Parametric Identification of a Dynamical System with Switching

Anna Golovkina(✉) and Vladimir Kozynchenko

Saint Petersburg State University, Saint Petersburg, Russia
a.golovkina@spbu.ru

Abstract. Many systems both in nature and created by humans give rise to time series data with complex, nonlinear dynamics. Moreover, such time series can cover different dynamical regimes which identification is of pivotal importance in system modeling. Ordinary differential equations with switching provide a tool for modeling physical phenomena whose time series data exhibit different dynamical modes. Usually, these modes are determined by changing unmeasured parameters of the system that should be learned from the available data. The paper proposes a novel learning structure in polynomial neural network (PNN) basis suitable for parametric identification of dynamical systems and doesn't require usage of numerical integration methods. The PNN weight matrices incorporate the information about the parameters of ODEs and vice versa the unknown parameters of ODEs can be recovered from the PNN weights. Transferring the knowledge about the particular states dependencies in ODEs to PNN can be carried out by finding the initial weight matrices of PNN. The paper proposes a method for PNN initialization based an iterative procedure for step by step non stationary ODE flow computing in the polynomial form. However, even when the ODEs are unknown, PNN can be learned from scratch and provide parameter identification for ODEs. We evaluate the proposed approach on synthetic dataset generated with the system of ODEs for an electrostatical deflector. As a result, PNN successfully uncovers different dynamical regimes and predict the switching dynamics for different initial conditions outside the training data range.

Keywords: Polynomial neural networks · Taylor mapping · System identification · Lie transformation · Dynamic systems

1 Introduction

A lot of dynamical processes in biology, physics, chemistry and so on can be modeled by ordinary differential equations (ODEs) usually derived from the first principles and knowledge of intrinsic nature of the considered process. So that the constructed equations adequately describe the observable process, they

Supported by Saint Petersburg State University, project ID: 90317740.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2022
O. Gervasi et al. (Eds.): ICCSA 2022 Workshops, LNCS 13380, pp. 557–569, 2022.
https://doi.org/10.1007/978-3-031-10542-5_38

are combined with the available measurements recorded as input-output signals to find the unknown/uncertain model parameters: *system identification*. This field has been a large active research area for many years and even with several specialized sub-domains.

The interest is explained by the fact that the identified ordinary differential equations can describe responses of a dynamical process to all possible inputs and initial conditions. Thus, they provide the basis for simulation of processes dynamics, short-term and mid-term prediction along with the core for the design of control systems. That is why recently we see a growing interest in discovering the governing equations from the experimental data rather then black-box modeling, regression or curve fitting.

In should be noted that many complex dynamical signals naturally has a compositional form, in the sense that their data generating process can be decomposed into different dynamical modes [1]. Discovering of these modes in the framework of ordinary differential equations usually means identification of the unobserved parameters in the system corresponding to the different regimes.

However, parametric identification of a model in the form of ODE and its usage for making prediction requires numerical integration to get particular solution for certain parameters and initial condition. Thus, the computations start over again if the initial condition or parameters change. What is more, in general, a model is a simplified representation of the real system. In practice, due to presence of multiple sources of uncertainties and assumptions made at the model construction stage, inevitably, ODEs can mimic only essential features of the real process.

These challenges motivated the development and usage of so called physics inspired neural networks (PINN), appealed to integrate modern NN architectures and domain knowledge represented in the form of ordinary (partial) differential equations [2]. The proposed benefits behind this structure are concluded in leveraging modern NN learning algorithms to fit a model to the data in the same time preserving physical consistency of the predictions.

Usually, the governing physics equations are embedded into PINN architectures either by constraining NN output with mathematical operators (loss function) forcing to follow the prescribed dynamics or by incorporating a network to traditional numerical methods for solving differential equation. For instance, instead of classical L_2 regularization during model learning, [3,4] require the NN output to satisfy Euler-Lagrange equation, [5,6] use Hamilton's equations for this purpose, while [3] proposes energy conservation law as additional regularization. Adepts of the second approach adopts Runge-Kutta or finite-difference methods to construct neural networks [7,8]. It also should be noted that many modern learning models (e.x. general purpose recurrent neural networks (RNN), residual neural networks etc.) can be viewed by observing obvious correspondence to many of these classical algorithms. For example, it has been pointed out that the Euler scheme has similarities with NN containing residual connections [9,10].

Summing up the aforementioned points, PINN can be perceived as an analogue to the model in form of ODE (PDE) but enabling convenient interface

for learning from new coming data. However, it should be noted that PINN is capable to identify only system parameters that are constant in time. In this work we overcome these limitations by exploiting the advantages of polynomial neural networks for nonstationary dynamical systems and as a particular case dynamical systems with switching.

