Integral Quadratic Constraints for Neural Networks

Gronqvist, Johan; Rantzer, Anders

DOI:
10.23919/ECC55457.2022.9838065

2022

Integral Quadratic Constraints for Neural Networks

Johan Grönqvist Anders Rantzer

***Abstract*— The formalism of Integral Quadratic Constraints (IQCs) is well-established in robust control. It has recently been used for systems with a neural network as one of its components, however, using only a small subset of the established techniques for obtaining IQC relations. We provide a larger set of IQCs relevant for the nonlinearities commonly used in neural networks, introduce new constraints for the rectified linear unit and the leaky rectified linear unit, and draw on the established literature to build a library of IQCs to use in connection with neural networks. Finally, our examples show how improved guarantees can be obtained with a larger library of IQCs.**

I. Introduction

In recent years, deep learning with neural networks has achieved impressive successes in many fields. Following in the footsteps of this progress, many groups have worked on robustness for systems involving neural networks. Several heuristic frameworks exist for training neural networks to improve robustness and some can provide formal guarantees. Within the field of automatic control, Integral Quadratic Constraints (IQCs) [6], building on previous methods in robust control [12], [14], [10], [11], can provide guarantees in the presence of nonlinearities, and problems formulated within the IQC-formalism reduce to semidefinite programming problems that can be solved efficiently.

Neural networks are well-suited to the IQC formalism, and previous work along this line draws on old results. We mention work on static properties like reachability [5] and Lipschitz constant estimation [3], which can be used as one part in a larger analysis of robustness of the full system, and such analyses can be used to bound the output variations for a given input variation, as in [4]. The closed loop system can also be analysed as a whole [13], [2], and provide guarantees of local or global stability in the presence of uncertain dynamics. A recurrent neural network (RNN) can also be viewed as a feedback system, and robust RNNs were studied in [8]. The IQC formalism has also been used in other contexts to provide other kinds of guarantees, and those kinds of guarantees would transfer to systems with neural networks. Its limitation is that the IQC-framework requires all nonlinear elements, as well as all properties, to be expressed in terms of quadratic forms, limiting the set of questions that the framework can answer. Its strength is that guarantees are global and hold even in the face of adversarial attacks on the network.

Here, we present a larger class of constraints that can be used for neural networks, than used in the works cited above. We also provide examples showing that a larger library of constraints can enable stricter bounds on the behaviour of neural networks, and that there is a trade-off in the choice of constraints. All our discussions and results refer to the continuous time case, but analogous results exist for the discrete time case, with the exception of the Popov IQC.

After introducing our notation, we schematically introduce neural networks in section II, and then describe the rectified linear unit (relu) activation function and the constraints that hold for the repeated relu-nonlinearity in section III. Bias terms in neural networks and studies of signals relative to a reference value are described in section IV, and section V presents the very short list of incremental IQCs. We then discuss constraints for the leaky rectified linear unit ($\mathrm{leakyrelu}$) and other activation functions in section VI and selection of constraints in section VII before finally presenting our two examples in section VIII. The first example considers the static gain of a neural network, construct, while the second considers closed-loop stability with our neural network in the loop.

A. Notation

Our function $\mathrm{sign}(x)$ takes values ± 1 for nonzero values, and 0 for vanishing x, and we say that real values can have three signs, ± 1 and 0. A hat denotes the Laplace transform of a function, as in $\hat{f}(z) = \int_0^\infty f(t)e^{-zt}dt$. The Hermitian conjugate of A is written $A^\star$, and a $\star$ is also used to denote a convolution, $(f \star g)(T) = \int_0^T f(T-t)g(t)dt$. We write relu for the rectified linear unit, $\mathrm{relu}(x) = \max(0, x)$. The terms static and soft constraint refer to ineuqalities of the forms

$$\begin{aligned} x(t)^T G y(t) &\geq 0 \\ \int_0^\infty \hat{x}(i\omega)^T \hat{G}(i\omega)\hat{y}(i\omega)\, d\omega &\geq 0 \end{aligned} \tag{1}$$

where the first holds for all t. We note that a static constraint implies soft constraint, using the same constant matrix G.

II. Neural Networks

A simple feedforward neural networks is a sequence of "linear" layers interspersed with nonlinear activation functions. More generally, the description would be in terms of a block diagram, where the blocks represent affine or nonlinear mappings. The nonlinear activation functions are often scalar and are then applied componentwise, but they can also be vector valued nonlinear functions, e.g., softmax.

