation may also avoid the full sequence of identifying a physical model and then redesigning the controller. Its primary benefit, however, is closed-loop uncertainty compensation. Without explicit mechanisms for coverage, memory, or retention, it remains closer to adaptation along the visited trajectory than to task-wide policy learning.

3 Key challenges

- Ideal-law approximation and information: the approximation domain, ε_{app}, available state or history information, and estimator error must be declared.

- Closed-loop adaptation versus learning: changing ϕ_t changes both the applied input and future data, while a trajectory-local update need not recover $\phi^\star$ or produce a reusable task-wide policy.
- Stability and constraints: bounded tracking error, bounded neural weights, admissible control inputs, and state safety are different claims.
- Excitation and online implementation: weak excitation can prevent parameter convergence, and sensitivity calculation, memory, and update time must remain compatible with the physical control rate.

4 A conventional approach — Lyapunov-derived neuro-adaptation

A conventional neuro-adaptive controller derives its parameter update from declared closed-loop error dynamics and a Lyapunov argument. Schematically,

$$\begin{aligned} u_k &= \mu_{\phi_{t(k)}}(\widehat{x}_k), \\ \phi_{t+1} &= \Pi_{\Phi_{\mathrm{adm}}}\left[\phi_t + \Delta\phi_t\big(e_{k(t)}, \widehat{x}_{k(t)}\big)\right], \end{aligned} \tag{5}$$

where e_k is the selected tracking or regulation error and $\Delta\phi_t$ is the update derived for the specific plant and controller. The projection $\Pi_{\Phi_{\mathrm{adm}}}$ is included only when the implemented method uses an admissible parameter set.

Under its stated approximation, gain, excitation, and initialization conditions, such an update may establish boundedness of selected closed-loop signals. It need not recover $\phi^\star$, learn a globally reusable policy, or enforce actuator and state constraints.

5 A constrained direction — optimization-based neuro-adaptation

A constrained formulation makes the online performance objective and admissibility conditions explicit. Let $k_t := k(t)$ denote the newest physical sample available at learner update t:

$$\begin{aligned} \min_{\phi} \quad & \mathcal{L}_{\mathrm{cl},t}(\phi) \\ \text{s.t.} \quad & c_{\phi,j}(\phi) \le 0, \qquad j = 1, \dots, n_\phi, \\ & c_{u,i}(\mu_\phi(\widehat{x}_{k_t}), \widehat{x}_{k_t}) \le 0, \qquad i = 1, \dots, n_u. \end{aligned} \tag{6}$$

$\mathcal{L}_{\mathrm{cl},t}$ is a closed-loop error or performance objective; it is not a supervised loss against the unavailable μ_{ideal}. Its dependence on a candidate ϕ must be supplied by a declared sliding-window or rollout prediction, forward sensitivity, or differentiable instantaneous surrogate. Previously observed errors alone are constant with respect to a new candidate ϕ and therefore do not define the displayed optimization. Example constraints include

$$c_{\phi,\ell}(\phi) = \|\phi_\ell\|_2^2 - \bar{\phi}_\ell^2 \le 0, \qquad c_u(\phi; \widehat{x}_{k_t}) = \|\mu_\phi(\widehat{x}_{k_t})\|_2^2 - \bar{u}^2 \le 0. \tag{7}$$

n_ϕ and n_u count the declared parameter and input constraints, ϕ_ℓ is a selected parameter group, and $\bar{\phi}_\ell, \bar{u} > 0$ are prescribed bounds.

The input constraint c_u is evaluated on the policy output at the current operating condition and can directly encode input amplitude or norm. An input rate constraint additionally requires the previous applied input, and an energy constraint requires an accumulated or horizon state. If the implemented actuator includes saturation or a separate safety filter, the constraint and analysis must refer to the input actually applied to the system, not only the raw neural-policy output.

Let c_j enumerate the declared parameter constraints $c_{\phi,j}$ and applied-input constraints $c_{u,i}$ after any required state augmentation. A conceptual primal–dual realization takes online steps toward the constrained target:

$$\begin{aligned} \mathscr{L}_t(\phi, \lambda) &= \mathcal{L}_{\mathrm{cl},t}(\phi) + \sum_j \lambda_j c_j(\phi; \widehat{x}_{k_t}), \\ \phi_{t+1} &= \phi_t - \eta_{\phi,t} \nabla_\phi \mathscr{L}_t(\phi_t, \lambda_t), \\ \lambda_{j,t+1} &= \left[\lambda_{j,t} + \eta_{\lambda_j,t} c_j(\phi_t; \widehat{x}_{k_t})\right]_+ . \end{aligned} \tag{8}$$

Here $\lambda_{j,0} \geq 0$, $\eta_{\phi,t}, \eta_{\lambda_j,t} > 0$, and $[\cdot]_+$ denotes projection onto the nonnegative reals. Transient primal–dual iterates need not be feasible; when every applied action must satisfy its constraints, an implemented projection, safety filter, saturation, or fallback and a corresponding closed-loop analysis remain necessary.

The **Constrained Optimization-Based Neuro-Adaptive Control (CoNAC)** preprint is a concrete related instance for uncertain Euler–Lagrange systems. It uses a DNN to approximate an ideal stabilizing law and formulates online weight adaptation with weight-norm and input constraints. The preprint reports bounded tracking errors and DNN weights under its stated assumptions, together with comparative numerical simulations. The scope of these guarantees is determined by the plant, network, feasibility, and adaptation assumptions stated in that work; extensions to other systems or architectures require corresponding analysis.

Evidence should distinguish tracking-error boundedness, weight boundedness, constraint satisfaction, parameter or KKT convergence, control effort, computation time, and performance outside the theorem conditions.

6 Application domains

- **Uncertain nonlinear vehicle and mobility subsystems** requiring online compensation under actuator limits.
- **Robotic and Euler–Lagrange systems**, including manipulators and motion platforms with tracking objectives.
- **Electric drives and mechatronic systems** subject to torque, current, voltage, or other input constraints.
- **Full-law or residual neural control** when a reusable physical model is unavailable or impractical to maintain online.

7 References

- M. Ryu, D. Hong, and K. Choi, "Constrained Optimization-Based Neuro-Adaptive Control (CoNAC) for Uncertain Euler–Lagrange Systems Under Weight and Input Constraints," TechRxiv preprint, v1, 2024.

References