Bounding the difference between model predictive control and neural networks

Ross Drummond ROSS.DRUMMOND@ENG.OX.AC.UK
Stephen R. Duncan STEPHEN.DUNCAN@ENG.OX.AC.UK
Department of Engineering Science, University of Oxford, 17 Parks Road, OX1 3PJ, Oxford, United Kingdom

Matthew C. Turner M.C.TURNER@SOTON.AC.UK
Department of Electronics and Computer Science, University of Southampton, Southampton, UK, SO17 1BJ

Patricia Pauli PATRICIA.PAULI@IST.UNI-STUTTGART.DE
Frank Allgöwer FRANK.ALLGOWER@IST.UNI-STUTTGART.DE
Institute for Systems Theory and Automatic Control, University of Stuttgart, 70569 Stuttgart, Germany

Editors: R. Firoozi, N. Mehr, E. Yel, R. Antonova, J. Bohg, M. Schwager, M. Kochenderfer

Abstract

There is a growing debate on whether the future of feedback control systems will be dominated by data-driven or model-driven approaches. Each of these two approaches has their own complimentary set of advantages and disadvantages, however, only limited attempts have, so far, been developed to bridge the gap between them. To address this issue, this paper introduces a method to bound the worst-case error between feedback control policies based upon model predictive control (MPC) and neural networks (NNs). This result is leveraged into an approach to automatically synthesize MPC policies minimising the worst-case error with respect to a NN. Numerical examples highlight the application of the bounds, with the goal of the paper being to encourage a more quantitative understanding of the relationship between data-driven and model-driven control.

Keywords: Neural network robustness, model predictive control, semi-definite programming.

1. Introduction

Two distinct approaches for the feedback control of dynamical systems are emerging, categorised into *model-driven* and *data-driven* methods. Model-driven approaches rely upon a physical model of the system to make predictions about its future behaviour and then compute a control input to shape this future response (Green and Limebeer (2012)). Data-driven methods, in particular model-free reinforcement learning, forgo the need of a predictive model and instead rely upon identifying patterns within the trajectory data itself to determine control policies (Sutton and Barto (2018)). The benefits and limitations of these two approaches are also quite distinct. Model-driven approaches are often equipped with performance certificates for properties such as closed-loop stability, optimality and robustness (Green and Limebeer (2012)), with these certificates making them more suited for safety-critical systems. But model-driven methods generally struggle in applications where existing models remain underdeveloped and when applied to large-scale systems as computing the control polices can become intractable (Šiljak and Zečević (2005)).

Data-driven methods, by contrast, do not require an accurate model of the system (Sutton and Barto (2018)) and can find control policies which would be unobtainable using model-based approaches (as they avoid the biases inherent within models), a point illustrated by the famous "move

37" from AlphaGo (Silver et al. (2017)) which confounded experts' analysis. A main drawback of most data-driven methods is their general lack of rigorous guarantees, e.g. robustness (Szegedy et al., 2013), as small changes in their inputs can lead to large changes in their outputs, and explaining their behaviour remains an open problem. Robustness issues are a primary reason why we have yet to fully see the heralded successes of data-driven methods transfer from controlled, laboratory settings to real-life practical applications.

The distinction between model and data driven control methods has driven interest in bridging the gap between them. In particular, there is a need to better characterise the relationship between each approach's benchmark methods which are, arguably, model predictive control (MPC) for model-based control and deep neural networks (NNs) for the data-driven approach. Relating these two approaches will not only bring greater understanding but will also encourage the development of new control methods that combine each approach's complementary advantages.

Contribution: The two main results of this paper are:

1. A method to bound the difference between a given MPC and a NN controller (Theorem 10).
2. A method to synthesise MPCs that robustly approximate neural networks (Theorem 11).

The overriding theme of these results is to quantify the similarity between MPC and neural network control actions, with the bounds holding robustly, in the sense that they hold for fairly generic neural network architectures and MPC cost functions considering pre-defined hyper-rectangles as input constraints. The bounds are obtained by solving a semi-definite program (SDP), a class of convex optimisation programs which have been extensively used for analysing the robustness of both NNs and MPC, as in Fazlyab et al. (2020); Li et al. (2006).

Literature: Several existing results have also been concerned with relating the disparate control schemes of MPC and NNs. Of particular note are the results on explicit MPC where the user takes advantage of the fact that the optimal solution to the linear MPC problem can be expressed as a piece-wise linear function (Bemporad et al. (2002)). By identifying which region of this piece-wise linear function the current state of the system is in, known as solving the "point location problem", then the MPC action can be obtained without online computation of the finite horizon optimal control problem. The main drawback of explicit MPC is that solving the point location problem can be challenging (since the complexity of the piece-wise affine function, measured by the number of piece-wise regions, scales exponentially with the state dimension and the number of constraints, Alessio and Bemporad (2009)), which has limited its use in practice. To avoid this computational bottleneck, it has been proposed to approximate the piece-wise linear optimal MPC policy with a neural network, Parisini and Zoppoli (1995); Zhang et al. (2019), as the approximating NN will also be a piece-wise linear function if a ReLU function is used for the activation function (Darup, 2020). Evaluating a NN is generally much simpler than solving the point location problem but can introduce an error which has proven difficult to bound. The bounds formulated in this paper are able to quantify the worst-case error of this approximation.