The approach is based on Taylor mapping technique for solving ODEs first time introduced in [11,12] and developed in [13]. This paper aims to present the algorithm for non stationary PNN weights initialization in the form admitting analysis of PNN convergence to general solution of non-autonomous ODE. The algorithm is illustrated for identification of a particular non-autonomous ODEs with parametric switching. The results are compared to identification with on recurrent neural network of LSTM type.

The rest of the paper is organized as follows: Sect. 2 introduces the problem setup, Sect. 3 contains the algorithm description for transferring non-autonomous ODEs to PNN architecture, while Sect. 4 demonstrates the results of numerical experiment on parametric identification and learning carried out for non-autonomous ODEs describing deflector dynamics.

2 Problem Setup

Let us consider a dynamical system described by a system of ordinary differential equations as follows:

$$\frac{d\mathbf{X}}{dt} = F(t, \mathbf{X}) \tag{1}$$

where t is an independent variable (time), $\mathbf{X} \in \mathbb{R}^n$ is a n-dimensional state of the system, F is a known vector field mapping $F : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^n$ and representing the prior knowledge of the system.

A curve $\mathbf{X}(t)$ solving (1) with initial condition $\mathbf{X}(t_0) = \mathbf{X}_0$ is an integral curve of a vector field F. For the fixed t,

$$\mathbf{X}(t, \mathbf{X}_0) = \mathcal{M}_F^t(\mathbf{X}_0) = \mathbf{X}_0 + \int_{t_0}^{t} F(\tau, \mathbf{X}(\tau))d\tau$$

is a flow generated by vector field F.

Solving system (1) means finding mapping $\mathcal{M}_F^t$. Traditional numerical solvers divide the interval $[t_0, t]$ into several steps and iteratively compute $\mathbf{X}(t_k)$ starting from $\mathbf{X}_0$ until reaching $\mathbf{X}_t$.

The simplest method using this general framework is explicit Euler method, in which a single step takes the following form:

$$\mathbf{X}(t_{k+1}) = \mathbf{X}(t_k) + hF(t_k, \mathbf{X}(t_k)). \tag{2}$$

More complicated numerical integration techniques preserves the same iterative structure (2) where the state vector values at the previous time step generally are not separable from the nonlinear function F. This fact means that another

initial condition $\mathbf{X}_0$ generates a new sequence of points in the phase space. Its computing each time requires repetition of procedure (2).

The papers aims to find an approximate representation of the flow $\mathcal{M}_F^t(\mathbf{X}_0)$ in the form

$$\mathbf{X}(t) = \sum_{i=1}^{m} R^{1i}(t, t_0)\mathbf{X}_0^{[i]} \tag{3}$$

and derive a procedure to construct unknown $R^{1i}(t, t_0)$, $i = \overline{1..m}$ matrices. Here $\mathbf{X}^{[k]}$ means k-th Kronecker power of vector X. For example, for $\mathbf{X} = (x_1, x_2)$ we have $\mathbf{X}^{[2]} = (x_1^2, x_1x_2, x_2^2)$, $\mathbf{X}^{[3]} = (x_1^3, x_1^2x_2, x_1x_2^2, x_2^3)$ after reduction of the same terms.

3 PNN Construction

3.1 PNN General Architecture

The accuracy of approximation (3) depends of the length of time interval $T = [t_0, t]$ and number of nonlinear terms m. Dividing the T in a smaller steps and increasing m, obviously, results into more precise numerical ODE solving. Flow $\mathcal{M}_F^t(\mathbf{X}_0)$ can be associated with a network consisting of a sequence of polynomial neurons shown in Fig. 1. When (1) is non autonomous, PNN's weight matrices $R^{1i}(t_j, t_{j-1})$ depend not only on the length of a time step $t_j - t_{j-1}$ but also on time itself. It means that, generally speaking, PNN's weights are not shared between the layers.

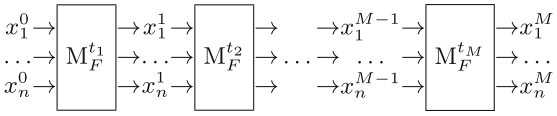

Fig. 1. Architecture of non stationary PNN with M layers

Thus, PNN architecture can represent a general solution of ordinary differential equation with the required degree of accuracy. To incorporate the knowledge about the system expressed in the ODE form into PNN, one should find weight matrices $R^{1i}(t_j, t_{j-1})$, $i = \overline{1..m}$, $j = \overline{1..M}$ directly from (1).

