2023 62nd IEEE Conference on Decision and Control (CDC)
December 13-15, 2023. Marina Bay Sands, Singapore

Nonlinear Data-Driven Predictive Control using Deep Subspace Prediction Networks

Mircea Lazar and Mihai-Serban Popescu and Maarten Schoukens

Abstract— **Indirect data-driven predictive control (DPC) algorithms for nonlinear systems typically employ multi-step predictors, which are identified from input-output data using neural networks. In this paper we put forward a unifying multi-step prediction network architecture, i.e., the deep subspace prediction network (DSPN). We then prove that the DSPN architecture specialized to multi-layer-perceptron neural networks recovers the linear predictor corresponding to subspace predictive control for a sufficient number of hidden layer neurons. Hence, we establish a well-posed generalization of subspace predictive control for nonlinear systems. Moreover, we develop a regularized DSPN architecture that embeds a linear subspace predictor to improve extrapolation properties for non-training data. Simulation results on a benchmark inverted pendulum show that nonlinear DPC based on DSPN achieves high control performance for both noiseless and noisy data.**

I. INTRODUCTION

Model predictive control (MPC) has become the preferred advanced control method in many emerging applications [1] due to constraints handling, anticipating control actions and optimal performance. Since reliable data becomes increasingly available in smart engineering systems, it is of interest to develop data-driven predictive control (DPC) algorithms. An indirect data-driven approach to predictive control for linear systems was already developed in [2], i.e., subspace predictive control (SPC). Therein, instead of identifying a system model, a multiple-input-multiple-output (MIMO) multi-step linear subspace predictor was directly identified using least squares.

Indirect approaches to data-driven predictive control for nonlinear systems have also been developed as follows. Initially, nonlinear autoregressive with exogeneous input (NARX) models based on neural networks were used to identify the system dynamics. Then, the resulting NARX models were used recurrently to predict future outputs over the prediction horizon, see, e.g., [3], [4], [5], [6], [7] and the references therein. However, the one-step ahead NARX predictors are not well suited for recurrent usage in prediction over long horizons, as the prediction error is propagated [5]. As such, multi-model multi-step predictors were developed in [8], [9], [5], which use different NARX models for predicting the output at each future time instant up to the prediction horizon. These papers use multi-layer-perceptron (MLP) neural networks, while more recently, [10] developed multi-model multi-step predictors using reproducing kernel functions. In [11], neural networks based multi-step predictors with a specific, input affine structure, were developed for nonlinear DPC. Therein, each predictor for a specific future output was allowed to depend on the output of the previous predictor, which yields a deep MIMO multi-step predictor that can be trained sequentially or jointly. Such a deep multi-step predictor combines measured output data with simulated/predictive output data, leading to a hybrid model, i.e., combining NARX with nonlinear output error (NOE) modelling principles.

Multi-step ahead predictors have also been developed for nonlinear system identification, i.e., not necessarily linked with predictive control. For brevity we refer only to [12], which uses reproducing kernel functions, and the comprehensive survey [13]. In [12], multi-model multi-step predictors [5] are referred to as multiple shallow prediction networks. Therein, two other multi-step predictors are introduced: iterated shallow prediction networks (ISPN), which optimize the multi-step prediction error of a one-step predictor, and deep prediction networks (DPN), which allow for a different predictor at each time step. Since in [12] it was already shown that DPN outperforms ISPN in terms of prediction accuracy, in this work we will consider only DPN for usage within nonlinear DPC.

Despite the many available formulations, a general categorisation of prediction networks (i.e., independent of the chosen representation) for nonlinear data-driven predictive control and an explicit link with the linear SPC predictor [2] are not yet available, to the best of our knowledge. These would be instrumental to better understand the relation between existing prediction networks, which often share similar features, despite using different representations, and to arrive at a well-posed formulation of nonlinear DPC.

Therefore, in this paper we introduce a unifying prediction network architecture, i.e., the deep subspace prediction network (DSPN). We prove that the DSPN architecture specialized to MLP neural networks recovers the linear predictor of SPC for a sufficient number of hidden layer neurons. Hence, we establish a well-posed generalization of SPC for nonlinear systems. Moreover, we develop a regularized DSPN predictor that embeds a linear SPC predictor to improve extrapolation properties for non-training data. Training methods for DSPN predictors are also briefly discussed. Simulation results on a benchmark inverted pendulum show that both DSPN and regularized DSPN predictors based on MLP neural networks achieve high tracking control performance in the presence of noisy data, while outperforming multi-step NARX [5], [10] and DPN [12] architectures.

The authors are with the Department of Electrical Engineering, Eindhoven University of Technology, The Netherlands, E-mails: m.lazar@tue.nl, m.popescu@student.tue.nl, m.schoukens@tue.nl.

II. Preliminaries

This section provides the basis for indirect nonlinear DPC and reviews the state-of-the-art in design of multi-step predictors. For any finite number $q \in \mathbb{N}_{\geq 1}$ of vectors $\{\xi_1, \ldots, \xi_q\} \in \mathbb{R}^{n_1} \times \ldots \times \mathbb{R}^{n_q}$ we will make use of the operator $\operatorname{col}(\xi_1, \ldots, \xi_q) := [\xi_1^\top, \ldots, \xi_q^\top]^\top$.

