Mapping back and forth between model predictive control and neural networks

Ross Drummond ROSS.DRUMMOND@SHEFFIELD.AC.UK
Pablo R. Baldivieso-Monasterios P.BALDIVIESO@SHEFFIELD.AC.UK
Dept. of Automatic Control and Systems Engineering, University of Sheffield, Sheffield, S1 3JD, UK.

Giorgio Valmorbida GIORGIO.VALMORBIDA@CENTRALESUPELEC.FR
L2S, CentraleSupélec, CNRS, Université Paris-Saclay, Inria-Saclay Projet DISCO, Gif–sur–Yvette 91192, France.

Editors: A. Abate, K. Margellos, A. Papachristodoulou

Abstract

Model predictive control (MPC) for linear systems with quadratic costs and linear constraints is shown to admit an exact representation as an implicit neural network. A method to “unravel” the implicit neural network of MPC into an explicit one is also introduced. As well as building links between model-based and data-driven control, these results emphasize the capability of implicit neural networks for representing solutions of optimisation problems, as such problems are themselves implicitly defined functions.

Keywords: Model predictive control, neural networks

Introduction

For control problems beyond the capabilities of the classical methods, such as PID controllers, two important benchmarks are model predictive control (MPC) and reinforcement learning policies built upon neural networks (NNs). These two approaches are generally quite distinct; MPC is a *model-based* method which uses a physical model of the system to predict how the trajectory of the system will evolve in the future and then solves an optimisation problem to optimally control the system whilst ensuring some control and safety constraints. Neural networks are the foundations of many *data-driven* control methods, with the neural networks trained to control the system by optimising over trajectory data from experiments. The properties of these two methods are also distinct; MPC can provide strong guarantees on robustness and optimality whilst NNs can be applied to problems where the problem’s objectives, constraints and models may be unknown.

In spite of their differences, there has been strong interest in linking MPC and NN controllers. Typically, the motivation to link the two has been driven by the need to improve the scalability of MPC, since evaluating a NN is typically much simpler than computing the solution of an MPC problem. Various schemes have been proposed to link MPC and NNs in this way, including Fahandezh-Saadi and Tomizuka (2020); Xu (2021); Karg and Lucia (2020); Ferlez and Shoukry (2020); Åkesson and Toivonen (2006); Kittisupakorn et al. (2009); Xiao et al. (2022); Hertneck et al. (2018); Soloperto et al. (2020) and Ahn et al. (2022) which used imitation learning. An overriding theme of these approaches is to exploit the fact that the optimal solution of linear quadratic MPC is a piece-wise linear function, and so a piece-wise linear function in the form of a neural network should approximate it well. As these studies show, this motivation is well-founded. However, even

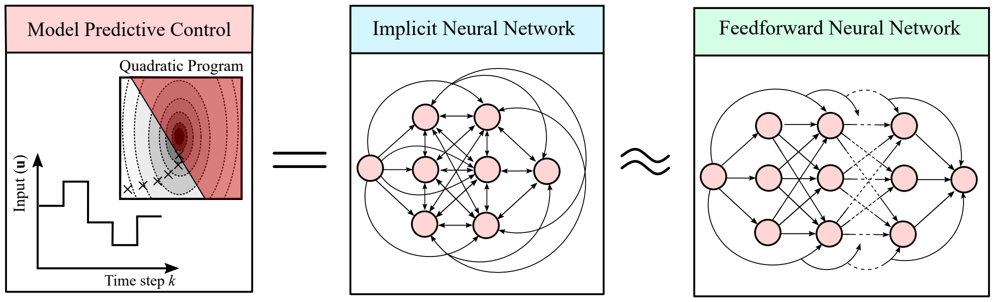

Figure 1: Linear quadratic model predictive control (MPC) is shown to be structured as an implicit neural network. This implicit neural network is then approximated by a feed-forward one. This neural network structure builds upon the results of Valmorbida and Hovd (2023) on the linear complementarity formulation of the *MPC-QP* problem.

then, approximating MPC with a NN generally leads to approximation errors, and it is only recently that such errors have been bounded Drummond et al. (2022); Fabiani and Goulart (2022); Schwan et al. (2023). In some special cases, exact representations of MPC policies using explicit NNs have been obtained; however strong restrictions on the system dynamics have been imposed in these cases Teichrib and Darup (2022), for example in Teichrib and Darup (2021) where the analysis was restricted to single state systems .

Together, these results highlight the growing interest in linking the disparate control policies based upon NNs and MPC. Not only will this help improve the scalability of MPC but it also promises to strengthen the bridges between data-driven and model-based control.

Contributions: The main results of this paper are:

1. An analytic representation for an MPC law for linear systems with a quadratic cost and linear input/state constraints in terms of implicit neural networks.

