

Imposing a Weight Norm Constraint for Neuro-Adaptive Control

Background and Contributions

Introduction to Neuro-Adaptive Control

Introduction to Neuro-Adaptive Control

- Neuro-adaptive control (NAC) is a control strategy that combines neural network (NN) with adaptive control [1].
- Features of both NN and adaptive control can be found in NAC.



1. adaptive controls adapts their parameters in real-time to systems.
 - Hence, they can handle uncertainties and disturbances.
2. Like adaptive control, NAC adapts its NN weights, as well.
3. Hence, it has both features of NN and adaptive control.

Imposing a Weight Norm Constraint for Neuro-Adaptive Control

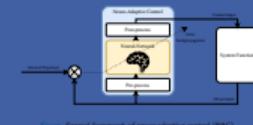
Background and Contributions

Introduction to Neuro-Adaptive Control

Introduction to Neuro-Adaptive Control

Advantages of Neuro-Adaptive Control

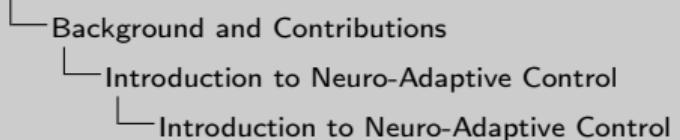
- Adaptability: NAC adapts its weights to changing environments and system dynamics.
- Stability Guarantee: The closed-loop stability is ensured using Lyapunov methods.
- Online Learning Capability: NAC adapts its weights to new data with stability guarantees.
- Robustness: NAC handles uncertainty and disturbances more effectively with adaptive control techniques.



1. The features are...

- It can adapt its weights to change environments.
- And the stability is ensured in the sense of Lyapunov.
- So, it has online learning capability.
- In addition, with various methods from adaptive control, NAC can be more robust than naive NN-based control methods.

Imposing a Weight Norm Constraint for Neuro-Adaptive Control



Challenges in NAC

Weight Boundedness

- Generally, NN weights are adapted by *gradient descent*.
- Gradient function typically consists of the control error.
- Hence, the NN weights can grow *unbounded*, leading to *numerical instability* and *control oscillations*.
- Unbounded weights can cause the NN to produce *excessive inputs*, which may lead to following challenges:

Control Saturation (unpredictable amplitude of NN outputs)

- Typical issue of control problems in physical systems.
- The NN outputs are *unbounded* and *non-interpretable*.
- These features—unbounded NN weights and unpredictable amplitude—can lead to *control oscillations* and *control saturation*.

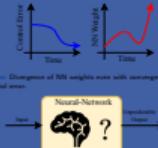


Figure: Divergence of NN weights even with converged control error.



Figure: Unpredictable amplitude of NN outputs.

- However, there are some challenges in NAC.
- The first challenge is weight boundedness.

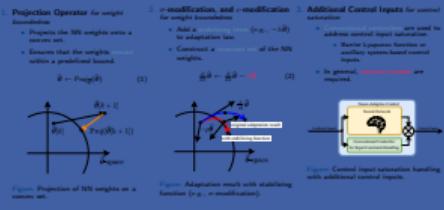
- If the NNs are used for control naively, the boundedness of NN weights is not ensured.
- For instance, despite the convergence of the control error, the NN weights can diverge like parameter drift.
- This divergence leads to the following issue.

- Second one is the NN's output can be excessively large control inputs.
- This is because, the output of NN is not interpretable and unpredictable.
- So, the input saturation problem can occur by them.

Imposing a Weight Norm Constraint for Neuro-Adaptive Control

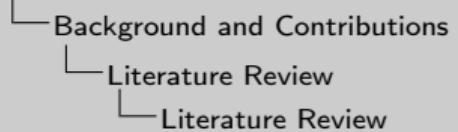
- └ Background and Contributions
 - └ Literature Review
 - └ Literature Review

Literature Review



1. To address these issues, several methods have been introduced.
2. For weight boundedness, the projection operator and modification methods are used.
 - they ensure the weight boundedness by projects onto a convex set or adding a stabilizing term.
 - Hence, the boundedness of NN weights can be easily ensured.
3. For control input saturation, additional control inputs are used.
 - Conventionally, auxiliary system-based control inputs or, more recently, barrier Lyapunov function-based control inputs are used.
 - For this, the nominal model knowledge is required.
 - These make the control design more complicated, but, have shown the effectiveness of the control input saturation handling.