3.2 PNN Initialization

Let us described the proposed algorithm for the case, when $F(t, \mathbf{X}) \in C^\infty(\mathbb{R} \times \mathbb{R}^n)$, so it can be expanded in Taylor series in the neighborhood of $(t_0, \mathbf{X}_0)$. We consider the finite number of terms in the expansion and for the following

inferences treat a non-stationary nonlinear system of ODEs with a polynomial right-hand side

$$\frac{d\mathbf{X}}{dt} = \sum_{k=0}^{N} \mathrm{P}^{1k}(t)\,\mathbf{X}^{[k]} \tag{4}$$

An iterative algorithm for constructing an approximate solution to nonlinear Eq. (4) in form (3) is based on the method of immersion in the space of phase moments. Let us introduce the notation for the vector of phase moment of m-th order $\mathbf{X}^{[m]}$ — Kroneker's degree of vector $\mathbf{X}$. Accepting this notation we can show that $\mathbf{X}^{[m]}$ satisfies the equation of form (5).

Theorem 1. *If* $\mathbf{X}$ *satisfies Eq.* (4)*, then vector* $\mathbf{X}^{[m]}$ *satisfies* (5) *for any* $m = \overline{1..N}$

$$\frac{d\mathbf{X}^{[m]}}{dt} = \sum_{k=m}^{N} \mathrm{P}^{mk}(t)\,\mathbf{X}^{[k]}, \tag{5}$$

where matrices $\mathrm{P}^{mk}(t)$ $k \geq m$ *can be found by the formula*

$$\mathrm{P}^{mk}(t) = \left\langle \mathrm{P}^{1k-m+1}(t) \otimes \mathrm{E}^{[m]} + \mathrm{E} \otimes \mathrm{P}^{m-1k-1}(t) \right\rangle. \tag{6}$$

A proof is carried out by induction within the order of the phase moment. Obviously, the basis ($m = 1$) is hold resulting in the matrices P^{1k} from the Eq. (4). Taking the induction step, we assume that the statement holds for any given case $m = l$ and prove that it must also hold for the next case $m = l + 1$. For sake of further computing convenience, let us illustrate the procedure for $m = 2$ and introduce the necessary matrix operations.

Here and further in the transformations, we use the property of derivative of tensor product. Thus, the derivative of the vector of second phase moments can be transformed in the following way:

$$\begin{aligned}
\frac{d\mathbf{X}^{[2]}}{dt} &= \frac{d\mathbf{X}}{dt} \otimes \mathbf{X} + \mathbf{X} \otimes \frac{d\mathbf{X}}{dt} \\
&= \left(\sum_{k=1}^{N} \mathrm{P}^{1k}(t)\,\mathbf{X}^{[k]} \right) \otimes \mathbf{X} + \mathbf{X} \otimes \left(\sum_{k=0}^{N} \mathrm{P}^{1k}(t)\,\mathbf{X}^{[k]} \right) \\
&= \sum_{k=1}^{N} \left(\left(\mathrm{P}^{1k}(t)\,\mathbf{X}^{[k]} \right) \otimes (\mathrm{E}\mathbf{X}) + (\mathrm{E}\mathbf{X}) \otimes \left(\mathrm{P}^{1k}(t)\,\mathbf{X}^{[k]} \right) \right) \\
&= \sum_{k=1}^{N} \left\langle \mathrm{P}^{1k} \otimes \mathrm{E} + \mathrm{E} \otimes \mathrm{P}^{1k} \right\rangle \mathbf{X}^{[k+1]} = \sum_{k=2}^{N} \mathrm{P}^{2k}(t)\,\mathbf{X}^{[k]}
\end{aligned} \tag{7}$$

where

$$\mathrm{P}^{2k}(t) = \left\langle \mathrm{P}^{1k}(t) \otimes \mathrm{E} + \mathrm{E} \otimes \mathrm{P}^{1k}(t) \right\rangle, k \geq 2 \tag{8}$$

While calculating matrix $\mathrm{P}^{2k}(t)$, we use the operation of tensor multiplication and the operation of matrix addition. Taking into account the Kronecker degree definition for a vector, during tensor multiplication of matrices, we sum the columns corresponding to the same monomials $x_1^{k_1}...x_n^{k_n}$. Then, after adding the matrices, we remove the duplicate rows. The result of joint use of tensor multiplication and matrix addition according to that rule, we enclose in brackets $\langle\rangle$ (see Eq. (6))

where

$$\mathrm{P}^{3k}(t) = \left\langle \mathrm{P}^{1k-2}(t) \otimes \mathrm{E}^{[2]} + \mathrm{E} \otimes \mathrm{P}^{2k-1}(t) \right\rangle, k \geq 3. \tag{9}$$

