Lie Transform Based Polynomial Neural Networks for Dynamical Systems Simulation and Identification

Andrei Ivanov, Sergei Andrianov, and Alena Sholokhova
Saint Petersburg State University, Faculty of Applied Mathematics and Control Processes, Russia
05x.andrey@gmail.com, sandrianov@yandex.ru, al.sholokhova@gmail.com

Abstract. In the article, we discuss the architecture of the polynomial neural network that corresponds to the matrix representation of Lie transform. The matrix form of Lie transform is an approximation of general solution of the nonlinear system of ordinary differential equations. Thus, it can be used for simulation and modeling task. On the other hand, one can identify dynamical system from time series data simply by optimization of the coefficient matrices of the Lie transform. Representation of the approach by polynomial neural networks integrates the strength of both neural networks and traditional model-based methods for dynamical systems investigation. We provide a theoretical explanation of learning dynamical systems from time series for the proposed method, as well as demonstrate it in several applications. Namely, we show results of modeling and identification for both well-known systems like Lotka–Volterra equation and more complicated examples from retail, biochemistry, and accelerator physics.

Keywords: matrix Lie transform, deep polynomial neural network, dynamical systems simulation, time series based system identification

1 Introduction

Modeling and control of complex dynamical systems require techniques for non-linearities and uncertainties consideration. On the face of it, artificial neural networks could provide a suitable approach for dynamical systems ‘learning’. The applications cover different problems like solving of ordinary differential equations [9,12], signal processing [10], feedback control systems [11], dynamical systems modeling and identification [3], and others.

One group of researchers suggests solving a known differential equation with predefined mathematical formulas by neural networks. Another one tries to utilize benefits of neural networks for system identification and inverse problem solving when the equations of the system’s dynamics are not defined. In both cases, the neural network is considered as a black-box model and its applicability for dynamical systems modeling is explained only by the universal approximation theorem.
In the article, we describe a polynomial neural network, that is suitable for simulation and identification of dynamical systems in a mathematically rigorous style. The proposed architecture is a neural network representation of Lie propagator for dynamical systems integration that is introduced in [4] and commonly is used in charged particles beam dynamics simulation. The neural network representation allows incorporating parallel architecture, performance increasing and uncertainties accounting with the rigorous theory of dynamical systems investigation. We consider dynamical systems that can be described by nonlinear ordinary differential equations:

\[
\frac{d}{dt}X = F(t, X),
\]

where \( t \) is independent variable, \( X \in \mathbb{R}^n \) is state vector. There is an assumption that function \( F \) can be expanded in Taylor series with respect to the components of \( X \). Note that independent variable \( t \) can arise in the equation as an arbitrary nonlinear function. Such nonlinear systems arise in different fields, namely automated control, robotics, mechanical and biological systems, chemical reactions, drug development, molecular dynamics and so on.

![Neural network representation of matrix Lie transform up to the 3rd order of nonlinearities.](image-url)
2 Proposed Neural Network

Proposed neural network implements map $\mathcal{M} : \mathbf{X} \rightarrow \mathbf{Y}$ using following polynomial transformation

$$
\mathbf{Y} = W_0 + W_1 \mathbf{X} + W_2 \mathbf{X}^{[2]} + \ldots + W_k \mathbf{X}^{[k]},
$$

where $\mathbf{X}, \mathbf{Y} \in \mathbb{R}^n$, $W_i$ are weights matrices, and $\mathbf{X}^{[k]}$ means $k$-th Kroneker power of vector $\mathbf{X}$. For instance Fig. 1 presents neural network representation of map $\mathcal{M}$ up to the third order of nonlinearities for 2 dimensional state space. In each layer the input vector $\mathbf{X} = (x_1, x_2)$ is consequently transformed into $\mathbf{X}^{[2]} = (x_2^1, x_1 x_2, x_2^2)$ and $\mathbf{X}^{[3]} = (x_1^3, x_1^2 x_2, x_1 x_2^2, x_2^3)$ where weighted sum is applied. The output $\mathbf{Y}$ equals to sum of results from every layers. Note in the figure we reduce Kroneker powers for decreasing of weights matrices dimension (e.g. $\mathbf{X}^{[2]} = (x_2^1, x_1 x_2, x_2^2) \rightarrow (x_1^2, x_1 x_2, x_2^2)$).