Imposing a Weight Norm Constraint for Neuro-Adaptive Control

**Limitation 1: Lack of Optimality**

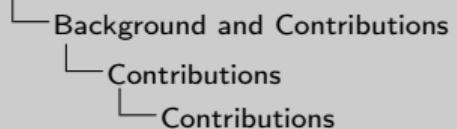
- The existing methods do not guarantee the **optimality** of the control input.
- Projection operator:
 - The projection operator **maps** the NN weights onto a convex set, regardless of the imposed **constraints**.
 - Moreover, if the convex set is non-convex or input saturation.
 - σ - and α -projection operators:
 - The existing methods **map** the NN weights towards the origin.
 - Therefore, the weights converge toward a **central point**.

Limitation 2: Disruption of Learning Process by Additional Control Inputs

- Feedback tracking error for learning is **disrupted** by additional control inputs.
 - The feedback error does not reflect the error induced by the NN, directly.
 - The additional control inputs may exceed the input limits, already.

1. the limitations of the existing methods can be summarized as follows.
2. Most of methods does not consider the optimality when the constraints are active.
 - Because, they only consider the boundedness of the weights or the control input saturation using additional operators or terms.
3. Using additional terms can lead to disruption of learning process.
4. It is, because, as the NN learns with respect to the feedback error, once the additional control inputs are applied, the feedback error does not reflect the error induced by the NN, directly.
5. Moreover, the additional control inputs may exceed the input limits, already.
6. Therefore, in the worst case, the NN may be forced to compensate for the additional terms.

Imposing a Weight Norm Constraint for Neuro-Adaptive Control



Contributions

Contribution 1: Unified Optimization Framework

- Trajectory tracking and constraint handling are formulated as a ~~conventional optimization problem~~ ~~convex constrained optimization problem~~.
- The ~~conventional controller~~ does not require:
 - Nominal model knowledge is not required for the conventional controllers.

Contribution 2: Online Learning Capability (Stability Guarantees)

- Stability are rigorously proven using ~~Lyapunov stability theory~~.
- Hence, ~~online learning with no prior system knowledge~~ is possible.

Contribution 3: Weight and Control Input Constraint Handling

- Weight and control input ~~constraints~~ are ~~in the optimization problem~~.
 - Any combination of convex input constraints can be handled.

1. So, here are the contributions of this work.
2. We, first, formulated a unified optimization framework for trajectory tracking and constraint handling.
3. Hence, we do not need the additional operators or conventional controllers, which may need the nominal model knowledge.
4. Second, we rigorously proved the stability of the proposed method using Lyapunov stability theory.
5. It means, that the proposed method can learn online without any prior system knowledge with stability guarantees.
6. Finally, as we explicitly consider the weight and control input constraints in the optimization problem, we can handle any combination of convex input constraints and weight constraint.

Imposing a Weight Norm Constraint for Neuro-Adaptive Control

Proposed Method

Architecture of the Proposed Method

Architecture of the Proposed Method

Architecture of the Proposed Method

Target Two-link Robotic Manipulator System:

- Control input saturation function $\text{sat}(\cdot)$
- Desired trajectory q_d is given.

$$\Delta q = V_{\omega} q + \dot{x} + \tilde{G} \times v_x = \text{sat}(w)$$

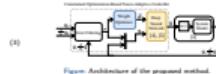


Figure: Architecture of the proposed method.

Control Input:

- DNN's output Φ is used as the control input.
- Consists of the estimated DNN angles $\hat{\theta}$.

$$w = \Phi(q_d, \hat{\theta})$$

Deep Neural Network (DNN):

- k layers with weights $\hat{\theta}_i := \text{vec}(\hat{W}_i)$.
- Activation function: $\phi(\cdot) = \tanh(\cdot)$.

$$\Phi(q_d, \hat{\theta}) = \begin{cases} \hat{W}_k^T \phi(\hat{W}_{k-1} \cdots \phi(\hat{W}_1 q_d)), & i \in \{1, \dots, k\}, \\ \hat{W}_k^T q_d, & i = 0. \end{cases}$$

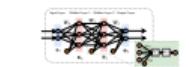


Figure: Architecture of the DNN.