Let us take an induction step and assume (7) holds for $m = l$, then the derivative of the vector of $l+1$ phase moments can be transformed in the similar way

$$\begin{aligned}
&\frac{d\mathbf{X}^{[l+1]}}{dt} = \frac{d}{dt}\left(\mathbf{X} \otimes \mathbf{X}^{[l]}\right) = \frac{d\mathbf{X}}{dt} \otimes \mathbf{X}^{[l]} + \mathbf{X} \otimes \frac{d\mathbf{X}^{[2]}}{dt} \\
&= \left(\sum_{k=1}^{N} \mathrm{P}^{1k}(t)\,\mathbf{X}^{[k]}\right) \otimes \mathbf{X}^{[l]} + \mathbf{X} \otimes \left(\sum_{k=l}^{N} \mathrm{P}^{lk}(t)\,\mathbf{X}^{[k]}\right) \\
&= \sum_{k=1}^{N} \left(\left(\mathrm{P}^{1l}(t)\,\mathbf{X}^{[k]}\right) \otimes (\mathrm{E}\mathbf{X})^{[l]} + (\mathrm{E}\mathbf{X}) \otimes \left(\mathrm{P}^{lk}(t)\,\mathbf{X}^{[k]}\right)\right) \\
&= \sum_{k=1}^{N} \left\langle \mathrm{P}^{1k-l+1} \otimes \mathrm{E}^{[l]} + \mathrm{E} \otimes \mathrm{P}^{lk} \right\rangle \mathbf{X}^{[k+1]} \qquad (10) \\
&= \sum_{k=l}^{N} \left\langle \mathrm{P}^{1k-l} \otimes \mathrm{E}^{[l]} + \mathrm{E} \otimes \mathrm{P}^{lk-1} \right\rangle \mathbf{X}^{[k]} = \sum_{k=l+1}^{N} \mathrm{P}^{lk}(t)\,\mathbf{X}^{[k]}
\end{aligned}$$

where

$$\mathrm{P}^{l+1k}(t) = \left\langle \mathrm{P}^{1k-l}(t) \otimes \mathrm{E}^{[2]} + \mathrm{E} \otimes \mathrm{P}^{lk-1}(t) \right\rangle, k \geq l+1. \tag{11}$$

Equation (10) together with (11) prove the theorem.

Finally, let us introduce an iterative routine for matrices R^{1i} computing, based on utilization of auxiliary matrices P^{mk} constructed with the Theorem 1. At first, we consider a linear equation ($N = 1$)

$$\frac{d\mathbf{X}}{dt} = \mathrm{P}^{11}(t)\,\mathbf{X}. \tag{12}$$

The solution of this equation can be found analytically or numerically (for example $\mathrm{R}^{11}(t, t_0) = \exp\left((t - t_0)\,\mathrm{P}^{11}\right)$ if $\mathrm{P}^{11}(t) = \mathrm{P}^{11}$) in the form

$$\mathbf{X}(t) = \mathrm{R}^{11}(t, t_0)\,\mathbf{X_0}, \qquad \mathbf{X_0} = \mathbf{X}(t_0) \tag{13}$$

Further, we consider an Eq. (4) with quadratic nonlinearity ($N = 2$) and introduce a procedure to find the solution

$$\frac{d\mathbf{X}}{dt} = \mathrm{P}^{11}(t)\,\mathbf{X} + \mathrm{P}^{12}(t)\,\mathbf{X}^{[2]} \tag{14}$$

Substitute solution (13) to linear Eq. (12) into the second term on the right-hand side of nonlinear Eq. (14)

$$\frac{d\mathbf{X}}{dt} = \mathrm{P}^{11}(t)\,\mathbf{X} + \mathrm{P}^{12}(t)\,\mathrm{R}^{22}(t,t_0)\,\mathbf{X_0}^{[2]}, \tag{15}$$
$$\mathrm{R}^{22}(t,t_0) = \left(\mathrm{R}^{11}(t,t_0)\right)^{[2]}$$

Equation (15) becomes linear with respect to $\mathbf{X}$ and inhomogeneous, thus its solution can be written in the form

$$\mathbf{X}(t) = \mathrm{R}^{11}(t,t_0)\,\mathbf{X_0} + \mathrm{R}^{12}(t,t_0)\,\mathbf{X_0}^{[2]}, \tag{16}$$

where

$$\mathrm{R}^{12}(t,t_0) = \int_{t_0}^{t} \mathrm{R}^{11}(t,\tau)\,\mathrm{P}^{12}(t)\,\mathrm{R}^{22}(\tau,t_0)\,d\tau \tag{17}$$