We consider MIMO nonlinear systems with inputs $u \in \mathbb{R}^m$ and measured noisy outputs $y \in \mathbb{R}^p$ that can be modeled by a discrete-time NARX model, i.e.,

$$\hat{y}(k+1) = f(\mathbf{u}_{\text{ini}}(k), \mathbf{y}_{\text{ini}}(k), u(k)) \tag{1}$$

where $k \in \mathbb{N}$, $f : \mathbb{R}^{(n_b-1)m} \times \mathbb{R}^{n_a p} \times \mathbb{R}^m \to \mathbb{R}^p$,

$$\begin{aligned}\mathbf{u}_{\text{ini}}(k) &:= \operatorname{col}(u(k-n_b+1), \ldots, u(k-1)) \in \mathbb{R}^{(n_b-1)m},\\ \mathbf{y}_{\text{ini}}(k) &:= \operatorname{col}(y(k-n_a+1), \ldots, y(k)) \in \mathbb{R}^{n_a p},\end{aligned}$$

and where $n_a, n_b \in \mathbb{N}$ define the order of the dynamics. *In what follows we will use $y(i|k)$ to denote future outputs predicted by model* (1) *based on measurements at time k, as typically done in MPC, i.e., $\hat{y}(k+1) = y(1|k)$.*

Next, we define a generic nonlinear predictive control problem for NARX models. Let N denote the prediction horizon and, for any positive definite matrix L, let $L^{\frac{1}{2}}$ denote its Cholesky factorization. For positive definite matrices Q, R of compatible dimensions, define the cost function:

$$\begin{aligned}J(\mathbf{u}_{\text{ini}}(k), \mathbf{y}_{\text{ini}}(k), \mathbf{u}(k)) := \|Q^{\frac{1}{2}} y(N|k)\|_2^2 +\\ \sum_{i=0}^{N-1} \|Q^{\frac{1}{2}} y(i|k)\|_2^2 + \|R^{\frac{1}{2}} u(i|k)\|_2^2.\end{aligned} \tag{2}$$

Above, $\mathbf{u}(k) := \operatorname{col}(u(0|k), \ldots, u(N-1|k))$ is the sequence of predicted inputs at time k.

Problem II.1 At time k, given the measured input-output data $\mathbf{u}_{\text{ini}}(k), \mathbf{y}_{\text{ini}}(k)$ solve the optimization problem:

$$\min_{\mathbf{u}(k)} J(\mathbf{u}_{\text{ini}}(k), \mathbf{y}_{\text{ini}}(k), \mathbf{u}(k)) \tag{3a}$$

subject to

$$y(i+1|k) = f(\mathbf{u}_{\text{ini}}(i|k), \mathbf{y}_{\text{ini}}(i|k), u(i|k)),\quad i = 0, \ldots, N-1, \tag{3b}$$

$$(u(i|k), y(i+1|k)) \in \mathbb{U} \times \mathbb{Y},\ i = 0, \ldots, N-1, \tag{3c}$$

where $\mathbf{u}_{\text{ini}}(0|k) := \mathbf{u}_{\text{ini}}(k)$, $\mathbf{y}_{\text{ini}}(0|k) := \mathbf{y}_{\text{ini}}(k)$ and assuming $N > \max(n_b, n_a) + 1$, the initial input-output data for each remaining predictor are constructed as:

$$\begin{aligned}\mathbf{u}_{\text{ini}}(1|k) &= \operatorname{col}(u(k-n_b+1), \ldots, u(k-1), u(0|k)),\\ \mathbf{u}_{\text{ini}}(2|k) &= \operatorname{col}(u(k-n_b+2), \ldots, u(0|k), u(1|k)),\\ &\vdots\\ \mathbf{u}_{\text{ini}}(N-1|k) &= \operatorname{col}(u(N-n_b|k), \ldots, u(N-2|k)),\end{aligned}$$

and similarly for $\mathbf{y}_{\text{ini}}(i|k)$, by replacing n_b with n_a:

$$\begin{aligned}\mathbf{y}_{\text{ini}}(1|k) &= \operatorname{col}(y(k-n_a+2), \ldots, y(0|k), y(1|k)),\\ \mathbf{y}_{\text{ini}}(2|k) &= \operatorname{col}(y(k-n_a+3), \ldots, y(1|k), y(2|k)),\\ &\vdots\\ \mathbf{y}_{\text{ini}}(N-1|k) &= \operatorname{col}(y(N-n_a|k), \ldots, y(N-1|k)).\end{aligned}$$

This approach yields a NARX model based nonlinear predictive controller. Typically, neural networks are used to identify the NARX model f, see, e.g., [4], [5] and the references therein. As mentioned in the Introduction, recurrent usage of NARX one-step ahead predictors propagates prediction errors, which is why multi-step predictors are preferred.