The transformation $\mathcal{M}$ can be considered as an approximation of evolution operator of the (1) for predefined initial time $t_0$ and time interval $\Delta t$. This means that with appropriate weights $W_i = W_i(t_0, \Delta t)$ the evolution of initial state vector $\mathbf{X}_0 = \mathbf{X}(t_0)$ during time $\Delta t$ can be approximately calculated as $\mathbf{Y} = \mathbf{X}(t_0, \mathbf{X}_0, \Delta t_1) = \mathcal{M} \circ \mathbf{X}_0$. If the system (1) is time independent then weights $W_i$ are constant for a predefined time interval.

3 Mathematical Background

The dynamics of vector $\mathbf{X}$ in (1) can be presented in form of Lie transform

$$
\mathcal{M}(t|t_0) = T \exp \left( \int_{t_0}^{t} \mathcal{L}_{\mathbf{F}}(\tau) d\tau \right),
$$

where $\mathcal{L}_{\mathbf{F}}(\tau)$ is Lie operator associated with vector function $\mathbf{F}$ in (1). Transformation $\mathcal{M}$ is presented in form of the time-ordered exponential operator and can be identified with the dynamical system itself.

On the assumption that the function $\mathbf{F}$ allows its expansion in Taylor series, the required solution of equation (1) in its convergence region can be also presented in form of series

$$
\mathbf{X}(t) = \mathcal{M} \circ \mathbf{X}_0 = \sum_{k=0}^{\infty} R_k \mathbf{X}^{[k]}, \quad \mathbf{F} = \sum_{k=0}^{\infty} P_k \mathbf{X}^{[k]}, \quad (2)
$$

In (2) it is shown how to calculate matrices $R_k$ either analytically or numerically. The main idea is replacing differential equation (1) by the equation

$$
R'_k(t|t_0) = \sum_{j=0}^{k} P_{ij}(t) R_j(t|t_0), \quad 1 \leq i < k,
$$

where $P_{ij} = P_{(j-i+1)(j-1)}$, and $P_{1k} = P_k, R_{1k} = R_k$.

Another way to estimate matrix coefficients ($W_k = R_k$) of a truncated map

$$
\mathbf{X}(t) = W_0(t) + W_1(t) \mathbf{X}_0 + W_2(t) \mathbf{X}_0^{[2]} + \ldots + W_k(t) \mathbf{X}_0^{[k]}, \quad (3)
$$
is utilizing an appropriate numerical step-by-step integration method. Taking derivative of the $X(t)$ with respect to the $(2)$ one can obtain a system of equations

$$
\frac{d}{dt} X = \frac{d}{dt} W_0(t) + \frac{d}{dt} W_1(t)X_0 + \frac{d}{dt} W_2(t)X_0^2 + \ldots + \frac{d}{dt} W_k(t)X_0^k,
$$

$$
\frac{d}{dt} X = P_0(t) + P_1(t)X + P_2(t)X^2 + \ldots + P_p(t)X^p =

P_0(t) +

P_1(t) \left( W_0(t) + W_1(t)X_0 + W_2(t)X_0^2 + \ldots + W_k(t)X_0^k \right) +

P_2(t) \left( W_0(t) + W_1(t)X_0 + W_2(t)X_0^2 + \ldots + W_k(t)X_0^k \right)^2 + \ldots +

P_p(t) \left( W_0(t) + W_1(t)X_0 + W_2(t)X_0^2 + \ldots + W_k(t)X_0^k \right)^p,
$$

which leads to a new system of ordinary differential equations with respect to the weight matrices $W_k$

$$
\frac{d}{dt} W_k(t) = \sum_{i=1}^{p} P_i(t) \frac{\partial X^i}{\partial (X_0^j)^2}, \quad k = 0, 1, 2, \ldots
$$

Since the right-hand sides of the given equations depend only on $W_i$, the system can be numerically integrated with initial condition $W_k(0) = 0, k \neq 1; W_1(0) = E$ during necessary time interval.