Then, we need to find expression for $\mathbf{X}^{[2]}$ to iteratively increase an order of non-linearity for the solution. For this, we use corresponding to (14) auxiliary system of equations with respect to vector of phase moments introduced in the Theorem 1. Substitute linear solution (13) into the second term on the right-hand side of nonlinear Eq. (5):

$$\frac{d\mathbf{X}^{[2]}}{dt} = \mathrm{P}^{22}(t)\,\mathbf{X}^{[2]} + \mathrm{P}^{23}(t)\,\mathrm{R}^{33}(t,t_0)\,\mathbf{X_0}^{[3]}, \tag{18}$$
$$\mathrm{R}^{33}(t,t_0) = \left(\mathrm{R}^{11}(t,t_0)\right)^{[3]}$$

Again, the Eq. (18) is linear with respect to $\mathbf{X}$ and inhomogeneous, this its solution can be written in the form:

$$\mathbf{X}^{[2]}(t) = \mathrm{R}^{22}(t,t_0)\,\mathbf{X_0}^{[2]} + \mathrm{R}^{23}(t,t_0)\,\mathbf{X_0}^{[3]}, \tag{19}$$

where

$$\mathrm{R}^{23}(t,t_0) = \int_{t_0}^{t} \mathrm{R}^{22}(t,\tau)\,\mathrm{P}^{23}(t)\,\mathrm{R}^{33}(\tau,t_0)\,d\tau. \tag{20}$$

To see the iterative pattern, let us show the introduced procedure for one step forward. We consider an Eq. (4) with cubic nonlinearity ($N = 3$)

$$\frac{d\mathbf{X}}{dt} = \mathrm{P}^{11}(t)\,\mathbf{X} + \mathrm{P}^{12}(t)\,\mathbf{X}^{[2]} + \mathrm{P}^{13}(t)\,\mathbf{X}^{[3]} \tag{21}$$

and repeat the steps to find its solution.

Substitute solution (13) into the third term on the right-hand side of nonlinear Eq. (21) and solution (19) into the second term of (21)

$$\frac{d\mathbf{X}}{dt} = \mathrm{P}^{11}(t)\,\mathbf{X} + \mathrm{P}^{12}(t)\,\mathrm{R}^{22}(t,t_0)\,\mathbf{X_0}^{[2]} + \mathrm{P}^{23}(t)\,\mathrm{R}^{33}(t,t_0)\,\mathbf{X_0}^{[3]}. \tag{22}$$

Again, these substitutions made (22) linear with respect to $\mathbf{X}$ and its solution can be written in the form:

$$\mathbf{X}(t) = \mathrm{R}^{11}(t,t_0)\,\mathbf{X_0} + \mathrm{R}^{12}(t,t_0)\,\mathbf{X_0}^{[2]} + \mathrm{R}^{13}(t,t_0)\,\mathbf{X_0}^{[3]}, \tag{23}$$

where

$$\mathrm{R}^{13}(t,t_0) = \int_{t_0}^{t} \mathrm{R}^{11}(t,\tau)\,\mathrm{P}^{12}(t)\,\mathrm{R}^{23}(\tau,t_0)\,d\tau + \int_{t_0}^{t} \mathrm{R}^{11}(t,\tau)\,\mathrm{P}^{13}(t)\,\mathrm{R}^{33}(\tau,t_0)\,d\tau. \tag{24}$$

Thus, now we can see a general pattern to find successively unknown matrices R^{1i}, $i = \overline{1..m}$. Increasing at each step the order $l \le N$ of non-linearity in the right side of the Eq. (4) and auxiliary Eq. (5), we try to get rid of the terms higher than linear (for $k = \overline{2..l}$) by substituting

- instead of $k = \overline{2..l-1}$: the expressions for $\mathbf{X}^{[k]}$ found at the previous steps solving (5);
- instead of $k = l$:a linear solution (13).

By these means, at each successive step, we get a linear inhomogeneous equation with respect to $\mathbf{X}$ that can be solved analytically. Finally, for $l = N$ we get

$$\mathbf{X}(t) = \sum_{k=1}^{N} \mathrm{R}^{1k}(t,t_0)\,\mathbf{X_0}^{[k]}, \tag{25}$$

where

$$\mathrm{R}^{ik}(t,t_0) = \sum_{j=i+1}^{k} \int_{t_0}^{t} \mathrm{R}^{ii}(t,\tau)\,\mathrm{P}^{ij}(t)\,\mathrm{R}^{jk}(\tau,t_0)\,d\tau, \tag{26}$$

$$\mathrm{R}^{ii}(t,t_0) = \left(\mathrm{R}^{11}(t,t_0)\right)^{[i]}.$$

Thus, this section introduces an iterative procedure to find the solution to (4) in polynomial form (3) and initialize the weights of the corresponding PNN.