A. Multi-step predictors overview

Multi-model multi-step predictors based on neural networks were developed in [8], [9], [5], where for each predicted output $y(i|k)$, a different predictor f_i is used, i.e.,

$$y(i+1|k) = f_{i+1}(\mathbf{u}_{\text{ini}}(0|k), \mathbf{y}_{\text{ini}}(0|k), \mathbf{u}(i|k)), \tag{4}$$

where $i = 0, \ldots, N-1$ and for any such i,

$$\mathbf{u}(i|k) := \operatorname{col}(u(0|k), \ldots, u(i|k)).$$

Notice that f_1 recovers the predictor f in (3b), but the predictors $f_2, f_3, \ldots$ can be different. Since all predictors depend on the measured system output and there is no coupling, they are NARX models. However, since they are not used recurrently, there is no propagation of prediction errors. A similar NARX multi-model multi-step predictor structure based on reproducing kernel functions was recently used in [10] where additionally, each predictor f_i is allowed to depend on the full input sequence $\mathbf{u}(k)$, i.e.

$$y(i+1|k) = f_{i+1}(\mathbf{u}_{\text{ini}}(0|k), \mathbf{y}_{\text{ini}}(0|k), \mathbf{u}(k)). \tag{5}$$

In [11], the predictors f_i where parameterized using an input affine structure and coupling with the output of the previous predictor was introduced, i.e.,

$$\begin{aligned}y(i+1|k) =& f_{i+1}^1(\mathbf{u}_{\text{ini}}(0|k), \mathbf{y}_{\text{ini}}(0|k)) +\\ & f_{i+1}^2(\mathbf{u}_{\text{ini}}(0|k), \mathbf{y}_{\text{ini}}(0|k))\mathbf{u}(i|k) + c_i y(i|k),\end{aligned} \tag{6}$$

where $i = 0, \ldots, N-1$, $c_0 = 0$, $c_i = 1$ for any other i and each f_i^1, f_i^2 mapping is modeled using a different MLP neural network.

In [12] ISPN and DPN architectures specialized to reproducing kernel functions are used. The ISPN predictor is

$$y(i+1|k) = f(\mathbf{u}_{\text{ini}}(i|k), \mathbf{y}_{\text{ini}}(i|k), u(i|k)), \tag{7}$$

and the DPN predictor is given by

$$y(i+1|k) = f_{i+1}(\mathbf{u}_{\text{ini}}(i|k), \mathbf{y}_{\text{ini}}(i|k), u(i|k)), \tag{8}$$

where $i = 0, \ldots, N-1$.

Another approach to multi-step predictors design has been recently developed in [14], [15]. Therein, and ISPN-like structure is used, which employs batch training of the same one-step predictor f. The difference in these works is that recurrent neural networks (RNNs) are used to model f and an additional neural network is used to model an encoder that estimates the state $x(i|k)$ of the RNN from input-output data $\mathbf{u}_{\text{ini}}(i|k), \mathbf{y}_{\text{ini}}(i|k)$.

III. Deep Subspace Prediction Networks

The multi-step predictors presented in the previous section can be unified/generalized by the following deep subspace prediction network (DSPN) architecture:

$$y(i+1|k) = f_{i+1}(\mathbf{u}_{\text{ini}}(0|k), \mathbf{y}_{\text{ini}}(0|k), \mathbf{u}(k), c_i \mathbf{y}(i|k)), \quad (9)$$

where $i = 0, \ldots, N-1$, $c_0 = 0$, $c_i = 1$ for all other i, and

$$\begin{aligned} \mathbf{y}(i|k) &:= \operatorname{col}(y(i|k), \ldots, y(i-(n_a-1)|k)),\ i \geq n_a, \\ \mathbf{y}(i|k) &:= \operatorname{col}(y(i|k), \ldots, y(1|k)),\ 1 \leq i < n_a. \end{aligned}$$

Above, each mapping f_i can be modeled using any model class, including MLP neural networks, RNNs and reproducing kernel functions. We observe that in the DSPN architecture, each predictor f_i (except for $i = 1$) has access to both measured and simulated/predicted output data and that each predictor f_i has access to the combined data/information available to the predictors (5) and (8). This yields a richer prediction network class, which combines NARX and NOE modeling principles. The DSPN archictecture can be readily employed in nonlinear DPC algorithms as in Problem II.1, with (3b) replaced by (9).

Next, we provide an explicit link between the DSPN and the linear predictor of SPC [2]. To this end, define the agregated DSPN architecture:

$$\mathbf{y}(k) := \mathbb{F}(\mathbf{u}_{\text{ini}}(0|k), \mathbf{y}_{\text{ini}}(0|k), \mathbf{u}(k)), \quad (10)$$

$\mathbf{y}(k) = \operatorname{col}(y(1|k), \ldots, y(N|k))$ and $\mathbb{F} := \operatorname{col}(f_1, \ldots, f_N)$. Note that since each f_i is a MIMO predictor, it is the aggregation of several MISO predictors, i.e., $f_i = \operatorname{col}(f_{i,1}, \ldots, f_{i,p})$ where each $f_{i,j}$ predicts the j-th output, i.e., for $i = 0, \ldots, N-1$

$$\begin{aligned} y_j(i+1|k) &= f_{i+1,j}(\mathbf{u}_{\text{ini}}(0|k), \mathbf{y}_{\text{ini}}(0|k), \mathbf{u}(k), c_i \mathbf{y}(i|k)), \\ y(i+1|k) &= \operatorname{col}(y_1(i+1|k), \ldots, y_p(i+1|k)), \end{aligned}$$

where $j = 1, \ldots, p$ and p is the number of outputs.