A. Network structure

A feedforward neural network, depicted in Fig. 1, shows the sequence of layers. Each layer, indexed by $k =$

johan.gronqvist@control.lth.se
anders.rantzer@control.lth.se

$$\zeta_0 \underset{b_1}{\overset{W_1}{\rightarrow}} \xi_1 \overset{\varphi}{\rightarrow} \zeta_1 \underset{b_2}{\overset{W_2}{\rightarrow}} \xi_2 \overset{\varphi}{\rightarrow} \zeta_2 \rightarrow \cdots \rightarrow \zeta_N \underset{b_{N+1}}{\overset{W_{N+1}}{\rightarrow}} \zeta_{N+1}$$

Fig. 1: A neural network, with input ζ_0, output ζ_{N+1}, and a sequence of N hidden layers with affine mappings $\xi_k = W_k\zeta_{k-1} + b_k$ and nonlinear mappings $\zeta_k = \varphi(\xi_k)$, for $k = 1, \ldots, N$. Compare (2).

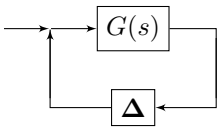


Fig. 2: Standard block diagram for IQC-formalism, where P is an LTI system, and $\mathbf{\Delta}$ contains everything else in the system.

$1, \ldots, N+1$, has an input ζ_{k-1}, an internal variable ξ_k, and an output ζ_k. Every layer but the last also has a nonlinear activation function, denoted φ, that acts componentwise on ξ_k, and our numbering scheme is chosen to have the activation function appear N times. Layer k (for $k = 1, \ldots, N$) is described by the two equations

$$\begin{aligned} \xi_k &= W_k\zeta_{k-1} + b_k \\ \zeta_k &= \varphi(\xi_k) \end{aligned} \tag{2}$$

where the scalar nonlinear function φ acts componentwise, so that i-th components of ζ_k and ξ_k satisfy $\zeta_{k,i} = \varphi(\xi_{k,i})$.

The input to the network is the input to its first layer, ζ_0. The output of the network is the output of the last layer, and is $\zeta_{N+1} = \xi_{N+1} = W_{N+1}\zeta_N + b_{N+1}$ as the final layer lacks an activation function.

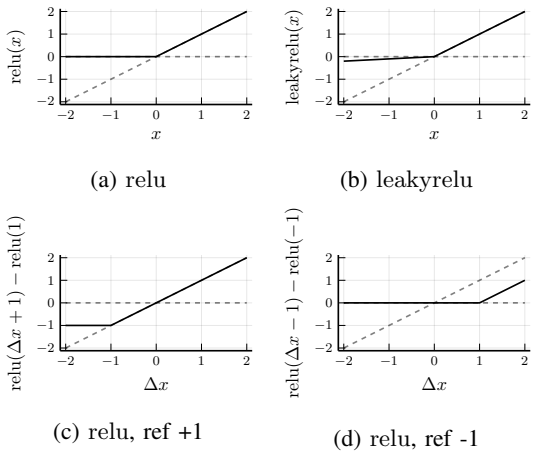
B. Structure of the Nonlinearity

The activation function, which we denote by φ, is a nonlinear function from $\mathbb{R}$ to $\mathbb{R}$, which componentwise on the vectors ξ_k for $k = 1, \ldots, N$, and we can combine all appearances of φ into a single vector structure as

$$\zeta = \begin{bmatrix} \zeta_{1,1} \\ \vdots \\ \zeta_{1,n_1} \\ \vdots \\ \zeta_{N,1} \\ \vdots \\ \zeta_{N,n_N} \end{bmatrix} = \begin{bmatrix} \varphi(\xi_{1,1}) \\ \vdots \\ \varphi(\xi_{1,n_1}) \\ \vdots \\ \varphi(\xi_{N,1}) \\ \vdots \\ \varphi(\xi_{N,n_N}) \end{bmatrix} = \varphi(\xi), \tag{3}$$

where the vectors ξ_k and ζ_k have n_k components each, labeled $\xi_{k,i}$ and $\zeta_{k,i}$ for $i = 1, \ldots, n_k$.

To analyse a closed loop system, e.g., the system shown in Fig. 4 and used in our second example, we first restructure it as a system in the form shown in Fig. 2. The nonlinear block $\mathbf{\Delta}$ will contain all our variables ξ_k and ζ_k, and we describe its nonlinea structure as a repetition of the activation function over all components of all the variables $\xi_{k,i}$ (for $k = 1, \ldots, N$). The nonlinearity is thus repeated (or "diagonal") on the vector $\xi = [\xi_1^T \xi_2^T \ldots \xi_N^T]^T$. When discussing repeated nonlinearities later, we may use the full vector ξ, or a smaller vector comprising a subset of its components.



