ML Enhanced Orbit Correction in Particle Accelerators

Andrei Ivanov
Hamburg, 03.02.2020

Overwiew

01 Problem Formulation

- Lattice imperfection and orbit correction
- Traditional methods

02 ML-based orbit correction

- Taylor maps for ODEs
- From Taylor maps to neural networks
- Regularization when learning dynamics with small datasets

03 Next steps

Problem Formulation

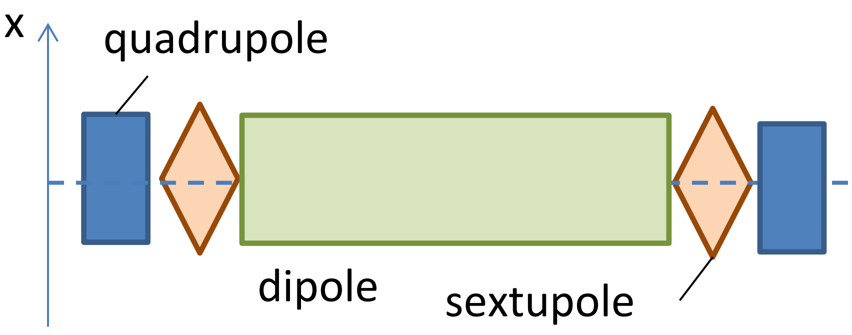
The part of the PETRA IV lattice is used in simulation:
34 magnets (166 elements), 11 beam position monitors and 10 corrector magnets

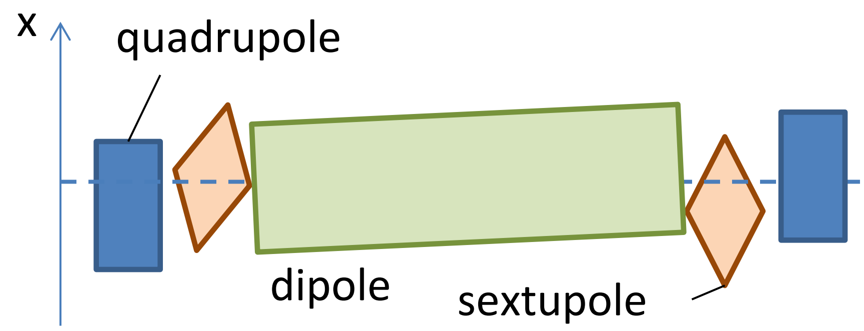
OCELOT framework is used for the dynamics simulation

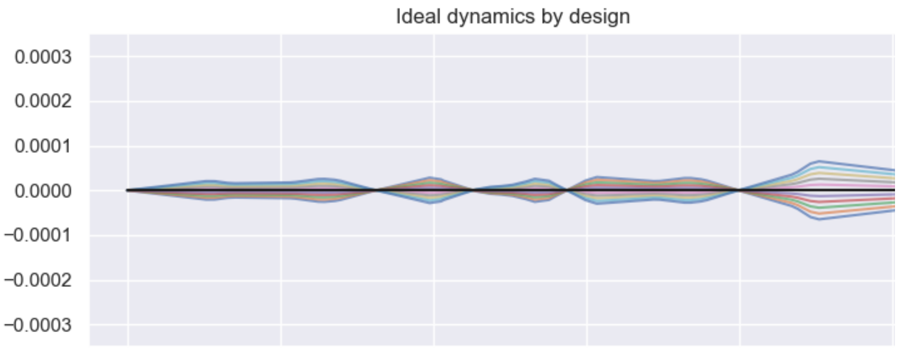
To simulate lattice imperfection, the random misalignments of magnets are generated

