ISBN: 978-3-95450-214-1 ISSN: 2673-5490 doi:10.18429/JACoW-IPAC2021-THPAB191

PHYSICS-ENHANCED REINFORCEMENT LEARNING FOR OPTIMAL CONTROL[*]

A. Ivanov[†], I. Agapov, A. Eichler, S. Tomin
Deutsches Elektronen Synchrotron DESY, Hamburg, Germany

Abstract

We propose an approach for incorporating accelerator physics models into reinforcement learning agents. The proposed approach is based on the Taylor mapping technique for the simulation of particle dynamics. The resulting computational graph is represented as a polynomial neural network and embedded into the traditional reinforcement learning agents. The application of the model is demonstrated in a nonlinear simulation model of beam transmission. The comparison of the approach with the traditional numerical optimization as well as neural networks-based agents demonstrates better convergence of the proposed technique.

INTRODUCTION

Traditionally, dynamical system behavior is represented by physics-based modeling, which relies on the theory of differential equations. Such models do not depend on data and perform well regardless of inputs. At the same time, exact models are difficult to build for real systems because of their complexity. This compels us to employ equations in approximate forms to meet scale-accuracy trade-offs, thus inducing a gap between the real and predicted dynamics.

Increasing interest in machine learning (ML) brought a bunch of methods to solve this problem [1–6]. Usually, authors propose either ML-based fitting of the parameters of ordinary (ODEs) or partial differential equations (PDEs), or substituting the equation-based models by surrogate neural networks.

To take into account the physics behind the black-box model, some authors suggest incorporating domain knowledge in the neural networks (NNs) or provide additional loss functions for the physical inconsistency of the predictions [7, 8]. Other approaches rely on the implementation of a traditional step-by-step integrating method in a NN framework [9].

The most relevant to our research is paper [10], where authors introduce the connection between polynomial ODEs and polynomial neural networks (PNN). Further, the PNNs were widely highlighted in the literature [11–13].

In opposite to these works, we do not solve or fit ODEs of interest with an NN-based learning procedure. We suggest a deterministic algorithm to translate ODEs to PNN without training. In [14] we demonstrated that if the dynamics of the system follow approximately given ODE, the Taylor mapping approach allows us to calculate weights of the PNN (TM-PNN).

In the paper [15], the proposed TM-PNN architecture is validated in practice for control one of the largest in the world X-ray source, PETRAIII [16]. The TM-PNN is initialized from the ODEs that describe the particle motion in a magnetic field. The resulting NN consists of 1519 hidden polynomial layers with unique weights that are fine-tuned with only one trajectory of the real system.

In this research, we present results of incorporating TM-PNN into a reinforcement learning (RL) agent, see Fig. 1. The simulation is based on the simplified but strongly nonlinear lattice described in the following section.

SIMULATION ENVIRONMENT

For simplicity, we consider particle motion in the horizontal plane for the lattice (d, c1, d, sf, d, qf, ap1, d, d, c2, d, sd, d, qd, ap2, d, d, m). There are two horizontal correctors c1 and c2 that are used as actuators to transfer the beam. It is assumed that each magnet and aperture have random misalignments in the horizontal plane. The full description of lattice elements is provided in Table 1.

Table 1: Lattice Specifications

Label	Element	Parameters
d	Drift	length=1
sf	Sextupole	length=0.2, k2=3000
sd	Sextupole	length=0.2, k2=-3000
qf	Quadrupole	length=0.2, k1=1, k2=20
qd	Quadrupole	length=0.2, k1=-1, k2=-20
ap1,ap2	Aperture	xmax=0.005

The lattice is implemented on OCELOT simulation software [17]. Due to the high nonlinearities, the considered system has a high sensitivity to misalignments of the magnets (see Fig. 2). Given the random errors in magnet placement, the response of the systems depends on the two control variables (horizontal correctors) and is calculated as transmitted beam rate ($\mathcal{T}$). In this formulation, the considered problem can be defined as a 2-dimensional single output optimization task:

$$\mathcal{T} = \mathcal{T}(c_1, c_2, errors) \rightarrow max, \tag{1}$$

where c_1, c_2 are strengths of the horizontal correctors.

