

Reconstruction and Identification of Dynamical Systems Based on Taylor Maps

Anna Golovkina, Vladimir Kozynchenko, and Nataliia Kulabukhova(✉)

Saint Petersburg State University, St. Petersburg, Russia
{a.golovkina,v.kozynchenko,n.kulabukhova}@spbu.ru

Abstract. In the process of conducting various physical experiments, a certain set of data is accumulated. Processing and interpreting the simulation results is a fundamental task for analyzing the behaviour of the model, predicting its future actions, and managing the entire system. This paper provides an overview of the currently existing approaches to identification of dynamic systems models: white, gray and black boxes. Special attention is paid to methods based on neural networks. The article suggests a combined approach that allows both preserving a physical consistency of the model and using modern methods for learning from data. A polynomial neural network of a special architecture, approximating the general solution of the system of ordinary differential equations (ODEs) in the form of Taylor map is considered. This model can work in the case of a small amount of initial data, which is a problem when exploiting traditional machine learning methods, and neural networks in particular. The paper presents a new learning approach for PNN based on two steps: reconstructing an ODEs system based on a single trajectory, and identifying a general solution to initialize the weights of a neural network. Neural network representation allows using traditional learning algorithms for additional fine-turning the weights in line with new measured data. A toy example of a nonlinear deflector demonstrates the power and generalization ability of the proposed algorithm.

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

1 Introduction

Dynamical system reconstruction and identification from time series data is crucial for many applied problems, industries and experimental research. This is explained by the necessity to interpreter the data, to understand the underlying process and predict its behaviour in future. Identification and reconstruction belong to the class of inverse problems that are usually ill-posed, meaning that many standard numerical methods cannot achieve high accuracy when solving

Supported by Saint Petersburg State University, project ID: 75206008.

© Springer Nature Switzerland AG 2021
O. Gervasi et al. (Eds.): ICCSA 2021, LNCS 12956, pp. 360–369, 2021.
https://doi.org/10.1007/978-3-030-87010-2_26

them. The approaches described in literature dedicated to dynamical system identification usually keep in line with one of these directions: white-box, grey-box and black-box models learning.

White-box models are based on first principles that are often expressed as systems of differential equations containing unknown parameters describing the concrete physical situation. These parameters are usually defined though optimization of least square error between the measured and the modelled data. Regularisation or other modification of optimization algorithms can improve the convergence. However, in many cases such models are overly complex and hard to obtain because of the nature of many systems and processes. In the same time their simplification can lead to a limited understanding of the underlying physical or technical setting.

Another very important approach in dynamical system identification are gray box models. They use both insights into the system e.g. in the form of conservation laws and black box models to infer hidden parameters. Usually, this group of methods requires both statistical data and numerical solvers to build the model and estimate its parameters.

On the other hand, applying black-box models usually based on machine learning (ML) methods allows avoiding numerical solvers completely and extracting the complex behaviour of the systems. Also, a strong motivation to use neural networks as a representation of system model comes from the universal approximation theorem, which states that a feed-forward network with a single hidden layer containing a finite number of neurons can approximate any continuous function on compact subsets of R^n, under mild assumptions on the activation function.

The idea of using neural networks (NNs) for physical systems learning is not new and elaborated in existing studies [6,13,15,16,21]. However, the approaches followed by the authors usually require large measured or simulated data sets for NN training. Otherwise, the prediction results are likely too far from the real physical behavior. Ordinary, the basic idea is either to build surrogate models that can replace physics-based models or incorporate NNs into the differential equation to approximate unknown terms (gray-box models). For example, [4] proposes dynamical systems learning from partial observations with ordinary differential equations (ODEs), while the [19] adds NNs to identify unknown coefficients in partial differential equations. Authors of [7] use a back-propagation technique through an ODE solver but this still requires traditional numerical solvers to simulate the dynamics.

Summarizing the generally adopted in the literature approaches, usually by the term physics-inspired NNs [11], authors mean either incorporating domain knowledge in the traditional NN architectures or providing additional loss functions for the physical inconsistency of the predictions [14]. Model-free systems learning and control considered in the papers [8,17] imply using various NN architectures. However, the generalisation ability of the constructed models for the inputs not presented in the training data range is not investigated.

