Physics-Based Neural Networks for Particle Accelerators

Intelligent Process Control Seminar

Andrei Ivanov
Hamburg, 09.02.2021

Overview

01 Physics-inspired neural networks

- Related works and proposed neural architecture (TM-PNN)
- Detailed example of translating ODE for pendulum oscillation into TM-PNN

02 Training

- From scratch: a general-purpose regression method for deterministic systems
- With ODE-based weights initialization

03 Application in particle accelerators

- Simulation of beam dynamics
- Data-driven model calibration (PETRAIII experiments)
- RL-enhanced control (simulated environment)

01 Physics-inspired neural networks

- Related works and proposed neural architecture (TM-PNN)
- Detailed example of translating ODE for pendulum oscillation into TM-PNN

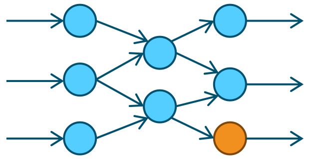
Physics-inspired neural networks

several methods to incorporate physical knowledge into predictive model exist

- **surrogate models:** train a black-box model with simulated data

$$\frac{d}{dt}\mathbf{X} = f(t,\, \mathbf{X}) \xrightarrow{\text{simulation}} \boxed{\text{training dataset}}$$

black-box

$\mathbf{X}_{input}$  $\mathbf{Y}_{pred}(\mathbf{W})$

Physics-inspired neural networks

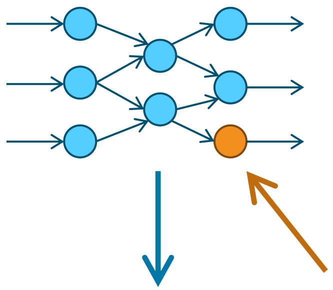
several methods to incorporate physical knowledge into predictive model exist

- **surrogate models:** train a black-box model with simulated data

$$\frac{d}{dt}\mathbf{X} = f(t,\, \mathbf{X}) \xrightarrow[\text{simulation}]{} \boxed{\text{training dataset}}$$

black-box

$\mathbf{X}_{\boldsymbol{input}}$



$\mathbf{Y}_{\boldsymbol{pred}}(\mathbf{W})$

- **parametrize** equation with NN or, oppositely, **include** equation into NN

$$\frac{d}{dt}\mathbf{X} = f(t,\, \mathbf{X},\, \boldsymbol{NN}) \qquad \frac{d}{dt}\mathbf{X} = f(t,\, \mathbf{X})$$

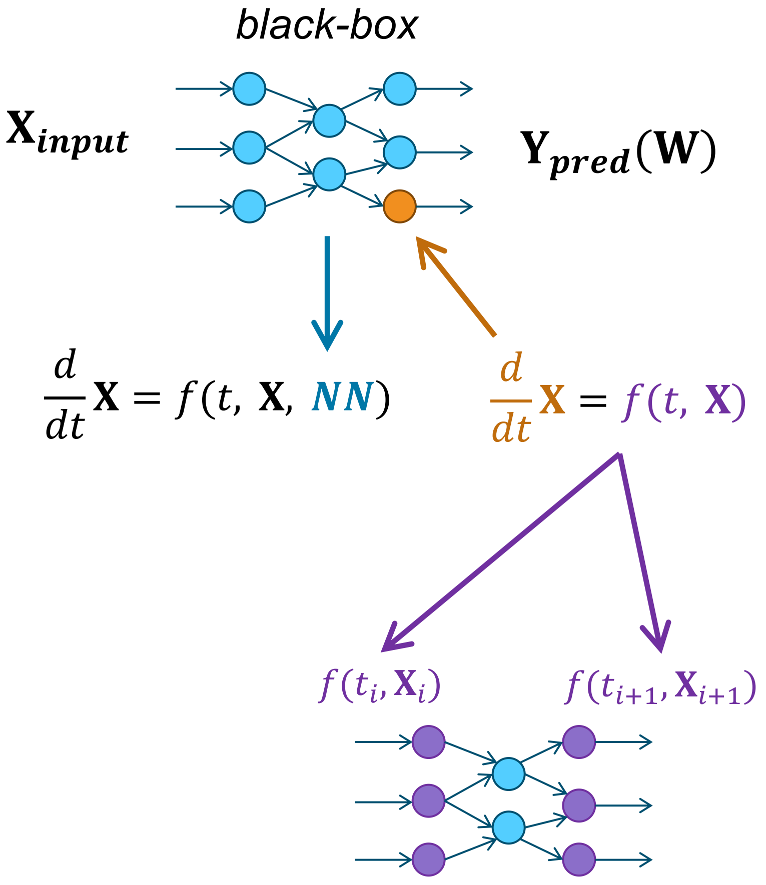
e.g. Hamiltonian Neural Networks

Physics-inspired neural networks

several methods to incorporate physical knowledge into predictive model exist

- **surrogate models:** train a black-box model with simulated data

$$\frac{d}{dt}\mathbf{X} = f(t,\, \mathbf{X}) \xrightarrow{\text{simulation}} \boxed{\text{training dataset}}$$



- **parametrize** equation with NN or, oppositely, **include** equation into NN

  e.g. Hamiltonian Neural Networks

- **implement a numerical scheme** in NN basis

  e.g. Neural ODE (requires numerical solvers)

Novel approach for constructing deep neural networks for beam dynamics

with the following key features:

- accurate simulation of dynamics **without training**
- model **fine-tuning** with limited measurements

ECAI 2020

oral presentation at the European Conference on Artificial Intelligence

Novel approach for constructing deep neural networks for beam dynamics

with the following key features:

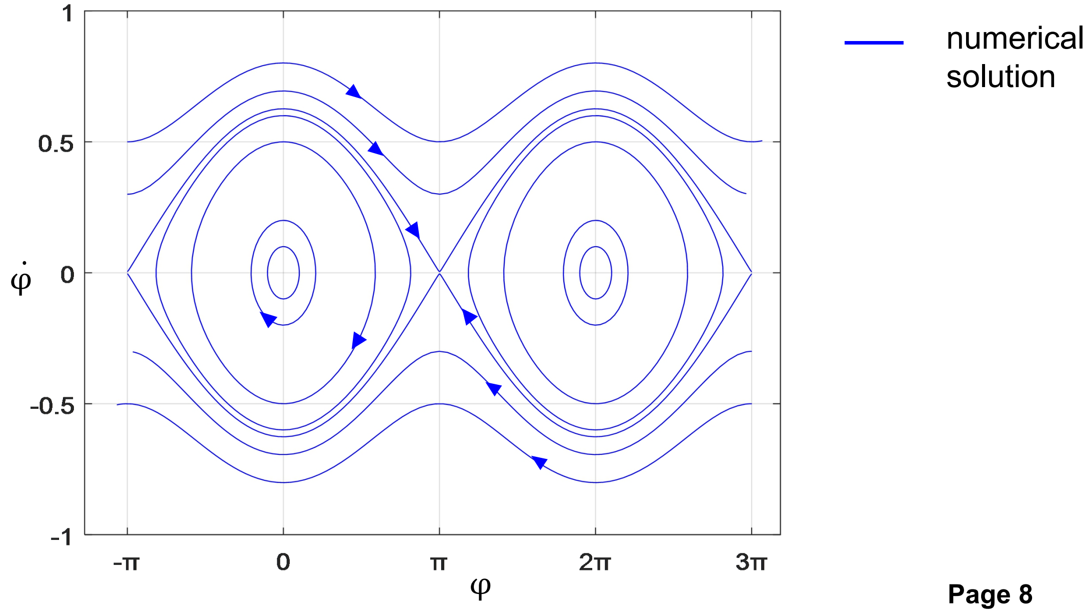
- accurate simulation of dynamics **without training**
- model **fine-tuning** with limited measurements

ECAI 2020 Santiago de Compostela

oral presentation at the European Conference on Artificial Intelligence

The key idea: If the dynamics of a system approximately follows a given differential equation, the Taylor mapping technique can be used to initialize the weights of a polynomial neural network

Pendulum oscillation: $\ddot{\varphi} = -\omega^2 \sin\varphi$



Novel approach for constructing deep neural networks for beam dynamics

with the following key features:

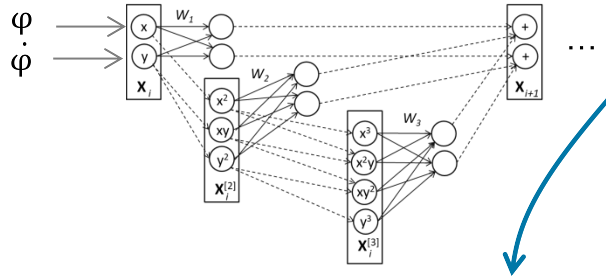
- accurate simulation of dynamics **without training**
- model **fine-tuning** with limited measurements

ECAI 2020

oral presentation at the European Conference on Artificial Intelligence

The key idea: If the dynamics of a system approximately follows a given differential equation, the Taylor mapping technique can be used to initialize the weights of a polynomial neural network

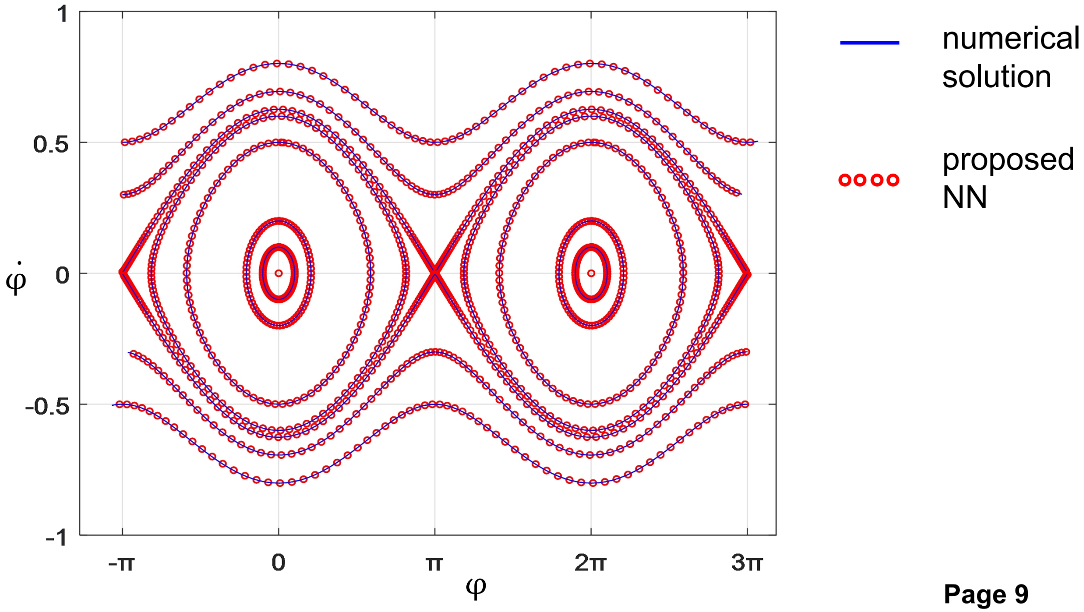
Pendulum oscillation: $\ddot{\varphi} = -\omega^2 \sin\varphi$



$$W_1 = \begin{pmatrix} 1 & 0.099 & -7.64E-06 & 0 \\ 0 & 1.000 & -1.54E-04 & 0 \end{pmatrix}$$

$$W_2 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & -1.58E-05 & 0 & 6.11E-10 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -4.80E-04 & 0 & 2.47E-08 & 0 \end{pmatrix}$$

$$W_3 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2.46E-05 & 0 & -2.28E-09 & 0 & 7.61E-10 & 5.87E-14 & 0 & -1.95E-14 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 9.95E-04 & 0 & -1.15E-07 & 0 & 3.84E-08 & 3.56E-12 & 0 & -1.18E-12 & 0 \end{pmatrix}$$