The presented results can be understood within the more general context of applying control theory to verify the robustness of neural networks. The theoretical framework for these methods are now classic, emerging from studies such as Chu and Glover (1999), Barabanov and Prokhorov (2002), Angeli (2009), and have been extended in recent years in studies such as Fazlyab et al. (2020). The presented results contrast with these existing robustness problems in that they are concerned with the robustness of a neural network with respect to MPC, not to itself. Some of the

authors used similar ideas to quantify the similarity of one neural network with respect to another (Li et al. (2021)), which allowed the approximation errors of pruned, quantised and reduced-order neural networks to be bounded (Li et al. (2021); Drummond et al. (2021)). The recently released study Fabiani and Goulart (2021) which was also concerned with bounding the error between MPC and a NN must also be highlighted. The method developed in that paper involved solving a mixed integer optimisation problem to bound the error between MPC and NNs with ReLU activation functions in particular. In contrast, the results of this paper solve a SDP to verify the robustness of fairly generic NN architectures with respect to MPC. Finally, it is highlighted how this paper is concerned with quantifying the similarity between MPC and a NN but it is known, that, in certain circumstances, these two policies can be shown to be equivalent, Darup (2020).

Examples of NN robustness failures (Szegedy et al. (2013)) have driven mounting interest in using semi-definite programming for NN analysis and synthesis Fazlyab et al. (2020). These results include the estimation of Lipschitz constants for NN mappings (Fazlyab et al., 2019; Pauli et al., 2021b) and the stability analysis of closed-loop systems using NN controllers Yin et al. (2021a). The stability analysis problem in particular can be linked to the problem of absolute stability (Desoer and Vidyasagar (2009); Barabanov and Prokhorov (2002); Chu and Glover (1999)), as a feed-forward neural network is a static, memoryless nonlinearity Pauli et al. (2021a). Within this context, Yin et al. (2021b) proposed approximating a MPC using NN controllers while satisfying LMI constraints to ensure closed-loop stability. Enforcing stability on a large-scale structure like a NN can be challenging, which motivates a workaround that finds and enforces bounds relating the two controller types. Doing so may lead to more elegant stability constraints to be formulated, as meeting these bounds have already been shown to imply stability guarantees Hertneck et al. (2018).

Notation

The set of real numbers is denoted $\mathbb{R}$ and the natural numbers are $\mathbb{N}$. For dimension n, the set of real vectors is $\mathbb{R}^n$, the set of vectors with non-negative elements is denoted $\mathbb{R}^n_+$ and the zero vector is $\mathbf{0}_n$. Element-wise positivity of a vector is denoted $>$ while positive definiteness of a matrix is denoted $\succ$, with similar definitions for negativity and semi-definiteness of matrices. The set of real matrices of dimension $n \times m$ is $\mathbb{R}^{n\times m}$ and the zero matrix is $\mathbf{0}_{n\times m}$. For dimension n, the space of positive (semi-)definite matrices is ($\mathcal{S}^n_{\succeq 0}$) $\mathcal{S}^n_{\succ 0}$, non-negative diagonal matrices is $\mathbb{D}^n_{\geq 0}$ and the identity matrix is I_n. We use the $\star$ notation for symmetric matrices, as in $\begin{bmatrix} A & B \\ B^\top & C \end{bmatrix} = \begin{bmatrix} A^\top & B^\top \\ B & C^\top \end{bmatrix} = \begin{bmatrix} A & B \\ \star & C \end{bmatrix}$. Several of the proofs of the lemmas are immediate and so are not shown.

2. Problem Set-up

This section introduces the general problem considered in the paper of computing bounds and approximations between MPCs and NNs.

2.1. Model predictive control

Consider input-constrained model predictive control with linear dynamics and a quadratic cost.

Definition 1 *Consider a system with linear dynamics*

$$x[k+1] = Ax[k] + Bu[k], \tag{1}$$

with state $x[k] \in \mathbb{R}^{n_x}$ *and input* $u[k] \in \mathbb{R}$. *For some finite horizon* $N \in \mathbb{N}$, *the input constrained MPC control law with linear dynamics is defined as the solution to*

$$z_{MPC}(x[k]) = arg \min_{u} \begin{bmatrix} x[k+1] \\ \vdots \\ x[k+N] \end{bmatrix}^\top Q \begin{bmatrix} x[k+1] \\ \vdots \\ x[k+N] \end{bmatrix} + \begin{bmatrix} u[k] \\ \vdots \\ u[k+N-1] \end{bmatrix}^\top R \begin{bmatrix} u[k] \\ \vdots \\ u[k+N-1] \end{bmatrix}, \tag{2a}$$

$$subject\ to\ Lu \leq b,\ b \geq 0, \tag{2b}$$

for some $Q \in \mathbb{S}^{Nn_x}_{\succeq 0}$, $R \in \mathbb{S}^{N}_{\succeq 0}$, $L \in \mathbb{R}^{n_{in} \times N}$ *and* $b \in \mathbb{R}^{n_{in}}_{+}$.

The linear dynamics of (1) mean that the optimisation problem (2) can be converted into a quadratic program (QP).

Definition 2 *The MPC control law defined by Definition 1 can be expressed as the solution to a constrained quadratic program*

$$z_{MPC}(x[k]) = arg \min_{u} u^\top H u + u^\top h x[k] \tag{3a}$$

$$subject\ to\ Lu \leq b,\ b \geq 0, \tag{3b}$$

for some $H \in \mathcal{S}^{N}_{\succ 0}$, $h \in \mathbb{R}^{N \times n_x}$ *parameterised by the matrices* $(A,\ B,\ Q,\ R)$ *of Definition 1 and* $b \in \mathbb{R}^{n_{in}}_{+}$. *The first instant of the QP solution from* (3) *is taken as the MPC control action, as in*

$$u_{MPC}(x[k]) = \begin{bmatrix} 1, & \mathbf{0}_{N-1}{}^\top \end{bmatrix} z_{MPC}(x[k]) = C_{MPC} z_{MPC}(x[k]). \tag{4}$$

2.2. Neural networks

The class of neural networks considered in this paper are those which can be structured in an implicit form (El Ghaoui et al. (2021)).