Traditional and state-of-the-art ML/NN models are suitable for either forecasting or building surrogate models. Time series forecasting means that the information in the past values can alone be used to predict the future values. While a surrogate model extrapolates dynamics to new inputs that stem from the same training data range.

The paper aims to combine the strengths of the mentioned techniques to build an approach suitable for identification of wide variety of dynamical process with small training data. Moreover, the recovered model is requested to predict the dynamics for new inputs beyond the training samples.

To solve this problem, we consider a polynomial neural network (PNN) of a special architecture, approximating the general solution of system of ODEs in the form of Taylor map [12]. This means that the physical description of the provided data is preserved, but it becomes possible to use modern methods of training neural networks to adjust the coefficients in accordance with the collected data. The paper proposes PNN learning algorithm consisting in two steps.

The rest of the paper is organized as follows. Section 2 presents a brief description of the Taylor mapping approach and its connection with PNN. Section 3 introduces a PNN learning algorithm having two successive steps. The first one is reconstruction of system of ODE with a polynomial right side with only one sample. Then, Taylor mapping representation of the general solution is used to find an initial approximation of PNN weights. Step two utilizes the other available training data to fine tune the weights for a better correspondence of the model to the real process. Section 4 describes simple example of nonlinear cylindrical deflector to demonstrate the achieved results.

2 PNN Architecture

The transformation $\mathcal{M} : \mathbf{X}_0 = \mathbf{X}(t_0) \rightarrow \mathbf{X}(t_1)$ defines a Taylor map in form of

$$\mathbf{X}(t_1) = W_0 + W_1\,\mathbf{X}_0 + W_2\,\mathbf{X}_0^{[2]} + \ldots + W_k\,\mathbf{X}_0^{[k]}, \tag{1}$$

where $\mathbf{X}, \mathbf{X}_0 \in R^n$, matrices W_i are weights, and $\mathbf{X}^{[k]}$ means the k-th Kronecker power of vector $\mathbf{X}$ with the same terms reduction. For example, if $\mathbf{X} = (x_1, x_2)$, then $\mathbf{X}^{[2]} = (x_1^2, x_1x_2, x_2^2)$ and $\mathbf{X}^{[3]} = (x_1^3, x_1^2x_2, x_1x_2^2,\ x_2^3)$. The transformation (1) is linear in weights W_i and nonlinear with respect to the $\mathbf{X}_0$. The transformation (1) sometimes may be named Taylor maps or models [5], tensor decomposition [9], matrix Lie transform [1], exponential machines [18], and others. In fact, transformation (1) is a polynomial regression with respect to the components of $\mathbf{X}$ that directly defines the polynomial neuron (see Fig. 1). The Taylor map (1) approximates the general solution of the system of differential equations [2,10]. If the systems of ODE is known, the weights in (1) can be calculated directly. The initialized from ODEs Taylor map accurately represents the dynamics of the system without the necessity of using numerical solver [12].

This property of PNN based on Taylor map allows using it as a representation of dynamical system. The following section describes the procedure of PNN learning from time-series data.

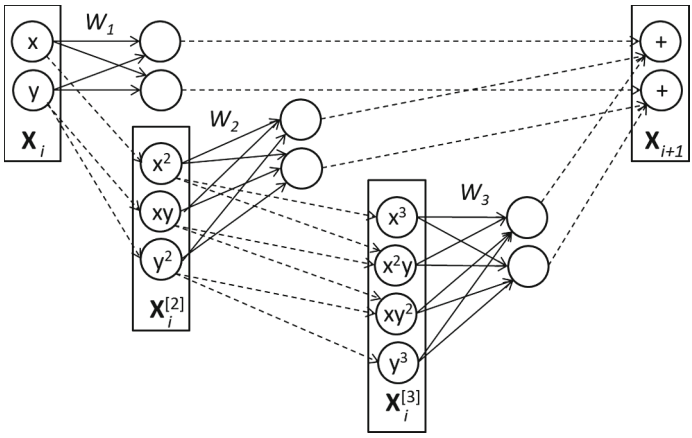

Fig. 1. Polynomial neuron of third order nonlinearity.