4 Numerical Example

To illustrate the proposed method for identification of non-autonomous ODEs, let us consider an electrostatic deflector equations. This system is described by the following ODEs:

$$\begin{aligned} \dot{x} &= y \\ \dot{y} &= -2x + \frac{x^2}{p(t)} \end{aligned} \tag{27}$$

where x and y are the state variables and p is a positive unknown parameter that can have different values over the time. The initial values of (27) are $x(0) = x_0$, $y(0) = y_0$. For the sake of simplicity, assume that $p(t)$ is a step function:

$$p(t) = \begin{cases} p_1, \text{ if } t \in [0, t_1] \\ p_2, \text{ if } t \in [t_1, t_2] \\ p_3, \text{ if } t \in [t_2, t_3] \end{cases}$$

The solution of the system (27) is continuous, meaning that at each switching point the following condition is hold $(x_i(t_i), y_i(t_i)) = (x_{i+1}(t_i), y_{i+1}(t_i))$, $i = 1..2$.

Let us select a set of p_i values and initial condition and generate training data using system (27) and any traditional numerical solver for ODE. We consider the time interval $T = [0; 3]$, $t_i = [1; 2; 3]$ $p_i = [10; 50; 100]$ and $(x_0; y_0) = (-2; 4)$. It should be noted, that this information is used only for generating training data, from which unknown parameters are identified.

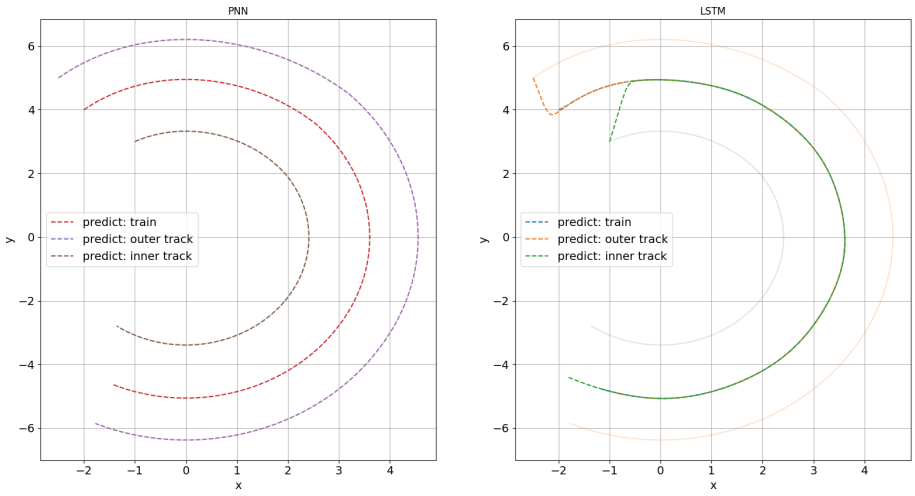

Fig. 2. Phase portrait of the reconstructed system using PNN (with weights initialization) and LSTM

Following the procedure for ODEs right hand side identification described in [14,15], we approximately find the unknown matrices P^{1i} for each of the switch intervals:

$$
\begin{aligned}
P_1^{11} &= \begin{pmatrix} 4.07 \cdot 10^{-6} & 0.99 \\ -1.99 & 4.38 \cdot 10^{-6} \end{pmatrix} P_1^{12} = \begin{pmatrix} 1.24 \cdot 10^{-7} & 3.28 \cdot 10^{-6} & 2.00 \cdot 10^{-9} \\ 0.09 & -9.71 \cdot 10^{-8} & 3.34 \cdot 10^{-6} \end{pmatrix} \\
P_2^{11} &= \begin{pmatrix} 2.4 \cdot 10^{-5} & 0.99 \\ -1.99 & 1.35 \cdot 10^{-5} \end{pmatrix} P_2^{12} = \begin{pmatrix} 1.68 \cdot 10^{-7} & 6.70 \cdot 10^{-7} & -4.07 \cdot 10^{-9} \\ 0.02 & -3.56 \cdot 10^{-8} & 6.95 \cdot 10^{-7} \end{pmatrix} \\
P_3^{11} &= \begin{pmatrix} -4.32 \cdot 10^{-5} & 0.99 \\ -2.00 & 1.45 \cdot 10^{-5} \end{pmatrix} P_3^{12} = \begin{pmatrix} -1.44 \cdot 10^{-7} & 3.91 \cdot 10^{-7} & 6.08 \cdot 10^{-9} \\ 9.99 \cdot 10^{-3} & 1.03 \cdot 10^{-7} & 3.40 \cdot 10^{-7} \end{pmatrix}
\end{aligned}
\tag{28}
$$

The maximal order of non-linearity in the right hand side of (27) is two, so we should build PNN of the second order nonlinearity having at least two weight matrices $R^{1i}(t)$, $i = \overline{1..2}$ at each layer. The Eq. (27) is step autonomous, so the matrices R^{1i} are shared through the time steps within each of the switching interval.