Definition 3 *The considered class of neural networks* $u_{NN}(x[k]) : \mathbb{R}^{n_x} \to \mathbb{R}$ *are those defined by the implicit form*

$$z_{NN} = \phi(W z_{NN} + W_0 x[k] + \beta), \quad \phi(\cdot) : \mathbb{R}^M \mapsto \mathbb{R}^M, \tag{5a}$$

$$u_{NN}(x[k]) = C_{NN} z_{NN} + D_{NN}, \tag{5b}$$

where $C_{NN} = \begin{bmatrix} \mathbf{0}^\top_{n_\ell}, & \dots, & \mathbf{0}^\top_{n_\ell}, & W^o \end{bmatrix}$, $D_{NN} \in \mathbb{R}$ *and*

$$z_{NN} = \begin{bmatrix} z^1_{NN} \\ z^2_{NN} \\ \vdots \\ z^\ell_{NN} \end{bmatrix}, W = \begin{bmatrix} W^{1,1} & W^{1,2} & \dots & W^{1,\ell} \\ W^{2,1} & W^{2,2} & \ddots & \vdots \\ \vdots & \ddots & \ddots & W^{\ell-1,\ell} \\ W^{\ell,1} & \dots & W^{\ell,\ell-1} & W^{\ell,\ell} \end{bmatrix}, W_0 = \begin{bmatrix} W^{1,0} \\ W^{2,0} \\ \vdots \\ W^{\ell,0} \end{bmatrix}, \beta = \begin{bmatrix} \beta^0 \\ \beta^1 \\ \vdots \\ \beta^\ell \end{bmatrix}. \tag{5c}$$

In the above, $\phi(\cdot)$ are the nonlinear activation functions which could be any function satisfying one or more of the function properties of Definition 13 (Appendix 1), acting element-wise upon their arguments. The NN will be assumed to have ℓ hidden layers and, for the sake of notational simplicity, it will be assumed that each layer will be of dimension n_ℓ, as in $z^j_{\text{NN}} \in \mathbb{R}^{n_\ell}$, $W^{i,j} \in \mathbb{R}^{n_\ell \times n_\ell}$, $W^{i,0} \in \mathbb{R}^{n_\ell \times n_x}$, $\beta^j \in \mathbb{R}^{n_\ell}$ and $W^{o\top} \in \mathbb{R}^{n_\ell}$ for $i = 1, \dots, \ell$, $j = 1, \dots, \ell$. The total length of the vector $z_{\text{NN}} \in \mathbb{R}^M$ is defined as $M = n_\ell \times \ell$.

Adopting the implicit NN structure of (5) in Definition 3, also known as an equilibrium network Revay et al. (2020); Bai et al. (2019), helps simplify the notation whilst also ensuring the results are applicable to a wide class of NN architectures, including deep recurrent and feed-forward NNs.

2.3. Bounds between MPC and NN

Bounds quantifying the similarity between the NN and MPC control policies described in Definitions 2 and 3 are sought. These bounds are formalised as solutions to the following problem.

Problem 1 *For any state $x[k] \in \mathcal{X}$, find $\gamma_x \geq 0$ and $\gamma \geq 0$ such that the error between the neural network and MPC control actions, namely $u_{NN}(x[k]) - u_{MPC}(x[k])$, is bounded by*

$$\|u_{NN}(x[k]) - u_{MPC}(x[k])\|_2^2 \leq \gamma_x \|x[k]\|_2^2 + \gamma, \quad \forall x[k] \in \mathcal{X}. \tag{6}$$

Remark 4 *If both controllers enforce $x[k] = 0$ as an equilibrium point of the dynamical system, then it is possible to obtain $\gamma = 0$ since the two control actions should be equivalent at that point.*

3. Quadratic Constraints

For the bounds of Problem 1 to hold robustly, as in for all inputs $x[k]$ in some set $\mathcal{X}$, characterisations of both the nonlinear activation functions of the neural network and the solution structure of the MPC control law have to be included within the problem formulation. Here, this information is incorporated using quadratic constraints.

3.1. Quadratic constraints for the neural network

The first step of this process is to determine the quadratic constraint of the nonlinear activation functions $\phi(.)$. Many different activation functions have been implemented within neural networks, including the tanh and ReLU functions, with each having their own characteristics. In general, each of these candidate activation functions satisfies certain properties which can be incorporated within a robust optimisation framework to compute the error bounds.

Examples of these function properties are given in Definition 13 (Appendix 1). Most commonly adopted activation functions, including the tanh and ReLU, satisfy at least one of these properties. For example, the tanh function is slope-restricted, bounded and sector bounded, the ReLU function is slope-restricted while both it and its complement are positive and satisfy the complementarity conditions.

By satisfying one or more of these function properties, the activation functions also satisfy quadratic constraints, as detailed in Lemma 14 (Appendix 2). These quadratic constraints are defined with the multipliers $\mathbf{T}^i$ ($i \in \{\mathrm{s}, \mathrm{sl}, +, c+, B, 0, \times, \otimes\}$), and in the following, the notation $\mathbf{T} \in \mathbb{T}$ will be used to collect all of the relevant $\mathbf{T}^i$'s ($i \in \{\mathrm{s}, \mathrm{sl}, +, c+, B, 0, \times, \otimes\}$) from Lemma 14 (Appendix 2), with the set $\mathbb{T}$ characterising the positivity conditions of the $\mathbf{T}^i$'s.