Novel approach for constructing deep neural networks for beam dynamics

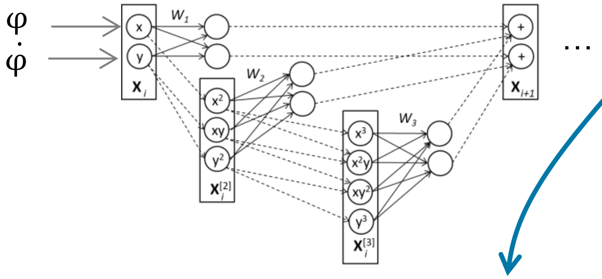
with the following key features:

- accurate simulation of dynamics **without training**
- model **fine-tuning** with limited measurements

oral presentation at the European Conference on Artificial Intelligence

The key idea: If the dynamics of a system approximately follows a given differential equation, the Taylor mapping technique can be used to initialize the weights of a polynomial neural network

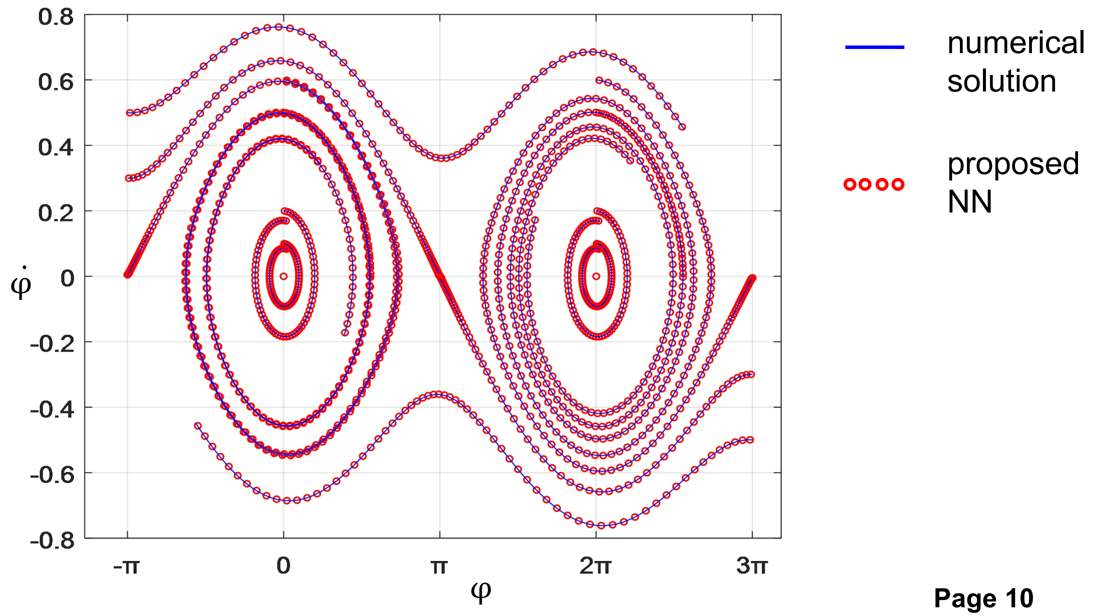
Pendulum oscillation: $\ddot{\varphi} = -\,\omega^2 \sin\varphi \boxed{-\,2\gamma\dot{\varphi}}$



$$W_1 = \begin{pmatrix} 1 & 0.099 & -7.79\mathrm{E}{-06} & 0 \\ 0 & 0.990 & -1.55\mathrm{E}{-04} & 0 \end{pmatrix}$$

$$W_2 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & -1.63\mathrm{E}{-05} & 0 & 6.36\mathrm{E}{-10} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -4.89\mathrm{E}{-04} & 0 & 2.54\mathrm{E}{-08} & 0 \end{pmatrix}$$

$$W_3 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2.56\mathrm{E}{-05} & 0 & -2.39\mathrm{E}{-09} & 0 & 7.99\mathrm{E}{-10} & 6.23\mathrm{E}{-14} & 0 & -2.07\mathrm{E}{-14} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.02\mathrm{E}{-03} & 0 & -1.19\mathrm{E}{-07} & 0 & 3.99\mathrm{E}{-08} & 3.74\mathrm{E}{-12} & 0 & -1.24\mathrm{E}{-12} & 0 \end{pmatrix}$$



Translating ODE of the pendulum into TM-PNN

1) transform ODE of mathematical pendulum to polynomial form

$$\varphi'' = -\,g\sin(\varphi)/L \quad \xleftrightarrow[y_1=\sin(\varphi),\, y_2=\cos(\varphi)]{} \quad \mathbf{X}' = \frac{d}{dt}\begin{pmatrix}\varphi\\ \varphi'\\ y_1\\ y_2\end{pmatrix} = \begin{pmatrix}\varphi'\\ -gy_1/L\\ y_2\varphi'\\ -y_1\varphi'\end{pmatrix} = P_1\mathbf{X} + P_2\mathbf{X}^2$$

2) represent the unknown solution as a Taylor map: $\mathbf{X} = W_{\mathbf{1}}\mathbf{X}_{\mathbf{0}} + W_{\mathbf{2}}\mathbf{X}_{\mathbf{0}}^{[\mathbf{2}]} + W_{\mathbf{3}}\mathbf{X}_{\mathbf{0}}^{[\mathbf{3}]}$

3) combine (1) and (2) and derive new system for $W_{\boldsymbol{i}}$:

$$W_1' = P_1W_1,$$

$$W_2' = P_1W_2 + P_2W_1^{[2]},$$

$$W_3' = P_1W_3 + 2P_2W_1 \otimes W_2,$$

Translating ODE of the pendulum into TM-PNN

1) transform ODE of mathematical pendulum to polynomial form

$$\varphi'' =- g\sin(\varphi)/L \quad \xleftrightarrow[y_1 = \sin(\varphi),\, y_2 = \cos(\varphi)]{} \quad \mathbf{X}' = \frac{d}{dt}\begin{pmatrix}\varphi\\ \varphi'\\ y_1\\ y_2\end{pmatrix} = \begin{pmatrix}\varphi'\\ -gy_1/L\\ y_2\varphi'\\ -y_1\varphi'\end{pmatrix} = P_1\mathbf{X} + P_2\mathbf{X}^2$$

2) represent the unknown solution as a Taylor map: $\mathbf{X} = W_\mathbf{1}\mathbf{X_0} + W_\mathbf{2}\mathbf{X}_\mathbf{0}^{[\mathbf{2}]} + W_\mathbf{3}\mathbf{X}_\mathbf{0}^{[\mathbf{3}]}$

3) combine (1) and (2) and derive new system for $W_{\boldsymbol{i}}$:

$$W_1' = P_1W_1,$$

$$W_2' = P_1W_2 + P_2W_1^{[2]},$$

$$W_3' = P_1W_3 + 2P_2W_1 \otimes W_2,$$

solving this system **at once** for predefined time interval result in weights suitable for arbitrary initial conditions $\longrightarrow$

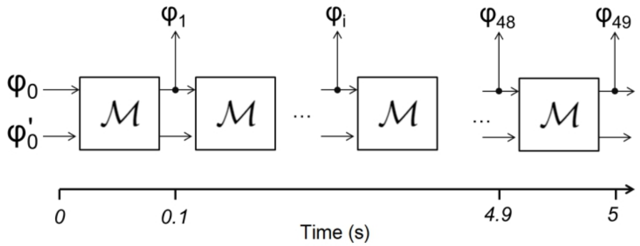
$$W_1 = \begin{pmatrix} 1 & 0.099 & -7.64\mathrm{E}{-06} & 0\\ 0 & 1.000 & -1.54\mathrm{E}{-04} & 0\end{pmatrix}$$

$$W_2 = \begin{pmatrix} 0\,0\,0\,0\,0\,0 & -1.58\mathrm{E}{-05} & 0 & 6.11\mathrm{E}{-10} & 0\\ 0\,0\,0\,0\,0\,0 & -4.80\mathrm{E}{-04} & 0 & 2.47\mathrm{E}{-08} & 0\end{pmatrix}$$

$$W_3 = \begin{pmatrix} 0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0 & 2.46\mathrm{E}{-05} & 0 & -2.28\mathrm{E}{-09} & 0 & 7.61\mathrm{E}{-10} & 5.87\mathrm{E}{-14} & 0 & -1.95\mathrm{E}{-14} & 0\\ 0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0 & 9.95\mathrm{E}{-04} & 0 & -1.15\mathrm{E}{-07} & 0 & 3.84\mathrm{E}{-08} & 3.56\mathrm{E}{-12} & 0 & -1.18\mathrm{E}{-12} & 0\end{pmatrix}$$

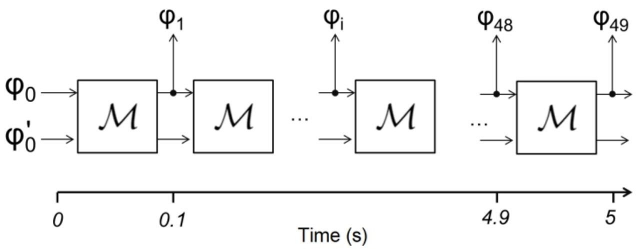
Taylor maps define a polynomial architecture

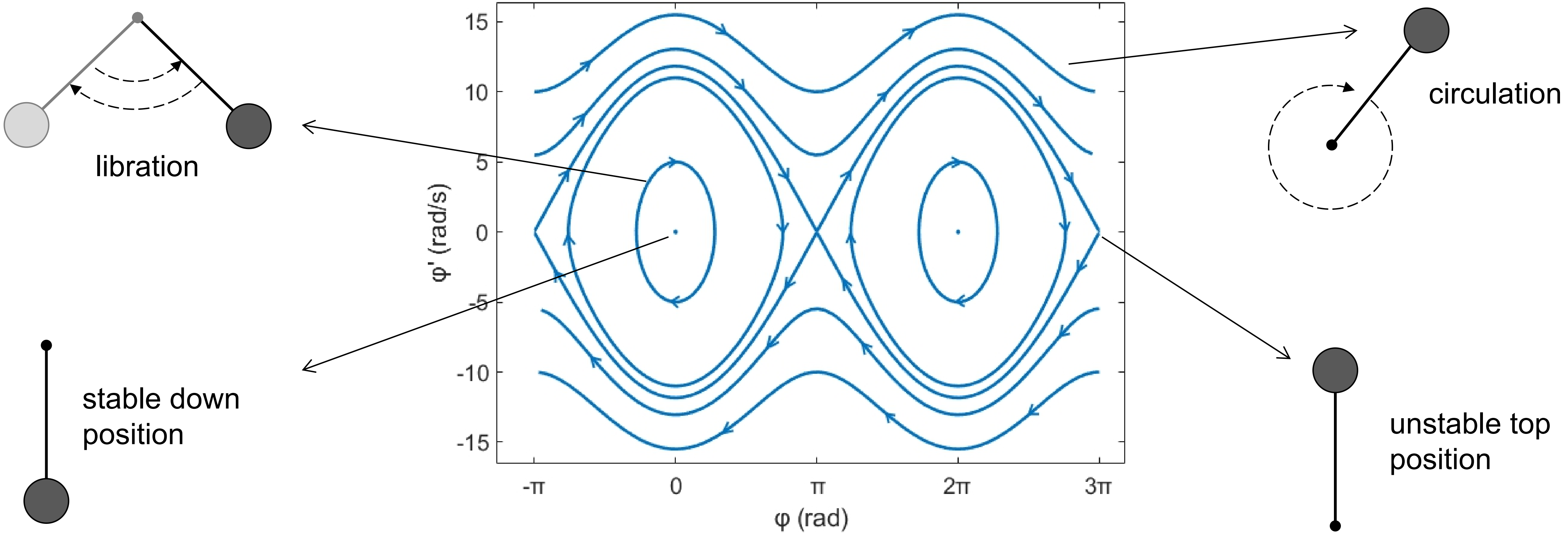
initialized with maps TM-PNN represents dynamics of the ODE with required level of accuracy for arbitrary inputs