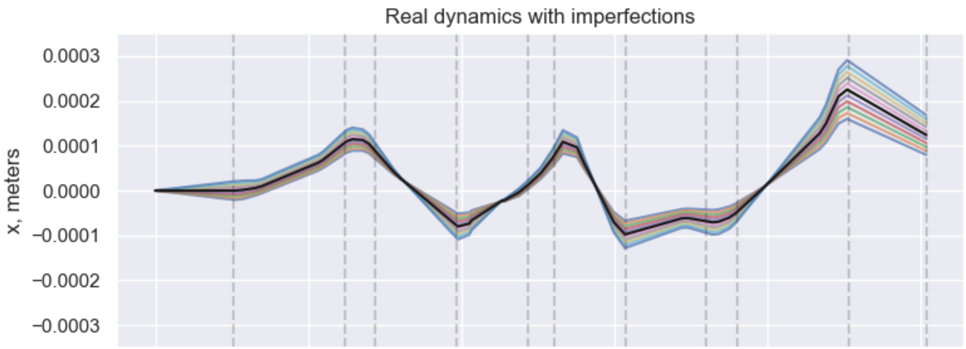
Particles moves along centers by design

but the actual orbit has deviations because of lattice imperfections



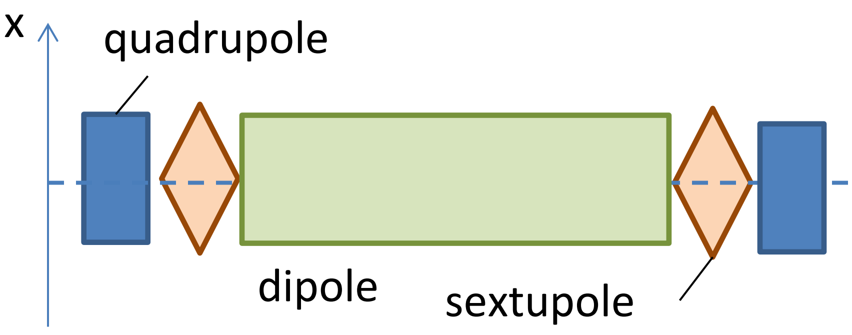


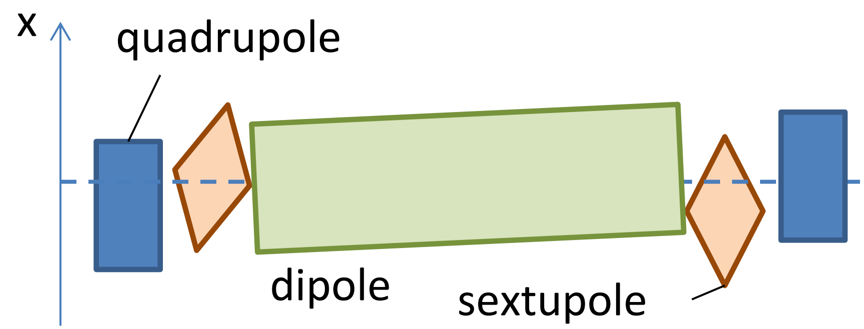


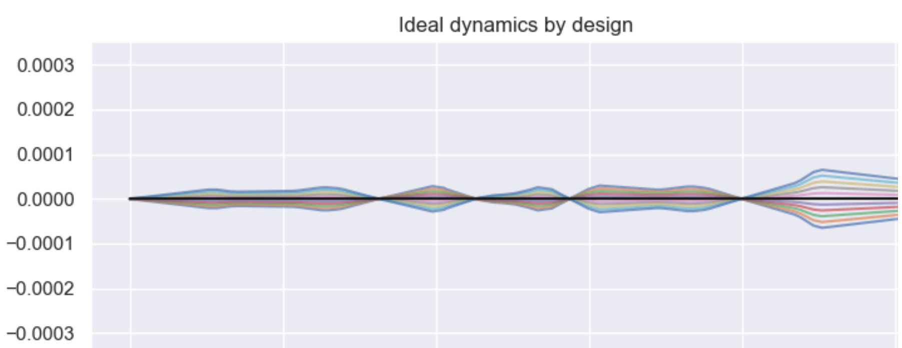


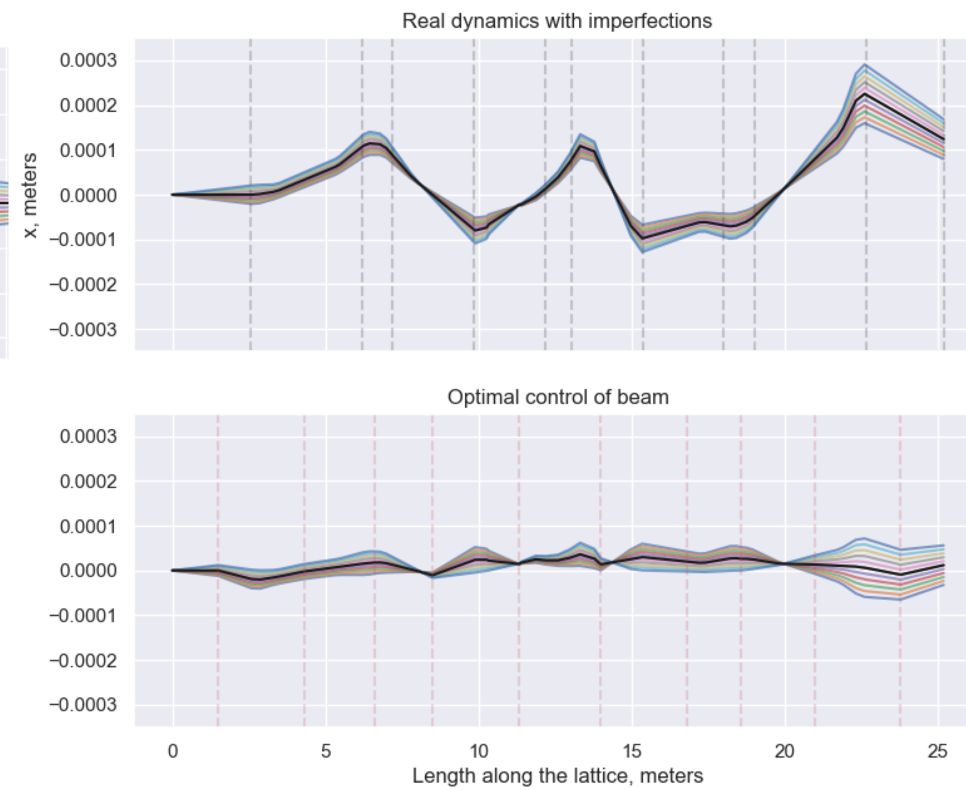
Particles moves along centers by design

but the actual orbit has deviations because of lattice imperfections









beam position monitors (BPMs)

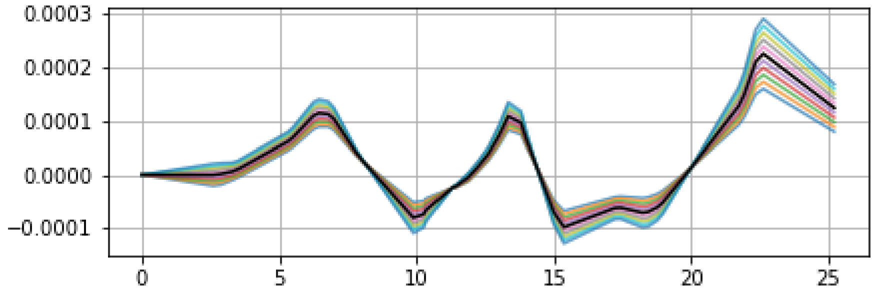
corrector magnets

Traditional methods in orbit correction

Coordinate descent or SVD

$$F = \sum_{i=0}^{10} ||x_i(c_0, c_1, \ldots, c_9)|| \to 0$$

where x_i is beam coordinate measured at i-th monitor, c_j is the strength of j-th corrector magnet.

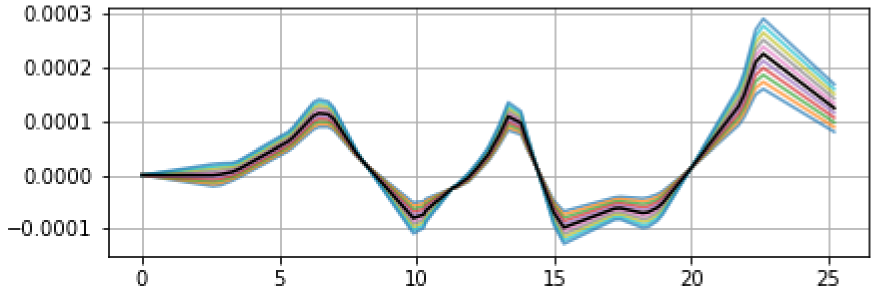


Traditional methods in orbit correction

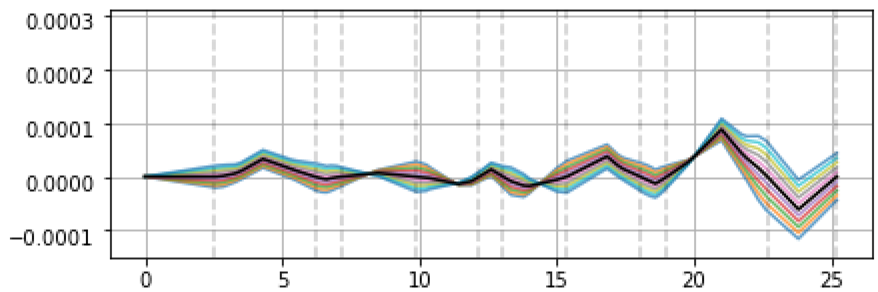
Coordinate descent or SVD

$$F = \sum_{i=0}^{10} ||x_i(c_0, c_1, \ldots, c_9)|| \to 0$$

where x_i is beam coordinate measured at i-th monitor, c_j is the strength of j-th corrector magnet.



