ISBN: 978-3-95450-184-7 doi:10.18429/JACoW-IPAC2018-THPAK088

MATRIX REPRESENTATION OF LIE TRANSFORM IN TensorFlow

A. Ivanov*, S. Andrianov, N. Kulabukhova, A. Sholokhova, E. Krushinevskii, E. Sboeva,
St. Petersburg State University, St. Petersburg, Russia

Abstract

In the article, we propose an implementation of the matrix representation of Lie transform using TensorFlow as a computational engine. TensorFlow allows easy description of deep neural networks and provides automatic code execution on both single CPU/GPU and cluster architectures. In this research, we demonstrate the connection of the matrix Lie transform with polynomial neural networks. The architecture of the neural network is described and realized in code. In terms of beam dynamics, the proposed technique provides a tool for both simulation and analysis of experimental results using modern machine learning techniques. As a simulation technique one operates with a nonlinear map up to the necessary order of nonlinearity. On the other hand, one can utilize TensorFlow engine to run map optimization and system identification problems.

INTRODUCTION

Charged particle accelerator consists of a number of physical equipment (e.g. quadrupoles, bending magnets and others, see Fig. 1). 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 is:

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

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 nonlinear matrix map for each physical element in an accelerator.

NONLINEAR MATRIX MAP FOR SOLVING OF SYSTEMS OF ODES

The dynamics of charged particles in elect systems that can be described by nonlinear ordinary differential equations:

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

where t is independent variable, $\mathbf{X} \in R^n$ is state vector. There is an assumption that function F can be expanded in Taylor series with respect to the components of $\mathbf{X}$. Note that independent variable t can arise in the equation as an arbitrary nonlinear function.

* 05x.andrey@gmail.com

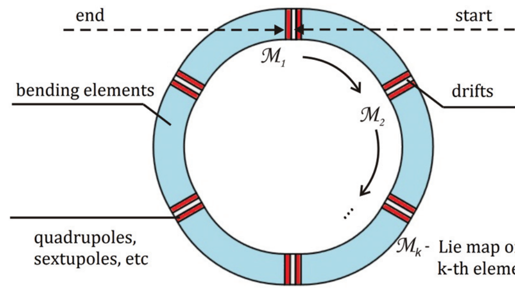

Figure 1: Schematic map based representation of circular accelerator.

In the articles [1] the mathematical models that can be utilized for dynamics description are presented. For instance, spin-orbit dynamics is described by Newton–Lorentz and T–BMT equations in curvilinear coordinate system. The coordinates corresponds to the design orbit and can be written as a state vector $\mathbf{X} = (x,\ y,\ t, p_x/p_0,\ p_y/p_0,\ \delta W,\ S_x,\ S_y,\ S_z,\ L)$, where x and y are transverse and vertical offsets of a particle, t is physical time of motion, p_x, p_y are transverse and vertical components of momentum, p_0 is the momentum, δW is energy deviation, $\mathbf{S} = (S_x,\ S_y,\ S_s)$ is vector of spin, and L is the length of a trajectory.

Matrix Representation of Lie Map

The dynamics of vector $\mathbf{X}$ in (1) can be presented in the form of a 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 as a series:

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

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

$$R'_{ik}(t|t_0) = \sum_{j=i}^{k} P_{ij}(t)R_{jk}(t|t_0),\ 1 \le i < k,$$

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

ISBN: 978-3-95450-184-7 doi:10.18429/JACoW-IPAC2018-THPAK088

Numerical Map Estimation

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

$$\mathbf{X}(t) = W_0(t) + W_1(t)\mathbf{X}_0 + W_2(t)\mathbf{X}_0^{[2]} + \cdots + W_k(t)\mathbf{X}_0^{[k]}, \quad (3)$$

is by utilizing an appropriate numerical step-by-step integration method. Taking derivative of the $\mathbf{X}(t)$ with respect to the (2) one can obtain a system of equations:

$$\begin{aligned}
\frac{d}{dt}\mathbf{X} &= \frac{d}{dt}W_0(t) + \ldots + \frac{d}{dt}W_k(t)\mathbf{X}_0^{[k]},\\
\frac{d}{dt}\mathbf{X} &= P_0(t) + P_1(t)\mathbf{X} + P_2(t)\mathbf{X}^{[2]} + \ldots + P_p(t)\mathbf{X}^{[p]}\\
&= P_0(t) +\\
&\quad P_1(t)\left(W_0(t) + W_1(t)\mathbf{X}_0 + \cdots + W_k(t)\mathbf{X}_0^{[k]}\right) +\\
&\quad P_2(t)\left(W_0(t) + W_1(t)\mathbf{X}_0 + \cdots + W_k(t)\mathbf{X}_0^{[k]}\right)^{[2]} +\\
&\quad \cdots +\\
&\quad P_p(t)\left(W_0(t) + W_1(t)\mathbf{X}_0 + \cdots + W_k(t)\mathbf{X}_0^{[k]}\right)^{[p]},
\end{aligned}$$