To clarify this approach let’s consider 1-dim equation

$$
x' = kx^2,
$$

with the known general solution

$$
x(t) = \frac{x_0}{1 - ktx_0}; \quad x_0 = x(0).
$$

Let’s calculate matrix Lie map using described above technique. This means that we want to find out coefficients of the transformation

$$
x(t) = W_0 + W_1x_0 + W_2x_0^2 + W_3x_0^3 + \ldots
$$

Taking derivative of $x(t)$ yields the equations

$$
x' = W'_0 + W'_1x_0 + W'_2x_0^2 + W'_3x_0^3 + \ldots
$$

$$
x' = k(W_0 + W_1x_0 + W_2x_0^2 + W_3x_0^3 + \ldots)^2,
$$

which after combining like terms lead to the initial value problem for $W_i$

$$
W'_0 = kW_0^2,
W'_1 = 2kW_0W_1,
W'_2 = k(2W_0W_2 + W_1^2),
W'_3 = k(2W_0W_3 + 2W_1W_2), \ldots
$$
with initial condition $W_1(0) = 1; W_i(0) = 0, i \neq 1$. Consequently solving given equations one can found map

$$W_0 = 0,$$

$$W_1 = 1,$$

$$W_2 = kt,$$

$$W_3 = k^2 t^2, \ldots$$

and the estimation of general solution

$$x = x_0 + ktx_0^2 + k^2 t^2 x_0^3 + \ldots$$

As we say above this series equals to the general solution inside the region of convergence. Indeed, taking into account formula $(1 - q)^{-1} = 1 + q + q^2 + q^3 + \ldots$ the precise solution (5) can be written in the same form

$$x(t) = \frac{x_0}{1 - ktx_0} = x_0 + ktx_0^2 + k^2 t^2 x_0^3 + \ldots$$

### 3.1 Invariants preserving

Any numerical computational process leads to distortion of qualitative properties (e.g. dynamical and kinematical invariants). These quantities can be evaluated using, for example, Casimir operators. According to the Lie groups theory, we can construct invariants using special forms and use these data for computational process control.

As an example let’s consider Hamiltonian systems that are very popular in physics problems. The Hamiltonian nature leads us to preserve of the symplecticity of the map $M(t|t_0)$

$$M^T(t|t_0)J_0M(t|t_0) = J_0,$$

where $J_0 = \begin{pmatrix} 0 & E \\ -E & 0 \end{pmatrix}$ and $M(t|t_0) = \partial (M(t|t_0) \circ X_0) / \partial X_0^T$.

For truncated map (3) one can apply order-by-order symplectification scheme [1]. This leads to linear algebraic homogeneous equations for matrix elements $W_k = \{ w_k^{ij} \}$. For instance for 2 dimensional state vector $X = (x, y)$ and second order map

$$X = W_1 X_0 + W_2 X_0^2 = \begin{pmatrix} w_{11}^1 & w_{12}^1 \\ w_{21}^1 & w_{22}^1 \end{pmatrix} \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} + \begin{pmatrix} w_{11}^2 & w_{12}^2 & w_{13}^2 \\ w_{21}^2 & w_{22}^2 & w_{23}^2 \end{pmatrix} \begin{pmatrix} x_0^2 \\ x_0 y_0 \\ y_0^2 \end{pmatrix}$$