coordinate descent: consequent varying of the correctors

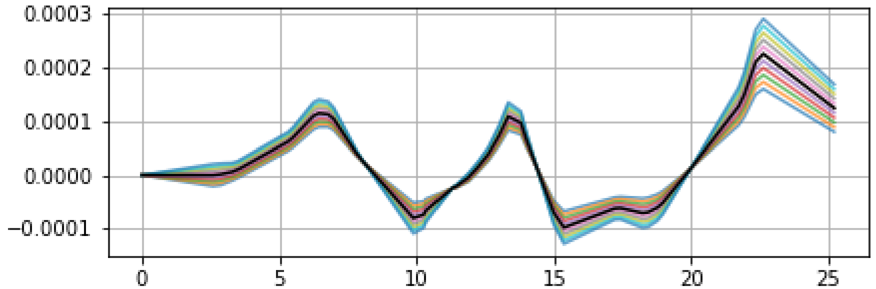


Traditional methods in orbit correction

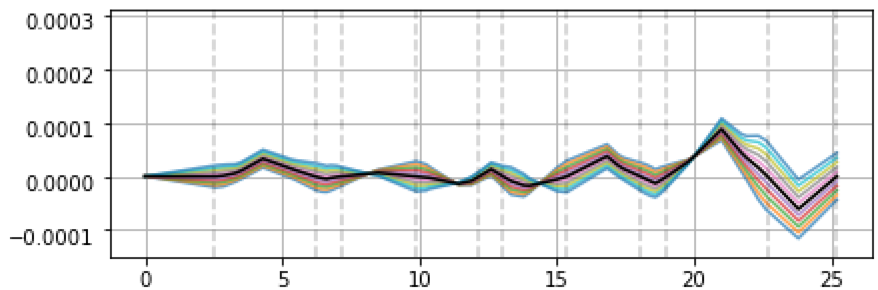
Coordinate descent or SVD

$$F = \sum_{i=0}^{10} ||x_i(c_0, c_1, \ldots, c_9)|| \to 0$$

where x_i is beam coordinate measured at i-th monitor, c_j is the strength of j-th corrector magnet.

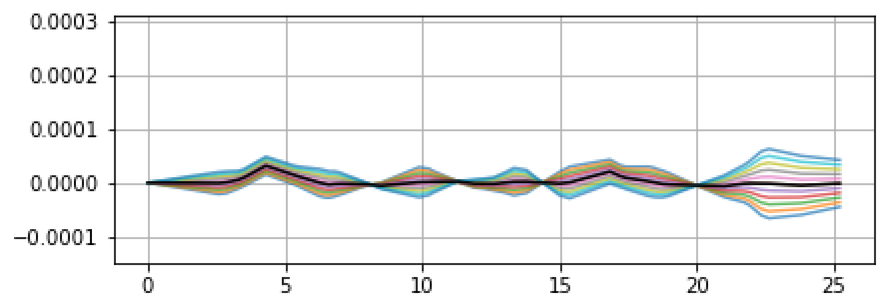


coordinate descent: consequent varying of the correctors



SVD for pseudoinverse

$$M\begin{pmatrix} C_1 \\ \cdots \\ C_{10} \end{pmatrix} = \begin{pmatrix} \Delta x_1 \\ \cdots \\ \Delta x_{11} \end{pmatrix}$$



The limitations of the traditional methods

1. **Coordinate descent is time-consuming procedure, may lead to local bumps in orbit, requires expertise**

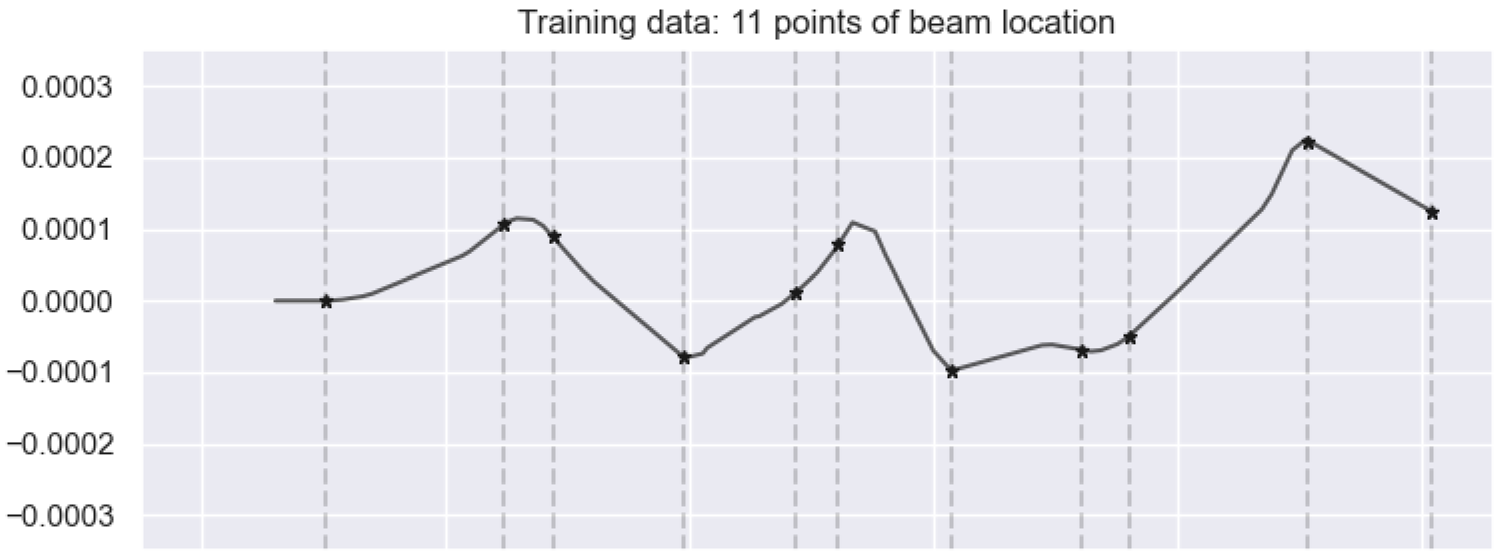
2. **SVD assumes linear dependencies between correctors and BPMs**

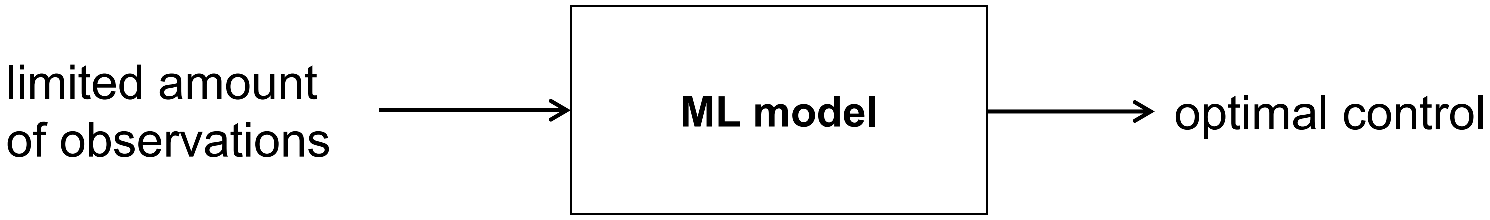
The main goal is to develop ML-based models that improve existing solutions