If the neural network's activation function satisfies more than one of these quadratic constraints, as in it satisfies one or more of the properties in Lemma 14 (Appendix 2), then each of these quadratic constraints can be combined into a single quadratic constraint defined by a matrix λ, as considered previously in results such as Drummond et al. (2021); Fazlyab et al. (2020). In this way, information about the nonlinear activation functions of the neural network can be incorporated into the robust optimisation problem for the error bounds.

Lemma 5 *If the activation function $\phi(\cdot)$ satisfies one or more of the quadratic constraints of Lemma 14 (Appendix 2), then there exists matrices*

$$\hat{\lambda}(\mathbf{T}) = \begin{bmatrix} \hat{\lambda}_{11} & \hat{\lambda}_{12} & \hat{\lambda}_{13} \\ \star & \hat{\lambda}_{22} & \hat{\lambda}_{23} \\ \star & \star & \hat{\lambda}_{33} \end{bmatrix}, \quad \lambda(\mathbf{T}) = \begin{bmatrix} \lambda_{11} & \lambda_{12} & \lambda_{13} \\ \star & \lambda_{22} & \lambda_{23} \\ \star & \star & \lambda_{33} \end{bmatrix}, \tag{7}$$

defined by the $\mathbf{T}^i$'s ($i \in \{s, sl, +, c+, B, 0, \times, \otimes\}$) of Lemma 14 (Appendix 2), such that with the vectors

$$\hat{\mu}(x[k]) = \left[(W z_{NN} + W_0 x[k] + \beta)^\top, \quad {z_{NN}}^\top, \quad 1\right]^\top, \tag{8a}$$

$$\mu(x[k]) = \left[x[k]^\top, \quad {z_{NN}}^\top, \quad 1\right]^\top, \tag{8b}$$

the following inequality holds

$$s_{NN}(x[k]) = \hat{\mu}(x[k])^T \hat{\lambda}(\mathbf{T}) \hat{\mu}(x[k]) = \mu(x[k])^T \lambda(\mathbf{T}) \mu(x[k]) \geq 0. \tag{9}$$

Proof Immediate from applying the S-procedure (Jönsson (2001)) on Lemma 14 (Appendix 2). ■

3.2. Quadratic constraints for model predictive control

Following (Li et al., 2006, Result 5), a quadratic constraint can also be characterised for the solution of an MPC control law defined by the QP of Definition 2. In Li et al. (2006), this quadratic constraint was introduced to characterise the closed-loop stability of linear systems controlled by MPC, using Zames-Falb multipliers to provide the stability certificates (Zames and Falb (1968); Turner and Drummond (2021)). Here, we exploit this quadratic constraint to include information about the solution structure of the MPC control law into the error bound computation.

Theorem 6 *[Li et al. (2006), Result 5] The MPC control law characterised by the QP of Definition 2 satisfies, for any $\tau_{QP} \geq 0$,*

$$s_{QP}(x[k]) = -\tau_{QP} \left({z_{MPC}}^\top H z_{MPC} + {z_{MPC}}^\top h x[k] \right) \geq 0. \tag{10}$$

On top of this quadratic constraint for the QP cost function (10), another quadratic constraint for the inequality bounds (3b) can also be devised since, for any positive vector $q \in \mathbb{R}^{n_{in}}_+$,

$$s_{in}(x[k]) = q^\top (b - L z_{MPC}) + (b - L z_{MPC})^\top q \geq 0. \tag{11}$$

The function properties of Definition 13 (Appendix 1) can be used to build satisfactory q. For example, if $\phi(.)$ and its complement are positive, as for the ReLU, then, for some $\tau_{in} \in \mathbb{R}^{n_{in}}_+$, $(\tau^+_{in}, \tau^{c+}_{in}) \in \mathbb{R}^{n_{in} \times M}_+$, one could define

$$q = \tau_{in} + (\tau^+_{in} + \tau^{c+}_{in}(I_M - W)) z_{\text{NN}} - \tau^{c+}_{in} W_0 x[k] - \tau^{c+}_{in} \beta \tag{12}$$

which links the NN and MPC vectors, z_{NN} and z_{MPC} within the quadratic constraint. The quadratic constraint for the inequality constraints $s_{in}(x[k])$ can then be combined with that of the cost function from Theorem 6 to give the following characterisation of the MPC solution.

Lemma 7 *For any $x[k] \in \mathbb{R}^{n_x}$, the MPC solution of the QP given in Definition 2 satisfies*

$$s_{MPC}(x[k]) = s_{QP}(x[k]) + s_{in}(x[k]) \geq 0, \tag{13}$$

with $s_{QP}(x[k])$ defined in (10) and $s_{in}(x[k])$ in (11).

3.3. Quadratic constraints for the input

Information about the input constraint set $x[k] \in \mathcal{X}$ must also be included within the robust optimisation problem for the bounds. By restricting the input space to a hyper-rectangle, the following lemma defines a quadratic constraint for this set containment.

Definition 8 *For some upper $\overline{x}$ and lower $\underline{x}$ bound, define $\mathcal{X} := \{x[k] : x[k] \geq \underline{x},\, x[k] \leq \overline{x}\}$.*

Lemma 9 (Fazlyab et al. (2020)) *If $x[k] \in \mathcal{X}$, then, for any $\tau_x \in \mathbb{D}_+^{n_x}$, the following holds*

$$s_{\mathcal{X}}(x[k]) = \begin{bmatrix} x[k]^\top & 1 \end{bmatrix} \begin{bmatrix} -2\tau_x & \tau_x(\overline{x} + \underline{x}) \\ \star & -(\overline{x}^\top \tau_x \underline{x} + \underline{x}^\top \tau_x \overline{x}) \end{bmatrix} \begin{bmatrix} x[k] \\ 1 \end{bmatrix} \geq 0. \tag{14}$$

4. Error bound between MPC and NN

The quadratic constraints of Lemmas 5, 7 and 9 allow information about the nonlinear activation functions, the solution structure of the MPC controller and the set containment $x[k] \in \mathcal{X}$ to be included within the error bound optimisation. In this way, the error bounds of Problem 1 can be computed by solving the following SDP.