the describe above symplectic conditions are

$$w_{11}^1 w_{22}^1 - w_{12}^1 w_{21}^1 = 1,$$

$$w_{11}^2 w_{22}^2 - w_{12}^2 w_{21}^2 + 2w_{12}^1 w_{22}^1 - 2w_{12}^1 w_{21}^2 = 0,$$

$$w_{12}^1 w_{21}^2 - w_{12}^1 w_{22}^1 + 2w_{12}^1 w_{21}^2 - 2w_{12}^1 w_{21}^2 = 0,$$

$$w_{11}^2 w_{23}^1 - w_{13}^2 w_{21}^2 = 0, w_{12}^2 w_{23}^1 - w_{13}^2 w_{22}^1 = 0.$$
This means that some of the elements of matrices $W_k$ are coupled with each other and whatever one computes these elements the described above condition should be satisfied. To clarify this statement, let’s consider linear system

$$
x' = -y,\nonumber$$

$$y' = x.\nonumber$$

One can estimate numerically linear map $\bar{W}$. Solving equations (4) by Runge–Kutta 4th order method with constant step size 0.02 results the map $M(\Delta t = 0.1)$ in form of

$$W_1 = \begin{pmatrix} 0.99517016 & -0.0998484 \\ 0.0998484 & 0.99517016 \end{pmatrix}.$$ 

Since the integration step is quite big the resulting map $W_1$ is not precise enough. This leads to physical property violation. However, we can use described above symplectic condition for linear map and correct weights

$$\bar{W}_1 = \begin{pmatrix} 0.99517016 & -0.09816492 \\ 0.09816492 & 0.99517016 \end{pmatrix}.$$ 

Fig. 2 demonstrates the simulation results of this linear system. Note that both $W_1$ and $\bar{W}_1$ maps correspond to the same time interval $\Delta t = 0.1$. Weights of $\bar{W}_1$ are slightly corrected to $\det(\bar{W}_1) = 1$, while $\det(W_1) = 1.0003$.

Thus, having the system of differential equations one can build matrix Lie map up to the necessary order of nonlinearity. This map can replace step-by-step integration method and be used for simulation tasks. On the other hand, matrix Lie map can be represented as a polynomial neural network and be a model for learning dynamical systems from data.
4 Learning simple models

4.1 Simple Dynamical Systems Simulation

The proposed Lie transform based neural network can be used as a computer model for dynamical systems and a numerical approach for dynamics simulation. Having the differential equations one can build matrix Lie map and use it for dynamics calculation. As an example, we consider three well-known systems. The parameters of the equation are selected in an arbitrary way just for example.

| Lotka–Volterra equation | Van der Pol oscillator | Saddle Point |
|--------------------------|-------------------------|--------------|
| $x' = -y - xy$,           | $x'' = (1 - x^2)x' - x$ | $x' = 0.5x + 0.5y$, |
| $y' = x + xy$            | ($y = x'$, $y' = x''$)  | $y' = 1.5x - 0.5y$ |

Fig. 3. Phase space simulation by Lie based neural network (blue points) and traditional Runge–Kutta based method (red line) for Lotka–Volterra equation (left), Van der Pol oscillator (center), and Saddle point.

For all equations, we compare the simulation results (see Fig. 3) of both Lie transform of 3rd order of nonlinearity and traditional Runge–Kutta based integration. Lie map is built in according to the described above algorithms in 0.01 time interval. Saddle point represents a linear system

$$W_0 = 0; W_1 = \begin{pmatrix} 1.005 & 0.005 \\ 0.015 & 0.995 \end{pmatrix}; W_2 = 0; W_3 = 0.$$