ML Enhanced Orbit Correction

ML methods are data-driven approaches for

recovering dynamics with misalignments from limited observations





ML methods are data-driven approaches for

recovering dynamics with misalignments from limited observations

1. A naive method is to parametrize equations of motion and tune parameters

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

ML methods are data-driven approaches for

recovering dynamics with misalignments from limited observations

1. A naive method is to parametrize equations of motion and tune parameters

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

2. Another solution is replacement of real data with simulated and training predictive models (RL-agents)

ML methods are data-driven approaches for

recovering dynamics with misalignments from limited observations

1. A naive method is to parametrize equations of motion and tune parameters

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

2. Another solution is replacement of real data with simulated and training predictive models (RL-agents)

3. Our approach is **combination of ODE-based dynamics and data-driven fitting with small data**

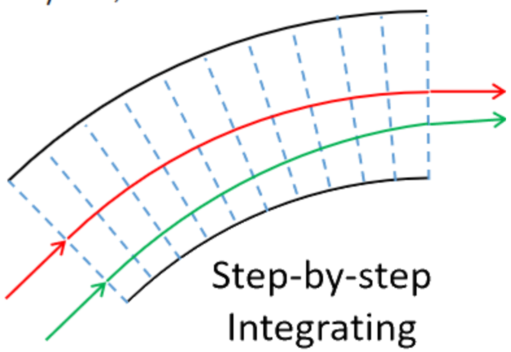
Taylor maps for ODEs

Instead of numerical solving of differential equations one can use maps

Example: equation of motion in a bending element

$$x' = y,$$
$$y' = -2x + x^2/R,$$

step-by-step integration
(Runge-Kutta solvers)
requires ~ 30 steps



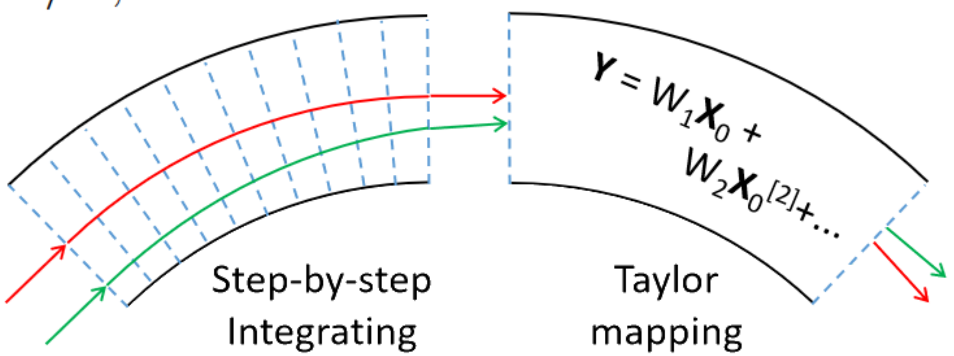
Taylor maps for ODEs

Instead of numerical solving of differential equations one can use maps

Example: equation of motion in a bending element

$$x' = y,$$
$$y' = -2x + x^2/R,$$

step-by-step integration (Runge-Kutta solvers) requires ~ 30 steps



Taylor map allows to compute particle coordinates at the end of the element in 1 step

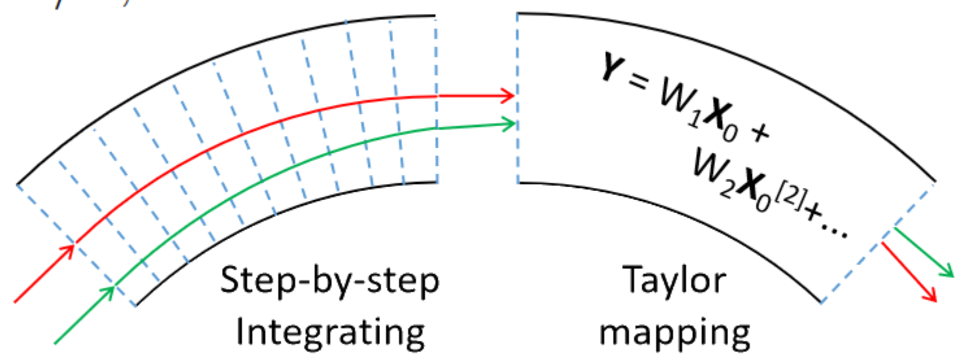
Taylor maps for ODEs

Instead of numerical solving of differential equations one can use maps

Example: equation of motion in a bending element

$$x' = y,$$
$$y' = -2x + x^2/R,$$

step-by-step integration (Runge-Kutta solvers) requires ~ 30 steps

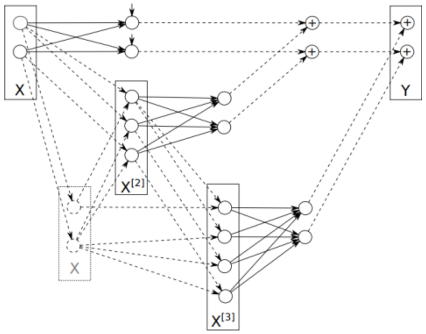


Taylor map allows to compute particle coordinates at the end of the element in 1 step

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0.44 & 0.63 \\ -0.13 \cdot 10 & 0.44 \end{pmatrix} \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} +$$
$$\begin{pmatrix} 0.23 \cdot 10^{-1} & 0.12 \cdot 10^{-1} & 0.26 \cdot 10^{-2} \\ 0.40 \cdot 10^{-1} & 0.35 \cdot 10^{-1} & 0.12 \cdot 10^{-1} \end{pmatrix} \begin{pmatrix} x_0^2 \\ x_0 y_0 \\ y_0^2 \end{pmatrix} +$$
$$\begin{pmatrix} 0.21 \cdot 10^{-3} & 0.17 \cdot 10^{-3} & 0.47 \cdot 10^{-4} & 0.56 \cdot 10^{-5} \\ 0.83 \cdot 10^{-3} & 0.95 \cdot 10^{-3} & 0.32 \cdot 10^{-3} & 0.47 \cdot 10^{-4} \end{pmatrix} \begin{pmatrix} x_0^3 \\ x_0^2 y_0 \\ x_0 y_0^2 \\ y_0^3 \end{pmatrix}$$