Theorem 10 *Consider a given NN structure of Definition 3 and MPC control law of Definition 2. For given weights ω_x and ω, if there exists a solution to*

$$\min_{\mathbf{T},\, \tau_{QP},\, \tau_{in},\, \gamma_x,\, \gamma,\, \tau_x} \quad \omega_x \gamma_x + \omega\gamma \tag{15a}$$

$$\text{subject to:} \quad \Lambda_{NN}(\mathbf{T}) + \Lambda_{MPC}(\tau_{QP}, \tau_{in}) + X(\tau_x) + \Omega(\gamma_x,\, \gamma) \preceq 0, \tag{15b}$$

$$\mathbf{T} \in \mathbb{T}, \tau_x \in \mathbb{D}_+^{n_x},\, \tau_{QP} \geq 0, \tau_{in} \in \mathbb{R}_+^{n_{in}},\, \gamma_x \geq 0, \gamma \geq 0, \tag{15c}$$

where the matrices $\Lambda_{NN}(\mathbf{T})$, $\Lambda_{MPC}(\tau_{QP}, \tau_{in})$, $X(\tau_x)$ and $\Omega(\gamma_x,\, \gamma)$ are defined in (23) of Appendix 3, then

$$\|u_{NN}(x[k]) - u_{MPC}(x[k])\|_2^2 \leq \gamma_x \|x[k]\|_2^2 + \gamma, \quad \forall x[k] \in \mathcal{X}. \tag{16}$$

Proof Multiplying (15b) on the left by

$$\zeta(x[k]) = \begin{bmatrix} x[k]^\top, & z_{\text{NN}}{}^\top, & z_{\text{MPC}}{}^\top, & 1 \end{bmatrix}^\top \tag{17}$$

and on the right by it's transpose implies that, when $x[k] \in \mathcal{X}$, then

$$s_{NN}(x[k]) + s_{\mathcal{X}}(x[k]) + s_{MPC}(x[k]) + \|u_{\text{NN}}(x[k]) - u_{\text{MPC}}(x[k])\|_2^2 - \gamma_x \|x[k]\|_2^2 - \gamma \leq 0. \tag{18}$$

Lemmas 5, 7 and 9 state that $s_{NN}(x[k]) \geq 0$, $s_{\mathcal{X}}(x[k]) \geq 0$, $s_{MPC}(x[k]) \geq 0$ when $x[k] \in \mathcal{X}$. Hence, the bound (16) must hold. ■

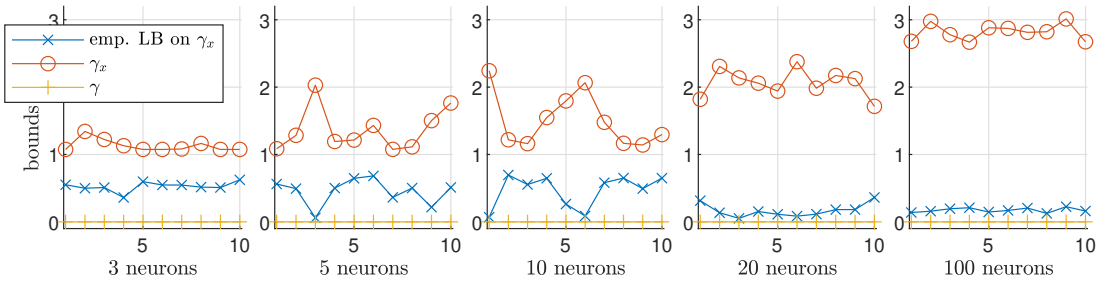

Figure 1: Bounds γ and γ_x and empirical lower bound on γ_x for respectively 10 NNs with $\{3, 5, 10, 20, 100\}$ neurons in the hidden layer.

5. Automatic synthesis of an MPC approximating a NN

Theorem 10 allows the error between *given* MPC and NN control policies to be bounded. This idea can be extended to *synthesize* MPC control laws that minimise the worst-case error to the NN.

Theorem 11 *Consider a given NN structure of* (5)*. For the given weights ω_x and ω, if there exists a solution to*

$$\min_{\mathbf{T}, H, h, L, b, \gamma_x, \gamma, \tau_x} \quad \omega_x \gamma_x + \omega\gamma \tag{19a}$$

$$\text{subject to:} \quad \Lambda_{NN}(\mathbf{T}) + \check{\Lambda}_{MPC}(H, h, L, b) + X(\tau_x) + \Omega(\gamma_x, \gamma) \preceq 0, \tag{19b}$$

$$\mathbf{T} \in \mathbb{T}, \tau_x \in \mathbb{D}_+^{n_x}, \gamma_x \geq 0, \gamma \geq 0, H \in \mathcal{S}_{\succ 0}^N, h \in \mathbb{R}^{N \times n_x}, b \in \mathbb{D}_+^{n_{in}}, L \in \mathbb{R}^{n_{in} \times N} \tag{19c}$$

with the matrices $\Lambda_{NN}(\mathbf{T})$, $\check{\Lambda}_{MPC}(H, h, L, b)$, $X(\tau_x)$ and $\Omega(\gamma_x, \gamma)$ defined in Appendix 3, then

$$\|u_{NN}(x[k]) - u_{MPC}(x[k])\|_2^2 \leq \gamma_x \|x[k]\|_2^2 + \gamma, \quad \forall x[k] \in \mathcal{X}. \tag{20}$$