The Lotka–Volterra system is presented by nonlinear equations. Therefore map will consist of higher order weight matrices. Van der Pol oscillator is also a nonlinear system that produces a limit cycle where trajectories in phase space spiral into it. The corresponded nonlinear Lie map approximately equals to

$$W_0 = 0; W_1 = \begin{pmatrix} 0.99995067 & 0.01004917 \\ -0.01004917 & 1.00099984 \end{pmatrix}; W_2 = 0; W_3 = \begin{pmatrix} 1.6 \cdot 10^{-7} & -4.9 \cdot 10^{-5} & -3.2 \cdot 10^{-7} & -7.9 \cdot 10^{-10} \\ 4.9 \cdot 10^{-5} & -1.0 \cdot 10^{-2} & -9.9 \cdot 10^{-5} & -3.3 \cdot 10^{-7} \end{pmatrix}.$$
4.2 Identification of Lotka–Volterra system

The simplest generalization property can be examined with a system of nonlinear oscillations. For instance, we considered differential equations:

\[
\begin{align*}
    x' &= -2x + xy, \\
    y' &= y - xy,
\end{align*}
\]

that describes the predator-prey population dynamics with nonlinear oscillations (see Fig. 4). Starting with an initial point \(X_0\) we calculated discrete states \(\{X_i\}_{i=1,n}\) by numerical integrating of the differential equation. Then the differential equation was not utilized in training procedure and identification process.

Fig. 4. Train and test data in phase space states for Lotka–Volterra system identification.

Fig. 5. Train and test data in the space states for Lotka–Volterra system identification.
According to the equations, we generated 4 different particular solutions that are presented as time series $X(t_i) = (x_i, y_i)$. As a training set, we used only one of them (blue line in the Fig. 4). For testing, three other solutions were selected, namely oscillation with higher amplitude, near linear oscillation, and fixed point. We constructed three neural network architectures: polynomial one based on 3rd order Lie transform, multilayer perceptron with sigmoid activation functions (MLP), and long short-term memory network (LSTM). As inputs and outputs for each neural network pairs $(X_i, X_{i+1})$ were used.

![Fig. 6. Results of system identification and simulation in unknown state space by Lie based neural network (left), MLP (center), and LSTM (right). Red points are predictions.](image)

After fitting on the training particular solution, both MLP and LSTM networks tend to learn the given track without any kind of generalization (see Fig. 6). At the same time proposed Lie based neural network can predict dynamics in both nonlinear area and near linear oscillation around stationary point. Moreover with the fitted matrix Lie map we can even estimate the stationary point simply solving numerically equation $X_{fixed} = M X_{fixed}$.

The point of this example is the demonstration of generalization property of Lie based neural network for physical data learning. Lie transform based polynomial neural network corresponds to a system of differential equations and recognizes the dynamical pattern from a single example. In the example, a Lie transform based neural network was fitted on the oscillation pattern, and tends to predict oscillations in other initial conditions. MLP and LSTM network require lots of data to correctly predict dynamics and tend to predict known solution from the training set.
5 Applications

5.1 Retail Price Identification

In the article [6] the authors investigate dynamics of iPhone and iPad sales with differential equations. They suggest analytical formulas for systems of nonlinear differential equations and fit parameters based on time-series data. This is a traditional approach for system identification. Using the described above technique one can identify dynamics utilizing matrix Lie transform without knowledge of appropriate equations at all.

For the example, one can generate data (two time series for iPad and iPhone sales) similar as in the article [6] and just fit these time series using 3rd order Lie based neural network. The built neural network can be used for the products specific market potential evaluation as well as other system parameters investigation. Note that in this case, we did not use any specific assumption on the possible view of equations as made in the article [6].