Definitions: $q \in \mathbb{R}^n$: Joint positions, M : Inertia matrix, C : Coriolis matrix, G : Gravity center, w : Control input, v_x : Displacement.

- The proposed method consists of a DNN and weight optimizer, inside.
- Let's start with the system.
 - The system is a two-link robotic manipulator system, which is controlled by the control input τ .
 - The control input is saturated by the saturation function $\text{sat}(\cdot)$.
 - The desired trajectory q_d is given.
- The control input is purely decided by the DNN, so the output of the DNN is the control input.
- And, the DNN is defined like (5), with k layers and weights, and $\tanh(\cdot)$ as the activation function.

Imposing a Weight Norm Constraint for Neuro-Adaptive Control

Proposed Method

Problem Formulation

Problem Formulation

- Optimization Problem Statement:**
- Find NN weights $\hat{\theta}$.
 - That minimizes objective function J_{λ}^* .
- $$J(x|\hat{\theta}) := \|\hat{\theta}\|^2 - \frac{1}{2}x^T x \quad (4)$$

- where $x := g(\hat{\theta}) + \Delta$ is **fitted training error**.
 - while satisfying the following **constraints**:
- Boundedness of the NN weights $\hat{\theta}$.
 - Saturation of the control input x .

- Considered Constraints:**
- Weight Boundedness for Each Layer:
 $a_k(\hat{\theta}) := \|\hat{\theta}_k\|^2 - \frac{1}{2}x_k^T x_k \leq 0, \forall i \in \{0, \dots, k\} \quad (5)$
 - Convex Control Input Saturation:
 - Input bound constraints for each control input:
 $a_{\pi_j}(\hat{\theta}) := x_j - \bar{x}_j \leq 0, \quad a_{\bar{\pi}_j}(\hat{\theta}) := \bar{x}_j - x_j \leq 0 \quad (6)$
 - Input norm constraint:
 $a_{\|x\|}(\hat{\theta}) := \|x\|^2 - \bar{x}^T x \leq 0 \quad (7)$

Assume: $\hat{\theta} \in \mathbb{R}_{+}^{n_{\text{NN}}}$, fitting matrix

- The control problem can be represented as a constrained optimization problem.
 - That is,
 - We need to find NN weights ...
- To explicitly consider the constraints, we define the constraint functions as follows, for weight boundedness and convex control input saturation of each control input and norm of the control input.

Imposing a Weight Norm Constraint for Neuro-Adaptive Control

Proposed Method

Problem Formulation

Problem Formulation

Problem Formulation Optimization Problem

Original Optimization Problem

- Constrained optimization problem to minimize the tracking error.
- Inequality constraints $c_j(\hat{\theta}) \leq 0$ for $j \in \mathcal{I}$.

$$\min_{\hat{\theta}} J(x, \hat{\theta}) \quad (10)$$

s.t. $c_j(\hat{\theta}) \leq 0, \forall j \in \mathcal{I}$

Define Lagrangian Function

$$L(x, \hat{\theta}, [\lambda_i]_{i \in \mathcal{I}}) := J(x, \hat{\theta}) + \sum_{j \in \mathcal{I}} \lambda_j c_j(\hat{\theta}) \quad (11)$$

Dual Problem

- The dual problem is to maximize the Lagrangian function with respect to the NN weights $\hat{\theta}$, while minimizing with respect to the Lagrange multipliers λ_j .
- The Lagrange multipliers λ_j are non-negative, i.e., $\lambda_j \geq 0$.

$$\min_{\hat{\theta}} \max_{[\lambda_i]_{i \in \mathcal{I}}} L(x, \hat{\theta}, [\lambda_i]_{i \in \mathcal{I}}) \quad (12)$$

- Then using the constraint functions, we can formulate the constrained optimization problem as in (10).
- And, we can define the Lagrangian function as in (11).
- So, the dual problem can be formulated as in (12).
- It means, that we can find the optimal solution by minimizing the Lagrangian function with respect to the NN weights, while maximizing it with respect to the Lagrange multipliers.