Proof From Lemma 7, the computed (H, h, L, b) define a quadratic program from Definition 2 for which $s_{MPC}(x[k]) \geq 0$ using $\tau_{\text{QP}} = 1$ and each element of τ_{in} being one. The proof is then immediate from the argument of Theorem 10. ■

Remark 12 *The main advantage of using Theorem 11 to generate an approximating MPC is that it includes the worst-case approximation error directly within the cost function being minimised in* (19)*.*

6. Numerical examples

In the following, we apply Theorem 10 to bound the difference between an MPC and a NN controller. All NNs were trained using Pytorch and all SDPs were solved in Matlab using the parser `YALMIP` (Lofberg (2004)) and solver `MOSEK` (Andersen and Andersen (2000)).

We consider the following linear time-invariant system

$$x[k+1] = \begin{bmatrix} 4/3 & -2/3 \\ 1 & 0 \end{bmatrix} x[k] + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u[k], \tag{21}$$

and solve (3) choosing the horizon length $N = 10$ and the input constraints $-0.1 \leq u_{\text{MPC}} \leq 0.1$ such that $L = \begin{bmatrix} 1 & -1 \end{bmatrix}^\top$, $b = \begin{bmatrix} 0.1 & 0.1 \end{bmatrix}^\top$. The MPC quadratic cost function of (2) was parameterised by

$$\widetilde{P} = \begin{bmatrix} 7.1667 & -4.2222 \\ -4.2222 & 4.6852 \end{bmatrix}, \ \widetilde{Q} = \begin{bmatrix} 1 & -2/3 \\ -2/3 & 3/2 \end{bmatrix}, \ \widetilde{R} = 1,$$

yielding $Q = \text{blkdiag}(\widetilde{Q}, \widetilde{Q}, \ldots, \widetilde{Q}, \widetilde{P})$ and $R = \text{blkdiag}(\tilde{R}, \ldots, \tilde{R})$. Subsequently, feedforward neural networks with one hidden layer and ReLU activation were trained using backpropagation to approximate this MPC controller. The results are detailed in Figure 1 where upper and lower bounds for the obtained approximation error are shown. The upper bound, defined in terms of γ and γ_x, was obtained by solving Theorem 10 while the empirical lower bound was obtained by sampling. For different numbers of neurons in the hidden layer, ten randomly initialised NNs were trained and their approximation errors are plotted in the figure. For all cases, it was found that $\gamma = 0$, agreeing with the prediction of Remark 4. The computed upper bound was always above the empirical lower bound, with the bounds being relatively tight for the small NNs. The larger gap between the upper and lower bounds for the larger NN examples is due to the increasing number of neurons causing an increased abstraction of the neural network mapping by the quadratic constraints, and hence an increase in the conservatism of the problem.

Conclusions

A method to bound the worst-case error between feedback control policies based upon neural networks (NNs) and model-predictive control (MPC) was introduced. Using these bounds, a method to synthesize MPC policies minimising the worst-case error with respect to a NN was developed. The overriding goal of this work was to improve our understanding of, and even quantify, the relationship between MPC and NNs, helping to bridge the gap between model-driven and data-driven control. Future work will explore using the bounds to generate closed-loop stability certificates of both the MPC and NN controllers, reducing the conservatism of the bounds and applying them to the verification of control systems used in practice.

7. Appendix

7.1. Appendix 1: Activation function properties

Definition 13 (Drummond et al. (2021)) *The activation function* $\phi(s) : \mathcal{S} \subset \mathbb{R} \to \mathbb{R}$ *satisfying* $\phi(0) = 0$ *is said to be* sector bounded *if* $\frac{\phi(s)}{s} \in [0, \delta] \ \forall s \in \mathcal{S}, \ \delta > 0$, *and* slope restricted *if* $\frac{d\phi(s)}{ds} \in [\underline{\beta}, \beta], \ \forall s \in \mathcal{S}, \ \beta > 0$. *If* $\underline{\beta} = 0$ *then the nonlinearity is* monotonic *and if* $\phi(s)$ *is slope restricted then it is also sector bounded. The activation function* $\phi(s)$ *is* bounded *if* $\phi(s) \in [\underline{c}, \overline{c}], \ \forall s \in \mathcal{S}$, *it is* positive *if* $\phi(s) \geq 0, \ \forall s \in \mathcal{S}$, *its* complement is positive *if* $\phi(s) - s \geq 0, \ \forall s \in \mathcal{S}$, *and it satisfies the* complementarity condition *if* $(\phi(s) - s)\phi(s) = 0, \ \forall s \in \mathcal{S}$.