3 PNN Learning

PNN learning consists of two successive steps. Initialization of weights and their fine-tuning based on the measured data. According to the described in Sect.2 properties and structure of PPN, initialization of weights requires the presence of the system of ordinary differential equations. Then Taylor map approach is used to approximate the general solution of the system that correspond to the PNN. The following subsections contain the description of reconstruction approach of a system of ODEs from the data and Taylor mapping technique to identify its general solution.

3.1 Reconstruction of the Right Side of the System of Differential Equations

Let us denote the set of the parameters describing the process as vector $\mathbf{X}$ with changing in time components $X_j(t)$, $j = \overline{1,n}$. And we suppose to know the values of the vector function $\mathbf{X}(t)$ measured in M discrete times $t_0, \ldots, t_{M+1}$: $\mathbf{X}(t_0), \ldots \mathbf{X}(t_{M+1})$.

The main our assumption about the collected time-series data describing the multi parametric dynamical process is that it approximately follows autonomous ODE system with polynomial right-hand side

$$\frac{d\mathbf{X}}{dt} = \sum_{k=0}^{N} \mathrm{P}^{1k}\mathbf{X}^{[k]} \tag{2}$$

where t is an independent variable, $\mathbf{X} \in R^n$ is a state vector corresponding to the parameters of the dynamical process, and $\mathbf{X}^{[k]}$ means k-th Kronecker power of vector $\mathbf{X}$. 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.

Matrices P^{1k} are usually unknown, so we should approximately find them from the measurements $\mathbf{X}(t_0), \ldots \mathbf{X}(t_{M+1})$. Let us replace the derivatives $\frac{d\mathbf{X}}{dt}$ in the left side of (2) with finite differences:

$$\frac{\mathbf{X}(t_{i+1}) - \mathbf{X}(t_{i-1})}{t_{i+1} - t_{i-1}} = \sum_{k=0}^{N} \mathrm{P}^{1k} \mathbf{X}^{[k]}(t_i), \quad i = \overline{1, M}. \tag{3}$$

System of equations (3) is a system of linear algebraic equations that can be written in a matrix form

$$AP = B, \tag{4}$$

where:

$$A = \begin{pmatrix} \mathbf{Y}(t_1) & \mathbf{Y}^{[2]}(t_1) & \ldots & \mathbf{Y}^{[N]}(t_1) \\ \mathbf{Y}(t_2) & \mathbf{Y}^{[2]}(t_2) & \ldots & \mathbf{Y}^{[N]}(t_2) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{Y}(t_M) & \mathbf{Y}^{[2]}(t_M) & \ldots & \mathbf{Y}^{[N]}(t_M) \end{pmatrix}, \quad \mathbf{Y}^{[j]}(t_i) = \left(\mathbf{X}^{[j]}(t_i)\right)^T,$$

$$P = \left(\mathrm{P}^{11}, \ldots, \mathrm{P}^{1N}\right)^T,$$

$$B = \left(\left(\mathbf{X}(t_2) - \mathbf{X}(t_0)\right) / (t_2 - t_0), \ldots, \left(\mathbf{X}(t_{M+1}) - \mathbf{X}(t_{M-1})\right) / (t_{M+1} - t_{M-1})\right)^T.$$

The system (4) contains $n * (n + n_2 + \ldots + n_N)$ unknowns and $n * M$ equations. In these formulas, n is the dimension of the vector $\mathbf{X}$, and $n_i, i \geq 2$ is the dimension of its Kronecker degree $\mathbf{X}^{[i]}$. Hence it follows that in order to obtain a system with an equal number of equations and unknowns, the following condition must be met for the number of measurements $3 * M$:

$$M = n + n_2 + \ldots + n_N. \tag{5}$$

Condition (5) defines the connection between the system dimension which is related to the number of process parameters and the quantity of measurements. If (5) is violated, its impossible to unambiguously reconstruct the polynomial right side of (2) for the requested order of non linearity.