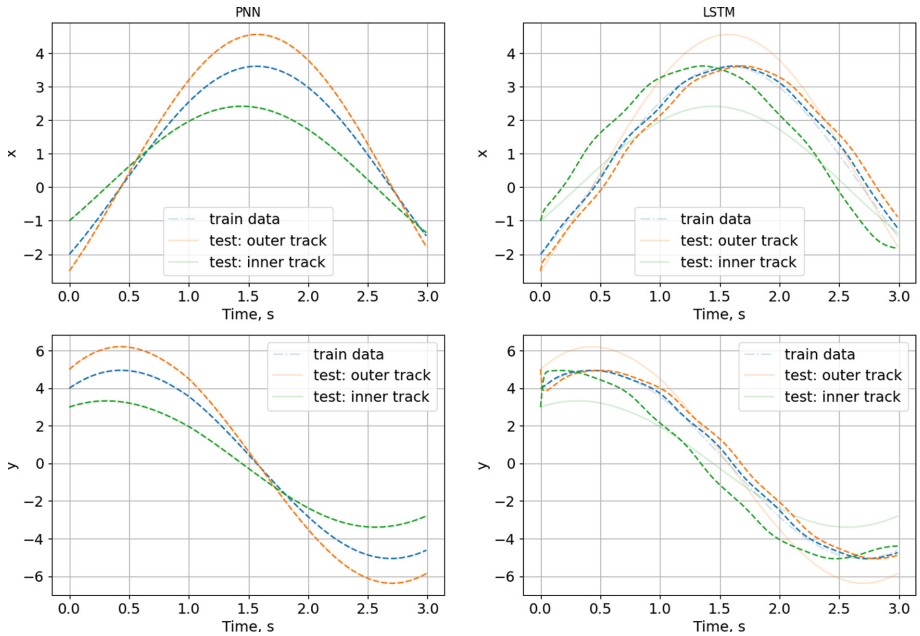

Fig. 3. System trajectories of the reconstructed system using PNN (with weights initialization) and LSTM

At first let us use the weights initialization procedure described in Sect. 3.2. We start from the system matrices obtained through the reconstruction procedure (28). To test the generalization ability of PNN, two test initial values are considered: $[-2.5; 5]$ and $[-1; 3]$. The PNN is required to reconstruct the

system trajectories started form these values. The results are compared with LSTM neural network because of its internal connection with a discrete time dynamical system.

Figures 2 and 3 depict the comparison of the phase portrait and system trajectories correspondingly for the identified system predicted for the train and test initial values.

It is clearly seen from the Figs. 2 and 3 that LSTM is unable to recover the dynamics of the system just memorizing the trajectory used for training. Even for different initial conditions it can predict only seen system behaviour, while PNN correctly reconstruct the dynamics and shows great generalization ability for the initial valued laying beyond the range used for training.

Now let us compare identification abilities of PNN and LSTM for another set of switching parameters that don't provide smoothness of the trajectories: $p_i = [2; 50; 5]$ and learn PNN from scratch without reconstruction of ODEs matrices and applying weights initialization procedure described above. The results are presented in the Fig. 4.

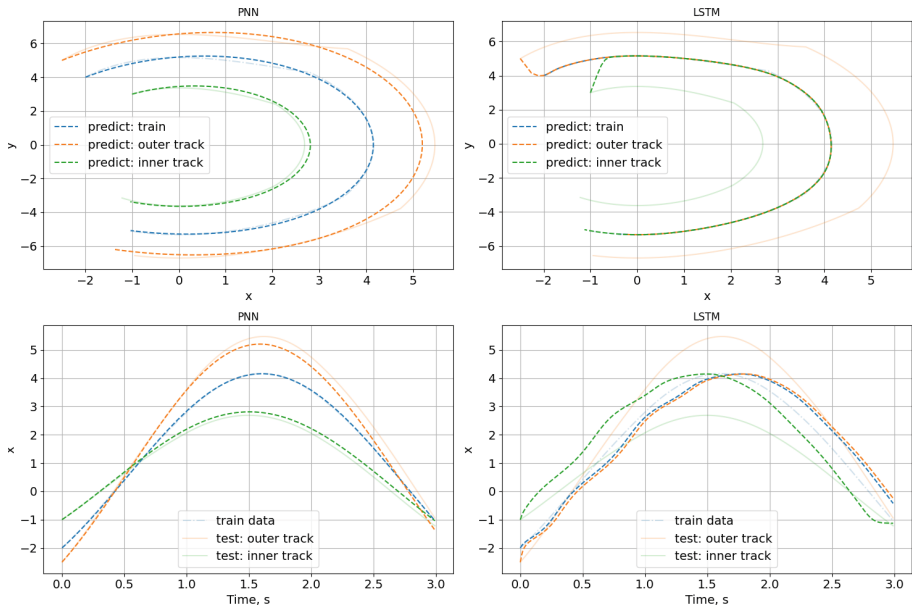

Fig. 4. Phase portrait and $x(t)$ trajectories for the reconstructed system using PNN (without weights initialization) and LSTM