Taylor maps define a polynomial architecture

initialized with maps TM-PNN represents dynamics of the ODE with required level of accuracy for arbitrary inputs





02 Training

- From scratch: a general-purpose regression method for deterministic systems
- With ODE-based weights initialization

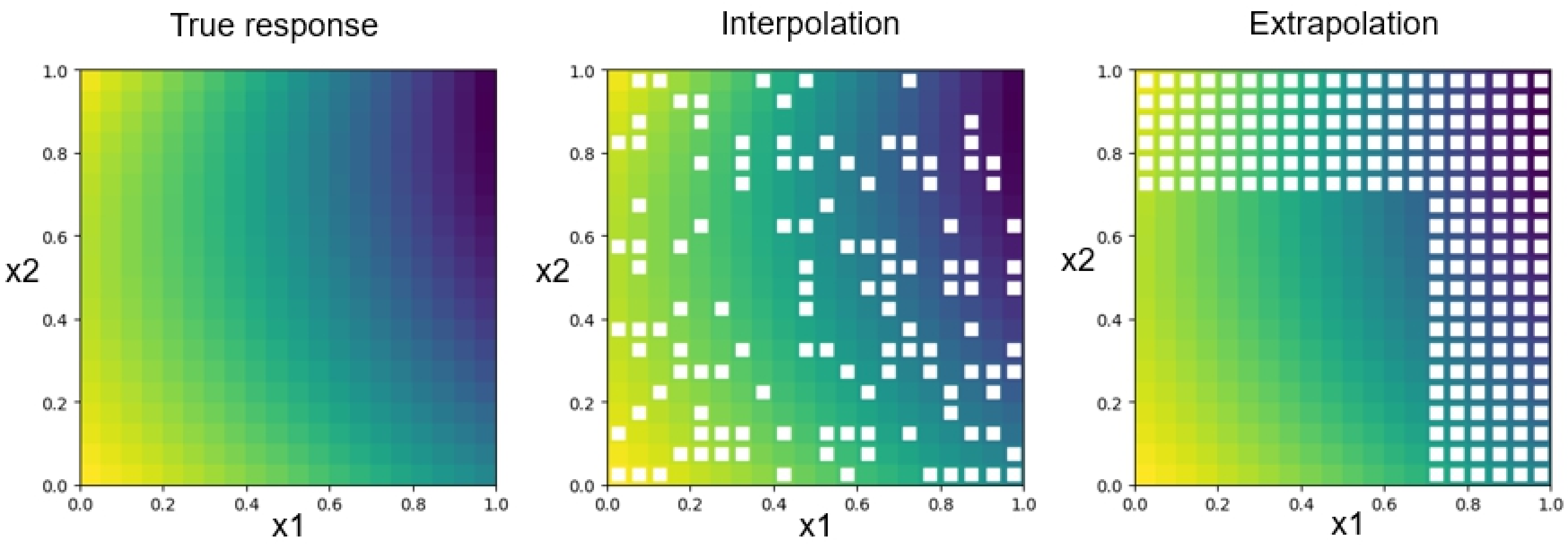
Training from scratch: a general-purpose regression method

If dataset generated by a physical system then developed model can be applied for a general purpose regression problem without a prior knowledge about ODEs

$$f:\ \{x_1, x_2, \ldots, x_n\} \rightarrow y.$$

UCI Machine Learning Repository:

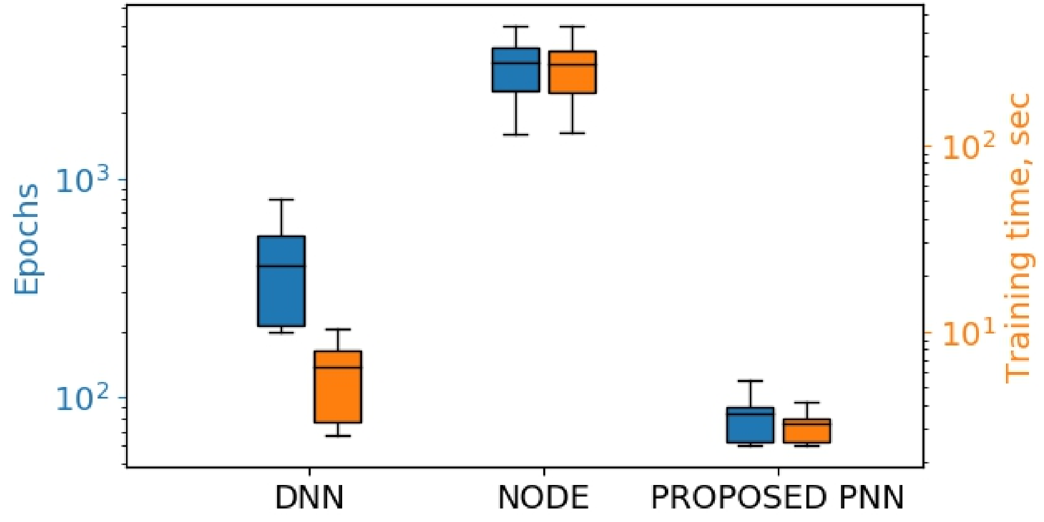
- Airfoil Self-Noise Data Set: NASA data set, obtained from a series of aerodynamic and acoustic tests of two and three-dimensional airfoil blade sections conducted in an anechoic wind tunnel.
- Yacht Hydrodynamics Data Set: Delft data set, used to predict the hydrodynamic performance of sailing yachts from dimensions and velocity.



Training from scratch: a general-purpose regression method

If dataset generated by a physical system then developed model can be applied for a general purpose regression problem without a prior knowledge about ODEs

METHODS	INTERPOLATION RMSE	INTERPOLATION R2	EXTRAPOLATION RMSE	EXTRAPOLATION R2
AIRFOIL SELF-NOISE DATASET (UCI, NASA)				
RIDGE REGRESSION	0.128	0.122	0.126	0.195
POLYNOMIAL REGR.	0.119	0.461	0.126	0.369
PNN	0.121	0.208	0.119	0.006
FM	0.134	< 0	0.217	< 0
GPR	0.079	0.761	0.130	0.557
SVR	0.086	0.682	0.144	< 0
XGBREGRESSOR	0.045	0.933	0.191	0.569
CATBOOSTREGR.	0.046	0.937	0.215	< 0
DNN	0.042	0.942	0.159	0.616
NODE	0.062	0.874	0.080	0.792
PROPOSED MODEL	0.077	0.811	0.106	0.733

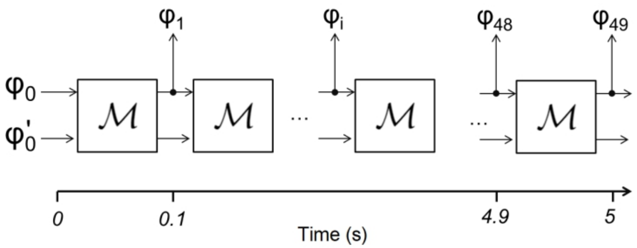


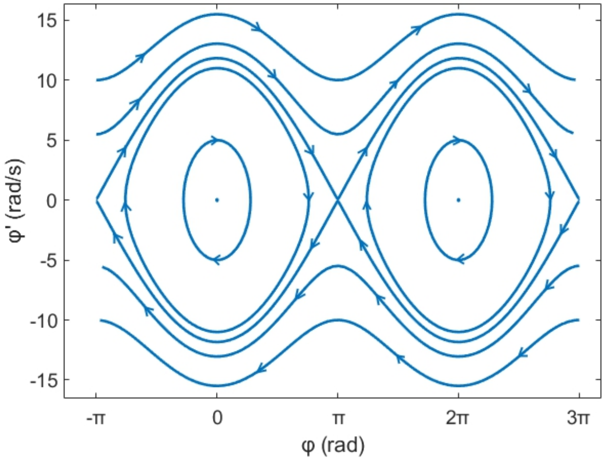
Training with ODE-based initialized weights

Having an approximate knoweledge about the system in form of ODE

$$\ddot{\varphi} = -\,\omega^2 \sin\varphi$$

one can build a TM-PNN with physical properties without training

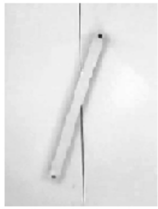




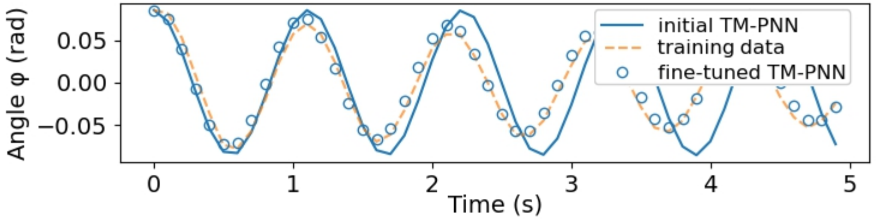
Fine-tuning of the TM-PNN with measurements

and recover true dynamics by fine-tuning of the weights

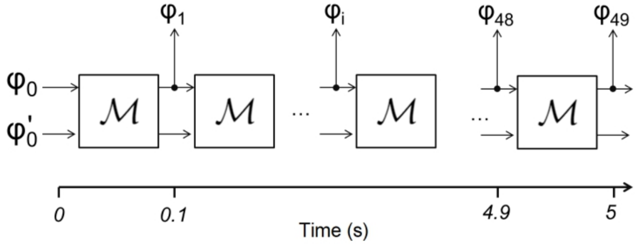
real dumped pendulum



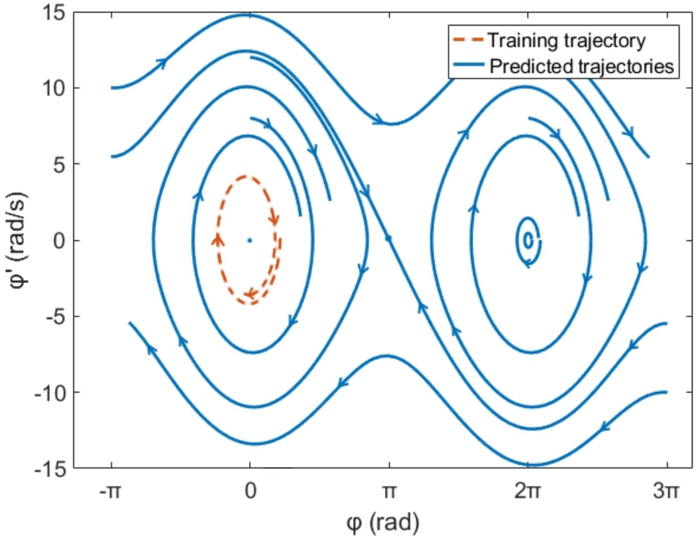
one training trajectory

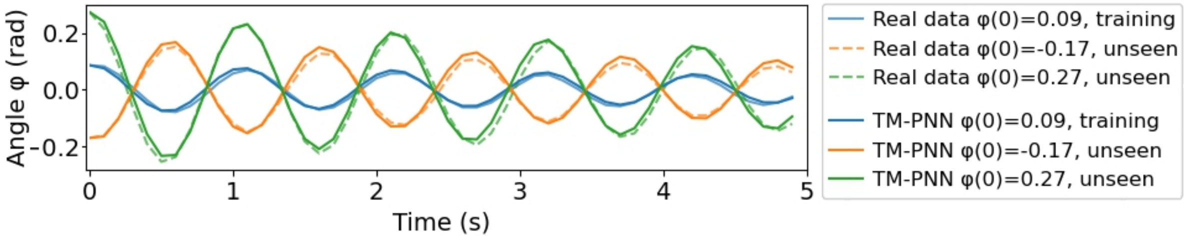


partial and noisy measurements



Prediction of the TM-PNN fine-tuned with one trajectory for unseen inputs:

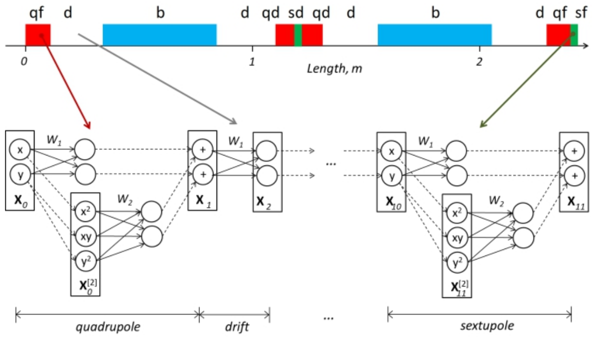


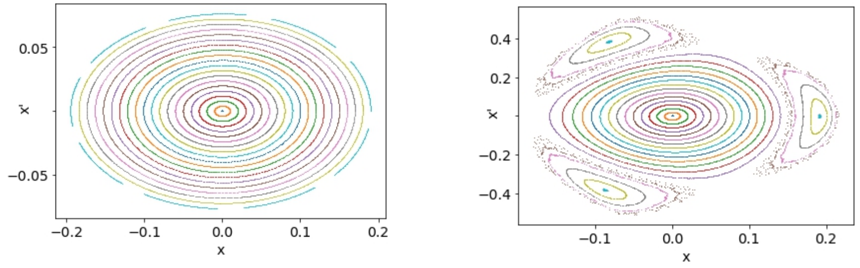


The fine-tuning of the TM-PNN with one oscillation not only increases the accuracy of the prediction for the given training oscillation but also recovers the physical property of the real pendulum for unseen inputs.

Weights initialization for particle accelerators

Each magnet is defined by a system of ODE





Initialized NN accurately represents the parametric dependency of dynamics on magnet strength, such as the appearance of a third-integer resonance

During training, the symplectic condition can be used

For Hamiltonian systems representing single-particle beam dynamics, the symplectic property can be used. The Hamiltonian structure of each layer is preserved for all new inputs which has a large impact on generalization.

$$W_1 = \begin{pmatrix} w_1^{11} & w_1^{12} \\ w_1^{21} & w_1^{22} \end{pmatrix}, \quad W_2 = \begin{pmatrix} w_2^{11} & w_2^{12} & w_2^{13} \\ w_2^{21} & w_2^{22} & w_2^{23} \end{pmatrix} \xrightarrow{\text{symplectic property}}$$

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

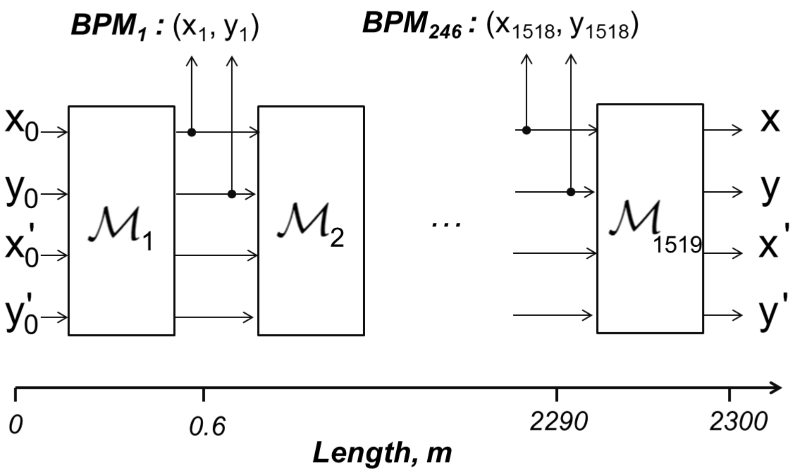
03 Application in particle accelerators

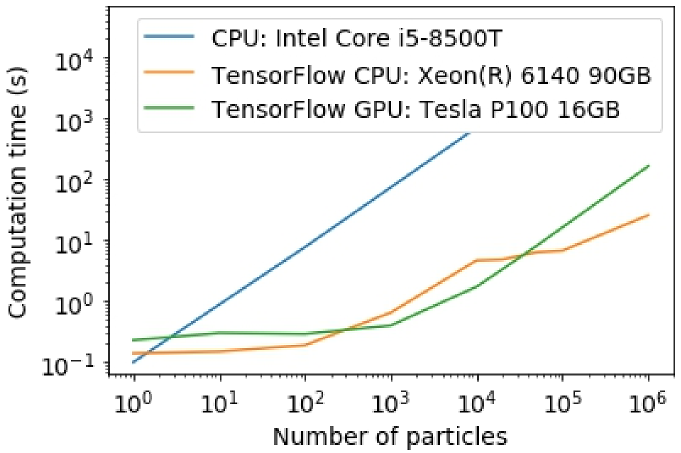
- Simulation of beam dynamics
- Data-driven model calibration (PETRAIII experiments)
- RL-enhanced control (simulated environment)

Simulation of beam dynamics

PETRAIII: deep neural network with **1519 layers** represents ideal lattice with fair accuracy

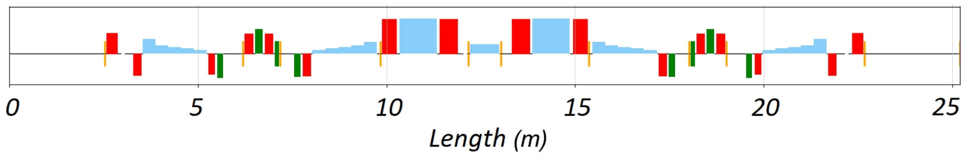
- 2,3 km length with **1519** magnets
- **210** horizontal and **194** vertical correctors
- **246** BMPS