3.2 Identification of the General Solution of the System of Differential Equations

We consider autonomous system of ODEs with polynomial right-hand side approximately reconstructed from data in the previous subsection

$$\frac{d\mathbf{X}}{dt} = \sum_{k=0}^{N} \mathrm{P}^{1k} \mathbf{X}^{[k]}$$

An iterative algorithm for constructing an approximate solution to nonlinear equation (1) is based on the technique introduced in [3]. Starting from the linear

term in the right side of (1), we successively increase the order of polynomials to the required degree of non linearity. At the first step we consider the linear equation

$$\frac{d\mathbf{X}}{dt} = P^{11}\mathbf{X} \tag{6}$$

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

$$\mathbf{X}(t) = \mathrm{R}^{11}(t,t_0)\,\mathbf{X_0}, \qquad \mathbf{X_0} = \mathbf{X}(t_0) \tag{7}$$

Then, let us consider the nonlinear equation with the second order polynomials in the right side

$$\frac{d\mathbf{X}}{dt} = \mathrm{P}^{11}\mathbf{X} + \mathrm{P}^{12}\mathbf{X}^{[2]}. \tag{8}$$

To find the solution of (8), we substitute the solution (7) of the linear equation (6) found at the previous step to the second term on the right side of nonlinear equation (8):

$$\frac{d\mathbf{X}}{dt} = \mathrm{P}^{11}\mathbf{X} + \mathrm{P}^{11}\mathrm{R}^{22}(t,t_0)\,\mathbf{X_0}^{[2]}, \qquad \mathrm{R}^{22}(t,t_0) = \left(\mathrm{R}^{11}(t,t_0)\right)^{[2]} \tag{9}$$

The solution to the linear inhomogeneous equation (9) can be written in the form:

$$\mathbf{X}(t) = \mathrm{R}^{11}(t,t_0)\,\mathbf{X_0} + \mathrm{R}^{12}(t,t_0)\,\mathbf{X_0}^{[2]}, \tag{10}$$

where

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

To obtain an approximate solution of the form

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

we substitute solution (10) in the nonlinear terms of the truncated equation

$$\frac{d\mathbf{X}}{dt} = \mathrm{P}^{11}\mathbf{X} + \mathrm{P}^{12}\mathbf{X}^{[2]} + \mathrm{P}^{13}\mathbf{X}^{[3]}. \tag{13}$$

After substitution, we obtain the linear equation

$$\begin{gathered}\frac{d\mathbf{X}}{dt} = \mathrm{P}^{11}\mathbf{X} + \mathrm{P}^{12}\left(\mathrm{R}^{11}(t,t_0)\,\mathbf{X_0} + \mathrm{R}^{12}(t,t_0)\,\mathbf{X_0}^{[2]}\right)^{[2]} \\ + \mathrm{P}^{13}\left(\mathrm{R}^{11}(t,t_0)\,\mathbf{X_0} + \mathrm{R}^{12}(t,t_0)\,\mathbf{X_0}^{[2]}\right)^{[3]} = \mathrm{P}^{11}\mathbf{X} + \mathrm{P}^{12}\mathrm{R}^{22}(t,t_0)\,\mathbf{X_0}^{[2]} \\ + \mathrm{P}^{12}\Big(\left(\mathrm{R}^{11}(t,t_0)\,\mathbf{X_0}\right)\otimes\left(\mathrm{R}^{12}(t,t_0)\,\mathbf{X_0}^{[2]}\right) \\ + \left(\mathrm{R}^{12}(t,t_0)\,\mathbf{X_0}^{[2]}\right)\otimes\left(\mathrm{R}^{11}(t,t_0)\,\mathbf{X_0}\right)\Big) + \mathrm{P}^{13}\mathrm{R}^{33}(t,t_0)\,\mathbf{X_0}^{[3]} + \ldots\end{gathered} \tag{14}$$

where

$$\mathrm{R}^{33}(t) = \left(\mathrm{R}^{11}(t)\right)^{[3]}. \tag{15}$$

Let us introduce the notation

$$\left(\mathrm{R}^{11}(t,t_0)\,\mathbf{Y}\right) \otimes \left(\mathrm{R}^{12}(t,t_0)\,\mathbf{Y}^{[2]}\right) + \left(\mathrm{R}^{12}(t,t_0)\,\mathbf{Y}^{[2]}\right) \otimes \left(\mathrm{R}^{11}(t,t_0)\,\mathbf{Y}\right) = f(t,t_0,\mathbf{Y}) \tag{16}$$

Let us solve the matrix system of linear algebraic equations with the matrix of unknowns R^{23}:

$$\mathrm{R}^{23}\mathrm{M}_\mathrm{Y} = \mathrm{M}_\mathrm{f} \tag{17}$$

In the matrix system (17): $\mathrm{M}_\mathrm{Y} = \left(\mathbf{Y_1}^{[3]}, \ldots, \mathbf{Y_p}^{[3]}\right)$, $\mathbf{Y_1}, \ldots, \mathbf{Y_p}$ – arbitrary linearly independent system of vectors, $\mathrm{M}_\mathrm{f} = \left(f(t,t_0,\mathbf{Y_1}), \ldots, f(t,t_0,\mathbf{Y_p})\right)$.