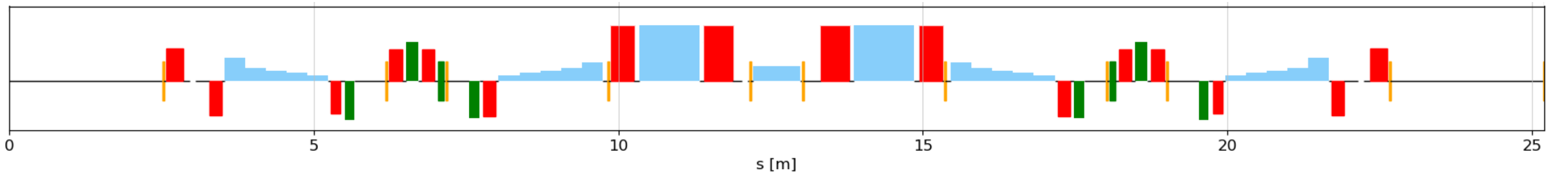
From Taylor maps to Neural Networks

Taylor map as a polynomial neuron



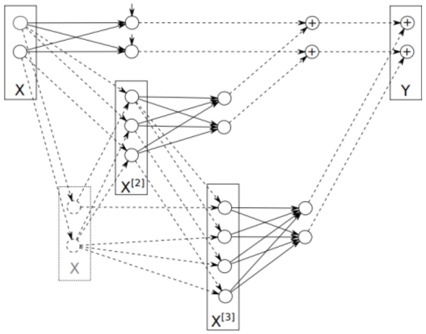
$$\mathbf{X}(t_1) = W_0 + W_1\,\mathbf{X}_0 + W_2\,\mathbf{X}_0^{[2]} + \ldots + W_k\,\mathbf{X}_0^{[k]}$$

Instead of tuning ODEs one can fit weights directly



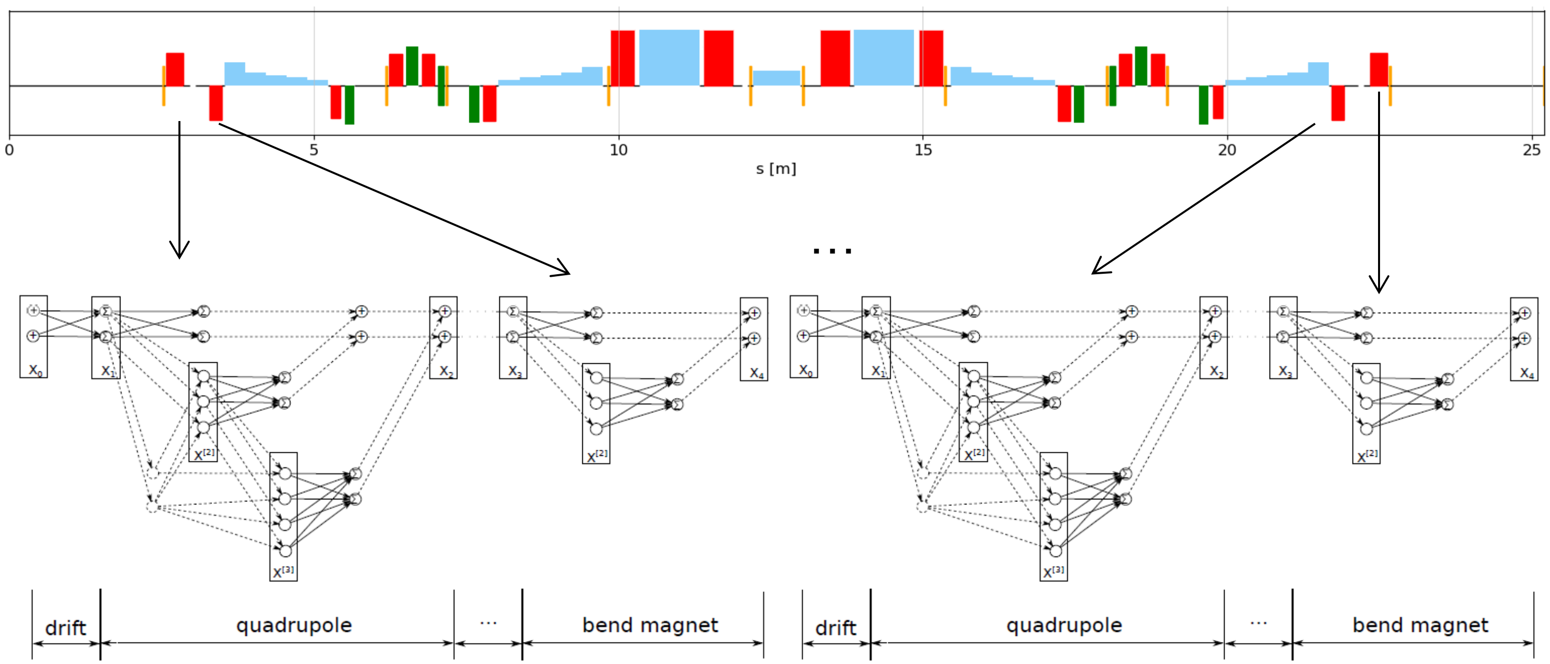
From Taylor maps to Neural Networks

Taylor map as a polynomial neuron



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

Instead of tuning ODEs one can fit weights directly



Regularization

Standart L1 or L2 regularization terms don't work

Since the Taylor maps consist of weights that are varied by several order of magnitudes, it makes impossible to use L1L2 regularization. It attempts to simply reduce weights magnitude *norm(W_i).*

$$
\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0.44 & \boxed{0.63} \\ -0.13 \cdot 10 & 0.44 \end{pmatrix} \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} +
$$

$$
\begin{pmatrix} 0.23 \cdot 10^{-1} & 0.12 \cdot 10^{-1} & 0.26 \cdot 10^{-2} \\ 0.40 \cdot 10^{-1} & 0.35 \cdot 10^{-1} & 0.12 \cdot 10^{-1} \end{pmatrix} \begin{pmatrix} x_0^2 \\ x_0 y_0 \\ y_0^2 \end{pmatrix} +
$$

$$
\begin{pmatrix} 0.21 \cdot 10^{-3} & 0.17 \cdot 10^{-3} & 0.47 \cdot 10^{-4} & \boxed{0.56 \cdot 10^{-5}} \\ 0.83 \cdot 10^{-3} & 0.95 \cdot 10^{-3} & 0.32 \cdot 10^{-3} & 0.47 \cdot 10^{-4} \end{pmatrix} \begin{pmatrix} x_0^3 \\ x_0^2 y_0 \\ x_0 y_0^2 \\ y_0^3 \end{pmatrix}
$$

Symplectic regularization

preserves physical properties of trained model

Symplecticity is the property of the Hamiltonian systems

$$\mathcal{M} : \mathbf{X}_0 \to \mathbf{X}$$

that can be written in form of

$$\left(\frac{\partial \mathbf{X}}{\partial \mathbf{X}_0}\right)^T J \, \frac{\partial \mathbf{X}}{\partial \mathbf{X}_0} - J = 0, \ \forall \mathbf{X}_0, \quad J = \begin{pmatrix} 0 & I \\ -I & 0 \end{pmatrix},$$