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

$$\frac{d}{dt}W_k(t) = \sum_{i=1}^{p} P_i(t)\frac{\partial \mathbf{X}^{[i]}}{\partial (\mathbf{X}_0^{[i]})^T}, \qquad k = 0, 1, 2, \ldots. \quad (4)$$

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.

Invariant 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 $\mathcal{M}(t|t0)$

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

where

$$J_0 = \begin{pmatrix} 0 & E \\ -E & 0 \end{pmatrix}$$

and $M(t|t_0) = \partial\left(\mathcal{M}(t|t_0) \circ \mathbf{X}_0\right)/\partial\mathbf{X}_0^T$.

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

$$\mathbf{X} = W_1\mathbf{X}_0 + W_2\mathbf{X}_0^{[2]} = \begin{pmatrix} w_{11}^1 & w_{12}^1 \\ w_{21}^1 & w_{22}^1 \end{pmatrix}\begin{pmatrix} x_0 \\ y_0 \end{pmatrix} + \begin{pmatrix} w_{11}^2 & w_{12}^2 & w_{13}^2 \\ w_{21}^2 & w_{22}^2 & w_{23}^2 \end{pmatrix}\begin{pmatrix} x_0^2 \\ x_0y_0 \\ y_0^2 \end{pmatrix},$$

the describe above symplectic conditions are:

$$\begin{aligned}
&w_{11}^1w_{22}^1 - w_{12}^1w_{21}^1 = 1,\\
&w_{11}^1w_{22}^2 - w_{21}^1w_{12}^2 + 2w_{22}^1w_{11}^2 - 2w_{12}^1w_{21}^2 = 0,\\
&w_{22}^1w_{12}^2 - w_{12}^1w_{22}^2 + 2w_{11}^1w_{23}^2 - 2w_{21}^1w_{13}^2 = 0,\\
&w_{11}^2w_{23}^2 - w_{13}^2w_{21}^2 = 0, w_{12}^2w_{23}^2 - w_{13}^2w_{22}^2 = 0.
\end{aligned}$$

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.

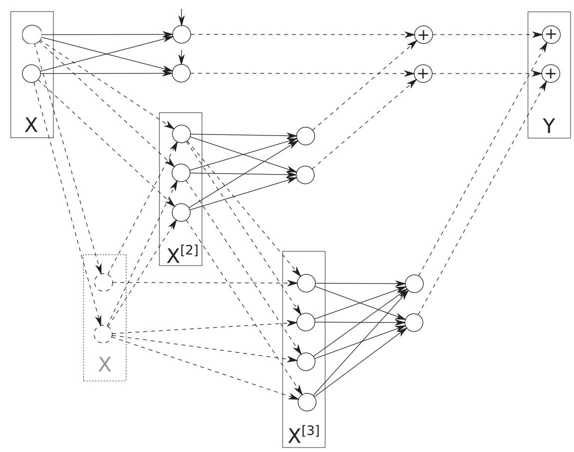

Figure 2: Polynomial neural network for 3rd order matrix map.

PROPOSED NEURAL NETWORK

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

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

where $\mathbf{X}, \mathbf{Y} \in R^n$, W_i are weights matrices, and $\mathbf{X}^{[k]}$ means k-th Kroneker power of vector $\mathbf{X}$. For instance Fig. 2 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 $\mathbf{X}^{[2]} = (x_1^2, x_1x_2, x_2^2)$ and $\mathbf{X}^{[3]} = (x_1^3, x_1^2x_2, x_1x_2^2, x_2^3)$ 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, for example

$$\mathbf{X}^{[2]} = (x_1^2, x_1x_2, x_2x_1, x_2^2) \to (x_1^2, x_1x_2, x_2^2).$$

The transformation $\mathcal{M}$ can be considered as an approximation of evolution operator of the (1) for predefined initial time