The multi-step NARX predictors (5) can be recovered as a special case of the DSPN architecture, where the predictors for each time instant are decoupled. *We call the mapping $\mathbb{F}$ consisting of f_i-s defined as in* (5) *a Subspace Prediction Network (SPN), to distinguish it from the DSPN architecture.*

In order to define the MLS predictor [2] we need to introduce some notation. For any $k \geq 0$ (starting time instant in the data vector) and $j \geq 1$ (length of the data vector), define

$$\begin{aligned} \bar{\mathbf{u}}(k,j) &:= \operatorname{col}(u(k), \ldots, u(k+j-1)), \\ \bar{\mathbf{y}}(k,j) &:= \operatorname{col}(y(k), \ldots, y(k+j-1)). \end{aligned}$$

Then we can define the Hankel data matrices:

$$\begin{aligned} \mathbf{U}_p &:= \begin{bmatrix} \bar{\mathbf{u}}(0, T_{\text{ini}}) & \ldots & \bar{\mathbf{u}}(T-1, T_{\text{ini}}) \end{bmatrix}, \\ \mathbf{Y}_p &:= \begin{bmatrix} \bar{\mathbf{y}}(1, T_{\text{ini}}) & \ldots & \bar{\mathbf{y}}(T, T_{\text{ini}}) \end{bmatrix}, \\ \mathbf{U}_f &:= \begin{bmatrix} \bar{\mathbf{u}}(T_{\text{ini}}, N) & \ldots & \bar{\mathbf{u}}(T_{\text{ini}}+T-1, N) \end{bmatrix}, \\ \mathbf{Y}_f &:= \begin{bmatrix} \bar{\mathbf{y}}(T_{\text{ini}}+1, N) & \ldots & \bar{\mathbf{y}}(T_{\text{ini}}+T, N) \end{bmatrix}. \end{aligned} \quad (11)$$

Next, by defining the matrix

$$\Theta := \mathbf{Y}_f \begin{bmatrix} \mathbf{U}_p \\ \mathbf{Y}_p \\ \mathbf{U}_f \end{bmatrix}^{\dagger} \in \mathbb{R}^{(pN) \times (mn_a + pn_b + mN)}, \quad (12)$$

we obtain the linear predictor used in SPC [2] as:

$$\mathbf{y}_{\text{LIN}}(k) = \Theta \begin{bmatrix} \mathbf{u}_{\text{ini}}(0|k) \\ \mathbf{y}_{\text{ini}}(0|k) \\ \mathbf{u}(k) \end{bmatrix}. \quad (13)$$

Next, define the input to the i-th predictor as $U_{\text{PN}}(i|k) = \operatorname{col}(\mathbf{u}_{\text{ini}}(0|k), \mathbf{y}_{\text{ini}}(0|k), \mathbf{u}(k), c_i \mathbf{y}(i|k))$ and define every predictor $f_{i,j}$ part of $\mathbb{F}$ using a MLP neural network with a single hidden layer, i.e.,

$$f_{i,j}(U_{\text{PN}}(i|k)) := \bar{w}_{i,j}^T \sigma_{\text{nn}} \left(\mathbf{W}_{i,j} U_{\text{PN}}(i|k) + \mathbf{b}_{i,j} \right) + \bar{b}_{i,j}, \quad (14)$$

where $\mathbf{W}_{i,j}$, $\mathbf{b}_{i,j}$ are a matrix and a vector which contain the weights and the biases of the hidden layer, respectively, $\bar{w}_{i,j}$ is a vector with the weights of the output layer and $\bar{b}_{i,j}$ is the bias of the output layer. Also, above σ_{nn} denotes a typical neural network activation function which is applied to the vector in between the round parentheses element wise.

Lemma III.1 *Consider the DSPN archictecture defined in* (10) *with $f_{i,j}$ defined as in* (14) *with the number of neurons in the hidden larger than or equal to $mn_a + pn_b + mN$. Then there exists an activation function σ_{nn} and a set of weights and biases such that the DSPN architecture recovers the linear predictor defined in* (12)-(13).

Proof: For brevity, we provide the proof for the number of neurons in the hidden layer equal to $mn_a + pn_b + mN$. Let σ_{nn} be a linear activation function, i.e., for any real number s, $\sigma_{\text{nn}}(s) = s$. Define

$$\mathbf{W}_{i,j} := \begin{bmatrix} I_{mn_a + pn_b + mN} & 0_{(mn_a + pn_b + mN) \times q_i} \end{bmatrix}$$

where

$$\begin{aligned} q_i &:= n_a p,\ i \geq n_a, \\ q_i &:= ip,\ i < n_a. \end{aligned}$$