2. A method to unravel the implicit neural network for MPC into an explicit NN with a bounded approximation error.

3. A procedure to reconstruct the MPC cost from an explicit NN.

The main insight of the paper is that linear quadratic MPC (as in a control policy where the control action is determined by solving a quadratic programming problem such as Definition 1) has an exact representation as an implicit NN (with the implicit NN structure detailed in El Ghaoui et al. (2021)- although also referred to as equilibrium networks Bai et al. (2019); Revay et al. (2023)). This is achieved by building upon the representation in terms of linear complementarity problems detailed in Valmorbida and Hovd (2023). Figure 1 illustrates this connection between MPC and NNs mapped out in this paper. The focus on implicit neural networks explored in this paper follows from the observation that most control policies defined by the solutions of optimisation problems, such as MPC, are themselves implicitly defined functions, and so implicit neural networks should approximate them well.

The presented results highlight the benefits of focusing on *implicit-to-implicit* mappings for linking MPC and NNs, rather than the *explicit-to-explicit* mappings considered previously, e.g. in

Fahandezh-Saadi and Tomizuka (2020); Xu (2021); Karg and Lucia (2020); Ferlez and Shoukry (2020); Åkesson and Toivonen (2006); Kittisupakorn et al. (2009); Hertneck et al. (2018); Soloperto et al. (2020); Kanavalau and Lall (2024). The main result of this paper is similar in spirit to Teichrib and Darup (2021) on generating exact NN representations of MPC, but by focusing on the implicit mapping, it does not impose strict restrictions on the system, such as the dimension of the state-space being one. With our second result, the obtained implicit MPC is "*unravelled*" into an approximating explicit NN, which can be simply evaluated and has the classical feed-forward NN architecture. Together, these results point towards a framework to help unify model based control and data-driven control methods built on top of neural networks. As well as helping to improve the computational scalability of MPC, bringing model-based and data-driven methods closer together also promises to improve the data-driven methods' explainability (by associating the black-box of the NN with a predictive model) and robustness (by connecting their stability analysis for control purposes to the well developed field of robust control).

Notation

Real matrices A of size $m \times n$ are denoted $A \in \mathbb{R}^{n\times m}$. Positive-definite matrices $A \succ 0$ of size n are denoted $A \in \mathbb{S}^n_{\succ 0}$. Vectors p of length n are $p \in \mathbb{R}^n$. The space of diagonal matrices of dimension n are $\mathbb{D}^n_+$. The identity matrix of dimension n is I_n. Symmetric matrices A of dimension n are denoted $A \in \mathbb{S}^n$. The i-th row of a matrix A is denoted $A_{(i,:)}$. The i-th element of a vector $x \in \mathbb{R}^n$ is likewise denoted x_i. The set of natural numbers with zero is denoted $\mathbb{N}_0 = \mathcal{N} \cup \{0\}$. Given a nonlinear function $\phi\colon \mathbb{R}^n \to \mathbb{R}^n$, the notation $\phi(s) \in \mathbb{R}^n$ for $s = [s_1, \ldots, s_n]^\top \in \mathbb{R}^n$ refers to a component-wise application, *i.e.*, $\phi(s) = [\phi_1(s_1), \ldots, \phi_n(s_n)]^\top$.

1. Linear Quadratic MPC

Consider the discrete linear time-invariant state-space model

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

where $x[k] \in \mathbb{R}^n$ and $u[k] \in \mathbb{R}^m$ are the state and control respectively evolving on time instants $k = 0, 1, 2, \ldots$. We use the notation $x[k] = x$ throughout to simplify the analysis. A state $x \in \mathbb{R}^n$ and an $N-$step sequence of control actions $\mathbf{u} = [u[k]^\top, \ldots, u[k+N-1]^\top]^\top$ generate the $N-$step state predictions sequences $\mathbf{x} = [x^\top, x[k+1]^\top, \ldots, x[k+N]^\top]^\top$. Here, the inputs $\mathbf{u}$ are constrained by the state-dependent polytopic set $\mathcal{U}(x) = \{\mathbf{u} : G\mathbf{u} \le S_u x + w\}$ for some $G \in \mathbb{R}^{n_u \times Nm}$, $S_u \in \mathbb{R}^{n_u \times n}$ and $w \in \mathbb{R}^{n_u}$.

1.1. MPC problem

From a given state $x \in \mathbb{R}^n$, the MPC control input sequence $\mathbf{u}(x)$ is computed from the quadratic programme (QP):

$$\min_{\mathbf{u}(x)} \; x[k+N]^\top P x[k+N] + \sum_{i=1}^{N-1} x[k+i]^\top Q_i x[k+i] + u[k+i]^\top R_i u[k+i] \tag{2a}$$

subject to $x[k] = x$,

$$G\mathbf{u} \le S_u x + w, \tag{2b}$$

$$x[k+1] = Ax[k] + \tilde{B}_i \mathbf{u}, \tag{2c}$$

where $\tilde{B}_i = [A^{i-1}B \ \ldots \ AB \ B \ \mathbf{0}_{n\times(N-i-1)m}] \in \mathbb{R}^{n\times mN}$, $Q_i \in \mathbb{R}^{n\times n}$, $P \in \mathbb{R}^{n\times n}$, $R_i \in \mathbb{R}^{m\times m}$, $Q_i \succeq 0$, $R_i \succ 0$, $\forall i = 1, \ldots, N-1$, and $P \succ 0$. Exploiting the linear dynamics of (1) allows the above to be equivalently written as the following QP (referred to here as the *MPC-QP*).