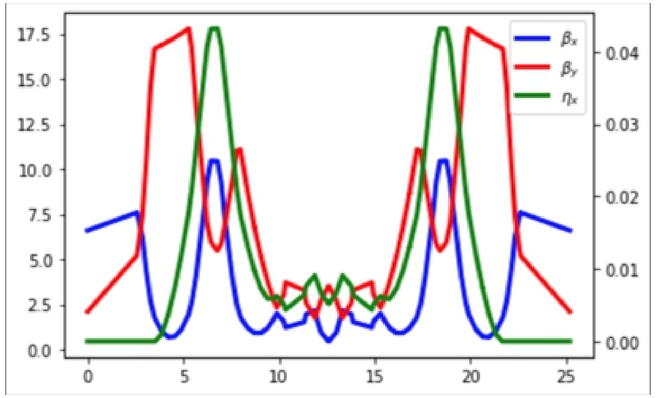


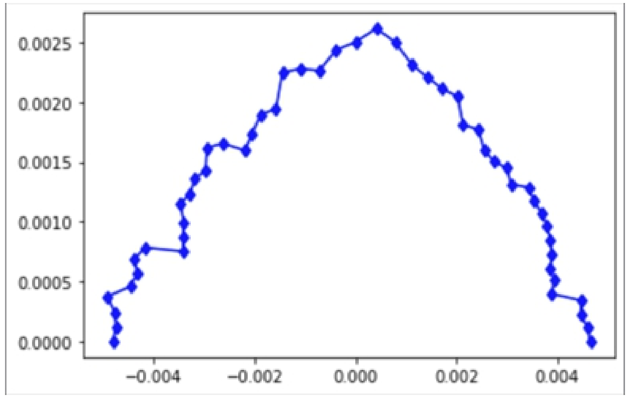
Simulation of beam dynamics

PETRAIV cell

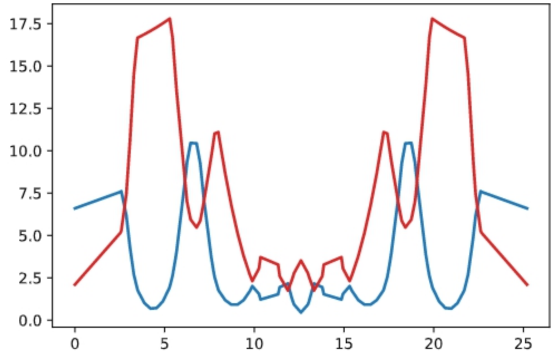


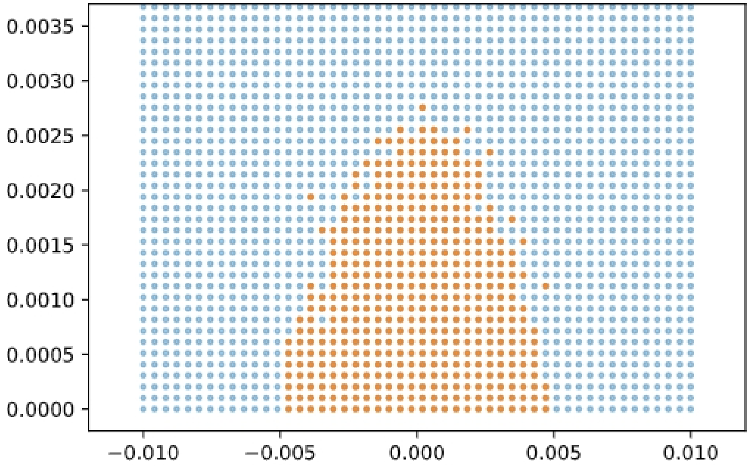
Elegant





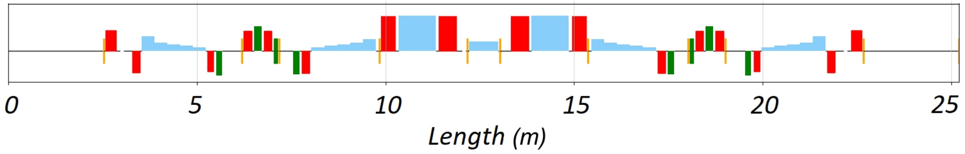
NN in TensorFlow



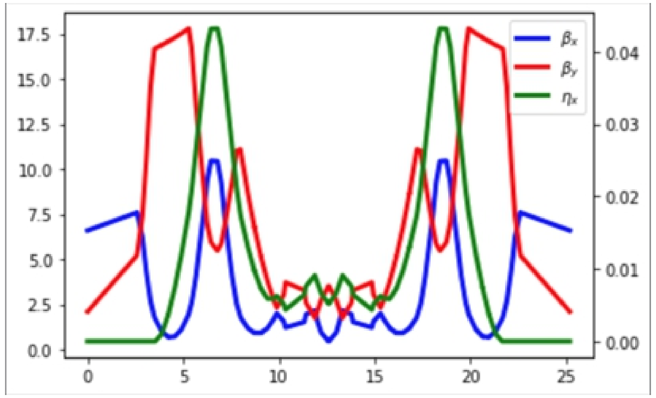


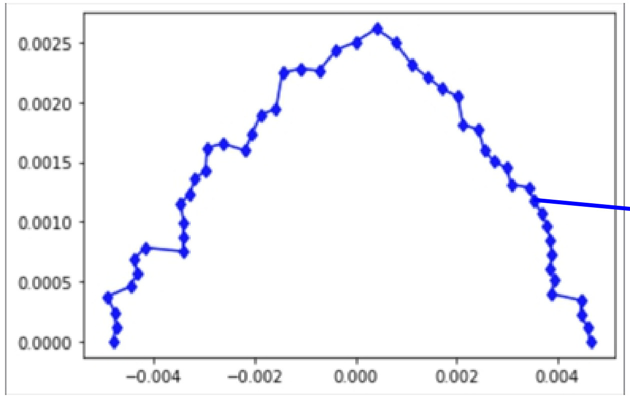
Simulation of beam dynamics

PETRAIV cell



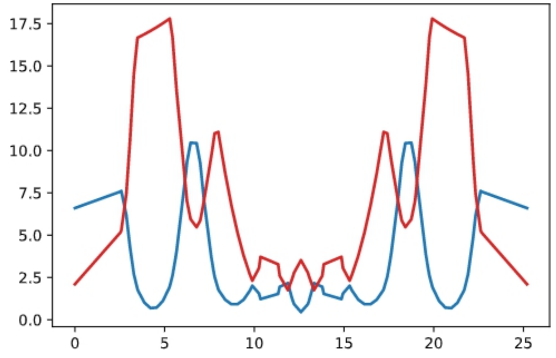
Elegant

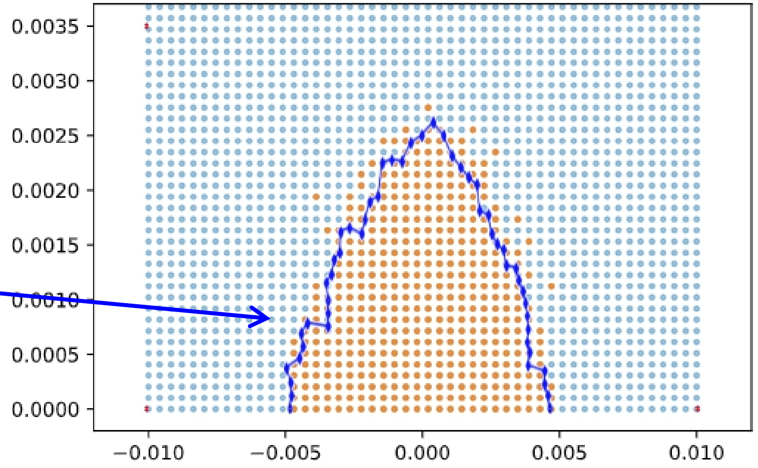




different grid search
and numerical maps

NN in TensorFlow

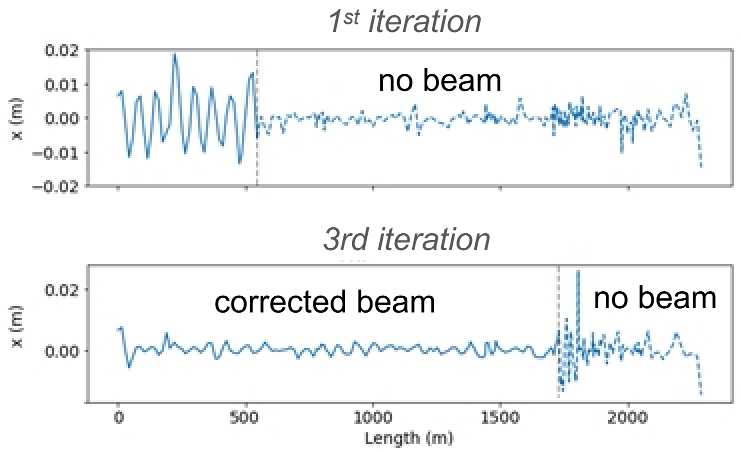




One-shot learning of PETRAIII in experiments

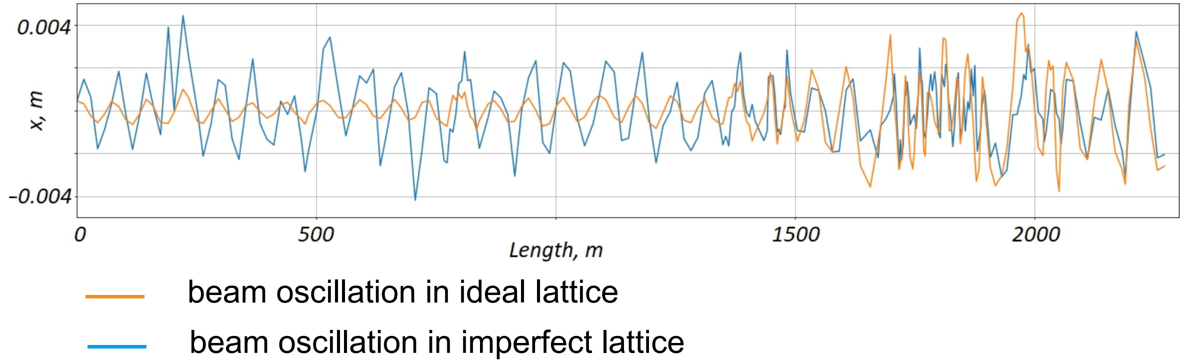
Beam threading

1. All corrector magnets are switched off
2. Beam is able to travel through only a part of the ring
3. **Neural Network predicts** an optimal control policy for beam propagation

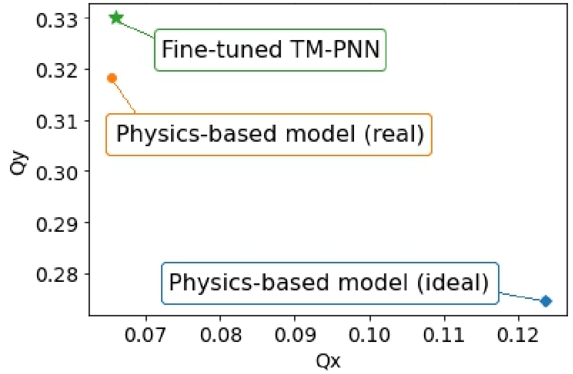


Tune recovering

1. Tune is the main multi-turn frequency of beam oscillation in the storage ring
2. The affected magnets cause the tune change from the designed values.
3. **Neural Network** is trained with only a **single-turn measurement** and estimates tunes with 95% accuracy.



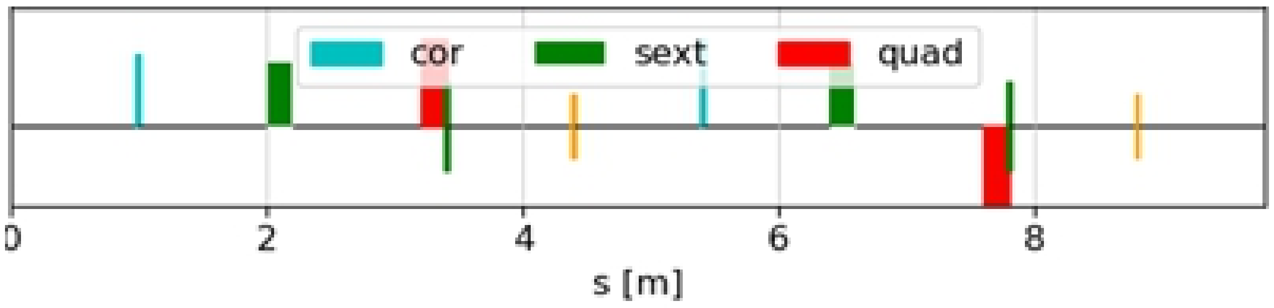
training NN with a single-turn measurements



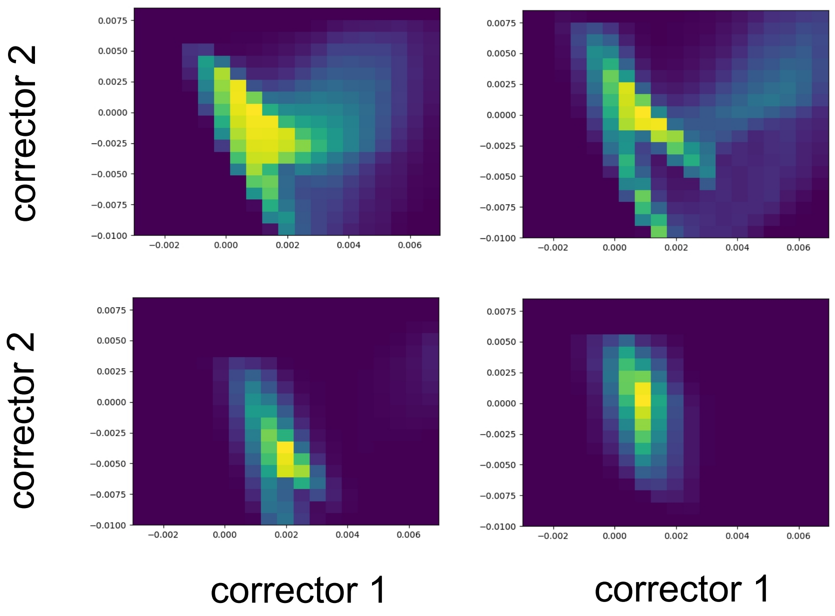
95% accuracy of the multi-turn prediction

RL-enhanced control

beam transmission: 2 actuators (correctors), 1 objective, sextupoles and apertures