(a) relu (b) leakyrelu

(c) relu, ref +1 (d) relu, ref -1

Fig. 3: Four variants of relu: plain, leaky, with positive reference value and with negative reference value, described in sections III, VI, IV and IV, respectively. Dashed lines show $y = 0$ and $y = x$.

III. Rectified Linear Unit

The rectified linear unit (relu) is a very common choice of nonlinearity, and we introduce the scalar function, before discussing constraints valid for repeated relu.

A. Scalar relu

We let $x \in \mathbb{R}$ and $y = \varphi(x) = \text{relu}(x) = \max(0, x) \in \mathbb{R}$ denote the scalar input and output of the relu function.

The relu function is shown in Fig. 3a, and it satisfies

$$\begin{aligned} &y = 0 \text{ if } x = 0 \\ &y \geq 0 \text{ and } y \geq x \\ &(y - x)y = 0 \end{aligned} \tag{4}$$

For any two inputs $x_i \in \mathbb{R}$ and $x_j \in \mathbb{R}$ with $x_i \neq x_j$, the function is "rate-limited" or "slope-restricted", in the sense that $(y_i - y_j)/(x_i - x_j) \in [0, 1]$. We use it in the form

$$(y_i - y_j)((x_i - x_j) - (y_i - y_j)) \geq 0 \tag{5}$$

which holds for all x_i, x_j.

B. Repeated Relu

We introduce two classes of constraints for the repeated relu-nonlinearity, and then combine these with established forms of constraints from the literature, to build our library of IQCs for the repeated relu-nonlinearity.

Input and output are now vectors $x \in \mathbb{R}^n$ and $y = \text{relu}(x) \in \mathbb{R}^n$, for some integer n.

The quadratic equality of the scalar case, $(y - x)y = 0$, now appears repeated n times, and in the vector case we use it as an inequality $(y - x)^T Gx \geq 0$ which holds for any diagonal matrix G. The lack of sign constraints on G reflects

the equality in the constraint. We list this constraint as IQC 1 below.

From the linear vector inequalities $y - x \succeq 0$ and $y \succeq 0$, we can form positive semidefinite quadratic forms

$$\begin{bmatrix} y-x \\ y \end{bmatrix}^T G \begin{bmatrix} y-x \\ y \end{bmatrix} \geq 0 \tag{6}$$

for any matrix G with nonnegative entries.

For a dynamic system, where $x(t)$ and $y(t) = \mathrm{relu}(x(t))$ are vector-valued signals, we additionally convolve our vector of nonnegative signals with a nonnegative convolution kernel, and we obtain a new vector of nonnegative signals. The procedure leads to the inequality

$$\int \begin{bmatrix} \hat{y}(i\omega) - \hat{x}(i\omega) \\ \hat{y}(i\omega) \end{bmatrix}^T \hat{H}(i\omega) \begin{bmatrix} \hat{y}(i\omega) - \hat{x}(i\omega) \\ \hat{y}(i\omega) \end{bmatrix} d\omega \geq 0 \tag{7}$$

which holds if the entries of the matrix $H(t)$ are nonnegative for all t. These inequalities are used in used in IQCs 2 and 6 below.

C. Constraints

We now list our full set of constraints for the repeated relu-nonlinearity, including both the two forms above, as well as other IQCs from the literature.

For static constraints, we assume vectors $x \in \mathbb{R}^n$ and $y = \mathrm{relu}(x) \in \mathbb{R}^n$, for some integer n. When we list dynamic constraints, we assume vector-valued signals x and y so that $x(t) \in \mathbb{R}^n$ and $y(t) = \mathrm{relu}(x(t)) \in \mathbb{R}^n$ for all t and for some integer n.

We refer to these IQCs as IQCs for the absolute setting, to distinguish from IQCs listed in sections IV and V.

From the quadratic equality (4), we obtain

IQC 1 (Absolute; Equality Constraint): For any diagonal matrix G, with $y = \mathrm{relu}(x)$,

$$y^T G(y - x) \geq 0 \tag{8}$$

holds as a static constraint, with the associated soft IQCs. The lack of sign constraints on G reflects the fact that we have an equality constraint.

Products of the linear inequalities (4), combined using nonnegative coefficients or nonnegative convolution kernels give a class of static constraints, and of soft IQCs.

IQC 2 (Absolute; Combination of linear): For any matrix G with nonnegative entries, and $y = \mathrm{relu}(x)$, the static constraint

$$\begin{bmatrix} y \\ y-x \end{bmatrix}^T G \begin{bmatrix} y \\ y-x \end{bmatrix} \geq 0 \tag{9}$$

holds.