The solution to system (17) is the matrix $\mathrm{R}^{23} = \mathrm{R}^{23}(t,t_0)$, which can also be obtained analytically by transforming the left-hand side of equality (16). As a result, we have

$$\left(\mathrm{R}^{11}(t,t_0)\,\mathbf{X_0}\right) \otimes \left(\mathrm{R}^{12}(t,t_0)\,\mathbf{X_0}^{[2]}\right) + \left(\mathrm{R}^{12}(t,t_0)\,\mathbf{X_0}^{[2]}\right) \otimes \left(\mathrm{R}^{11}(t,t_0)\,\mathbf{X_0}\right) = \mathrm{R}^{23}(t,t_0)\,\mathbf{X_0}^{[3]} \tag{18}$$

Substituting equation (18) into equation (14) and taking into account the terms up to the third order, we obtain the linear equation

$$\frac{d\mathbf{X}}{dt} = \mathrm{P}^{11}\mathbf{X} + \mathrm{P}^{12}\mathrm{R}^{22}(t,t_0)\,\mathbf{X_0}^{[2]} + \left(\mathrm{P}^{12}\mathrm{R}^{23}(t,t_0) + \mathrm{P}^{13}\mathrm{R}^{33}(t,t_0)\right)\mathbf{X_0}^{[3]}, \tag{19}$$

The solution to this system has the form

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

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

Continuing iterations up to order N, we get:

$$\mathbf{X}(t) = \mathrm{R}^{11}(t,t_0)\,\mathbf{X_0} + \textstyle\sum_{k=2}^{N} \mathrm{R}^{1k}(t)\,\mathbf{X_0}^{[k]},$$

$$\mathrm{R}^{1k}(t) = \textstyle\sum_{j=2}^{k} \int_{t_0}^{t} \mathrm{R}^{11}(t,\tau)\,\mathrm{P}^{1j}\mathrm{R}^{jk}(\tau,t_0)\,d\tau, \quad \mathrm{R}^{ii}(t,t_0) = \left(\mathrm{R}^{11}(t,t_0)\right)^{[i]}, \tag{20}$$

$$\mathrm{R}^{11}(t,t_0) = \exp\left((t-t_0)\,\mathrm{P}^{11}\right)$$

The matrices $\mathrm{R}^{jk}(t,t_0)$ are calculated using an algorithm similar to the algorithm for calculating the matrices $\mathrm{R}^{23}(t,t_0)$. The matrix $\mathrm{R}^{jk}(t,t_0)$ can also be obtained analytically by transforming the left-hand side of equality

$$\sum_{\sum_{i=1}^{j} k_i = k} \left(\mathrm{R}^{1k_1}(t,t_0)\,\mathbf{X_0}^{[k_i]}\right) \otimes \ldots \otimes \left(\mathrm{R}^{1k_j}(t,t_0)\,\mathbf{X_0}^{[k_j]}\right) = \mathrm{R}^{jk}(t,t_0)\,\mathbf{X_0}^{[k]}$$

3.3 Example

Let us consider an approximation [20] of particle motion in cylindrical deflector written in the form:

$$\begin{aligned} x^{'} &= y \\ y^{'} &= -2x + x^2/R \end{aligned} \tag{21}$$

We generate a training data as a particular solution of the system (21) with the initial condition $\mathbf{X_0} = (-2, 4)$ (Fig. 2, green points). The training set includes 27 points ($R = 10$). After this solution was generated, the equation is not used further.