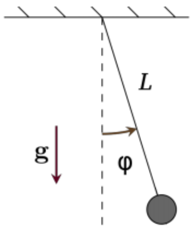
For example, for the second-order Taylor map it results

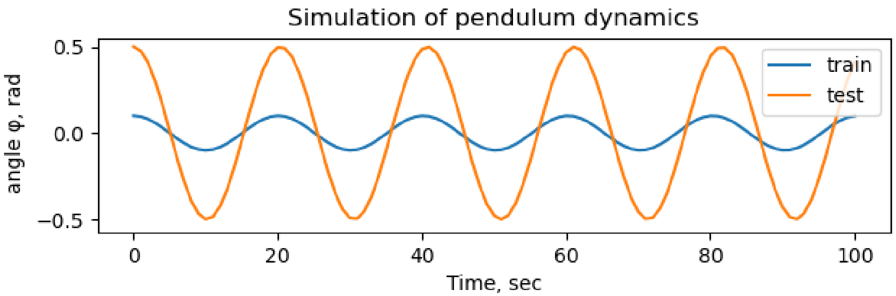
$$\begin{aligned}
&w_1^{11} w_1^{22} - w_1^{12} w_1^{21} - 1 = 0, \\
&w_1^{11} w_2^{22} - w_1^{21} w_2^{12} + 2 w_1^{22} w_2^{11} - 2 w_1^{12} w_2^{21} = 0, \\
&w_1^{22} w_2^{12} - w_1^{12} w_2^{22} + 2 w_1^{11} w_2^{23} - 2 w_1^{21} w_2^{13} = 0, \\
&w_2^{11} w_2^{23} - w_2^{13} w_2^{21} = 0, \ w_2^{12} w_2^{23} - w_2^{13} w_2^{22} = 0.
\end{aligned}$$

Symplectic regularization

preserves physical properties of trained model

Example: mathematical pendulum



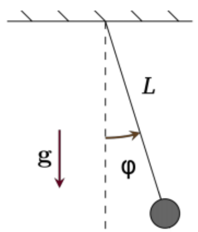


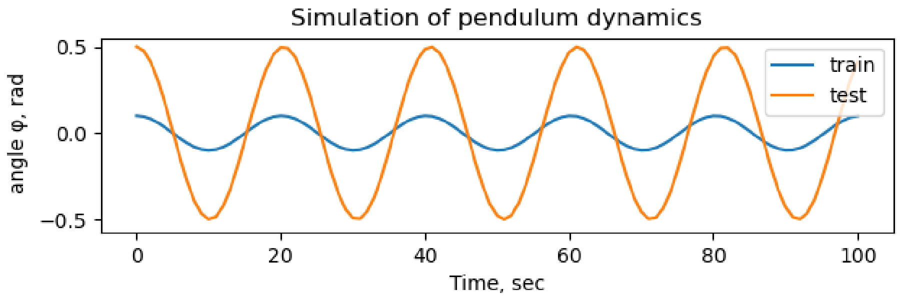
Model: second-order Taylor map: $$\begin{pmatrix}\phi_{i+1}\\ \phi'_{i+1}\end{pmatrix} = W_1\begin{pmatrix}\phi_i\\ \phi'_i\end{pmatrix} + W_2\begin{pmatrix}\phi_i^2\\ \phi_i\phi'_i\\ {\phi'_i}^2\end{pmatrix}$$

Symplectic regularization

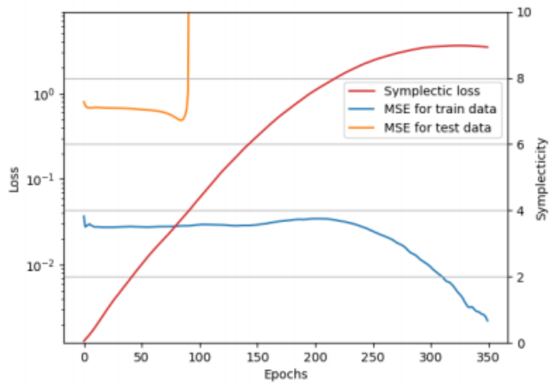
preserves physical properties of trained model

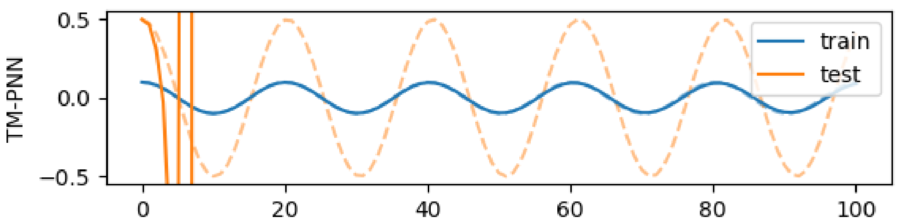
Example: mathematical pendulum





Model: second-order Taylor map: $$\begin{pmatrix}\phi_{i+1}\\ \phi'_{i+1}\end{pmatrix} = W_1 \begin{pmatrix}\phi_i\\ \phi'_i\end{pmatrix} + W_2 \begin{pmatrix}\phi_i^2\\ \phi_i\phi'_i\\ {\phi'_i}^2\end{pmatrix}$$



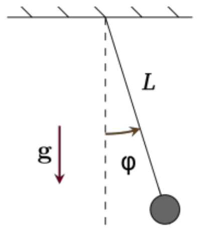


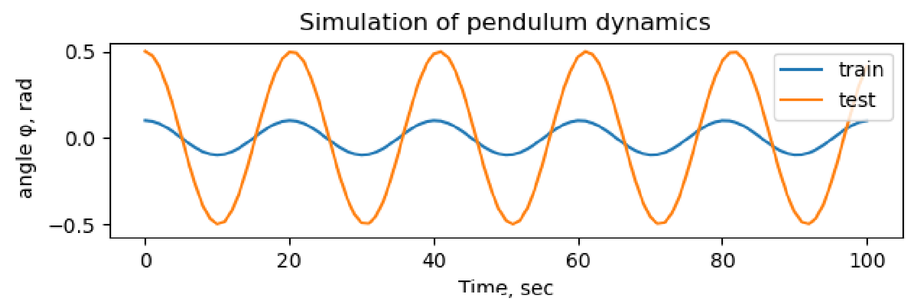
Training without symplectic regularization

Symplectic regularization

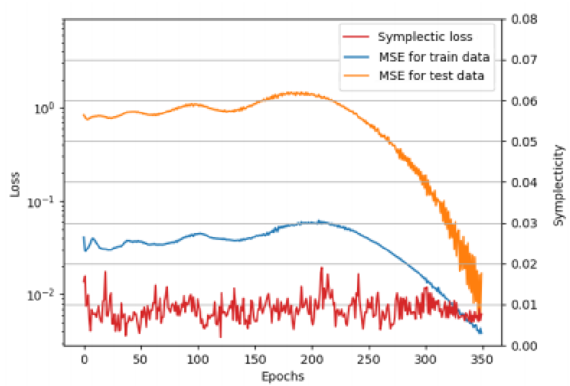
preserves physical properties of trained model