For any matrix-valued convolution kernel $H(t)$ with nonnegative entries for all t,

$$\int_{-\infty}^{\infty} \begin{bmatrix} \hat{y}(i\omega) \\ \hat{y}(i\omega) - \hat{x}(i\omega) \end{bmatrix} \hat{H}(i\omega) \begin{bmatrix} \hat{y}(i\omega) \\ \hat{y}(i\omega) - \hat{x}(i\omega) \end{bmatrix} d\omega \geq 0 \tag{10}$$

holds, for $y(t) = \mathrm{relu}(x(t))$.

To use the rate-limit inequalities, (5), we let e_i be the i-th cartesian basis vector and (following [3]) define the matrix $E_{ij} = (e_i - e_j)(e_i - e_j)^T$. This give us

$$y^T E_{ij} x = (y_i - y_j)^T (x_i - x_j) \tag{11}$$

and enables us to use the rate-limit as

IQC 3 (Absolute; Rate-limit): For a matrix G with nonnegative elements G_{ij}, with $y = \mathrm{relu}(x)$,

$$y^T E_{ij} G_{ij} (x - y) \geq 0 \tag{12}$$

holds for all i, j, as well as the corresponding soft IQC.

IQC 4 (Absolute; Popov): For any diagonal matrix G, and $y(t) = \mathrm{relu}(x(t))$,

$$\int_{-\infty}^{\infty} \hat{x}(i\omega)^\star (i\omega G) \hat{y}(i\omega)\, d\omega \geq 0 \tag{13}$$

holds.

There are several formulations that build on the original result of Zames and Falb [15] to obtain IQCs for repeated nonlinearities, and we quote the result from [1]. We use a static variant, as well as the full result.

IQC 5 (Zames-Falb): For a symmetric matrix G with nonpositive off-diagonal entries, and for which $G_{ii} \geq \sum_{j \neq i} |G_{ij}|$ for all i, the following hold for $y = \mathrm{relu}(x)$ and for any $\alpha < 0$ and $\beta > 1$.

$$y^T G x \geq 0 \tag{14}$$

$$(y - \alpha x)^T G (\beta x - y) \geq 0 \tag{15}$$

Additionally, for a matrix-valued real symmetric convolution kernel $H(t)$ with nonnegative entries for all t, and a stronger constraint on the diagonal elements of G, $G_{ii} \geq \sum_{j \neq i} |G_{ij}| + \sum_j \|H_{ij}\|_1$ for all i, the two soft IQCs

$$\int_{-\infty}^{\infty} \hat{y}(i\omega)^\star \left[G - \hat{H}(i\omega) \right] \hat{x}(i\omega)\, d\omega \geq 0 \tag{16}$$

$$\int_{-\infty}^{\infty} [\hat{y}(i\omega) - \alpha \hat{x}(i\omega)]^\star \left[G - \hat{H}(i\omega) \right] [\beta \hat{x}(i\omega) - \hat{y}(i\omega)]\, d\omega \geq 0 \tag{17}$$

hold, where $y(t) = \mathrm{relu}(x(t))$.

IV. Bias and reference values

In many cases, we are interested in the behaviour of a system relative to a known, constant, reference value. We may be tracking the deviations from a stationary point, but reference values can also be used as a trick to handle bias terms in the neural network.

The relative setting considers deviation of the system relative to a constant reference point, looking for constraints on the mapping from $\Delta\xi = \xi - \xi^{(\mathrm{ref})}$ to $\Delta\zeta = \zeta - \zeta^{(\mathrm{ref})} = \mathrm{relu}(\xi) - \mathrm{relu}\left(\xi^{(\mathrm{ref})}\right) = \mathrm{relu}\left(\Delta\xi + \xi^{(\mathrm{ref})}\right) - \mathrm{relu}\left(\xi^{(\mathrm{ref})}\right)$.

The repetitive structure used in section III-B is lost, as each scalar relu has its own reference value, and two relus with different reference values are no longer "the same" nonlinearity. We deal with this issue in section IV-B.

The relative setting provides a middle ground between the absolute setting discussed in section III and the much smaller set of incremental IQCs (listed in section V).