NUMERICAL EXPERIMENTS

In the research, we used two different approaches for solving the optimization problem. The first one is numerical optimization. The second one reinforcement learning agents with two architectures (see Fig. 1). The first one directly

[*] Work supported by The Innovation Pool project AMALEA of the Helmholtz association of Germany
[†] andrei.ivanov@desy.de

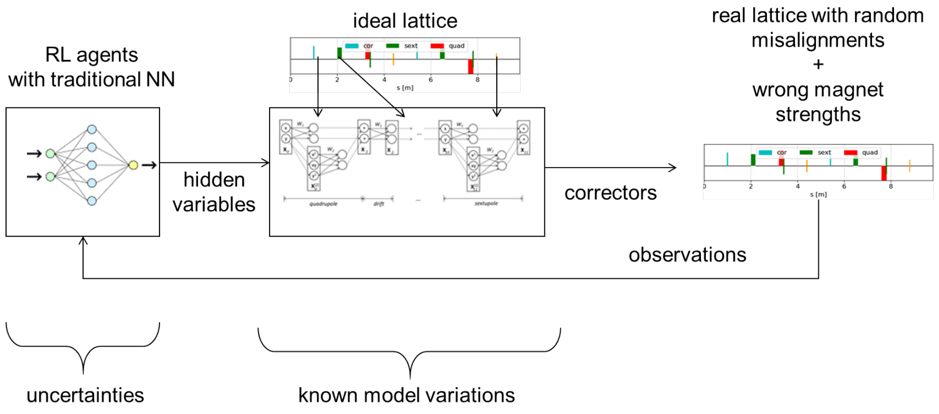

Figure 1: Reinforcement learning agent joint with the physics-based polynomial neural network.

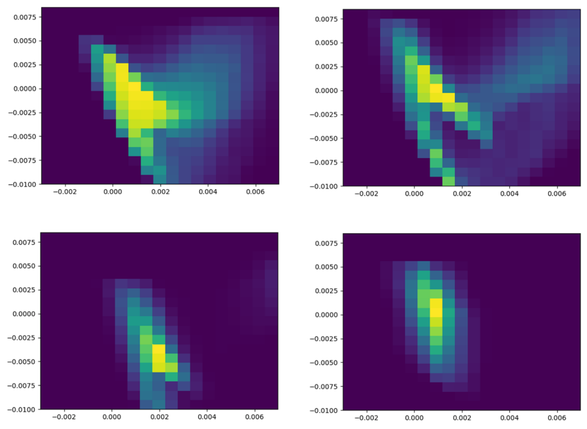

Figure 2: Response matrix of the transmission model for different random misalignments.

predicts the optimal control variables, while the second one use combination of NN with TM-PNN generated from an ideal lattice.

Numerical Optimization

For benchmarking, we use traditional optimization methods. For simplicity, we use Powell's conjugate direction method to optimize function from Eq. (1). The implementation taken from scipy python package [18] requires 40 iteration steps on average. This means that given the random errors in magnets placements and starting at random initial point $(c1, c2)$ the optimal transmission rate can be achieved after 40 measurements. Since the goal of our research is to find out an optimal strategy that can outperform traditional techniques, we use this step limit as the stopping condition for reinforcement learning algorithms.

Model-free Reinforcement Learning

For implementing an optimal control policy we used model from [19]. To be compatible with its interface, we wrapped the optimization problem from Eq. (1) with Open-AI GYM interface for RL environments.

The following models, algorithms, and hyperparameters were tested during training the model:

- Models: TD3, DQN.
- Policies: MlpPolicy, CnnPolicy, FeedForwardPolicy, LnMlpPolicy.
- Hyperparameters: learning rate, gamma.

During each training epochs, the neural network inside the RL agent tries to predict the optimal control to maximize the transmission rate. After 40 iterations, the environment is reset, i.e. new random errors are applied for magnets. Unfortunately, we did not achieve convergence for the considered task. Figure 3 presents the result of training for the approach of model-free prediction.

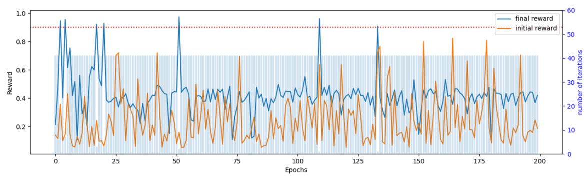

Figure 3: Training of the model-free RL agents: orange line for initial transmission rate and a blue one for the final result.

Physics-enhanced Reinforcement Learning

To introduce additional physics knowledge to RL agents, we propose to combine black-box NN and TM-PNN architecture. Using the Taylor mapping techniques, the ideal lattice without random misalignments is transformed to polynomial NN that is introduced as an intermediate layer between RL agent and environment (see Fig. 4). In this architecture, the RL agent does not predict control but rather reconstruct hidden variables of the system. This approach can be also considered as a tuning of the TM-PNN with the RL agent.

Note also that such a model does not tune every particular displacement in the lattice. Instead, it tunes all weights of the TM-PNN in a data-driven way. After training, the average number of iteration to achieve a threshold of 0.9 for the transmission rate is equal to 5. This significantly outperforms the traditional numerical optimization, though requires additional training with the given environment.