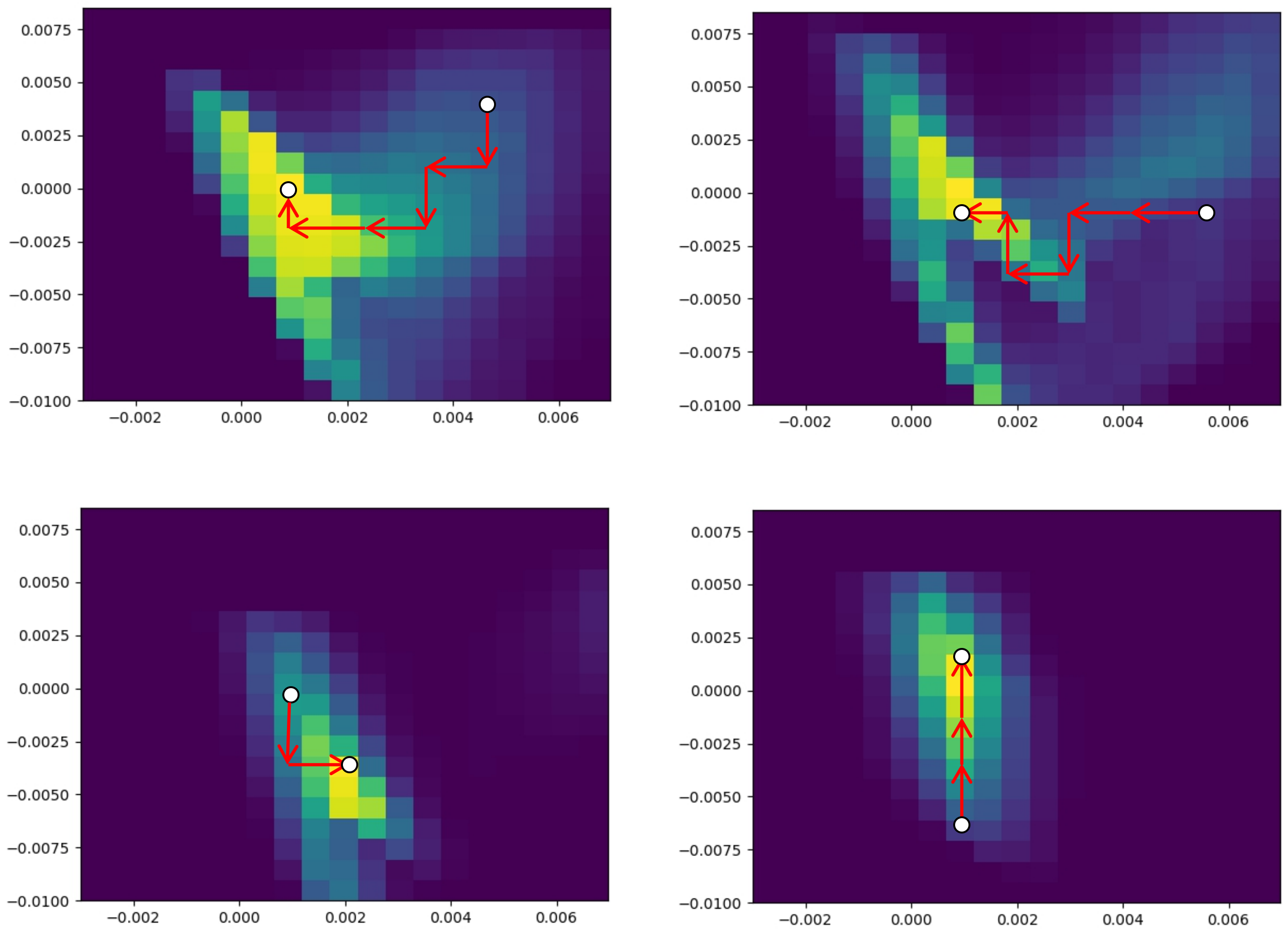


nonlinear response concerning the random misalignments of magnets



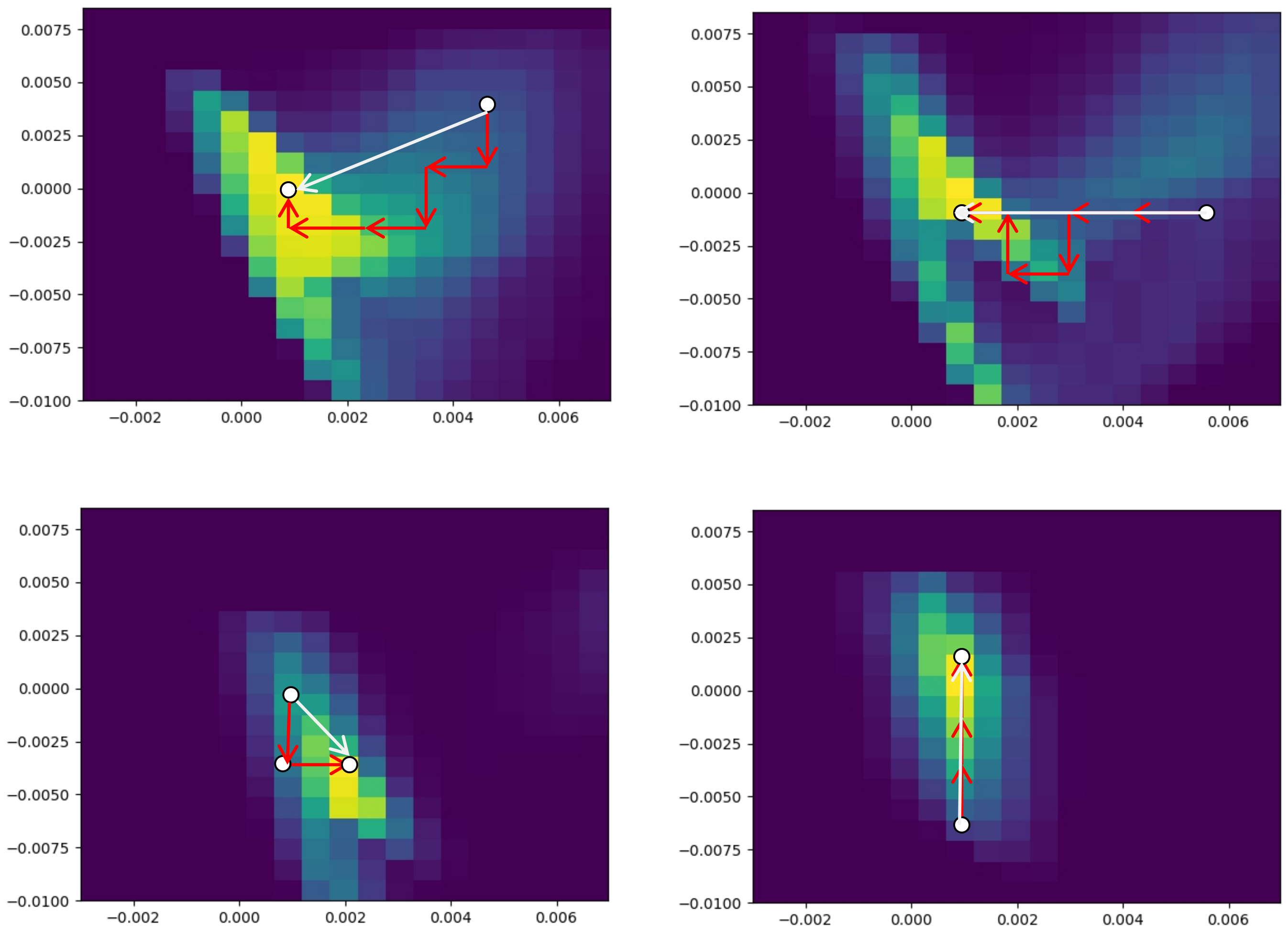
Numerical optimization

using traditional optimizers one can iteratively find out optimal corrector's values



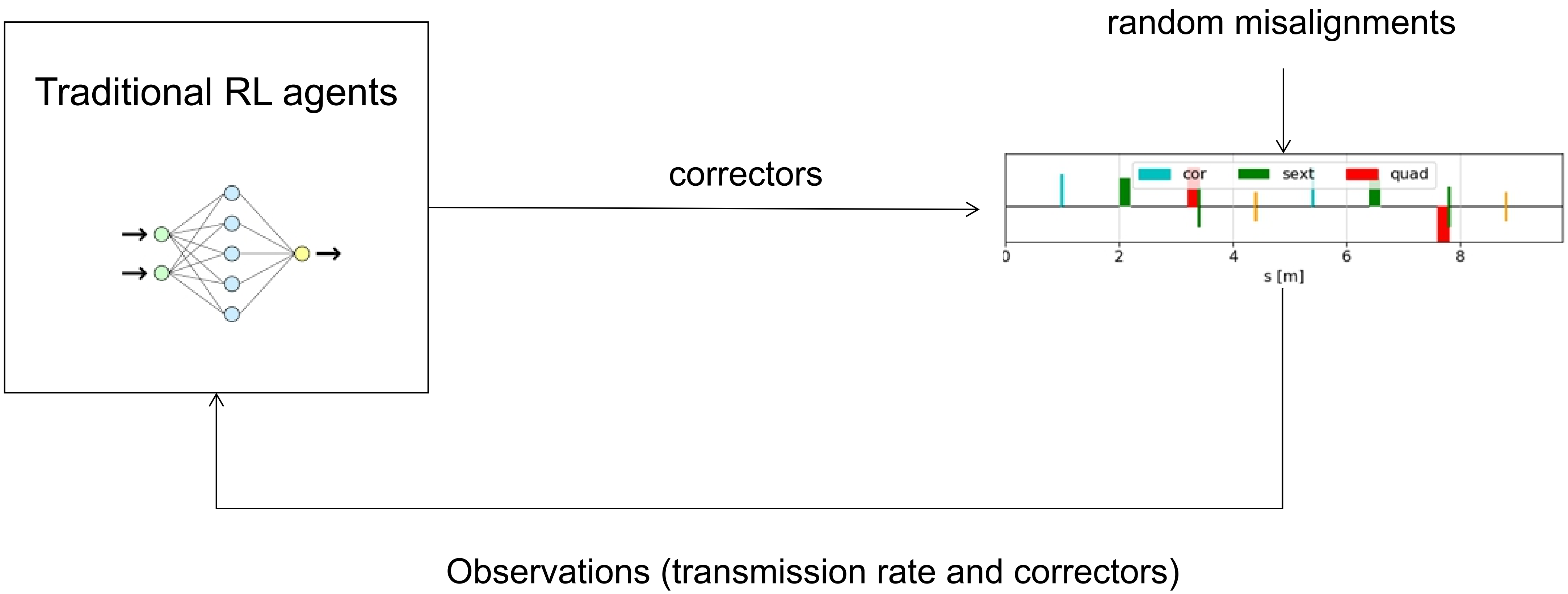
Numerical optimization

using traditional optimizers one can iteratively find out optimal corrector's values