A. Bias

The affine term, b, in (2), poses a problem for analyses that are limited to linear and quadratic forms. To amend this, we introduce reference values. For an input ζ_0, we choose a reference input ${\zeta_0}^{(\text{ref})}$ and let $\xi^{(\text{ref})}$ and $\zeta^{(\text{ref})}$ denote vectors satisfying (2) with ${\zeta_0}^{(\text{ref})}$ as the input. In terms of $\Delta\xi = \xi - \xi^{(\text{ref})}$ and $\Delta\zeta = \zeta - \zeta^{(\text{ref})}$, the relations read

$$\begin{aligned} \Delta\zeta_k &= \zeta_k - {\zeta_k}^{(\text{ref})} = \operatorname{relu}(\xi_k) - \operatorname{relu}\left({\xi_k}^{(\text{ref})}\right) \\ \Delta\xi_k &= \xi_k - {\xi_k}^{(\text{ref})} = W_k \Delta\zeta_{k-1} \end{aligned} \tag{18}$$

and we find the equations of a neural network with different nonlinear mappings, but without bias terms.

B. Scaling and Grouping

We consider rescaled variables for any vector component with nonvanishing reference value

$$\begin{bmatrix} \Delta\xi' \\ \Delta\zeta' \end{bmatrix} = \frac{1}{\left|\xi^{(\text{ref})}\right|} \begin{bmatrix} \Delta\xi \\ \Delta\zeta \end{bmatrix}, \tag{19}$$

and we find

$$\Delta\zeta' = \operatorname{relu}\left(\Delta\xi' + \operatorname{sign}\left(\xi^{(\text{ref})}\right)\right) - \operatorname{relu}\left(\operatorname{sign}\left(\xi^{(\text{ref})}\right)\right). \tag{20}$$

After rescaling every relu unit with nonzero reference value, the mapping from $\Delta\xi'$ to $\Delta\zeta'$ behaves as relu units with reference values that are either ± 1, or 0.

Grouping relus according to the sign of their reference values (positive, vanishing or negative), and rescaling as above, we now have three groups of repeated nonlinearities, behaving as relus with reference values 1, 0 and -1, respectively. The three functions are shown in Figs. 3a, 3c and 3d.

C. Constraints

For our static constraints, we assume vectors $x^{(\text{ref})}$, Δx and Δy. After grouping according to the sign of $x^{(\text{ref})}$ and rescaling as described in section IV-B, we have vectors denoted by $x_+^{(\text{ref})}$, $x_0^{(\text{ref})}$, $x_-^{(\text{ref})}$, Δx_+, Δx_0, Δx_-, Δy_+, Δy_0, Δy_-, satisfying

$$\begin{aligned} {x_+}^{(\text{ref})} &\succ 0; \;\; {x_0}^{(\text{ref})} = 0; \;\; {x_-}^{(\text{ref})} \prec 0 \\ \Delta y_+ &= \operatorname{relu}\left(\Delta x_+ + {x_+}^{(\text{ref})}\right) - \operatorname{relu}\left(x_+^{(\text{ref})}\right) \\ \Delta y_0 &= \operatorname{relu}(x_0) \\ \Delta y_- &= \operatorname{relu}\left(\Delta x_- + {x_-}^{(\text{ref})}\right) - \operatorname{relu}\left({x_-}^{(\text{ref})}\right) \\ \Delta x'_{+,i} &= \frac{\Delta x_{+,i}}{{x_{+,i}}^{(\text{ref})}}; \;\; \Delta y'_{+,i} = \frac{\Delta y_{+,i}}{{x_{+,i}}^{(\text{ref})}} \\ \Delta x'_{0,i} &= \Delta x_{0,i}; \;\; \Delta y'_{0,i} = \Delta y_{0,i} \\ \Delta x'_{-,i} &= \frac{\Delta x_{-,i}}{\left|{x_{-,i}}^{(\text{ref})}\right|}; \;\; \Delta y'_{-,i} = \frac{\Delta y_{-,i}}{\left|{x_{-,i}}^{(\text{ref})}\right|}. \end{aligned} \tag{21}$$

where the last three equations are componentwise for component i.

For dynamic systems, with vector-valued signals instead of vectors, we group and rescale analogously, remembering that the reference values are constant and known in advance.

While the linear inequalities in (4) still hold for Δx_0 and Δy_0, we only have $\Delta y_+ \succeq \Delta x_+$ and $\Delta y_- \preceq 0$ for the other components, and we use all the linear inequalities to construct static and dynamic constraints analogous to IQC 2

IQC 6 (Relative; Combinations of linear): For any matrix G with nonnegative entries,

$$\begin{bmatrix} \Delta y_+ - \Delta x_+ \\ \Delta y_0 - \Delta x_0 \\ \Delta y_0 \\ \Delta y_- \end{bmatrix}^T G \begin{bmatrix} \Delta y_+ - \Delta x_+ \\ \Delta y_0 - \Delta x_0 \\ \Delta y_0 \\ \Delta y_- \end{bmatrix} \geq 0, \tag{22}$$

holds.