Imposing a Weight Norm Constraint for Neuro-Adaptive Control

Proposed Method

Adaptation Law Derivation

Adaptation Law Derivation

MicLab

Adaptation Law Derivation Gradient Descent/Ascent Method

To solve the dual problem,

$$\min_{\hat{\theta}} \max_{\lambda} L(\hat{\theta}, [\lambda_i]_{i \in \mathcal{Z}}). \quad (13)$$

The first-order gradient descent/ascent method is used to derive the adaptation law.

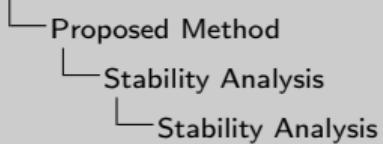
Adaptation Law
Gradient Descent Method for weights $\hat{\theta}$:
$\Delta \hat{\theta} = -\alpha \frac{\partial}{\partial \hat{\theta}} L = -\alpha \left(\frac{\partial}{\partial \hat{\theta}} + \sum_{i \in \mathcal{Z}} \lambda_i \frac{\partial}{\partial \hat{\theta}_i} \right), \quad (14)$
Gradient Ascent Method for Lagrange multipliers $\lambda_i, \forall i \in \mathcal{Z}$:
$\Delta \lambda_i = \beta_i \frac{\partial}{\partial \lambda_i} = \beta_i \eta_i, \quad (15)$
For non-negativity of the Lagrange multipliers,
$\lambda_i \leftarrow \max(\lambda_i, 0) \quad (16)$

α – adaptation gain (learning rate); β_i – update rate of the Lagrange multipliers

1. To solve this problem, we employed the first-order gradient descent/ascent method.
2. For minimizing the Lagrangian function with respect to the NN weights, the gradient descent method is used.
 - We have a adaptation gain α to control the learning rate.
3. For maximizing the Lagrangian function with respect to the Lagrange multipliers, the gradient ascent method is used.
 - We have an update rate β_j for each Lagrange multiplier.
4. Finally, the max operator is used to ensure the non-negativity of the Lagrange multipliers.
5. Therefore, according to the adaptation law, when the constraints are active, the Lagrange multipliers are updated to be positive.
6. And, the NN weights are updated to minimize the constraint violation while minimizing the tracking error.
7. After the constraints are satisfied, *negative, the Lagrange multipliers are updated to zero, and the NN weights are updated to minimize the tracking error only.*

Imposing a Weight Norm Constraint for Neuro-Adaptive Control

2025-06-20



Theorem 3.10:
 For the dynamical system described in [2], the neuro-adaptive controller in (4) with the weight adaptation laws in (24), (28) and (29) require the existence of the function φ and the weight vector $\hat{\theta}$ under the control input constraints satisfying Assumption 1 and 2. This holds under the weight norm constraint (7).

Assumption 1 (Convex Input Constraint):
 The constraint functions $\alpha_i(\hat{\theta}), i \in \mathcal{I}_c$, are convex in the $\hat{\theta}$ -space and satisfy $\alpha_i(\hat{\theta}) \leq 0$ and $\alpha_i(\hat{\theta}^*) \leq 0$.

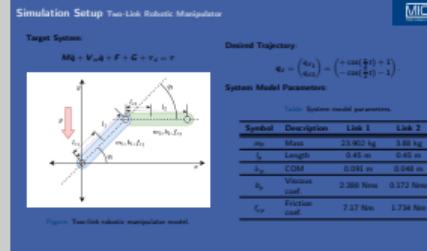
Assumption 2 (Linear Independence Constraint Qualification (LICQ):
 The selected constraints satisfy the Linear Independence Constraint Qualification (LICQ) [8, Chap. 12 Def. 12.1].

Proof of Theorem 3 is omitted due to space limitations. The detailed proof can be found in [2].

1. The stability of the proposed methods is proven using Lyapunov stability theory.
2. The requirements for the stability are as follows.
3. First, the weight norm constraints should be imposed.
4. And, two assumptions for convex input constraints and LICQ conditions should be satisfied.
5. The proof is omitted, but the detailed proof can be found in the reference [2].