After fitting of neural network, one can receive a Lie map

\[
\begin{pmatrix}
    x \\
    y
\end{pmatrix} = \begin{pmatrix}
    3.1838244e - 05 \\
    3.8284583e - 03
\end{pmatrix} + \begin{pmatrix}
    0.6154481 & -0.36771438 \\
    -0.16011019 & 0.45261806
\end{pmatrix} \begin{pmatrix}
    x_0 \\
    y_0
\end{pmatrix} + \\
\begin{pmatrix}
    -0.36270088 & -0.2199135 & -0.2199135 & -1.2503444 \\
    -0.09446148 & -0.26765344 & -0.26765344 & -1.0930712
\end{pmatrix} \begin{pmatrix}
    x_0^2 \\
    x_0 y_0 \\
    y_0 x_0 \\
    y_0^2
\end{pmatrix} + \ldots ,
\]

where \( x_0, y_0 \) are initial sales for iPhone and iPad respectively. Note that during fitting coefficients with respect to the terms \( x_0 y_0 \) and \( y_0 x_0 \) are equal to each other since they have the same impact on optimization criteria. The built polynomial neural network is a white-box model of the dynamical systems. Moreover, it is possible to reconstruct differential equation that corresponds to this map using the inverse problem solving for the algorithm of map calculation.
5.2 Biochemical Reaction Simulation

In this example, we provide an analysis of a biochemical system that is described in [5] and represents the influence of the Raf Kinase Inhibitor Protein (RKIP). In the article, the influence of RKIP is investigated via numerical analysis of nonlinear ordinary differential equations using Matlab ode45 function that is based on step by step integration (see Fig. 8).

![Figure 8](image-url)

**Fig. 8.** Simulation of RKIP dynamics using Runge–Kutta based method. Figure is taken from [5].

Instead of using a step-by-step numerical integration method one can build a polynomial neural network and utilize it for system simulation. Moreover, based on this model one can further solve identification problem and clarify the form of equations based on measured data.

The described in the article system of differential equation consists of 11 nonlinear equations that describe the biochemical network. We built a 2nd order Lie map for this system and used it for simulation with initial condition from [5]. The results of the simulation are slightly different to [5] due to the application of different integration methods and orders of nonlinearity.

![Figure 9](image-url)

**Fig. 9.** Simulation of biochemistry reaction utilizing Lie transform based neural network of 2 order of nonlinearity.
5.3 Charged Particle Accelerators

Charged particle accelerator consists of a number of physical equipment (e.g. quadrupoles, bending magnets and others, see Fig. 10). Design of accelerators and nonlinear dynamics investigation require accurate computer model of such complicated system.

Each of the physical equipment can be described by a system of differential equation that has a complex nonlinear form. For instance, the equation of radial motion looks like

$$x'' = \frac{qH}{m_0\gamma v} \left( H(E_x - x'E_z) / v - (1 + x'^2)B_y + y'(x'B_x + B_z) \right),$$

where electromagnetic fields and particle state vector are incorporated.

For some problems, such as modeling of long-term dynamics, the traditional step-by-step integration methods are not suitable due to the performance limitation. Instead of solving differential equations directly one can estimate matrix Lie map for each physical element in an accelerator. Combining such maps consequently one can obtain a deep polynomial neural network representation of whole accelerator ring (see Fig. 11).

The proposed Lie transform based mapping approach was used, for example, for nonlinear dynamics investigation of spin-orbit dynamics simulation in EDM search project (see for example [14]). The articles [7, 8, 13] describe problem formulation and simulation results that are achieved with the application of the matrix Lie maps for simulation of the systems of nonlinear differential equations in accelerated physics.

6 Conclusion

In the article, we demonstrate how one can build neural network representation of dynamical systems. The greatest advantage of the proposed Lie transform based neural network is its equivalence to the differential equations. This neural network is not a black-box model since each weight matrix corresponds to the certain order of nonlinear effects in the real system and has a physical explanation. Another important issue is the ability of proposed architecture easy...
guarantee physical properties of the system by additional constraints on weights. Note also that if the form of the differential equation is known, one can compute weight matrices and utilize them as an initial point for optimization procedure in neural network fitting.