For any matrix-valued convolution kernel $H(t)$ with nonnegative entries for all t,

$$\begin{aligned} &\int_{-\infty}^{\infty} \\ &\begin{bmatrix} \widehat{\Delta y_+}(i\omega) - \widehat{\Delta x_+}(i\omega) \\ \widehat{\Delta y_0}(i\omega) - \widehat{\Delta x_0}(i\omega) \\ \widehat{\Delta y_0}(i\omega) \\ \widehat{\Delta y_-}(i\omega) \end{bmatrix}^\star \hat{H}(i\omega) \begin{bmatrix} \widehat{\Delta y_+}(i\omega) - \widehat{\Delta x_+}(i\omega) \\ \widehat{\Delta y_0}(i\omega) - \widehat{\Delta x_0}(i\omega) \\ \widehat{\Delta y_0}(i\omega) \\ \widehat{\Delta y_-}(i\omega) \end{bmatrix} \\ &\qquad d\omega \geq 0 \end{aligned} \tag{23}$$

holds.

The rate-limit is used in two ways, giving us two different constraints. Firstly, we get one copy of IQC 3 for each sign in $x^{(\text{ref})}$.

IQC 7 (Relative; Rate-limit I): For any sign $\sigma \in \{+, 0, -\}$, the vectors $\Delta x'_\sigma$ and $\Delta y'_\sigma$ satisfy IQC 3.

Secondly, each component of in Δx and Δy is a difference between a value and a reference value, giving us a static constraint, with its associated soft IQC.

IQC 8 (Relative; Rate-limit II): For any diagonal matrix G with nonnegative elements,

$$\Delta y^T G(\Delta x - \Delta y) \geq 0 \tag{24}$$

holds.

IQC 9 (Relative; Popov): For a sign $\sigma \in \{+, 0, -\}$, IQC 4 holds for Δx_σ and Δy_σ.

For each sign of reference values, our mapping from $\Delta x'$ to $\Delta y'$ satisfies the conditions required for IQC 5 to hold.

IQC 10 (Relative; Zames-Falb): For a sign $\sigma \in \{+, 0, -\}$, IQC 5 holds for $\Delta x'_\sigma$ and $\Delta y'_\sigma$.

V. Incremental Constraint

In the incremental case, the rate-limit is our only remaining source of inequalities, in the variant without mixing

IQC 11 (Incremental; Rate-limit): For vectors $x^{(1)}$, $x^{(2)}$, $\Delta x = x^{(1)} - x^{(2)}$, $\Delta y = \operatorname{relu}\left(x^{(1)}\right) - \operatorname{relu}\left(x^{(2)}\right)$, IQC 8 holds for Δx and Δy.

Zames-Falb constructions cannot be used, as the mapping from Δx to Δy is not a function (we can have, e.g., $\Delta x = 1$ together with any value of Δy between 0 and 1).

VI. Other Activation functions

Feedforward networks often use the relu or $\mathrm{leakyrelu}$ activation functions for internal layers, but other activation functions are also used, and we first discuss how to use our full list of IQCs for $\mathrm{leakyrelu}$, before discussing what subset we can retain for other activations functions.

For internal layers, relu is somtimes replaced by a different componentwise function, and the vector structure of the repeated nonlinearity is retained. These functions typically satisfy a rate-limit and vanish for vanishing input, and constraints based on rate-limits are still available to us.

The final layer somtimes uses a softmax nonlinearity, defined by

$$\zeta_{N+1,i} = \frac{e^{\xi_{N+1,i}}}{\sum_\ell e^{\xi_{N+1,\ell}}} \tag{25}$$

The softmax nonlinearity is not a repetition of a scalar function, and its set of IQCs differs from those of repeated nonlinearities. The output of softmax is nonnegative, and it is rate-limited to $[0, 1]$.

A. Leaky Relu

The $\mathrm{leakyrelu}$ activation function is parametrized by a scalar value $a \in (0, 1)$, and is defined as

$$\mathrm{leakyrelu}_a(x) = \max(ax, x) = ax + (1-a)\,\mathrm{relu}(x), \tag{26}$$

As a sum of a linear term and a relu nonlinearity, any $\mathrm{leakyrelu}$ can be reformulated in terms of the relu-nonlinearity, and the full set of IQCs for relu is still available.

B. Other scalar nonlinearities

Many other activation functions satisfy rate-limits, and IQCs 3, 5, 7, 8, 10 and 11 hold for these as well.

As an example, we conisider the $\tanh$ activation function which satisfies $\tanh(0) = 0$ and has a derivative $\tanh'$ satisfying $0 \leq \tanh'(x) \leq 1$. The bounds are identical to the bounds on relu, and the $\tanh$ activation function satisfies IQCs 3, 5 and 11 in the same way as relu. In the relative formulation, our rescaling trick no longer works, and we are limited to IQC 11.