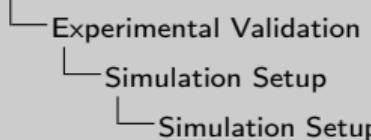
Imposing a Weight Norm Constraint for Neuro-Adaptive Control

- └ Experimental Validation
 - └ Simulation Setup
 - └ Simulation Setup



1. For validation, we simulated a two-link robotic manipulator system.
2. A simple cosine trajectory is given as the desired trajectory.

Imposing a Weight Norm Constraint for Neuro-Adaptive Control



Simulation Setup Controllers Setting

- Only weight norm constraint is considered in this presentation (for ECC).
- Input saturation constraints are not considered.
- For simplicity, weight holder function $\| \cdot \|$ is used.

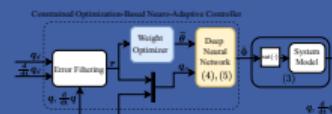
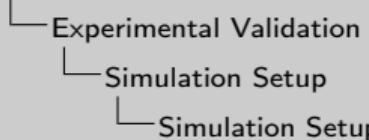


Figure: Architecture of the controller.

1. For this conference paper, we only considered the weight norm constraint.
2. So, please ignore the input saturation.

Imposing a Weight Norm Constraint for Neuro-Adaptive Control



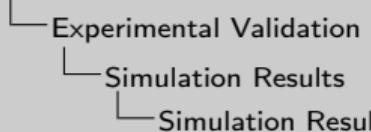
Name	Description	Adaptation Law
NAC-L2	NAC with L_2 -regularization (equal to σ -modification)	$\frac{d\hat{\theta}}{dt} = -\alpha \left(\frac{\partial L}{\partial \hat{\theta}} + \lambda \hat{\theta} \right)$
	(stabilizes $\hat{\theta}$ towards origin)	
NAC-eMod	NAC with ϵ -modification	$\frac{d\hat{\theta}}{dt} = -\alpha \left(\frac{\partial L}{\partial \hat{\theta}} + \rho \ \mathbf{r}\ \hat{\theta} \right)$
	(stabilizes proportionally to filtered error \mathbf{r})	
NAC-CO (proposed)	Constrained Optimization-based NAC	$\frac{d\hat{\theta}}{dt} = -\alpha \left(\frac{\partial L}{\partial \hat{\theta}} + \sum_j \lambda_j \frac{\partial \lambda_j}{\partial \hat{\theta}} \right)$
	(determines λ_j , adaptation speed)	$\frac{d\lambda_j}{dt} = \beta_j, \lambda_j \leftarrow \max(\lambda_j, 0)$

Simulation Objective

By varying the parameters, i.e., β_j , λ_j , and ρ , the parameter dependencies will be investigated.

- For comparative study, we used two existing methods: L2-regularization (NAC-L2) and ϵ -modification (NAC-eMod).
- Please, note that the L2-regularization is equal to the σ -modification.
- In summary, the NAC-L2 and NAC-eMod use the stabilizing terms $-\lambda\hat{\theta}$ and $\rho\|\mathbf{r}\|\hat{\theta}$, respectively, to ensure the boundedness of the weights.
- While the NAC-CO is the proposed method, which uses the Lagrange multipliers λ_j to ensure the boundedness of the weights.
- The simulation is mainly conducted to investigate dependencies of the crucial parameters, i.e., β_j , λ , and ρ .

Imposing a Weight Norm Constraint for Neuro-Adaptive Control



Simulation Results Box-and-Whisker Plots

Parameter Dependencies Investigation

- The parameters ranged from 0.001 to 1 across 10 samples.
- NAC-L2 shows the **worst performance** with high variance.
- NAC-eMod (proposed) shows the best performance and lowest variance.
- This result is because:
- NAC-L2 and NAC-eMod are biased towards the origin, $\hat{\theta} = -\rho \frac{1}{\lambda} \hat{x}_1^2$ (NAC-L2) or $\hat{\theta} = c_1 \hat{x}_1$ (NAC-eMod).
- NAC-CO (proposed) removes the bias and variance.
- $c_1 < 0$ in NAC-CO (proposed) (i.e., $\hat{\theta} = -c_1 \hat{x}_1 + \lambda_1 \hat{x}_1^2$)
- otherwise when constraints are inactive (i.e., $c_1 < 0$), and $\lambda_1 = \beta_1 \lambda_2$ and $\beta_1 := \max(\lambda_1, 0)$.