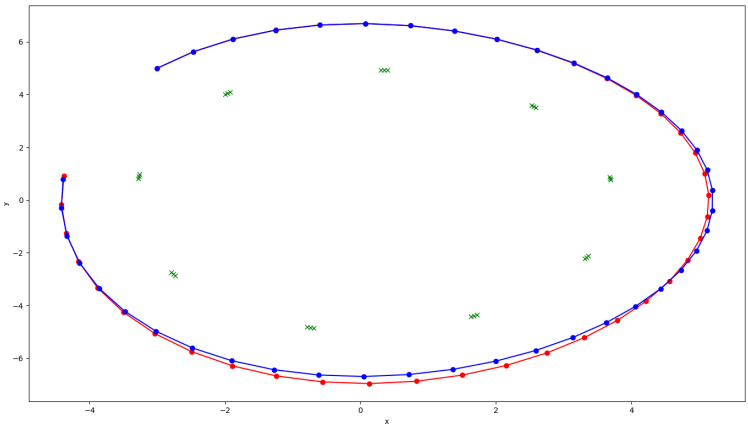

Fig. 2. Training data used for ODEs reconstruction (greed dots), system trajectory generated by PNN for another initial condition (red line), real system trajectory (blue line) (Color figure online)

Then, this generated trajectory is used to reconstruct the system of ordinary differential equations till the third order of non linearity according to the algorithm described in Sect. 3.1:

$$\begin{aligned} \frac{d\mathbf{X}}{dt} &= \begin{pmatrix} 4.06732625 \cdot 10^{-6} & 0.999964923 \\ -1.99993049 & -4.38301027 \cdot 10^{-6} \end{pmatrix} \mathbf{X} \\ &+ \begin{pmatrix} 1.24062940e \cdot 10^{-7} & 3.28166404 \cdot 10^{-6} & 2.00339676 \cdot 10^{-9} \\ 0.0999900634 & -9.71187750 \cdot 10^{-8} & 3.33946651 \cdot 10^{-6} \end{pmatrix} \mathbf{X^{[2]}} \\ &+ 10^{-7} \begin{pmatrix} -3.26578011 & 1.48318573 & -1.79415514 & 0.72283912 \\ 1.07008208 & 3.66009614 & -1.31485102 & 1.79102455 \end{pmatrix} \mathbf{X^{[3]}} \end{aligned} \tag{22}$$

Then, using the approximate system (22), we find the matrices of weights $\mathrm{R}^{11}(t, t_0)$ and $\mathrm{R}^{12}(t, t_0)$ for different values of t ($t_0 = 0$) according to the algorithm introduced in Sect. 3.2.

For example, for $t = 1.1$, the approximate solution to Eq. (21) for any initial values of $\mathbf{X_0}$ constructed up to the second order non linearity has the form:

$$\mathbf{X}\,(t = 1.1) = \begin{pmatrix} 0.0152181223886654 & 0.707024692103253 \\ -1.41404983954638 & 0.0152121475825109 \end{pmatrix} \mathbf{X_0} + \begin{pmatrix} 0.0330738670919230 & 0.0232052217767992 & 0.00808282751349695 \\ 0.0242876681630190 & 0.0338195709561698 & 0.0232073819255883 \end{pmatrix} \mathbf{X_0^{[2]}} \quad (23)$$

An example of comparing the numerical solution of Eq. (21) with the Runge-Kutta method (blue line) and the solution we obtained for the initial conditions $\mathbf{X_0} = (-3, 5)$ (red line) is shown in Fig. 2.

The Fig. 2 illustrates that the PNN with initial weights found according to the proposed algorithm already gives good prediction results for initial values beyond the training data set. Moreover, these weights can be fine-tuned with additionally measured data using standard optimization algorithms. It can provide even better model correspondence to the real system.

4 Conclusion

Recently, deep learning techniques have gained great popularity for their widespread success in many fields. Exploiting deep neural networks to help solve inverse problems particularly related to the reconstruction and identification of dynamical systems has been explored recently. However, standard machine learning algorithms can hardly provide physical interpretation of the considered dynamical process. Moreover, usually they require massive datasets to learn end-to-end mappings from the measurement domain to the target system domain. This paper proposes an architecture and learning algorithm for polynomial neural networks based on Taylor maps for dynamical system learning with small datasets. The Taylor mapping technique provides transformation of the dynamic system represented in the form of ODEs to the initial weights of the polynomial neural network. Thus, as a first step the algorithm requires reconstructing the right side of ODEs, that we propose to do in the polynomial form. We shown in a toy example of nonlinear deflector that the PNN with the initial weights found according to the proposed algorithms allows PNN to achieve great generalisation ability. However, the results can be improved even more by fine-tuning the initial weights from additional data using the standard optimization algorithms.