Let $\mathbf{b}_{i,j} = 0$, $\bar{b}_{i,j} = 0$. Furthermore, for any $i = 1, \ldots, N$ and $j = 1, \ldots, p$ define $\bar{w}_{i,j} := \Theta^T_{[j+(i-1)p,\, :]}$, where $\Theta_{[j+(i-1)p,\, :]}$ is the $j + (i-1)p$ line in the matrix Θ. Then by substitution in (14) and the $\operatorname{col}(f_1, \ldots, f_N)$ operator definition we obtain that

$$\mathbb{F}(\mathbf{u}_{\text{ini}}(0|k), \mathbf{y}_{\text{ini}}(0|k), \mathbf{u}(k)) = \Theta \begin{bmatrix} \mathbf{u}_{\text{ini}}(0|k) \\ \mathbf{y}_{\text{ini}}(0|k) \\ \mathbf{u}(k) \end{bmatrix}.$$

For more than $mn_a + pn_b + mN$ hidden neurons, a zero matrix of suitable dimensions must be included in $\mathbf{W}_{i,j}$. ■

From the proof we observe that the DSPN architecture can recover the linear SPC predictor even if the predictors f_i do not depend on the output of previous predictors, i.e., the SPN architecture suffices. However, a reason to allow for this dependency is that it is likely to require a less complex MLP neural network for learning/identifying f_i, as it uses the outputs of previous predictors.

A. DSPN architectures with a linear part

In what follows we present a variant of all above presented prediction networks that includes a linear predictor, which we call regularized prediction networks. This is beneficial because the embedded linear predictor reduces extrapolation errors to non-training data, compared to black-box neural networks. Also, many nonlinear systems consist of a known linear part that is dominant at steady-state and some unknown, nonlinear dynamics active under certain operating conditions, e.g., interconnected oscillators with sinusoidal coupling.

To define the regularized DSPN prediction, let Θ_i be real matrices of suitable dimensions and define:

$$\begin{aligned} y_r(i+1|k) &= y(i+1|k) + y_{\text{LIN}}(i+1|k) \\ &= y(i+1|k) + \Theta_{i+1} U_{\text{PN}}(i|k),\ i = 0, \dots N-1, \end{aligned} \tag{15}$$

where for DSPN,

$$U_{\text{PN}}(i|k) = \text{col}(\mathbf{u}_{\text{ini}}(0|k), \mathbf{y}_{\text{ini}}(0|k), \mathbf{u}(k), c_i \mathbf{y}(i|k)),$$

for SPN $U_{\text{PN}}(i|k) = \text{col}(\mathbf{u}_{\text{ini}}(0|k), \mathbf{y}_{\text{ini}}(0|k), \mathbf{u}(k))$ and for DPN $U_{\text{PN}}(i|k) = \text{col}(\mathbf{u}_{\text{ini}}(i|k), \mathbf{y}_{\text{ini}}(i|k), u(i|k))$. The above regularized prediction networks can be regarded as a multi-step extension of the linear (ARX model based) regularization for NARX models introduced in [16] for modeling inverse dynamics in the context of feedforward control.

IV. Training of various prediction networks

The first step of the training is to generate persistently exciting input-output data from the system of interest. This can be achieved by applying a white noise or PRBS signal to the system and measuring the output or by designing a rich multisine input signal [17].

The problem of learning the coefficients of all predictors f_{i+1} part of any of the discussed prediction networks (DPN, SPN, DSPN) can be posed as the optimization problem

$$\begin{aligned} \Omega^* = \arg\min_{\Omega} & \sum_{k=\max\{n_a,n_b\}}^{k_{sim}-N} L(Y_k, \hat{Y}_k) \\ \text{subject to: } & \\ y(i+1|k) &= f_{i+1}(\Omega^{i+1}, U_{\text{PN}}(i|k)) \\ \Omega &= \text{col}(\Omega^1, \dots, \Omega^N) \\ \hat{Y}_k &= \text{col}(y(1|k), \dots, y(N|k)) \\ Y_k &= \text{col}(y(k+1), \dots, y(k+N)) \\ i &= 0, \dots, N-1, \end{aligned} \tag{16}$$

where $L(Y_k, \hat{Y}_k) = \frac{1}{Np}\|Y_k - \hat{Y}_k\|_2^2$ is the mean squared error (MSE). Ω^i contains all weights and biases of the i-th neural network predictor, whose size varies with the number of hidden layers and number of neurons per layer. Note that for the different prediction networks, the training procedure remains similar, however, the size of Ω^i must be adjusted accordingly. In general, either sequential or batch training can be performed, but sequential training scales better with the prediction horizon. In this case, for DPN and DSPN, $u(i|k)$ will be replaced by $u(k+i)$ and outputs of previously trained predictors will be used in $U_{\text{PN}}(i|k)$. For SPN, since the predictors are not coupled, sequential or separate training in parallel can be performed.

In the case of regularized prediction networks with a linear part, we first identify Θ_i^* using least squares on the same data set, for each i-th predictor. Then we leave Θ_i as a free matrix variable and add the squared Frobenius norm regularization cost to the data fit error, i.e.

$$L(Y_k, \hat{Y}_k) = \frac{1}{Np}\|Y_k - \hat{Y}_k\|_2^2 + \lambda \sum_{i=1}^{N} \|\Theta_i - \Theta_i^*\|_{\text{F}}^2. \tag{17}$$

This regularized training procedure prevents competition among the nonlinear and linear predictors, as originally shown in [16].