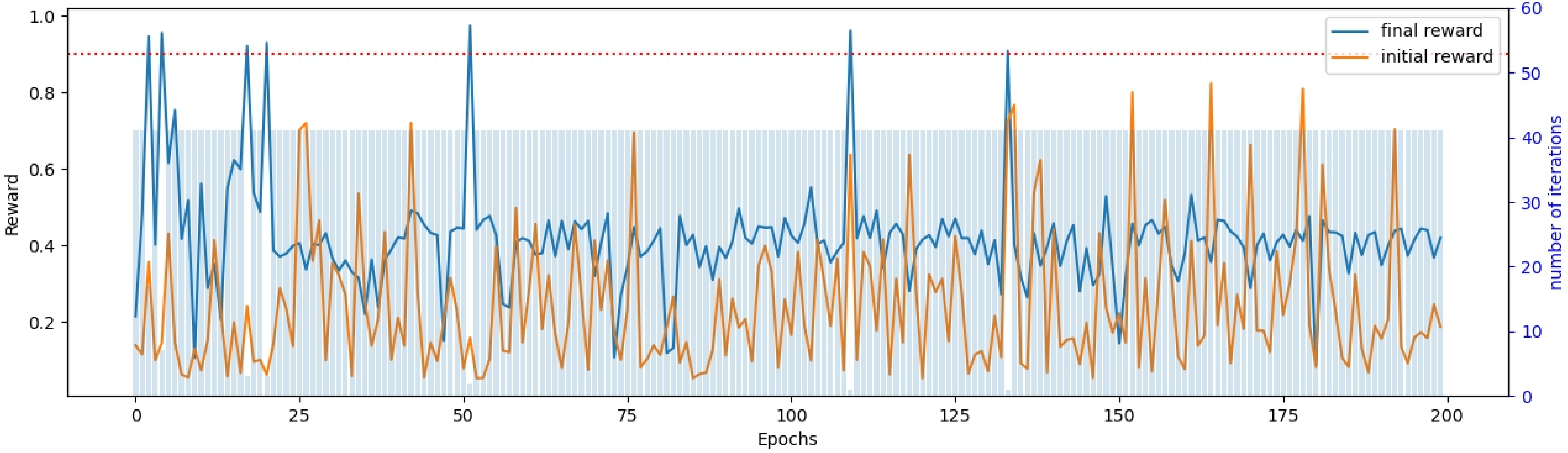
RL for control

NN is trained with ‘historical data’ and learns an optimal policy



RL for control

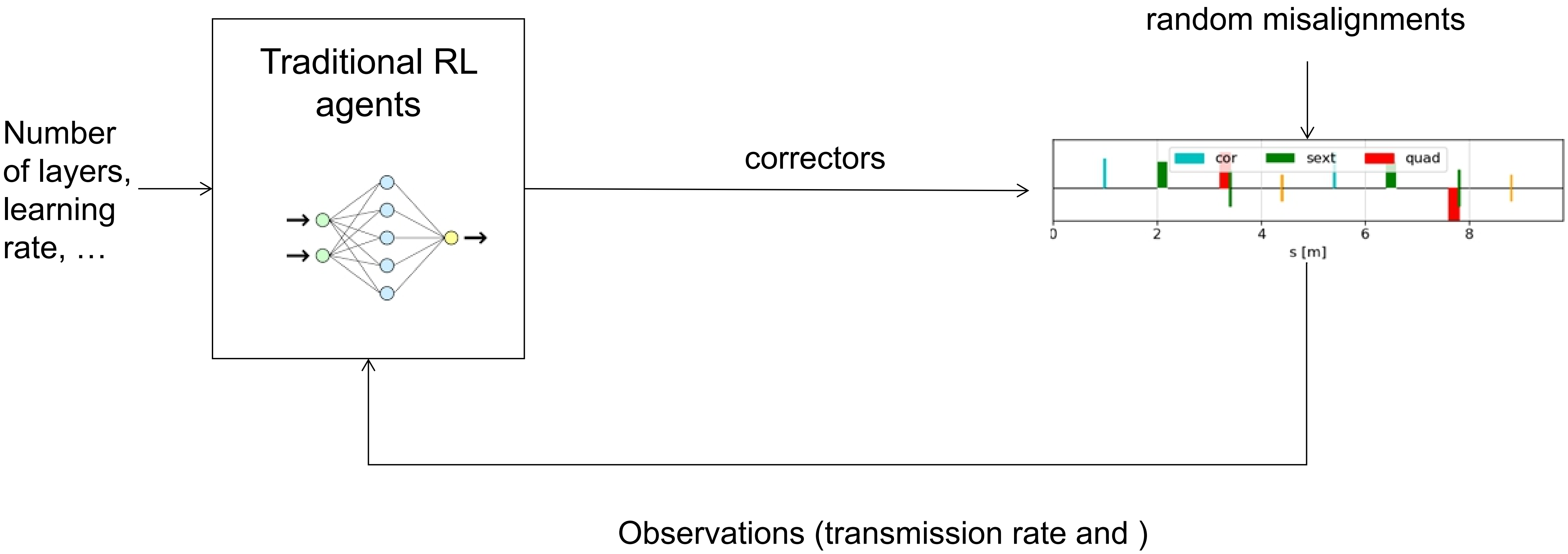
It is hard to achieve meaningful results with black-box models



During each epoch NN is trained with simulated data for the given random misalignments and tries to maximize initial state (orange line). After max. 40 iterations the procedure begins again for new random misalignments.

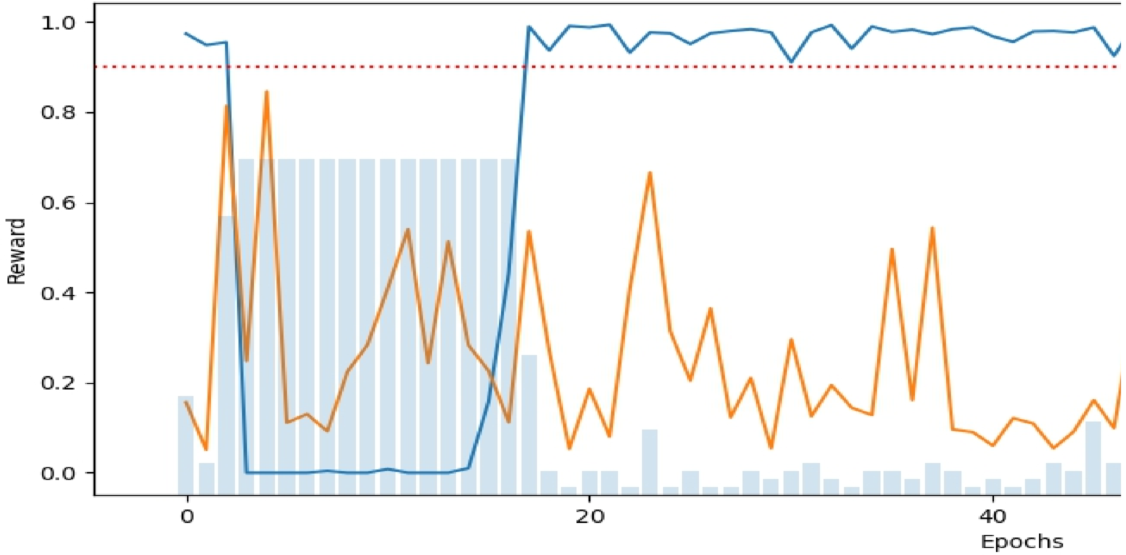
RL for control

To fix this issue ML methods provide possibility to tune hyper parameters of the NN



RL for control

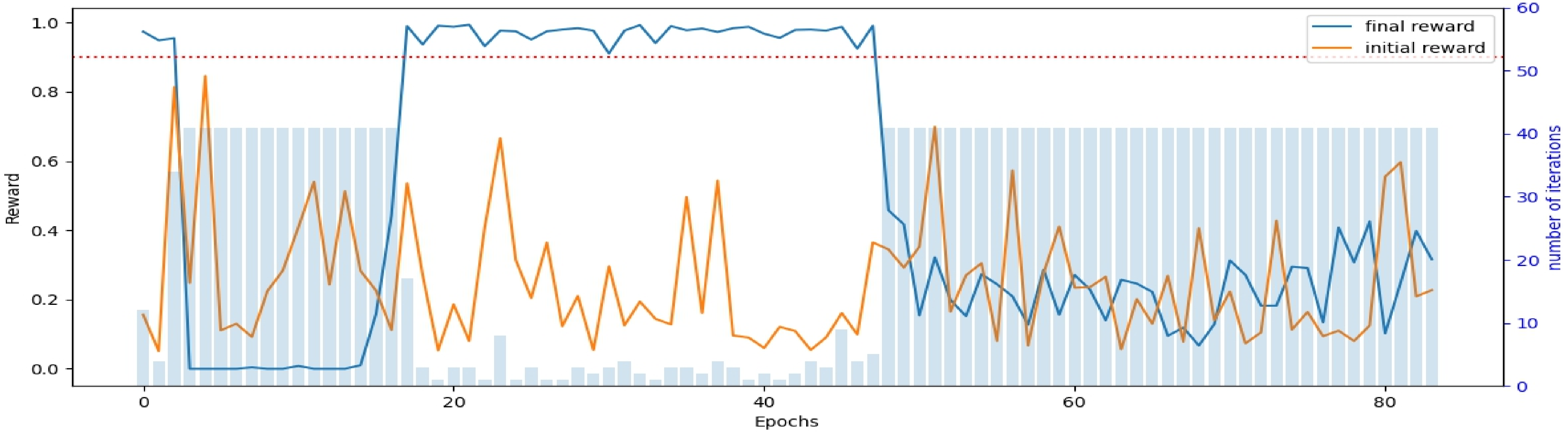
To fix this issue ML methods provide possibility to tune hyper parameters of the NN



looks like a convergence

RL for control

To fix this issue ML methods provide possibility to tune hyper parameters of the NN