Neural network representation of Lie transform combines the benefit of deep learning techniques and traditional modeling theory for dynamical systems identification. It allows to avoid numerical iterative time-stepping methods and use mapping approach instead of them. As soon as the proposed neural architecture has a good coincidence with traditional modeling methods, it can be useful for investigation of dynamics in unknown parameter space. Since weight coefficients in the proposed neural architecture describe dynamical systems, they can be utilized as features for other machine learning models. For example, one can build matrix representation of Lie transform for time-series data in unsupervised mode and use weight coefficients for supervised methods.

The described above examples are implemented in Keras\Tensorflow using Python. The code can be found at GitHub repository [https://github.com/andiva/DeepLieNet](https://github.com/andiva/DeepLieNet). Directory `core` consists of Keras layer that implements matrix Lie transform up to the necessary order of nonlinearity, as well as an algorithm for matrix Lie map estimation based on a predefined system of differential equations. Directory `demo` contains realizations of described above examples.

References

1. Andrianov, S.: Order-by-order symplectification of truncated lie maps. Proc. of the 2001 Particle Accelerator Conference, Chicago, USA 6244/2010, 1787–1789 (2001)
2. Andrianov, S.: A role of symbolic computations in beam physics. Computer Algebra in Sc, Comp., Lecture Notes in Computer Science 6244/2010, 19–30 (2010)
3. Chen, S., Billings, S.A.: Neural networks for nonlinear dynamic system modelling and identification. International Journal of Control 56(2), 319–346 (1992)
4. Dragt, A.: Lie Methods for Nonlinear Dynamics with Applications to Accelerator Physics (2011), [inspirehep.net/record/955313/files/TUC28Nov2011.pdf](https://inspirehep.net/record/955313/files/TUC28Nov2011.pdf)
5. Gilbert, D., Heiner, M.: From petri nets to differential equations - an integrative approach for biochemical network analysis. Lecture Notes in Comp. Sc. pp. 181–200 (01 2006)
6. Guidolin, M., Guseo, R.: On product cannibalization, a new lotka-volterra model for asymmetric competition in the icts. Tech. rep. (2016), http://paduaresearch.cab.unipd.it/9738/1/GUIDOLINGUSEO_SETT2016.pdf
7. Ivanov, A., Andrianov, S., Senichev, Y.: Simulation of spin-orbit dynamics in storage rings. Journal of Physics: Conference Series 747(1) (2016)
8. Ivanov, A., Senichev, Y.: Matrix integration of odes for spin-orbit dynamics simulation. Proc. of the Particle Accelerator Conference pp. 400–402 (2014)
9. Lagaris, I.E., Likas, A., Fotiadis, D.I.: Artificial neural networks for solving ordinary and partial differential equations. Tech. rep. (1997), https://arxiv.org/pdf/physics/9705023.pdf
10. Lapedes, A., Farber, R.: Nonlinear signal processing using neural networks: Prediction and system modelling. In: Langley, P. (ed.) IEEE International Conference on Neural Networks. San Diego, CA, USA (1987)
11. Lewis, F.L., Ge, S.S.: Neural Networks in Feedback Control Systems, in Mechanical Engineers’ Handbook: Instrumentation, Systems, Controls, and MEMS, vol. 2. John Wiley and Sons, Hoboken, NJ, USA, third edition (ed m. kutz) edn. (2005)
12. Mall, S., Chakraverty, S.: Comparison of artificial neural network architecture in solving ordinary differential equations. Advances in Artif. Neural Syst. (2013)
13. Senichev, Y., Ivanov, A., Lehrach, A., Maier, R., Zyuzin, D., Andrianov, S.: Spin tune parametric resonance investigation. Proc. of the Particle Accelerator Conference pp. 3020–3022 (2014)
14. Senichev, Y., Lehrach, A., Maier, R., Zyuzin, D., Berz, M., Makino, K., S.Andrianov, Ivanov, A.: Storage ring edm simulation: methods and results. Proceedings of ICAP2012, Rostock, Germany pp. 99–103 (2012)