1) Softmax: The softmax function gives rise to a constraint like IQCs 2 and 6 due to its nonnegativity. Its rate-limit also enables constraints like IQCs 3, 7, 8, 11, but only when mixing components of the same softmax unit.

In the relative and incremental setting, we have constraints analogous to IQCs 8 and 11.

As softmax is not a diagonal nonlinearity, we cannot directly use the result of [1]. Zames-Falb-style results for rotation-free nonlinearities (like softmax) are available from [9], but those results are not applicable to softmax, as $\mathrm{softmax}(0) \neq 0$.

VII. Selection of constraints

The library of constraints provided above aims at completeness, but, in practice, we want to limit our use to a subset of the available constraints.

Assembling all inputs to all relu-units in a single vector, as in (3), a constraint on the full vector introduces a large number of decision variables in the resulting SDP problem. A sparse SDP problem with a smaller number of decision variables is obtained by using one set of constraints per layer, only using ξ_k and ζ_k for a single k in each constraint. However, our constraint now only expresses knowledge about correlations withinthe same layer, and our resulting set of guarantees will not be as strong as with the full constraint.

As an intermediate step, we select constraints that mix adjacent layers, or, more generally, adjacent blocks in our block diagram. The structure of the neural network already couples adjacent layers, and our constraints follow this structure. This choice obtains good expressivity in the contraints, and retains a high degree of sparsity in the resulting SDP problem.

VIII. Examples

A. Static Gain of a Deep Network

We consider a simple network with a single hidden layer defined by the equations

$$\begin{aligned} x_1 &= \begin{pmatrix} 1 \\ -1 \end{pmatrix} y_0; \;\; y_1 = \mathrm{relu}(x_1) \\ x_2 &= \begin{pmatrix} 1 & -1 \end{pmatrix} y_1 \end{aligned} \tag{27}$$

where y_0 and x_2 are the scalar input and output, respectively.

The network is designed so that x_2 is identically equal to y_0 in all cases, and we compare analyses of the static gain of this network in the absolute and incremental settings.

In the incremental setting, we have two sets of signals with different superscripts $y_i^{(1)}$ and $y_i^{(2)}$, and we focus on the static gain from $\Delta y_0 = y_0^{(1)} - y_0^{(2)}$ to $\Delta x_2 = x_2^{(1)} - x_2^{(2)}$. In the incremental setting, all we have access to is the rate-limit, and we cannot exclude that $\mathrm{relu}(x) = x$ might hold. In that case, the full network behaviour would be

$$\Delta x_2 = \begin{pmatrix} 1 \\ -1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & -1 \end{pmatrix} \Delta y_0 = 2\Delta y_0 \tag{28}$$

and a gain of 2 cannot be excluded in the incremental setting.

In the absolute setting, we have access to the equality $y(x - x) = 0$, telling us that y is either 0 or x, as well as to $y \geq 0$ and $y \geq x$, allowing us to deduce which of the two cases actually appear.

The numerical procedure uses the constraints and the S-procedure to transform the gain-bound claim $x_2^2 - \lambda^2 y_0^2 \leq 0$ to an SDP problem. Numerically, we do find the values 1 and 2 for the absolute and incremental setting, respectively.

We conclude that the static constraints available in the absolute setting have more expressive power than those available in the incremental setting.

Consequently, a deep network build by stacking several identical layers on top of each other gives us a gain bound

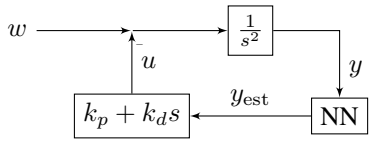


Fig. 4: Block diagram of system used in Closed Loop example.

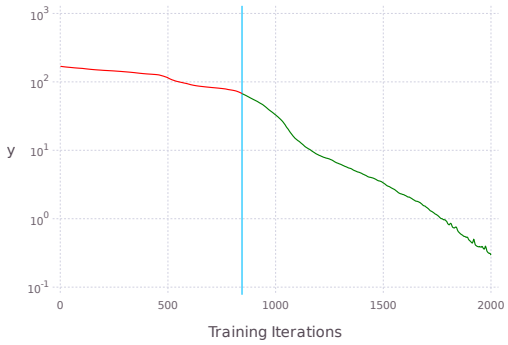


Fig. 5: Training loss vs. number of gradient steps, with guaranteed stability achieved after 884 steps.

that grows exponentially with depth in the incremental setting, while it remains constant in the absolute setting.