Figure 4 shows that even starting from zero initial weights, PNN is able to recover the dynamics correctly. Of course, there are discrepancies between predicted and true trajectory for test initial values around the switching points, but nevertheless in general the system behaviour is captured correctly. Unfortunately, in the same time LSTM again just memorize the training data.

5 Conclusion

This paper presents an approach for system identification based on PNN that doesn't require multiple solving of ODEs with different parameters. By the design PNN can incorporate prior knowledge of dynamical system by direct transferring ODE to neural polynomial architecture (PNN). The approach is based on Taylor mapping technique for solving ODEs. It provides transformation of the dynamic system represented in the form of ODEs to the initial weights of the polynomial neural network. This paper aims to present the algorithm for non stationary PNN weights initialization in the form admitting analysis of PNN convergence to general solution of non-autonomous ODE. However, even when we don't have information about the dynamical system generated the system trajectory, PNN can be learned from scratch and still provide physically consistent results. The identification ability of PNN is illustrated in the example of ODEs with switches describing the dynamics of electrostatics deflector. Prediction results using PNN (with and without weights initialization) are compared with LSTM. They show PNN ability to recover hidden system dynamics and generalize prediction for the initial values beyond the range of the training data.

References

1. Ojeda, C., Georgiev, B., Cvejoski, K., Schucker, J., Bauckhage, C., Sánchez, R.J.: Switching dynamical systems with deep neural networks. In: 2020 25th International Conference on Pattern Recognition (ICPR), 2021, pp. 6305–6312 (2021)
2. Karniadakis, G.E., Kevrekidis, I.G., Lu, L., Perdikaris, P., Wang, S., Yang, L.: Physics-informed machine learning. Nat. Rev. Phys. **3**(6), 422–440 (2021)
3. Lutter, M., Ritter, C., Peters, J.: Deep Lagrangian networks: using physics as model prior for deep learning (2019)
4. Roehrl, M.A., Runkler, T.A., Brandtstetter, V., Tokic, M., Obermayer, S.: Modeling system dynamics with physics-informed neural networks based on Lagrangian mechanics, vol. 53 (2020)
5. Greydanus, S., Dzamba, M., Yosinski, J.: Hamiltonian neural networks, vol. 32 (2019)
6. Han, C.-D., Glaz, B., Haile, M., Lai, Y.-C.: Adaptable Hamiltonian neural networks, vol. 3 (2021)
7. Wang, Y.-J., Lin, C.-T.: Runge-Kutta neural network for identification of dynamical systems in high accuracy. IEEE Trans. Neural Netw. **9**(2), 294–307 (1998)
8. Dufera, T.T.: Deep neural network for system of ordinary differential equations: vectorized algorithm and simulation. Mach. Learn. Appl. **5**, 100058 (2021)
9. Chen, R.T.Q., Rubanova, Y., Bettencourt, J., Duvenaud, D.K.: Neural ordinary differential equations, vol. 31 (2018)
10. Rubanova, Y., Chen, R.T.Q., Duvenaud, D.K.: Latent ordinary differential equations for irregularly-sampled time series, vol. 32 (2019)
11. Andrianov, S.: Dynamical Modeling of Control Systems for Particle Beams. SPb: Saint Petersburg State University (2004)
12. Andrianov, S.: Symbolic computation of approximate symmetries for ordinary differential equations. Math. Comput. Simul. **57**(3–5), 147–154 (2001)

13. Ivanov, A., Golovkina, A., Iben, U.: Polynomial neural networks and Taylor maps for dynamical systems simulation and learning. Front. Artif. Intell. Appl. **325**, 1230–1237 (2019)
14. Golovkina, A., Kozynchenko, V., Kulabukhova, N.: Reconstruction and identification of dynamical systems based on Taylor maps. In: Gervasi, O., et al. (eds.) ICCSA 2021. LNCS, vol. 12956, pp. 360–369. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-87010-2_26
15. Golovkina, A., Kozynchenko, V., Kulabukhova, N.: Reconstruction of ordinary differential equations from irregularly distributed time-series data. In: 9th International Conference "Distributed Computing and Grid Technologies in Science and Education". Crossref, December 2021