7.2. Appendix 2: Quadratic constraints for the activation functions

Lemma 14 (Drummond et al. (2021)) *Consider the vectors $y, y_1 \in \mathbb{R}^{n_y}$, and $\upsilon \in \mathbb{R}^{n_\upsilon}$ that are mapped component-wise through the activation functions $\phi(\cdot) : \mathbb{R}^{n_y} \to \mathbb{R}^{n_y}$ and $\tilde{\phi}(\cdot) : \mathbb{R}^{n_\upsilon} \to \mathbb{R}^{n_\upsilon}$. If $\phi(y)$ is* sector-bounded*, then*

$$(\delta y - \phi(y))^T \mathbf{T}^s \phi(y) \geq 0, \quad \forall y \in \mathbb{R}^{n_y},\ \mathbf{T}^s \in \mathbb{D}_+^{n_y}; \tag{22a}$$

slope-restricted *then*

$$(\beta(y - y_1) - (\phi(y) - \phi(y_1))^T \mathbf{T}^{sl}(\phi(y) - \phi(y_1) - \underline{\beta}(y - y_1)) \geq 0; \quad \forall \{y,\, y_1\} \in \mathbb{R}^{n_y},\ \mathbf{T}^{sl} \in \mathbb{D}_+^{n_y}; \tag{22b}$$

bounded *then*

$$(\overline{c} - \phi(y))^T \mathbf{T}^B (\phi(y) - \underline{c}) \geq 0, \quad \forall y \in \mathbb{R}^{n_y},\ \mathbf{T}^B \in \mathbb{D}_+^{n_y}; \tag{22c}$$

positive *then*

$$(\mathbf{T}^+)^T \phi(y) \geq 0, \quad \forall y \in \mathbb{R}^{n_y},\ \mathbf{T}^+ \in \mathbb{R}_+^{n_y}; \tag{22d}$$

such that its complement is positive *then*

$$(\mathbf{T}^{c+})^T (\phi(y) - y) \geq 0, \quad \forall y \in \mathbb{R}^{n_y},\ \mathbf{T}^{c+} \in \mathbb{R}_+^{n_y}. \tag{22e}$$

If $\phi(y)$ satisfies the complementary *condition then*

$$(\phi(y) - y)^T \mathbf{T}^0 \phi(y) = 0, \quad \forall y \in \mathbb{R}^{n_y},\ \mathbf{T}^0 \in \mathbb{D}^{n_y}. \tag{22f}$$

Additionally, if both $\phi(y)$ and $\tilde{\phi}(\upsilon)$ and their complements are positive then so are the cross terms

$$\tilde{\phi}(\upsilon)^T \mathbf{T}^\times (\phi(y) - y) \geq 0,\ \forall \upsilon \in \mathbb{R}^{n_\upsilon}, y \in \mathbb{R}^{n_y},\ \mathbf{T}^\times \in \mathbb{R}_+^{n_\upsilon \times n_y}, \tag{22g}$$

$$\tilde{\phi}(\upsilon)^T \mathbf{T}^\otimes \phi(y) \geq 0,\ \forall \upsilon \in \mathbb{R}^{n_\upsilon}, y \in \mathbb{R}^{n_y},\ \mathbf{T}^\otimes \in \mathbb{R}_+^{n_\upsilon \times n_y}. \tag{22h}$$

7.3. Appendix 3: Matrix definitions

$$\Omega = \begin{bmatrix} -\gamma_x I_{n_x} & \mathbf{0}_{n_x \times M} & \mathbf{0}_{n_x \times N} & \mathbf{0}_{n_x} \\ \star & {C_{\text{NN}}}^\top C_{\text{NN}} & -{C_{\text{NN}}}^\top C_{\text{MPC}} & \mathbf{0}_M \\ \star & \star & {C_{\text{MPC}}}^\top C_{\text{MPC}} & -{C_{\text{MPC}}}^\top D_{\text{NN}} \\ \star & \star & \star & -\gamma \end{bmatrix},\ \Lambda_{NN} = \begin{bmatrix} \lambda_{11} & \lambda_{12} & \mathbf{0}_{n_x \times N} & \lambda_{13} \\ \star & \lambda_{22} & \mathbf{0}_{M \times N} & \lambda_{23} \\ \star & \star & \mathbf{0}_{N \times N} & \mathbf{0}_N \\ \star & \star & \star & \lambda_{33} \end{bmatrix}, \tag{23a}$$

$$\Lambda_{MPC} = \begin{bmatrix} \mathbf{0}_{n_x \times n_x} & \mathbf{0}_{n_x \times M} & -(\tau_{QP} h^\top)/2 & \mathbf{0}_{n_x} \\ \star & \mathbf{0}_{M \times M} & \mathbf{0}_{M \times N} & \mathbf{0}_M \\ \star & \star & -\tau_{QP} H & -L^\top \tau_{in} \\ \star & \star & \star & b^\top \tau_{in} + \tau_{in}^\top b \end{bmatrix}, \tag{23b}$$

$$X = \begin{bmatrix} -2\tau_x & \mathbf{0}_{n_x \times M} & \mathbf{0}_{n_x \times N} & \tau_x(\overline{x} + \underline{x}) \\ \star & \mathbf{0}_{M \times M} & \mathbf{0}_{M \times N} & \mathbf{0}_M \\ \star & \star & \mathbf{0}_{N \times N} & \mathbf{0}_N \\ \star & \star & \star & -(\overline{x}^\top \tau_x \underline{x} + \underline{x}^\top \tau_x \overline{x}) \end{bmatrix},\ \check{\Lambda}_{\text{MPC}} = \begin{bmatrix} \mathbf{0}_{n_x \times n_x} & \mathbf{0}_{n_x \times M} & -h^\top/2 & \mathbf{0}_{n_x} \\ \star & \mathbf{0}_{M \times M} & \mathbf{0}_{M \times N} & \mathbf{0}_M \\ \star & \star & -H & -L^\top \\ \star & \star & \star & b + b^\top \end{bmatrix}. \tag{23c}$$

Acknowledgments

Ross Drummond would like to thank the Royal academy of engineering for funding this research through a UKIC Fellowship. This work was partly funded by Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy - EXC 2075 - 390740016. We acknowledge the support by the Stuttgart Center for Simulation Science (SimTech). The authors thank the International Max Planck Research School for Intelligent Systems (IMPRS-IS) for supporting Patricia Pauli.