ISBN: 978-3-95450-184-7 doi:10.18429/JACoW-IPAC2018-THPAK088

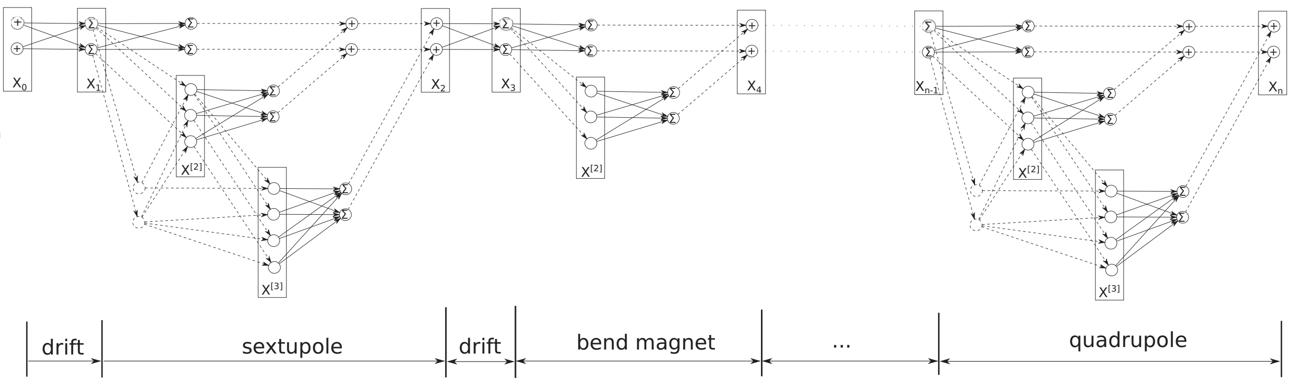

Figure 3: Lie transform based neural network for circular accelerator representation.

t_0 and time interval Δt. 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 Δt can be approximately calculated as $\mathbf{Y} = \mathbf{X}(t_0, \mathbf{X}_0, \Delta t,) = \mathcal{M} \circ \mathbf{X}_0$. If the system (1) is time independent then weights W_i are constant for a predefined time interval.

IMPLEMENTATION IN TensorFlow

The described above method was implemented on Keras API with TensorFlow backend. TensorFlow [7] is an open source software library for high-performance numerical computation. Its flexible architecture allows easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices. Keras [8] is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano.

The described above map $\mathcal{M}$ was implemented as user-defined Layer in Keras. This allows building neural networks with Lie transform based architecture and utilize optimization and computational techniques that are already implemented in TensorFlow.

Note that given technique allows both building maps for an arbitrary system of nonlinear differential equations, and solve identification problems in case of unknown equations.

CONCLUSION

The proposed technique allows building a map for each element in an accelerator.Combining such maps consequently one can obtain a polynomial neural network representation of whole accelerator ring (see Fig. 3).

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 [3]). The articles [4–6] 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.

The described above method are implemented in Keras Tensorflow using Python. The code can be found at GitHub repository `https://github.com/andiva/DeepLieNet`. Directory core consists of both Keras layer that implements matrix Lie transform up to the necessary order of nonlinearity, and algorithm for matrix Lie map estimation based on a predefined system of differential equations. Directory demo contains realizations of demo examples, like simple FODO structure modeling, as well as the application of described techniques in other areas like retail and biochemistry.

REFERENCES

[1] A. Ivanov and N. Kulabukhova, "An IDE for spin-orbit dynamics simulation", Proceedings of IPAC2013, China, 2013. pp. 921–923.

[2] S. Andrianov, "A role of symbolic computations in beam physics", Computer Algebra in Sc, Comp., Lecture Notes in Computer Science 6244/2010, 2010. pp. 19–30.

[3] Y. Senichev *et al.*, "Storage ring edm simulation: methods and results", Proceedings of ICAP2012, Germany, 2012. pp. 99–103.

[4] A. Ivanov, S. Andrianov, and Y. Senichev, "Simulation of spin-orbit dynamics in storage rings", Journal of Physics: Conference Series 747(1), 2016.

[5] A. Ivanov and Y. Senichev, "Matrix integration of ODEs for spin-orbit dynamics simulation", Proc. of IPAC2014, Germany, 2014. pp. 400–402.

[6] Y. Senichev, A. Ivanov, A. Lehrach, R. Maier, D. Zyuzin, and S. Andrianov, "Spin tune parametric resonance investigation", Proc. of IPAC2014, Germany, 2014. pp. 3020–3022.

[7] An open source machine learning framework. `https://www.tensorflow.org`

[8] Keras: The Python deep learning library. `https://keras.io`