Definition 1 (MPC-QP) *With a positive-definite Hessian $H \in \mathbb{S}^N_{\succ 0}$, the solution of the MPC problem at a state $x \in \mathbb{R}^n$ defined in (2) is given by*

$$\mathbf{u}^*(x) = \arg\min_{\mathbf{u}} \mathbf{u}^\top H\mathbf{u} + 2x^\top F^\top \mathbf{u}, \tag{3a}$$

$$\textit{subject to } G\mathbf{u} \leq S_u x + w. \tag{3b}$$

The solution of *MPC-QP* for a given state $x \in \mathbb{R}^n$ is referred to here as *implicit MPC*, since the computed optimal input sequence is an implicit function of the current value of the state x, *i.e.*, $\mathbf{u}^*(x)$. To obtain MPC control action $u_{\text{mpc}} \in \mathbb{R}^m$ from the vector $\mathbf{u}^*(x)$, we use

$$u_{\text{mpc}} = [1, 0, \ldots, 0]\mathbf{u}^*(x). \tag{4}$$

Assumption 1 It is assumed that a unique solutions $\mathbf{u}^*(x)$ exists for the MPC-QP problem of (3) for all $x \in \mathbb{R}^n$. This feasibility assumption will translate into an assumption on the well-posedness of the equivalent implicit neural network derived in this paper.

2. Implicit representation of MPC

As discussed in the introduction, there has been a concentrated effort to link the disparate control schemes of MPC and NN controllers, so as to accelerate, scale and ease the implementation of MPC. As far as the authors are aware, existing studies relating NNs and MPC, such as Fahandezh-Saadi and Tomizuka (2020); Xu (2021); Karg and Lucia (2020); Ferlez and Shoukry (2020); Åkesson and Toivonen (2006); Kittisupakorn et al. (2009); Hertneck et al. (2018); Soloperto et al. (2020), are motivated by the fact that the explicit MPC policy is a piece-wise affine function, so should be well-approximated by an *explicit NN*, in particular one that is also a piece-wise affine function with *ReLU* activation functions.

Definition 2 *An explicit NN $f\colon \mathbb{R}^n \to \mathbb{R}^{Nm}$ with $x \mapsto \mathbf{u}^*(x)$ is defined by the recurrence*

$$z[j+1] = \phi\left(\sum_{i=0}^{j} W^{j,i} z[i] + Y^j x + b^j\right), \tag{5a}$$

$$\mathbf{u}^*(x) = W_f z[J] + Y_f x + b_f, \tag{5b}$$

for some activation function $\phi(\cdot)$, hidden layer depth $j = 0, \ldots, J$, $W^{j,i} \in \mathbb{R}^{n_j\times n_i}$ (where n_i and n_j are the dimensions of layers i and j, respectively), $Y^j \in \mathbb{R}^{n_j\times n}$, $b^j \in \mathbb{R}^{n_j}$, $W_f \in \mathbb{R}^{Nm\times n_J}$, $Y_f \in \mathbb{R}^{Nm\times n}$ and $b_f \in \mathbb{R}^{Nm}$.

Several methods have been proposed to train explicit neural networks that approximate MPC, e.g. in Zhang et al. (2020); Chen et al. (2022); Karg and Lucia (2021). But, in general, these methods all require the standard training step to obtain the NN from data generated by sampling the MPC solution. Training in this way can introduce errors which are challenging to bound Drummond et al. (2022); Fabiani and Goulart (2022); Schwan et al. (2023).

By contrast, the focus of this paper is on *equating* MPC policies to the *implicit* neural networks of El Ghaoui et al. (2021). The following definition of an implicit neural network is used here.

Definition 3 *An implicit NN $f\colon \mathbb{R}^n \to \mathbb{R}^{Nm}$ with $x \mapsto \mathbf{u}^*(x)$ is defined by the solution of the following algebraic system of equations*

$$y(x) = W\phi(y(x)) + Yx + b, \tag{6a}$$
$$\mathbf{u}^*(x) = W_f\phi(y(x)) + Y_f x, \tag{6b}$$

for some activation function $\phi(\cdot)$, weights $W \in \mathbb{R}^{M\times M}$, $Y \in \mathbb{R}^{M\times n}$, $W_f \in \mathbb{R}^{Nm\times M}$, $Y_f \in \mathbb{R}^{Nm\times n}$ and biases $b \in \mathbb{R}^M$, $b_f \in \mathbb{R}^{Nm}$.