References

Alessandro Alessio and Alberto Bemporad. A survey on explicit model predictive control. In *Nonlinear model predictive control*, pages 345–369. Springer, 2009.

Erling D Andersen and Knud D Andersen. The MOSEK interior point optimizer for linear programming: an implementation of the homogeneous algorithm. In *High Performance Optimization*, pages 197–232. Springer, 2000.

David Angeli. Convergence in networks with counterclockwise neural dynamics. *IEEE Transactions on Neural Networks*, 20(5):794–804, 2009.

Shaojie Bai, J Zico Kolter, and Vladlen Koltun. Deep equilibrium models. *arXiv preprint arXiv:1909.01377*, 2019.

Nikita E Barabanov and Danil V Prokhorov. Stability analysis of discrete-time recurrent neural networks. *IEEE Transactions on Neural Networks*, 13(2):292–303, 2002.

Alberto Bemporad, Manfred Morari, Vivek Dua, and Efstratios N Pistikopoulos. The explicit linear quadratic regulator for constrained systems. *Automatica*, 38(1):3–20, 2002.

Yun-Chung Chu and Keith Glover. Bounds of the induced norm and model reduction errors for systems with repeated scalar nonlinearities. *IEEE Transactions on Automatic Control*, 44(3): 471–483, 1999.

Moritz Schulze Darup. Exact representation of piecewise affine functions via neural networks. In *Procs. of the European Control Conference*, pages 1073–1078. IEEE, 2020.

Charles A Desoer and Mathukumalli Vidyasagar. *Feedback systems: Input-output properties*. SIAM, 2009.

Ross Drummond, Mathew C Turner, and Stephen R Duncan. Reduced-order neural network synthesis with robustness guarantees. *arXiv preprint arXiv:2102.09284*, 2021.

Laurent El Ghaoui, Fangda Gu, Bertrand Travacca, Armin Askari, and Alicia Tsai. Implicit deep learning. *SIAM Journal on Mathematics of Data Science*, 3(3):930–958, 2021.

Filippo Fabiani and Paul J Goulart. Reliably-stabilizing piecewise-affine neural network controllers. *arXiv preprint arXiv:2111.07183*, 2021.

Mahyar Fazlyab, Alexander Robey, Hamed Hassani, Manfred Morari, and George J Pappas. Efficient and accurate estimation of Lipschitz constants for deep neural networks. *arXiv preprint arXiv:1906.04893*, 2019.

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*, 2020.

Michael Green and David JN Limebeer. *Linear robust control*. Courier Corporation, 2012.

Michael Hertneck, Johannes Köhler, Sebastian Trimpe, and Frank Allgöwer. Learning an approximate model predictive controller with guarantees. *IEEE Control Systems Letters*, 2(3):543–548, 2018.

Ulf Jönsson. Lecture notes on integral quadratic constraints. 2001.

Guang Li, William P Heath, and Barry Lennox. The stability analysis of systems with nonlinear feedback expressed by a quadratic program. In *Procs. of the Conference on Decision and Control*, pages 4247–4252. IEEE, 2006.

Jiaqi Li, Ross Drummond, and Stephen R Duncan. Robust error bounds for quantised and pruned neural networks. In *Learning for Dynamics and Control*, pages 361–372. PMLR, 2021.

Johan Lofberg. YALMIP: A toolbox for modeling and optimization in MATLAB. In *International Conference on Robotics and Automation*, pages 284–289. IEEE, 2004.

Thomas Parisini and Riccardo Zoppoli. A receding-horizon regulator for nonlinear systems and a neural approximation. *Automatica*, 31(10):1443–1451, 1995.

Patricia Pauli, Dennis Gramlich, Julian Berberich, and Frank Allgöwer. Linear systems with neural network nonlinearities: Improved stability analysis via acausal Zames-Falb multipliers. *arXiv preprint arXiv:2103.17106*, 2021a.

Patricia Pauli, Anne Koch, Julian Berberich, Paul Kohler, and Frank Allgower. Training robust neural networks using Lipschitz bounds. *IEEE Control Systems Letters*, 2021b.

Max Revay, Ruigang Wang, and Ian R Manchester. Lipschitz bounded equilibrium networks. *arXiv preprint arXiv:2010.01732*, 2020.

Dragoslav D Šiljak and AI Zečević. Control of large-scale systems: Beyond decentralized feedback. *Annual Reviews in Control*, 29(2):169–179, 2005.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of Go without human knowledge. *Nature*, 550(7676):354–359, 2017.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Matthew C Turner and Ross Drummond. Discrete-time systems with slope restricted nonlinearities: Zames-Falb multiplier analysis using external positivity. *International Journal of Robust and Nonlinear Control*, 31(6):2255–2273, 2021.

He Yin, Peter Seiler, and Murat Arcak. Stability analysis using quadratic constraints for systems with neural network controllers. *IEEE Transactions on Automatic Control*, 2021a.

He Yin, Peter Seiler, Ming Jin, and Murat Arcak. Imitation learning with stability and safety guarantees. *IEEE Control Systems Letters*, 2021b.

George Zames and PL Falb. Stability conditions for systems with monotone and slope-restricted nonlinearities. *SIAM Journal on Control*, 6(1):89–108, 1968.

Xiaojing Zhang, Monimoy Bujarbaruah, and Francesco Borrelli. Safe and near-optimal policy learning for model predictive control using primal-dual neural networks. In *Procs. of the American Control Conference*, pages 354–359. IEEE, 2019.