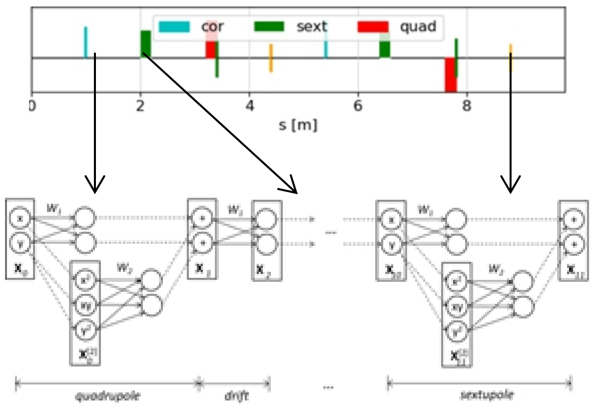
looks like a convergence

no guarantee that NN worsk for new parameters

RL for control enhanced by physics-based NN

Incorporate a priory knowledge in form of a trainable NN

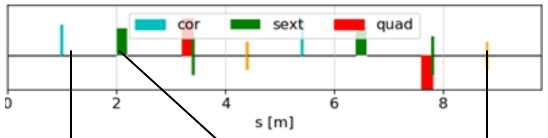
ideal lattice

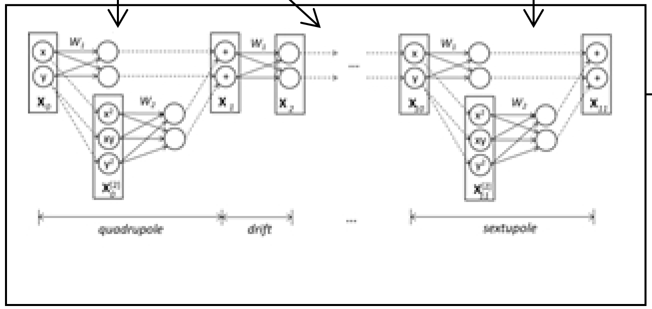


RL for control enhanced by physics-based NN

Incorporate a priory knowledge in form of a trainable NN

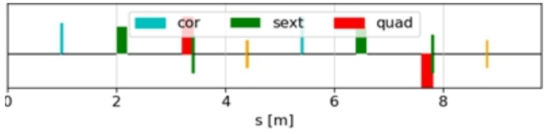
ideal lattice

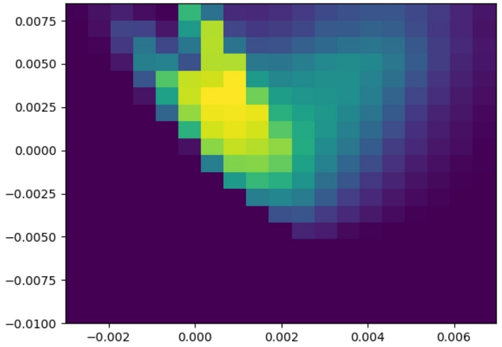


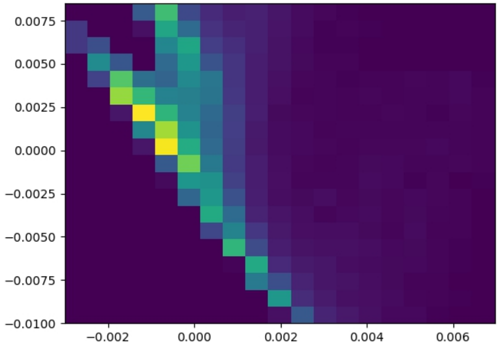


correctors

real lattice with random misalignments

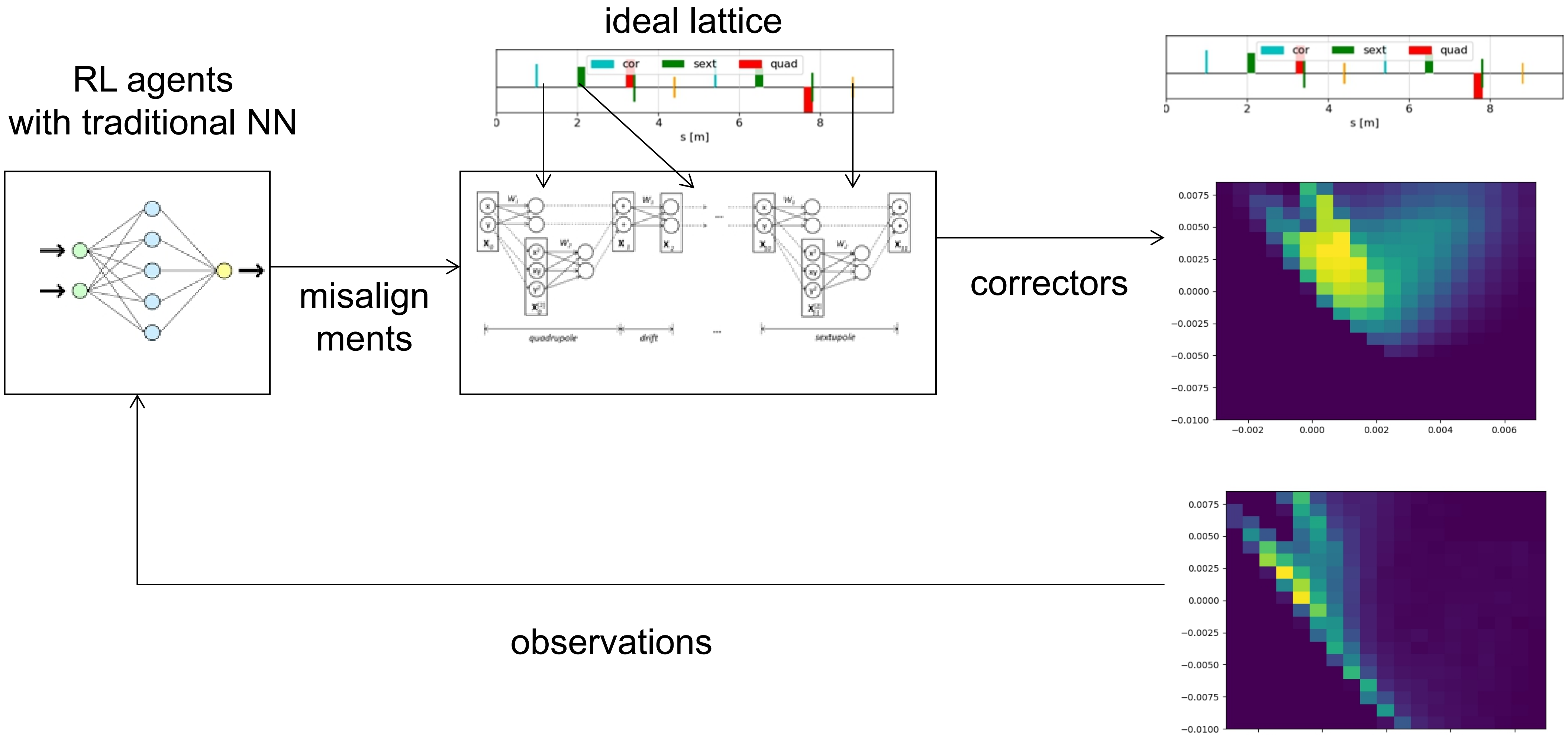






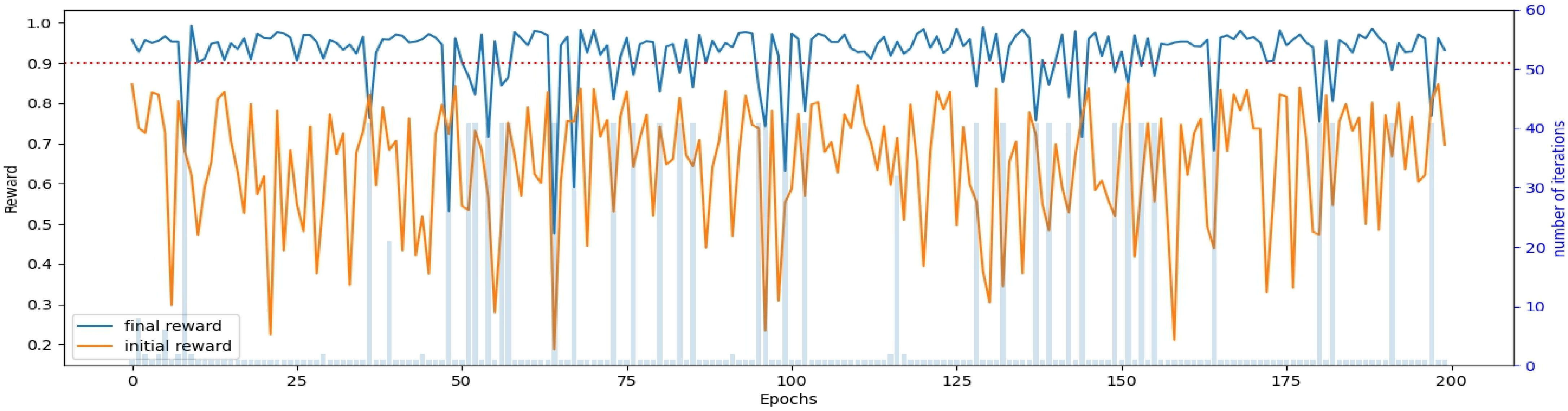
RL for control enhanced by physics-based NN

Incorporate a priory knowledge in form of a trainable NN



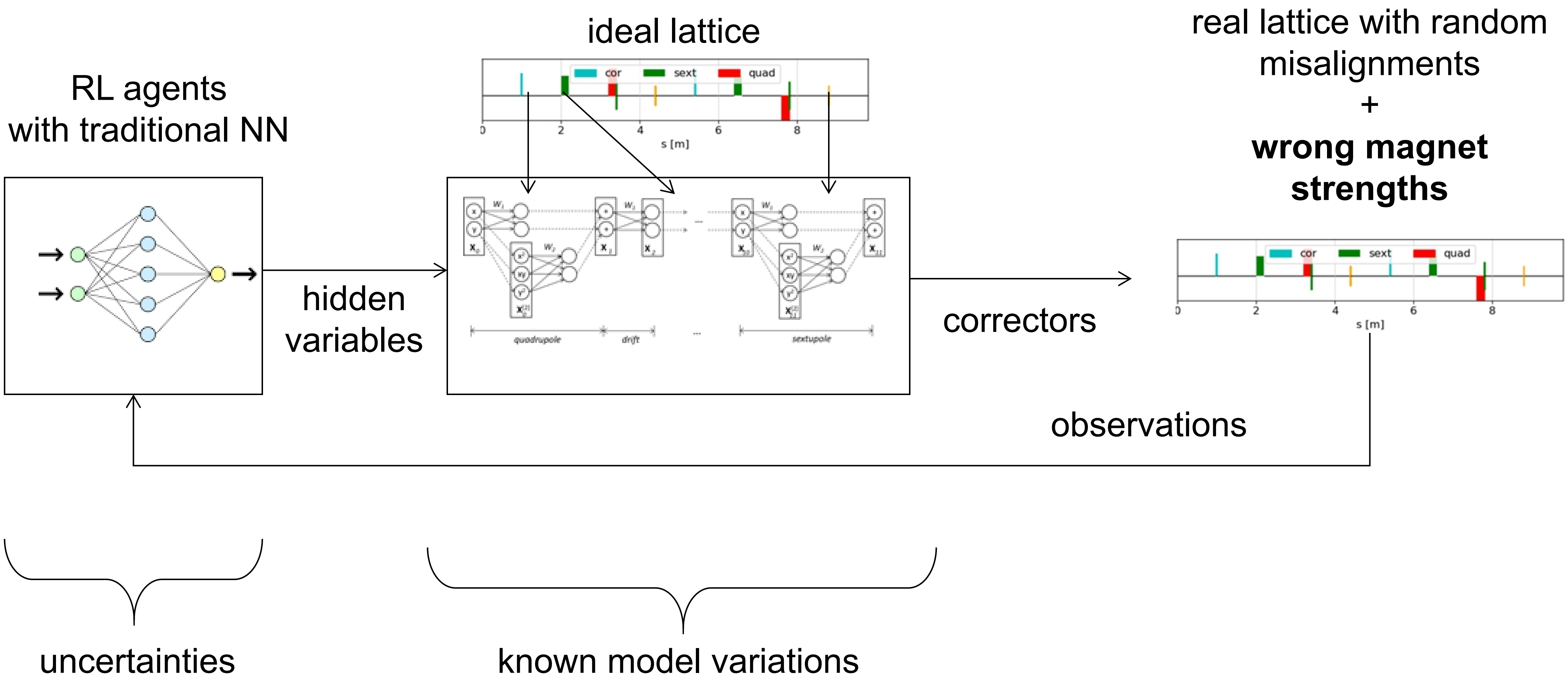
RL agent recovers misalignments distribution from data and provides an optimal strategy

Similar to a traditional optimizer that utilizes knowledge from historical data and uses adaptive steps during objective maximization



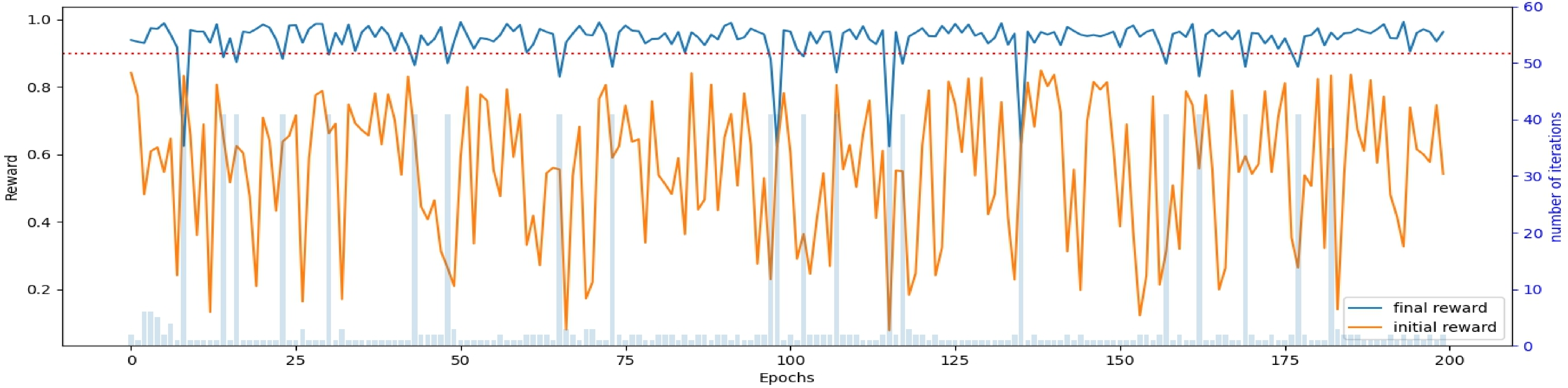
RL for control enhanced by physics-based NN

Incorporate a priory knowledge in form of trainable NN



RL agent + Taylor map-based NN approximates true system

Taylor maps are calculated for the ideal lattice, but true lattice consists of magnets with strengths reduced by 20%



Results

Paper in European Conference on Artificial Intelligence

Paper in Physical Review AB

01 Novel architecture of deep NN incorporating physical knowledge from ODEs

02 The NN is validated on both general-purpose regression tasks and specific accelerator problems

03 RL-enhanced optimal control with physics incorporating

Thank you

Contact

DESY. Deutsches
Elektronen-Synchrotron

www.desy.de

Andrei Ivanov
andrei.ivanov@desy.de