V. Simulation results

In order to compare different multi-step predictors, we consider the following inverted pendulum model [12]:

$$\begin{aligned} \begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} &= \begin{bmatrix} 1 - \frac{bT_s}{J} & 0 \\ T_s & 1 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} \frac{T_s}{J} \\ 0 \end{bmatrix} u(k) \\ &\quad - \begin{bmatrix} \frac{MLgT_s}{2J} \sin(x_2(k)) \\ 0 \end{bmatrix} \\ y(k) &= x_2(k) + e(k), \end{aligned} \tag{18}$$

where $u(k)$ and $y(k)$ are the system input torque and pendulum angle at time instant k, while $J = \frac{ML^2}{3}$, $M = 1kg$ and $L = 1m$ are the moment of inertia, mass and length of the pendulum. Moreover, $g = 9.81m/s^2$ is the gravitational acceleration, $b = 0.1$ is the friction coefficient and the sampling time $T_s = 0.033$s. The performance of the presented predictors is evaluated over a horizon $N = 40$. To do so, a training data set $\mathcal{D}$ of 20000 samples is collected by exciting (18) with a multisine signal [17]. For the noise scenario, white Gaussian measurement noise was added to the output with a standard deviation of 0.005 [rad] which ensures a SNR of 40 dB. The testing data set consists of 400 samples collected from a sinusoidal input whose frequency was not excited in the training data set.

The orders of the input and output delays are chosen as $n_a = n_b = 5$ for all multi-step predictors, as in [12]. All compared multi-step predictors are modeled using MLP neural networks with two hidden layers and *tansig* activation function, followed by a linear output layer. The number of neurons in the hidden layer varies per predictor as follows. For DSPN and DSPN with linear part, the number of neurons in each hidden layer is equal to the number of inputs of a fully connected predictor in the prediction network, which yields 54 neurons. For DPN and SPN, the number of neurons is computed such that all prediction networks presented have a similar complexity, which is given by the number of free parameters. This yields 71 neurons per layer for DPN and 56 for SPN.

In what follows we will present results obtained by training sequentially DPN, SPN, DSPN and DSPN with linear part predictors in Python using PyTorch [18] and ADAM.

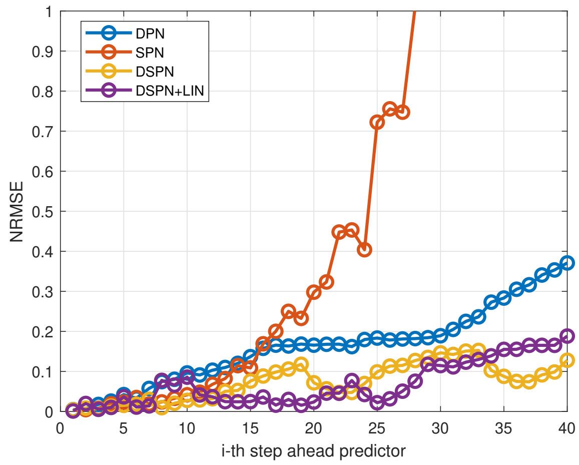

Fig. 1: NRMSE validation for all i-th step ahead predictors in the multi-step prediction networks using noiseless data. For SPN, the NRMSE grows above 2 after the 27^{th} predictor.

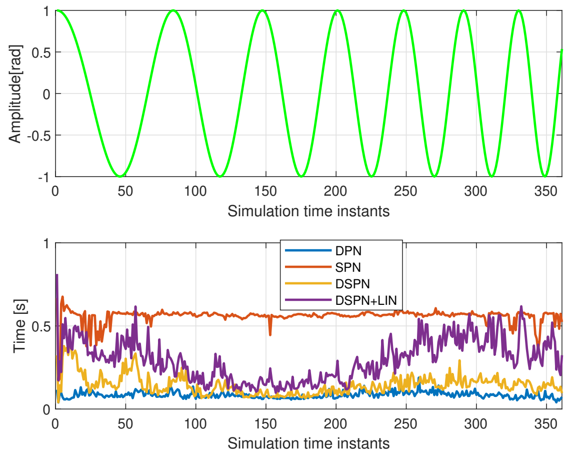

Fig. 2: Sweep sine reference and CPU time during nonlinear DPC simulation for all prediction networks for noiseless data with $N = 40$.

To reduce the computational burden induced by the long prediction horizon $N = 40$, all prediction networks were trained in a sequential manner. Thus individual training was performed for each predictor of the networks. In the case where the predictors are interconnected, the outputs of previous predictors are stored and used as inputs for the next ones. For each training, the learning rate was chosen as 0.0001 for 30000 epochs. For DSPN with linear part, λ from (17) was set to 0.1. For DSPN with linear part, the linear part is initialized with reference Θ_i^* values from (17), while for the neural network, the output layer is initialized to zero while the other layers are initialized randomly.

The performance of the identified prediction network on the validation set was quantified via the normalized root mean squared error (NRMSE) [15]. The NRMSE for all prediction networks in the noiseless case is illustrated in Fig. 1, where we can see that DSPN and DSPN with linear part clearly outperform DPN and SPN. Moreover, the NRMSE on the validation data set for certain chosen i-th step ahead predictors is presented for all prediction networks for both noiseless and noisy data in Table I and II.