The *implicit* term comes from the fact that (6a) is an implicit equation in general (in certain cases, such as when the weight matrix $W \in \mathbb{R}^{M\times M}$ has a strict upper- or lower-triangular structure, then $\mathbf{u}^*(x)$ again becomes an explicit function of x, as discussed in El Ghaoui et al. (2021)). Compared to explicit NNs, such as the classical feed-forward and recurrent architectures, evaluating an implicit NN may involve solving a system of nonlinear equations, since the weight matrices $W \in \mathbb{R}^{M\times M}$ may be dense. Implicit equations can be difficult to solve and conditions for the well-posedness of the implicit equation must be verified. Sufficient conditions to establish well-posedness of the implicit neural network of (6) can be derived by adapting the method from El Ghaoui et al. (2021) involving the weight matrix W.

Remark 4 *A key aspect of this paper is the observation that, in general, the solution of an optimal control problem is the output of an implicitly defined function. Only in particular cases, such as the robust control problems of Pates et al. (2019), are explicit solutions to control problems known. For the other cases, numerical algorithms are required to extract the solutions. The implicit nature of solving optimal control problems suggests that implicit neural networks should provide a more natural framework to express the solutions than the more commonly used explicit neural networks.*

The main result of this paper is to show that MPC problems defined by the solution of the *MPC-QP* admit an exact interpretation in terms of an *implicit neural network* of the form of Definition 3. The rest of the paper is concerned with showing this. The main idea is to build on top of the recent result from Valmorbida and Hovd (2023) where it was shown that the *MPC-QP* can be expressed as a linear complimentarity problem using *ReLU* functions.

Definition 5 *A function $r(s) : \mathbb{R}^{n_s} \to \mathbb{R}^{n_s}$ is a* ReLU *function if it satisfies*

$$r(s) = \begin{cases} 0 & \text{if } s < 0, \\ s & \text{if } s \geq 0. \end{cases} \tag{7}$$

This function is also known as the ramp function.

Theorem 6 *Valmorbida and Hovd (2023) The solution of the MPC-QP (3) at a state $x \in \mathbb{R}^n$ can be expressed as the solution of the piece-wise affine system of equations*

$$\mathbf{u}^*(x) = -H^{-1}F^\top x - H^{-1}G^\top r(y(x)), \tag{8a}$$

where $y(x) : \mathbb{R}^n \to \mathbb{R}^{n_c}$ is the solution to the following implicit equation

$$y(x) - (I - GH^{-1}G^\top)r(y(x)) = -(Sx + w), \tag{8b}$$

with $S = S_u + GH^{-1}F$.

In the above theorem, the state x and the optimal MPC control input $\mathbf{u}^*(x)$ are connected through the implicitly defined piece-wise affine function $y(x)$ which solves (8b).

Remark 7 *The expression* (8b) *is obtained from the MPC-QP Karush-Kuhn-Tucker optimality conditions*

$$H\mathbf{z} + G^\top \lambda = 0, \tag{9a}$$

$$\lambda_i(S_{(i,\cdot)}x + w_i - G_{(i,\cdot)}\mathbf{z}) = 0, \tag{9b}$$

$$Sx + w - G\mathbf{z} \geq 0, \tag{9c}$$

which, thanks to the invertibility of H, gives the following Linear Complementarity Problem

$$0 \leq \lambda \perp (Sx + w + GH^{-1}G^\top \lambda) \geq 0. \tag{10}$$

Relating the Lagrange multipliers λ of the MPC-QP to the ramp function of some functions of x, namely $\lambda = r(y(x))$, then the complementarity conditions stated above can be connected to the complementarity conditions defining the ramp function, giving (8b)*. See Valmorbida and Hovd (2023) for details.*

Equation (8b) allows the MPC-QP feedback law to be characterised as an implicit function. An iterative method to solve (8b) is proposed in Valmorbida and Hovd (2023), providing an alternative to using interior-point methods to solve the point-location problem of explicit MPC Bemporad et al. (2002); Bayat et al. (2011). In the rest of the paper, we exploit the ReLU function in (8b) to propose an approximate solution to the implicit equation (8b).

By equating terms between Equation (6) of Definition 3 and Equation (8) of Theorem 6, it follows that the MPC control action defined by the solution of the *MPC-QP* (Definition 1) can be uniquely represented as an implicit neural network of the form of Definition 3 with

$$W = (I_m - GH^{-1}G^\top),\ Y = -S,\ b = -w, \tag{11a}$$

$$W_f = -H^{-1}G^\top,\ Y_f = -H^{-1}F^\top,\ b_f = 0, \tag{11b}$$

and with a *ReLU* activation function $\phi(\cdot) = r(\cdot)$. In fact, from this point on, the activation functions considered in the paper will be *ReLU*s. The matrices in (11) show that to link MPC and NNs, an exact "*implicit-to-implicit*" mapping between MPC and NNs can be obtained, unlike the "*explicit-to-explicit*" mappings usually considered which either involve approximation errors, *e.g.* in Fahandezh-Saadi and Tomizuka (2020); Karg and Lucia (2020), or are restrictive, e.g. for single state systems Teichrib and Darup (2021). This observation connecting the implicit neural network weights (11) to the *MPC-QP* is the main result of this paper.