ISBN: 978-3-95450-214-1 ISSN: 2673-5490 doi:10.18429/JACoW-IPAC2021-THPAB191

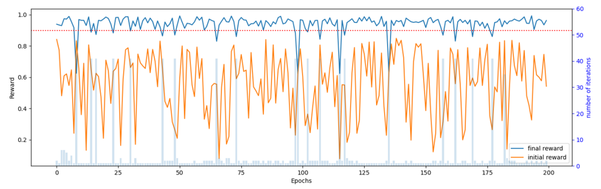

Figure 4: Training results of the RL agents combined with TM-PNN.

CODE

The implementation of the TM-PNN architecture in Keras/TensorFlow for the considered lattice and the described examples are available under `https://github.com/andiva/TM-PNN`

CONCLUSION

In our experiments, model-free RL agents do not perform well. This can be explained by the high sensitivity of the chosen lattice to the random error distribution. For a new set of random errors, the behavior of the system significantly differs from the previous one.

Introducing an intermediate level with certain knowledge about the system increases the convergence of the optimization approach. In the proposed model, the RL agents play the role of a generalization technique that does not require knowledge of the parameters of the system but rather helps to perform a data-driven tuning of the model.

REFERENCES

[1] N. Mohajerin and S. L. Waslander, "Multistep prediction of dynamic systems with recurrent neural networks", *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, pp. 3370-3383, 2019. doi:10.1109/TNNLS.2019.2891257

[2] X. Jia *et al.*, "Physics guided RNNs for modeling dynamical systems: A case study in simulating lake temperature profiles", in *Proc. 2019 SIAM International Conference on Data Mining*, Calgary, Canada, May 2019, pp. 558-566. doi:10.1137/1.9781611975673.63

[3] K. Bieker *et al.*, "Deep model predictive flow control with limited sensor data and online learning", *Theor. Comput. Fluid Dyn.*, vol. 34, pp. 577–591, 2020. doi:10.1007/s00162-020-00520-4

[4] R. Yu, S. Zheng, A. Anandkumar, and Y. Yue, "Long-term forecasting using higher order tensor RNNs", 2019. arxiv:1711.00073

[5] A. Nagabandi, G. Kahn, R. Fearing, and S. Levine, "Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning", 2017. arxiv:1708.02596

[6] Y. Chen, Y. Shi, and B. Zhang, "Optimal control via neural networks: A convex approach", 2019. arxiv:1805.11835

[7] A. Alvaro Sanchez-Gonzalez, V. Bapst, K. Cranmer, and P. Battaglia, "Hamiltonian graph networks with ode integrators", 2019. arxiv:1909.12790

[8] A. Karpatne, W. Watkins, J. Read, and V. Kumar, "Physics-guided neural networks (pgnn): An application in lake temperature modeling", 2018. arxiv:1710.11431

[9] A. A. Anastassi, "Constructing runge–kutta methods with the use of artificial neural networks", *Neural Comp. Appl.*, vol. 25, pp. 229-236, 2014. doi:10.1007/s00521-013-1476-x

[10] V. Lopez, R. Huerta, and J. R. Dorronsoro, "Reccurent and feedforward polynomial modeling of coupled time series", *Neural Comp.*, vol. 5, pp. 795-811, 1993. doi:10.1162/neco.1993.5.5.795

[11] L. Zjavka, "Differential polynomial neural network", *J. Artif. Intell.*, vol. 4, pp. 89-99, 2011. doi:10.3923/jai.2011.89.99

[12] Y. Yang, M. Hou, and J. Luo, "A novel improved extreme learning machine algorithm in solving ordinary differential equations by legendre neural net-work methods", *Adv. Differ. Equations*, vol. 4, p. 469, 2018. doi:doi:10.1186/s13662-018-1927-x

[13] V. Schetinin, "Polynomial neural networks learnt to classify EEG signals", 1997. arxiv:0504058

[14] A. Ivanov, A. Golovkina, and U. Iben, "Polynomial neural networks and taylor maps for dynamical systems simulation and learning", 2020. arXiv:1912.09986

[15] A. Ivanov and I. Agapov, "Physics-based deep neural networks for beam dynamics in charged particle accelerators", *Phys. Rev. Accel. Beams*, vol. 23, p. 074601, 2020. doi:10.1103/physrevaccelbeams.23.074601

[16] K. Balewski *et al.*, "PETRA III: A low emittance synchrotron radiation source", DESY, Hamburg, Germany, Rep. DESY-04-035, 2004.

[17] I. Agapov, G. Geloni, S. Tomin, and I. Zagorodnov, "OCELOT: A software framework for synchrotron lightsource and FEL studies", *Nucl. Instrum. Meth.*, vol. 768, pp. 151-156, 2014. doi:10.1016/j.nima.2014.09.057

[18] P. Virtanen *et al.*, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python", *Nat. Methods*, vol. 17, pp. 261-272, 2020. doi:10.1038/s41592-019-0686-2

[19] A. Hill *et al.*, Stable Baselines, https://github.com/hill-a/stable-baselines