B. Closed Loop Stability

We consider a double integrator controlled by a stabilizing PD-controller. The output is filtered through a neural network before passing to the controller. Our network structure is a few-layer variant of the network from the previous example, with their first-layer weights fixed, and with remaining weights trained from random initialization. We use the main theorem of [6] together with the KYP lemma [7] to obtain closed-loop stability guarantees based on the IQC formalism.

As the previous example showed that a gain-bound of 1 could be obtained with only the static constraints, we use only the IQCs that immediately follow from those constraints, and we only use constraints that mix adjacent layers in the netural network.

We train the network on synthetic data y with loss function $(y - y_{\text{est}})^2$. As our controller stabilizes the double integrator, a perfect network will ensure closed loop stability. Our training progress is shown in Fig. 5 and shows that the loss function decreases throughout training. After each gradient step we use the IQC framework to test for closed loop stability, and we obtain closed loop stability at each gradient step after the 884th one.

While the first example showed that the absolute setting provides greater expressivity than the incremental, in terms of its constraints, this example shows that the full power of the absolute setting is not always needed.

IX. Conclusion

We have provided a list of constraints that can be used for obtaining guarantees for systems involving neural networks. Many of the relations have been published in prior work while some are new.

We have split the discussion into three analysis settings, and provided two small examples showing that the absolute setting can provide stronger guarantees than the incremental setting, but that the full power of the absolute setting is often not needed.

X. Acknowledgement

This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

References

[1] F.J. D'Amato, M.A. Rotea, A.V. Megretski, and U.T. Jönsson. New results for analysis of systems with repeated nonlinearities. *Automatica*, 37(5):739 – 747, 2001.

[2] Mahyar Fazlyab, Manfred Morari, and George J. Pappas. Safety verification and robustness analysis of neural networks via quadratic constraints and semidefinite programming. *IEEE Transactions on Automatic Control*, 67(1):1–15, 2022.

[3] Mahyar Fazlyab, Alexander Robey, Hamed Hassani, Manfred Morari, and George J. Pappas. Efficient and accurate estimation of lipschitz constants for deep neural networks. *CoRR*, abs/1906.04893, 2019.

[4] Navid Hashemi, Mahyar Fazlyab, and Justin Ruths. Performance bounds for neural network estimators: Applications in fault detection. In *2021 American Control Conference (ACC)*, pages 3260–3266, 2021.

[5] Haimin Hu, Mahyar Fazlyab, Manfred Morari, and George J. Pappas. Reach-sdp: Reachability analysis of closed-loop systems with neural network controllers via semidefinite programming. *2020 59th IEEE Conference on Decision and Control (CDC), Decision and Control (CDC), 2020 59th IEEE Conference on*, pages 5929 – 5934, 2020.

[6] A. Megretski and A. Rantzer. System analysis via integral quadratic constraints. *IEEE Transactions on Automatic Control*, 42(6):819–830, 1997.

[7] Anders Rantzer. On the kalman—yakubovich—popov lemma. *Systems & Control Letters*, 28(1):7–10, 1996.

[8] Max Revay, Ruigang Wang, and Ian R. Manchester. A convex parameterization of robust recurrent neural networks. *IEEE Control Systems Letters*, 5(4):1363–1368, 2021.

[9] M.G. Safonov and V.V. Kulkarni. Zames-falb multipliers for mimo nonlinearities. *Proceedings of the 2000 American Control Conference. ACC (IEEE Cat. No.00CH36334), American Control Conference, 2000. Proceedings of the 2000*, 6:4144, 2000.

[10] Jan C. Willems. Dissipative dynamical systems part i: General theory. *Archive for Rational Mechanics and Analysis*, 45(5):321–351, 1972.

[11] Jan C. Willems. Dissipative dynamical systems part ii: Linear systems with quadratic supply rates. *Archive for Rational Mechanics and Analysis*, 45(5):352–393, 1972.

[12] V. A. Yakubovich. Frequency conditions for the absolute stability of control systems with several nonlinear or linear nonstationary units. *Automat. Telemech.*, page 5–30, 1967.

[13] He Yin, Peter Seiler, and Murat Arcak. Stability analysis using quadratic constraints for systems with neural network controllers. 2020.

[14] G. Zames. On the input-output stability of time-varying nonlinear feedback systems part one: Conditions derived using concepts of loop gain, conicity, and positivity. *IEEE Transactions on Automatic Control*, 11(2):228–238, 1966.

[15] G. Zames and P. L. Falb. Stability conditions for systems with monotone and slope-restricted nonlinearities. *SIAM Journal on Control*, 6(1):89–108, 1968.