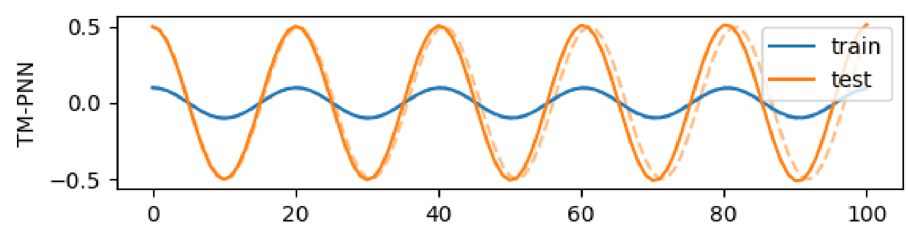
Example: mathematical pendulum





Model: second-order Taylor map: $$\begin{pmatrix}\phi_{i+1}\\ \phi'_{i+1}\end{pmatrix} = W_1 \begin{pmatrix}\phi_i\\ \phi'_i\end{pmatrix} + W_2 \begin{pmatrix}\phi_i^2\\ \phi_i\phi'_i\\ \phi_i'^2\end{pmatrix}$$



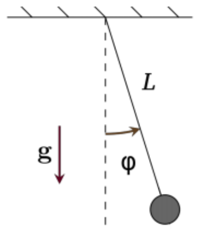


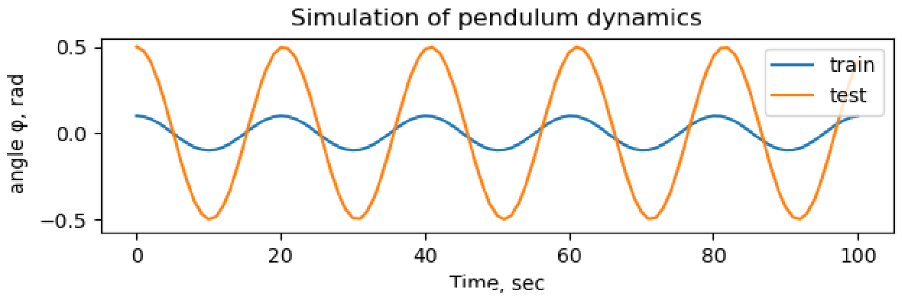
Training with symplectic regularization

Symplectic regularization

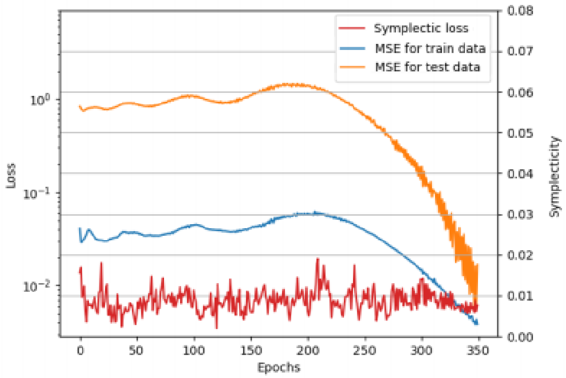
preserves physical properties of trained model

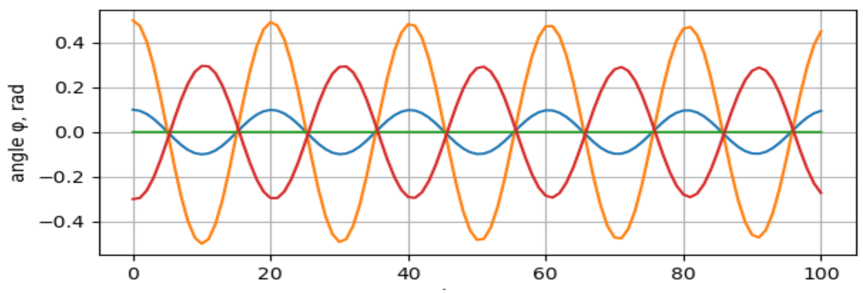
Example: mathematical pendulum





Model: second-order Taylor map: $$\begin{pmatrix}\phi_{i+1}\\ \phi'_{i+1}\end{pmatrix} = W_1\begin{pmatrix}\phi_i\\ \phi'_i\end{pmatrix} + W_2\begin{pmatrix}\phi_i^2\\ \phi_i\phi'_i\\ \phi_i'^2\end{pmatrix}$$



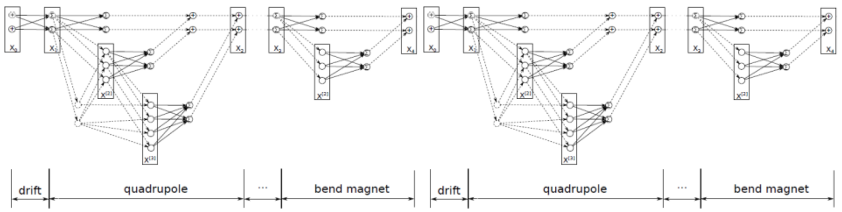


Training with symplectic regularization

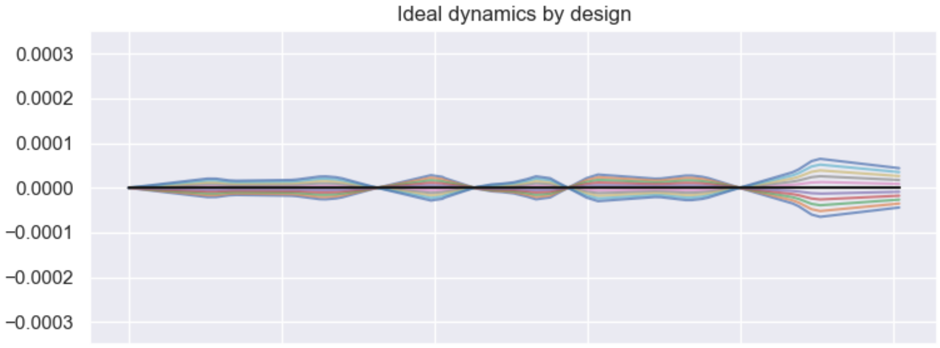
Training

with limited observation

1. Use mapping to initialize weights of polynomial NN. We extracted weights from OCELOT framework.



2. After Step 1 the constructed NN has 166 layers and accurately represents particle dynamics in the ideal lattice without misalignments