We observe that DSPN and DSPN with linear part outperform DPN, while SPN performs very poorly for horizons longer than 10. To ensure a fair comparison with the SPN architecture, which uses the same input data $U_{\text{PN}}(i|k)$ for all predictors $i = 1, \ldots, N$, we have increased the number of neurons in the hidden layers up to 80, but the performance remained similar to the one reported in Fig. 1.

Prediction Network	NRMSE results: blue - best; purple - second best				
	$N = 5$	$N = 10$	$N = 20$	$N = 30$	$N = 40$
DPN	0.042	0.096	0.165	0.189	0.371
SPN	0.023	0.042	0.297	1.624	1.578
DSPN	0.015	0.027	0.071	0.146	0.127
DSPN+LIN	0.036	0.085	0.023	0.114	0.188

TABLE I: NRMSE validation for chosen i-th step ahead predictors for all prediction networks using noiseless data.

Prediction Network	NRMSE results: blue - best; purple - second best				
	$N = 5$	$N = 10$	$N = 20$	$N = 30$	$N = 40$
DPN	0.077	0.112	0.144	0.177	0.345
SPN	0.068	0.085	0.257	1.094	1.589
DSPN	0.062	0.082	0.130	0.196	0.213
DSPN+LIN	0.063	0.079	0.057	0.078	0.312

TABLE II: NRMSE validation for chosen i-th step ahead predictors for all prediction networks using noisy data.

Next, the trained predictors are employed in a nonlinear DPC algorithm that minimizes the cost function in (2) and where various predictors are used to obtain the future predicted outputs. The cost function weights are $Q = 200$, $R = 0.5$ and $N = 40$. The chosen DPC trajectory is the swept sine that is presented in the top part of Fig. 2. For all predictors, the predictive control problem is solved in MATLAB using $fmincon$ with SQP as the nonlinear solver. The simulations were conducted on a desktop computer with the following specifications: Intel® Core™ i7-12700K Processor 5.0 GHz, 64GB RAM, 64–bit OS. The performance of different predictors in nonlinear predictive control are evaluated via four metrics: the integral squared error $J_{ISE} = \sum_{k=1}^{T_{\text{end}}} \|y(k) - r(k)\|_2^2$, the integral absolute error $J_{IAE} = \sum_{k=1}^{T_{\text{end}}} \|y(k) - r(k)\|_1$, the input cost $J_u = \sum_{k=1}^{T_{\text{end}}} \|u(k)\|_1$ and the DPC cost $J_{DPC} = \sum_{k=1}^{T_{end}} \|Q^{\frac{1}{2}}(y(k) - r_y(k))\|_2^2 + \|R^{\frac{1}{2}}(u(k) - r_u(k))\|_2^2$.

The CPU during the simulation of nonlinear DPC with various prediction networks and noiseless data is shown in Fig. 2. Despite using a general purpose SQP solver and a long prediction horizon, the corresponding nonlinear DPC problem is solved on average under $10T_s = 0.33$ seconds. FPGA implementation of SQP solvers coupled with parallel shooting SQP for nonlinear predictive control [19] is capable of speeding up calculations 10 times, which is promising for real-time implementation.

The performance of the prediction networks for noiseless and noisy data is presented in Table III and IV. It can be

Prediction Network	J_{ISE}	J_{IAE}	J_U	J_{DPC}
DPN	1.39	13.29	481	771
SPN	1.75	15.71	520	914
DSPN	0.91	9.05	489	698
DSPN + LIN	0.48	7.79	487	628

TABLE III: Nonlinear DPC performance for noiseless data (blue - best; purple - second best).

Prediction Network	J_{ISE}	J_{IAE}	J_U	J_{DPC}
DPN	1.61	15.69	478	803
SPN	2.13	16.87	515	990
DSPN	1.12	10.49	486	741
DSPN + LIN	1.18	10.88	468	716

TABLE IV: Nonlinear DPC performance for noisy data (blue - best; purple - second best).

noticed from these tables that DSPN and DSPN with a linear part outperform the classical DPN and SPN in both noisy and noise-free cases. The control performance was expected since the identification results showed a significant improvement during the model validation.

In order to illustrate the improved control performance of DSPN, the MPC tracking error is presented in Fig. 3. It can be seen that DSPN with a linear part outperforms the other prediction networks. It is important to mention that SPN resulted in inferior tracking performance, which is why it was not included in the figure, as it would obscure the other results.

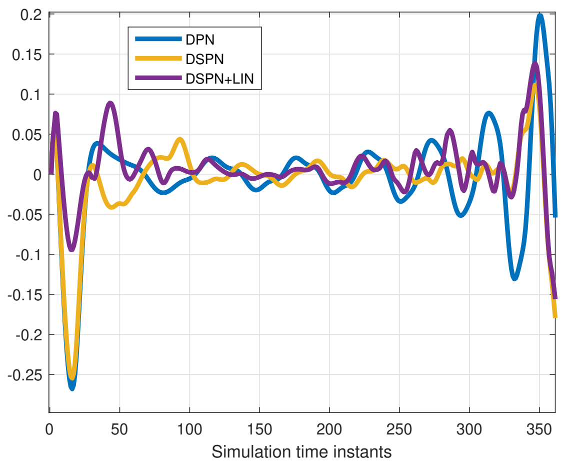

Fig. 3: Nonlinear DPC tracking error.