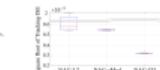
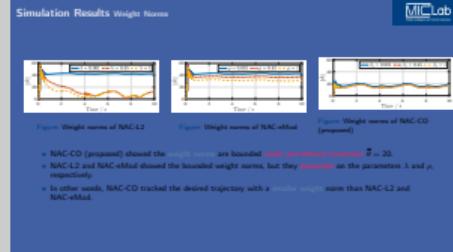
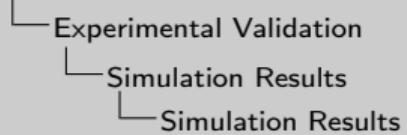


Figure: Box-and-whisker plots of the tracking error ISE.

Reproduced with the tracking error ISE (Integral of Squared Error), i.e., $\sqrt{\frac{1}{T} \int_0^T (\hat{x}_1 - x_1)^2 dt}$, where T denotes a simulation time.

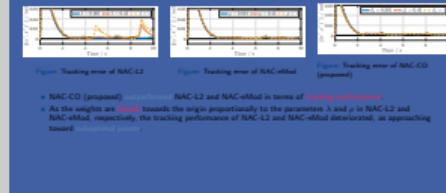
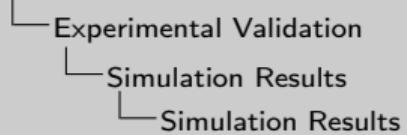
1. The parameters were varied from 0.001 to 1 across 10 samples.
2. The box-and-whisker plots show the distribution of the tracking error ISE (Integral of Squared Error).
3. The NAC-L2 shows the worst performance with high variance, while the NAC-CO (proposed) shows the best performance with the lowest variance.
4. The reason is that the NAC-L2 and NAC-eMod are biased towards the origin, especially, proportionally to the parameters λ and ρ , respectively.
5. On the other hand, the term that biases the weights towards the origin in NAC-CO, is removed when the constraints are inactive, i.e., $c_j < 0$.
6. Therefore, the NAC-CO can track the desired trajectory without being biased, when the constraints are inactive.

Imposing a Weight Norm Constraint for Neuro-Adaptive Control



1. In this slide, the weight norms are shown.
2. We can see that the NAC-CO's weights are bounded under the pre-defined constraint $\bar{\theta} = 20$.
3. The other two methods shows larger weight norms, and biased weights towards the origin, which may lead to suboptimal performance.
4. Moreover, the dependencies also can be seen, as the distribution of the weight norms of NAC-L2 and NAC-eMod are wider than that of NAC-CO.

Imposing a Weight Norm Constraint for Neuro-Adaptive Control



1. The tracking performance of the three methods are shown.
2. The NAC-CO (proposed) outperformed the other two methods in terms of tracking performance.
3. As the weights are biased towards the origin proportionally to the parameters λ and ρ in NAC-L2 and NAC-eMod, respectively, the tracking performance of NAC-L2 and NAC-eMod deteriorated, as approaching toward suboptimal points.

Imposing a Weight Norm Constraint for Neuro-Adaptive Control

└ Experimental Validation

└ Simulation Results

└ Real-time Implementation Result

Real-time Implementation Result

This video demonstrates:

• The applicability of the proposed method to real-time control (under 4 ms sampling time).

• Convex input constraints handling.



1. And, until the last slice, we have shown the conference paper's results.
2. In this slide, we show the recent result with real-time implementation.
3. The video demonstrates the applicability of the proposed method to real-time control, under 4 ms sampling time.
4. And, the convex input constraints are handled, here.

Temporary page!

`LATEX` was unable to guess the total number of pages correctly. As there was some unprocessed data that should have been added to the final page this extra page has been added to receive it.

If you rerun the document (without altering it) this surplus page will go away, because `LATEX` now knows how many pages to expect for this document.