3. Explicit approximation of the implicit neural network

While the implicit NN of Definition 3 parameterised by (11) exactly captures the solution of the *MPC-QP*, it can be argued that implicit NNs are currently not yet widely used in practice. This is primarily because evaluating them requires solving an implicit system of equations which can be challenging. Another issue with implicit neural networks is that most of the common NN architectures used for control, such as recurrent and feed-forward neural networks, are explicit and have many well-established tools for training, evaluating and pruning them, unlike for implicit NNs. These limitations motivate the following results on "*unravelling*" the implicit NN of Equation (11) into an approximating explicit NN.

The first step towards unravelling the implicit NN of Definition 3 with (11) is to write (8b) as

$$y(x) = D\phi(y(x)) + \zeta \tag{12}$$

with $D \in \mathbb{S}^m = (I_m - GH^{-1}G^\top)$ and $\zeta \in \mathbb{R}^m = -(Sx + w)$. The interpretation of the implicit NN as the equilibrium of a forced dynamical system, El Ghaoui et al. (2021); Revay et al. (2023), is adopted to generate an approximating explicit NN. With this interpretation, the approximating explicit NN is structured as

$$w[j+1] = D\phi(w[j]) + \zeta + f(w[j]), \tag{13}$$

with the hidden layer index $j = 0,\, 1,\, \ldots,\, J$. In the above, the function $f(\cdot)$ is a feedback control policy chosen by the user to accelerate the convergence of $w[j]$ towards $y(x)$. The benefits of accelerated convergence are that shallower neural networks can be used for a given level of solution accuracy.

3.1. Error dynamics

The problem is then to design the control action $f(\cdot)$ such that the dynamics of the approximating explicit NN (13) converge to the output of the implicit one (12). For this, the solution of the implicit NN (12) is subtracted from the iterates of (13) to give the error $e[j] = w[j] - y(x)$ evolving through the layers j by $e[j+1] = D(\phi(w[j]) - \phi(y(x)) + f(w[j])$. In this paper, the controller is structured as $f(w[j]) = K(w[j] - D\phi(w[j]) - \zeta)$ for some gain $K \in \mathbb{R}^{M\times M}$. If the error dynamics are convergent with this controller, then the implicit neural network can be understood as the output of the infinitely long explicit neural network. Truncating this infinitely long explicit neural network after a certain number of time steps leads to a feed-forward neural network with the explicit structure of Definition 2 and a bounded approximation error.

To show this, define

$$\tilde{\phi}(e[j]) = \phi(w[j]) - \phi(y(x)), \tag{14}$$

such that the iterates evolve according to

$$w[j+1] = D\phi(w[j]) + \zeta + K(w[j] - D\phi(w[j]) - \zeta), \tag{15}$$

and the error dynamics satisfy

$$\begin{aligned} e[j+1] =& D(\phi(w[j]) - \phi(y(x)) + K(w[j] - D\phi(w[j]) - \zeta), \\ =& Ke[j] + (I_m - K)D\tilde{\phi}(e[j]). \end{aligned} \tag{16}$$

This is a nonlinear system but notice that $\tilde{\phi}(e[j])$ is a slope-restricted nonlinearity.

Lemma 8 *The function $\tilde{\phi}(e[j]) : \mathbb{R}^{n_s} \to \mathbb{R}^{n_s}$ defined in (14) satisfies $\tilde{\phi}(e[j])^\top T(e[j] - \tilde{\phi}(e[j])) \geq 0$ for all $T \in \mathbb{D}^m_+$.*

Proof First, note that the *ReLU* function satisfies the equality

$$\phi(s)^\top T(s-\phi(s)) = 0, \quad \forall s \in \mathbb{R}^m, T \in \mathbb{D}_+^m. \tag{17}$$

Then

$$\begin{aligned}\tilde{\phi}(e[j])^\top T(e[j]-\tilde{\phi}(e[j])) &= -(\phi(w[j]))^\top T(y(x)-\phi(y(x))), \qquad \text{(18a)}\\ &= -\phi(y(x)))^\top T(w[j]-\phi(w[j])) \geq 0,\end{aligned}$$

since $T \in \mathbb{D}_+^m$, the *ReLU* function satisfies $\phi(s) \geq 0$, $s - \phi(s) \leq 0$ for all $s \in \mathbb{R}$ and using (17). ■

The error dynamics of (16) can then be understood as a discrete-time Lurie system. As such, for a given gain K, the linear matrix inequalities from papers such as Haddad and Bernstein (1993); Carrasco et al. (2019); Drummond and Valmorbida (2023) could then be used to certify asymptotic convergence and the errors bounded. If these LMIs are satisfied, then the feed-forward neural network of (15) is guaranteed to converge towards the solution of the *MPC-QP* of Definition 1. Page limits of this paper prevent an in-depth analysis of the computation of the gains K which is left to future work.