Acknowledgements. The authors would like to thank Saint Petersburg State University for the research grant ID: 75206008.

References

1. Andrianov, S.: A matrix representation of the lie transformation. In: Proceedings of the Abstracts of the International Congress on Computer Systems and Applied Mathematics, vol. 14 (1993)

2. Andrianov, S.: Symbolic computation of approximate symmetries for ordinary differential equations. Math. Comput. Simul. **57**(3–5), 147–154 (2001)
3. Andrianov, S.: Dynamical modeling of control systems for particle beams. Saint Petersburg State University, SPb (2004)
4. Ayed, I., Bezenac, E., Pajot, A., Brajard, J., Gallinari, P.: Learning dynamical systems from partial observations. arxiv:1902.11136 (2019)
5. Berz, M.: From Taylor series to Taylor models (1997). https://bt.pa.msu.edu/pub/papers/taylorsb/taylorsb.pdf
6. Bieker, K., Peitz, S., Brunton, S., Kutz, J., Dellnitz, M.: Deep model predictive control with online learning for complex physical systems. arxiv:1905.10094 (2019)
7. Chen, T.Q., Rubanova, Y., Bettencourt, J., Duvenaud, D.K.: Neural ordinary differential equations. In: Advances in Neural Information Processing Systems, pp. 6571–6583 (2018)
8. Chen, Y., Shi, Y., Zhang, B.: Optimal control via neural networks: a convex approach (2019). https://arxiv.org/pdf/1805.11835.pdf
9. Dolgov, S.: A tensor decomposition algorithm for large odes with conservation laws. Comput. Meth. Appl. Math. **19**(1), 23–38 (2019)
10. Dragt, A.: Lie methods for nonlinear dynamics with applications to accelerator physics (2011). https://inspirehep.net/record/955313/files/TOC28Nov2011.pdf
11. Group, T.: Physics-based deep learning (2019). https://github.com/thunil/Physics-Based-Deep-Learning
12. Ivanov, A., Golovkina, A., Iben, U.: Polynomial neural networks and Taylor maps for dynamical systems simulation and learning. ECAI2020 Accepted, not published yet. arxiv:1912.09986 (2019)
13. Jia, X., et al.: Physics guided RNNs for modeling dynamical systems: a case study in simulating lake temperature profiles. In: Proceedings of the 2019 SIAM International Conference on Data Mining, pp. 558–566. SIAM (2019)
14. Karpatne, A., Watkins, W., Read, J., Kumar, V.: Physics-guided neural networks (PGNN): an application in lake temperature modeling (2018). https://arxiv.org/pdf/1710.11431.pdf
15. Koppe, G., Toutounji, H., Kirsch, P., Lis, S., Durstewitz, D.: Identifying nonlinear dynamical systems via generative recurrent neural networks with applications to fMRI. arxiv:1902.07186 (2019)
16. Mohajerin, N., Waslander, S.L.: Multistep prediction of dynamic systems with recurrent neural networks. IEEE Trans. Neural Netw. Learn. Syst. **30**(11), 3370–3383 (2019)
17. Nagabandi, A., Kahn, G., Fearing, R., Levine, S.: Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning (2017). https://arxiv.org/pdf/1708.02596.pdf
18. Novikov, A., Trofimov, M., Oseledets, I.: Exponential machines. arxiv:1605.03795 (2017)
19. Raissi, M., Perdikaris, P., Karniadakis, G.E.: Physics informed deep learning (part II): data-driven discovery of nonlinear partial differential equations (2017)
20. Senichev, Y., Møller, S.: Beam dynamics in electrostatic rings (2000). http://accelconf.web.cern.ch/AccelConf/e00/PAPERS/MOP1B04.pdf
21. Yu, R., Zheng, S., Anandkumar, A., Yue, Y.: Long-term forecasting using higher order tensor RNNs. arxiv:1711.00073 (2019)