VI. Conclusions

In this paper we introduced a novel, unifying multi-step prediction network for indirect nonlinear data-driven predictive control, called deep subspace prediction network. We proved that DSPN predictors based on MLP neural networks can recover the subspace predictive control predictor for linear systems as a specific, linear network architecture. Based on this insight, we have derived a new regularization method for DSPN predictors. Simulation results on an inverted pendulum example showed that both DSPN and DSPN with linear part predictors based on MLP neural networks achieve high tracking control performance in the presence of noisy data. Future work will deal with efficient solvers for nonlinear DPC based on DSPN, formal analysis of training and prediction accuracy in the presence of noisy data and input-to-state closed-loop stability guarantees.

Acknowledgements: The first author acknowledges the NWO project PGN Mechatronics, project number 17973.

References

[1] F. Lamnabhi-Lagarrigue, A. Annaswamy, S. Engell, A. Isaksson, P. Khargonekar, R. M. Murray, H. Nijmeijer, T. Samad, D. Tilbury, and P. Van den Hof, “Systems & Control for the future of humanity, research agenda: Current and future roles, impact and grand challenges,” *Annual Reviews in Control*, vol. 43, pp. 1–64, 2017.
[2] W. Favoreel, B. D. Moor, and M. Gevers, “SPC: Subspace predictive control,” *IFAC Proceedings Volumes*, vol. 32, no. 2, pp. 4004–4009, 1999, 14th IFAC World Congress 1999, Beijing, Chia, 5-9 July.
[3] M. Nørregard, O. Ravn, N. K. Poulsen, and L. K. Hansen, *Neural Networks for modelling and control of dynamic systems*. Springer, London, 2000.
[4] M. Lazar and O. Pastravanu, “A neural predictive controller for nonlinear systems,” *Mathematics and Computers in Simulation*, vol. 60, no. 3-5, pp. 315–324, 2002.
[5] M. Ławryńczuk, “Neural networks in model predictive control,” *Intelligent systems for knowledge management*, pp. 31–63, 2009.
[6] S. Emami and A. Banazadeh, “Online identification of aircraft dynamics in the presence of actuator faults,” *Journal of Intelligent & Robotic Systems*, vol. 96, pp. 541–553, 2019.
[7] K. Patan, “Neural network-based model predictive control: Fault tolerance and stability,” *IEEE Transactions on Control Systems Technology*, vol. 23, no. 3, pp. 1147–1155, 2015.
[8] C. Greco, G. Menga, E. Mosca, and G. Zappa, “Performance improvements of self-tuning controllers by multistep horizons: The MUSMAR approach,” *Automatica*, vol. 20, no. 5, pp. 681–699, 1984.
[9] D. Liu, S. L. Shah, and D. G. Fisher, “Multiple prediction models for long range predictive control,” *IFAC Proceedings Volumes*, vol. 32, no. 2, pp. 6775–6780, 1999.
[10] L. Huang, J. Lygeros, and F. Dörfler, “Robust and kernelized data-enabled predictive control for nonlinear systems,” *arXiv*, p. 2206.01866, 2022.
[11] D. Masti, F. Smarra, A. D’Innocenzo, and A. Bemporad, “Learning affine predictors for mpc of nonlinear systems via artificial neural networks,” *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 5233–5238, 2020.
[12] A. Dalla Libera and G. Pillonetto, “Deep prediction networks,” *Neurocomputing*, vol. 469, pp. 321–329, 2022.
[13] G. Pillonetto, A. Y. Aravkin, D. Gedon, L. Ljung, A. H. Ribeiro, and T. Schon, “Deep networks for system identification: a survey,” *ArXiv*, vol. abs/2301.12832, 2023.
[14] D. Masti and A. Bemporad, “Learning nonlinear state-space models using deep autoencoders,” in *2018 IEEE Conference on Decision and Control (CDC)*, 2018, pp. 3862–3867.
[15] G. I. Beintema, M. Schoukens, and R. Tóth, “Deep subspace encoders for nonlinear system identification,” *Automatica*, vol. 156, p. 111210, 2023.
[16] M. Bolderman, M. Lazar, and H. Butler, “On feedforward control using physics–guided neural networks: Training cost regularization and optimized initialization,” in *2022 European Control Conference (ECC)*, 2022, pp. 1403–1408.
[17] J. Schoukens and L. Ljung, “Nonlinear system identification: A user-oriented road map,” *IEEE Control Systems Magazine*, vol. 39, no. 6, pp. 28–99, 2019.
[18] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, “Pytorch: An imperative style, high-performance deep learning library,” in *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035.
[19] P. C. N. Verheijen, M. Haghi, M. Lazar, and D. Goswami, “Parallel shooting sequential quadratic programming for nonlinear MPC problems,” *arxiv.org/abs/2307.10868*, 2023.