Remark 9 *Several results, such as Karg and Lucia (2020); Cao and Gopaluni (2020), have emphasized the importance of explicit neural network depth, as in the number of hidden layers, to gain a good approximation of an MPC control law. The analysis discussed in this section reinforces this notion, by showing how an infinitely deep feed-forward neural network, of the form of (15), can exactly represent the solution of the MPC-QP.*

4. Converse results: From neural networks to MPC

In this section, a converse result to the statement of Section 2 is introduced. Specifically, the following problem is considered: Given an implicit neural network $f \colon \mathbb{R}^n \to \mathbb{R}^{Nm}$ of the form of Definition 3, can the cost defining the MPC control law be recovered? The answer to this question is affirmative. The argument is the following: the piece-wise affine functions are dense in the set of continuous functions, therefore there exists a piece-wise linear approximation $\xi \colon \mathbb{R}^n \to \mathbb{R}^{Nm}$ to $f(\cdot)$ such that $\|f - \xi\| < \epsilon$ in a suitable norm. The function $\xi(\cdot)$ generates a partition of the state space $\tilde{P}_i$ for $i \in \tilde{\mathcal{I}}$ and a collection of affine control laws $\{\tilde{E}_i, \tilde{\omega}_i\}_{i \in \tilde{P}}$. To recover the cost matrices, an observation regarding the partition of the state space is made.

Lemma 10 *There exists a non-empty proper subset $\emptyset \neq \mathcal{J} \subset \mathcal{I}$ such that the associated control laws satisfy $j \in \mathcal{J}$, $\kappa_j(x) = \omega_j$ for each $x \in \tilde{P}_j$.*

The importance of the above lemma is that focus only has to be given to the interior of the feasible region, with the outer parts representing a "saturation" of the control input. In the interior of $\mathcal{X}$, there exists a subset of indices $\mathcal{K} = \mathcal{I} \setminus \mathcal{J}$ such that $\kappa_i(x) = K_i x + \beta_i$ for all $i \in \mathcal{K}$. The closed loop system in each of the regions P_i is characterised by $\Phi_i = (A + BE_i)$ which satisfy the following LMI:

$$\Phi_i^\top P_i \Phi_i - P_i \succ -(Q + E_i^\top R E_i), \; \forall i \in \mathcal{K}, \tag{19}$$

with $P_i, Q, R \prec 0$ for all $i \in \mathcal{K}$. The solution of this system of LMIs allow us to build common cost matrices (Q, R) that are part of the MPC formulation.

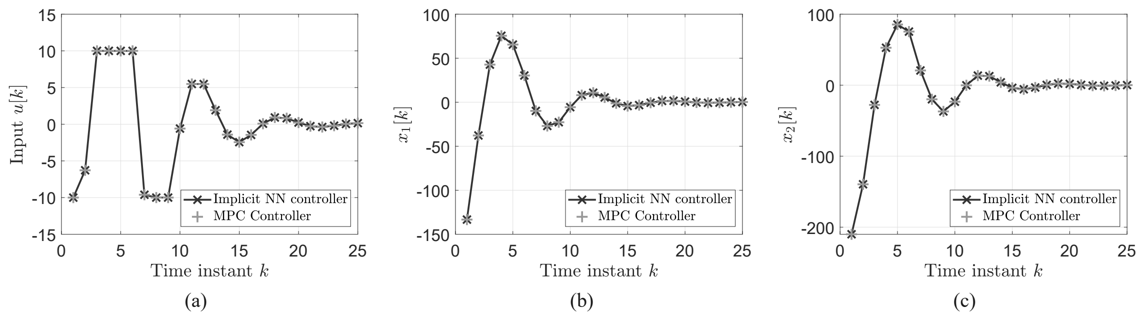

Figure 2: Comparison between the MPC controller and the implicit neural network defined by (11) for the numerical example. The input sequences $u[k]$, and states $x_1[k]$, $x_2[k]$ are shown for both policies. The equivalence between the two simulations validates the results of the paper. Note that the implicit neural network captures the input constraints imposed by the MPC policy, as predicted.

5. Numerical Example

Consider the linear time-invariant system from Drummond et al. (2022)

$$\begin{bmatrix} x_1[k+1] \\ x_2[k+1] \end{bmatrix} = \begin{bmatrix} 4/3 & -2/3 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x_1[k] \\ x_2[k] \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u[k], \tag{20}$$

to be controlled using an MPC defined by the *MPC-QP* of Definition 1. Set the horizon length $N = 10$ and the control action to be saturated at $-10 \leq \mathbf{u} \leq 10$, so $G = I_m \otimes \begin{bmatrix} 0.1 & -0.1 \end{bmatrix}^\top$, $w = [1,\, 1,\, \dots\, 1,\, 1]^\top$ and $S_u = 0$. The MPC quadratic cost function of (2) is parameterised by

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

The goal of the numerical example is to verify the claim that this MPC problem can be represented as the implicit neural network of Definition 3 defined by (11) which can itself be approximated by the explicit one of (15)[1].