Training

with limited observation

3. Generate training data (simulation of dynamics with random generated misalignments)

4. Define 11 outputs of the NN where BPMs are located and fit all 166 layers with symplectic regularization

$$Loss = MSE(x, x_{train}) + \lambda S(W_1, W_2)$$

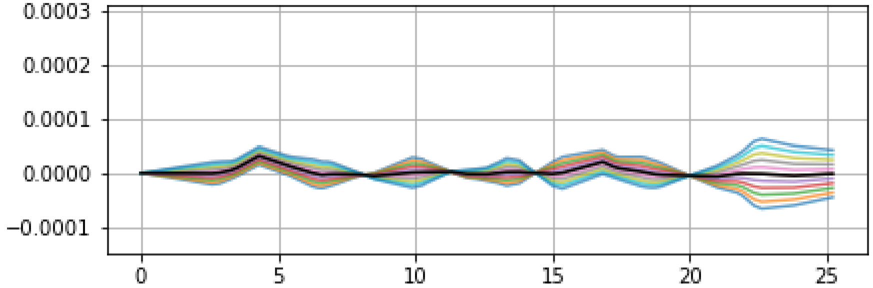
5. After training, the NN recovers imperfect dynamics in lattice

Optimal control

trained NN can be used for both optimal control and simulation

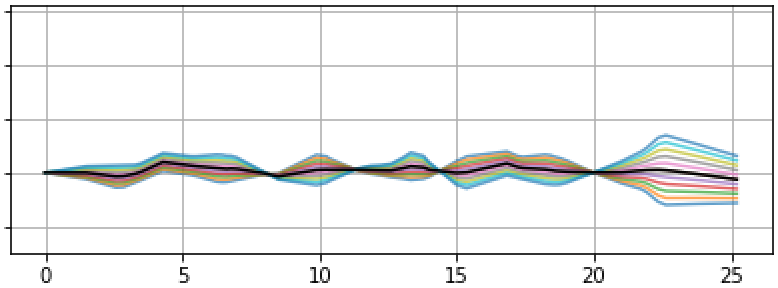
6. Since the trained NN preserves physical properties
   - Initialize weights from ODEs
   - Symplectic regularization

the model can be used for solving optimal control problems by varying parameters



SVD-based orbit correction

Only linear dependency between correctors and BPMs



NN-based orbit correction

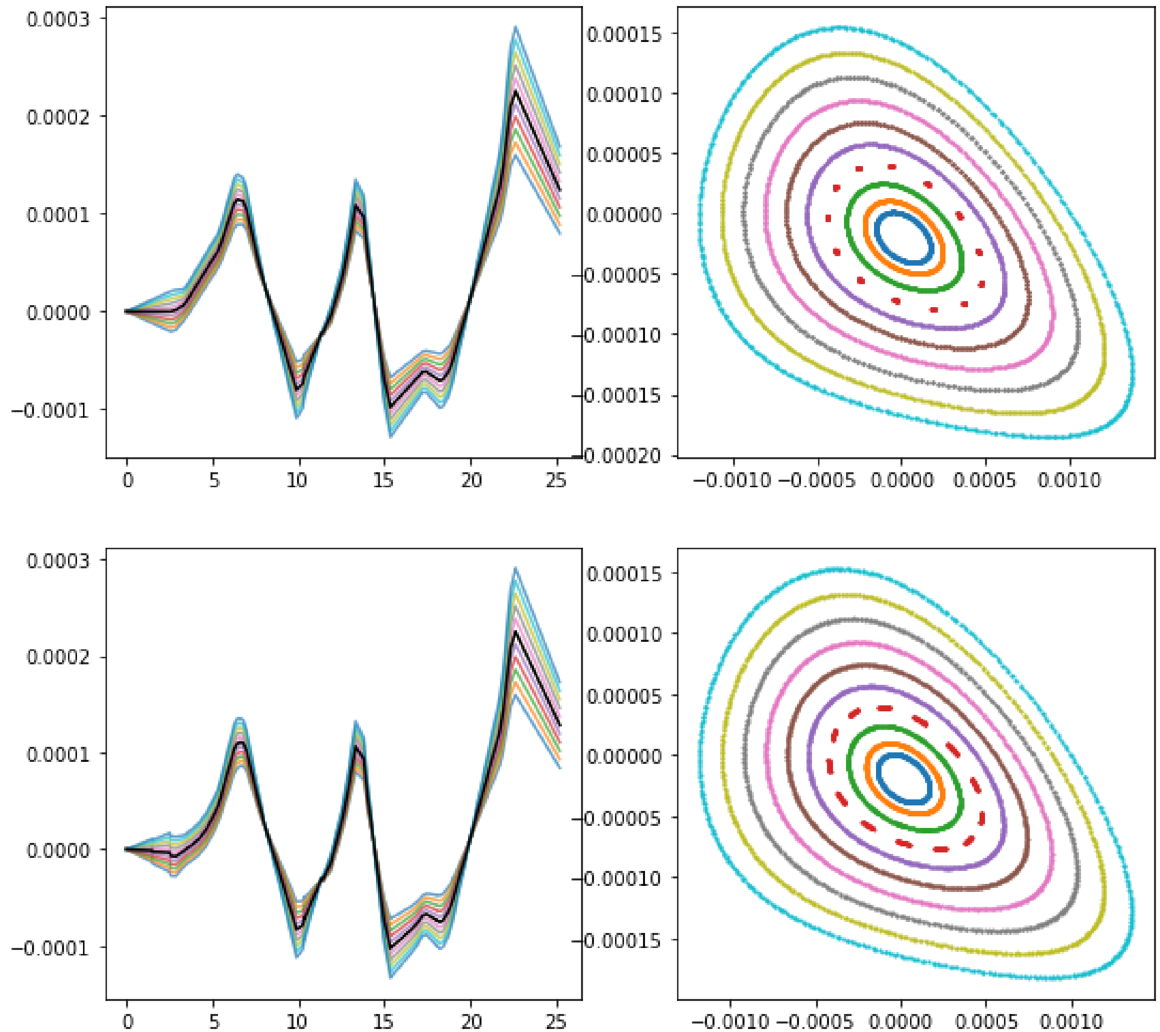
Nonlinear dynamics along complete lattice

Recovering of nonlinear dynamics

trained NN can be used for both optimal control and simulation

Lattice with known misalignments

Trained model



Results

Taylor map-based NN tuning along with the symplectic regularization allows to recover dynamics with limited observations

Various application: orbit and optics correction, optimizers

ECAI 2020

European Conference on Artificial Intelligence

Ivanov, A. et al. Polynomial Neural Networks and Taylor maps for Dynamical Systems Simulation and Learning (Full-paper, oral presentation)

Next steps

Ocelot Gym

OCELOT: open source project for beam dynamics simulation of the whole machine in modern electron-based x-ray sources

Gym environment + RL

Work with real data

in terms of data acquisition and interfaces to be able to introduce implemented models into existing control environment

Thank you

Contact

DESY. Deutsches
Elektronen-Synchrotron

www.desy.de

Andrei Ivanov
MPY
andrei.ivanov@desy.de