The first step is to show that the explicit neural network of (15) does indeed converge towards the unique solution of the implicit neural network of (12). From the initial condition $w[0] = [1, 1]^\top$, the curve in Figure 3(a) shows the convergence rates of the various methods defined by the "control gains" K. Three different gains are considered, $K = -0.9I_m$, $K = 0$ and $K = 0.2I_m$. The figure shows the convergence of the residuals $w[j] - D\phi(w[j]) - \zeta$ for each gain, indicating that, for a sufficient layer depth J, the explicit neural networks did converge to the unique solutions of the implicit neural network. Fastest convergence was achieved with $K = -0.9I_m$. LMI-based conditions to optimise the gains K will be developed in future work.

With the explicit neural networks of (15) shown to give a good approximation of the implicit neural networks of (12) (as long as they are sufficiently deep), the next step is to show that this implicit neural network (parameterised according to (11)) represents the MPC control action defined by the *MPC-QP* of Definition 1. This is illustrated in the simulation of Figure 2 as well as in Figures 3(b) and 3(c) which also show the equivalence, with the two control actions compared for

1. Code to generate these results can be found here: `https://github.com/r-drummond/MPC_vs_NN`.

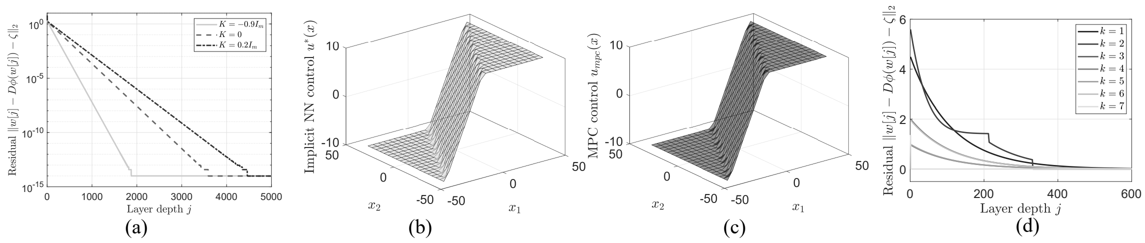

Figure 3: (a) Residuals from unravelling the implicit neural network into an explicit one using the gains $K = -0.9I_m$, $K = -0.9I_m$ & $K = 0.2I_m$. (b) The control action from the implicit neural network of the numerical example as a function of the state-space. (b) The control action from the MPC of the numerical example as a function of the state-space. (d) Residuals of the implicit neural network during the simulation as k evolves.

various different state values. For this simulation, the initial condition was $x = [-200, \, -200]^\top$, and controllers defined by the MPC of Definition 1 and the explicit neural network of (15) were run. Using the depth $J = 1000$, Figure 2, 3(b) and 3(c) shows the equivalence between the inputs $u[k]$ generated by both the MPC and NN controllers (with both saturating at certain instants k), which then also generated equivalent state dynamics for both $x_1[k]$ and $x_2[k]$. The equivalence between these two closed-loop trajectories reinforces the conclusion of Equation (11) that the *MPC-QP* can be represented as the implicit neural network of (12), and that the analytic expression of Equation (11) for the weights and biased hold.

Finally, Figure 3(d) shows the convergence of the approximating explicit neural network's residuals during the simulation (as in, as k increased). In the simulation, the explicit neural network was warm started, meaning that the initial condition $w[0]$ was based upon the final value $w[J]$ obtained from the previous time instant k. Figure 3(d) shows the accelerated convergence of these residuals as the simulation progressed with k increasing, and these residuals were found to be zero for $k > 7$. This figure shows the accelerated convergence rates of the explicit neural network as the state dynamics converge towards their equilibrium.

Conclusions

Linear quadratic model predictive control (MPC) was shown to admit a representation as an implicit neural network. Analytic expressions for the weights and biases of this implicit neural network were given and a method to "*unravel*" the implicit neural network into an explicit one was also described. Expressions for the weights and biases of the approximating explicit neural network were stated and it was shown that the depth of the explicit network defined the approximation error with respect to the implicit neural network. A numerical example illustrated the application of the results in practice. A procedure to recover the MPC cost matrices from piece-wise linear approximations of explicit neural networks was also developed. Together, these results give a constructive way to represent linear quadratic model predictive control using neural networks, providing an explicit link between model-based and data-driven control.

Acknowledgments

Ross Drummond was supported by a UK Intelligence Community Research Fellowship from the Royal Academy of Engineering.

References

Kwangjun Ahn, Zakaria Mhammedi, Horia Mania, Zhang-Wei Hong, and Ali Jadbabaie. Model predictive control via on-policy imitation learning. *arXiv preprint arXiv:2210.09206*, 2022.

Bernt M Åkesson and Hannu T Toivonen. A neural network model predictive controller. *Journal of Process Control*, 16(9):937–946, 2006.

Shaojie Bai, J Zico Kolter, and Vladlen Koltun. Deep equilibrium models. *Advances in Neural Information Processing Systems*, 32, 2019.

Farhad Bayat, Tor Arne Johansen, and Ali Akbar Jalali. Using hash tables to manage the time-storage complexity in a point location problem: Application to explicit model predictive control. *Automatica*, 47:571–577, 3 2011. ISSN 0005-1098.

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

Yankai Cao and R Bhushan Gopaluni. Deep neural network approximation of nonlinear model predictive control. *IFAC-PapersOnLine*, 53(2):11319–11324, 2020.

Joaquin Carrasco, William P Heath, Jingfan Zhang, Nur Syazreen Ahmad, and Shuai Wang. Convex searches for discrete-time Zames–Falb multipliers. *IEEE Transactions on Automatic Control*, 65 (11):4538–4553, 2019.

Steven W Chen, Tianyu Wang, Nikolay Atanasov, Vijay Kumar, and Manfred Morari. Large scale model predictive control with neural networks and primal active sets. *Automatica*, 135:109947, 2022.

Ross Drummond and Giorgio Valmorbida. Generalised Lyapunov functions for discrete-time Lurie systems with slope-restricted nonlinearities. *IEEE Transactions on Automatic Control*, 2023.

Ross Drummond, Stephen Duncan, Mathew Turner, Patricia Pauli, and Frank Allgower. Bounding the difference between model predictive control and neural networks. In *Learning for Dynamics and Control Conference*, pages 817–829. PMLR, 2022.

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. *IEEE Transactions on Automatic Control*, 2022.

Saman Fahandezh-Saadi and Masayoshi Tomizuka. In proximity of ReLU DNN, PWA function, and explicit MPC. *arXiv preprint arXiv:2006.05001*, 2020.

James Ferlez and Yasser Shoukry. AReN: Assured ReLU NN architecture for model predictive control of LTI systems. In *Procs. of the International Conference on Hybrid Systems: Computation and Control*, pages 1–11, 2020.

Wassim M Haddad and Dennis S Bernstein. Explicit construction of quadratic Lyapunov functions for the small gain, positivity, circle, and Popov theorems and their application to robust stability. Part I: Continuous-time theory. *International Journal of Robust and Nonlinear Control*, 3(4): 313–339, 1993.

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.

Andrei Kanavalau and Sanjay Lall. Safe explicit MPC by training neural networks through constrained optimization. In *Procs. of the UKACC International Conference on Control (CONTROL)*. IEEE, 2024.

Benjamin Karg and Sergio Lucia. Efficient representation and approximation of model predictive control laws via deep learning. *IEEE Transactions on Cybernetics*, 50(9):3866–3878, 2020.

Benjamin Karg and Sergio Lucia. Approximate moving horizon estimation and robust nonlinear model predictive control via deep learning. *Computers & Chemical Engineering*, 148:107266, 2021.

Paisan Kittisupakorn, Piyanuch Thitiyasook, Mohd Azlan Hussain, and Wachira Daosud. Neural network based model predictive control for a steel pickling process. *Journal of process control*, 19(4):579–590, 2009.

Richard Pates, Carolina Bergeling, and Anders Rantzer. On the optimal control of relaxation systems. In *Procs. of the Conference on Decision and Control (CDC)*, pages 6068–6073. IEEE, 2019.

Max Revay, Ruigang Wang, and Ian R Manchester. Recurrent equilibrium networks: Flexible dynamic models with guaranteed stability and robustness. *IEEE Transactions on Automatic Control*, 2023.

Roland Schwan, Colin N Jones, and Daniel Kuhn. Stability verification of neural network controllers using mixed-integer programming. *IEEE Transactions on Automatic Control*, 2023.

Raffaele Soloperto, Johannes Köhler, and Frank Allgöwer. Augmenting mpc schemes with active learning: Intuitive tuning and guaranteed performance. *IEEE Control Systems Letters*, 4(3):713–718, 2020.

Dieter Teichrib and Moritz Schulze Darup. Tailored neural networks for learning optimal value functions in MPC. In *Procs. of the Conference on Decision and Control (CDC)*, pages 5281–5287. IEEE, 2021.

Dieter Teichrib and Moritz Schulze Darup. Tailored max-out networks for learning convex PWQ functions. *arXiv preprint arXiv:2206.06826*, 2022.

Giorgio Valmorbida and Morten Hovd. Quadratic programming with ramp functions and fast online QP-MPC solutions. *Automatica*, 153:111011, 2023.

Xuesu Xiao, Tingnan Zhang, Krzysztof Choromanski, Edward Lee, Anthony Francis, Jake Varley, Stephen Tu, Sumeet Singh, Peng Xu, Fei Xia, et al. Learning model predictive controllers with real-time attention for real-world navigation. *arXiv preprint arXiv:2209.10780*, 2022.

Jun Xu. Lattice piecewise affine approximation of explicit linear model predictive control. In *Procs. of the Conference on Decision and Control (CDC)*, pages 2545–2550. IEEE, 2021.

Xiaojing Zhang, Monimoy Bujarbaruah, and Francesco Borrelli. Near-optimal rapid MPC using neural networks: A primal-dual policy learning framework. *IEEE Transactions on Control Systems Technology*, 29(5):2102–2114, 2020.