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

▪ surrogates models: train a black-box model with simulated data

\[
\frac{d}{dt} \mathbf{X} = f(t, \mathbf{X}) \quad \xrightarrow{\text{simulation}} \quad \text{training dataset}
\]
Physics-inspired neural networks

several methods to incorporate physical knowledge into predictive model exist

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

\[
\frac{d}{dt} X = f(t, X) \quad \text{simulation} \quad \text{training dataset}
\]

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

  e.g. Hamiltonian Neural Networks

\[
\frac{d}{dt} X = f(t, X, NN) \quad \frac{d}{dt} X = f(t, X)
\]
Physics-inspired neural networks

several methods to incorporate physical knowledge into predictive model exist

▪ surrogates: train a black-box model with simulated data

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

▪ 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
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 with the following key features:

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

Pendulum oscillation: \( \ddot{\phi} = -\omega^2 \sin \phi \)
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

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: \[ \dot{\phi} = -\omega^2 \sin \phi \]

**Numerical solution**

**Proposed NN**
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

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: 

\[ \dot{\phi} = -\omega^2 \sin \phi - 2\gamma \phi \]

\[
W_1 = \begin{pmatrix} 1 & 0.099 & -7.79E-05 & 0 \\ 0 & 0.990 & -1.55E-04 & 0 \end{pmatrix}
\]

\[
W_2 = \begin{pmatrix} 0 & 0 & 0 & 0 & 1.63E-05 & 6.36E-10 \\ 0 & 0 & 0 & 0 & -4.89E-04 & 2.54E-08 \\ 0 & 0 & 0 & 0 & -2.99E-07 & 3.99E-08 \\ 0 & 0 & 0 & 0 & -1.19E-06 & 3.74E-12 \\ 0 & 0 & 0 & 0 & -1.24E-12 & 0 \end{pmatrix}
\]

\[
W_3 = \begin{pmatrix} 0 & 0 & 0 & 0 & 2.56E-05 & 0 \\ 0 & 0 & 0 & 0 & -2.39E-09 & 0 \\ 0 & 0 & 0 & 0 & -1.19E-07 & 0 \\ 0 & 0 & 0 & 0 & -1.24E-12 & 0 \end{pmatrix}
\]
Translating ODE of the pendulum into TM-PNN

1) transform ODE of mathematical pendulum to polynomial form

\[ \phi'' = -g \sin(\phi)/L \]

\[ y_1 = \sin(\phi), y_2 = \cos(\phi) \]

\[ \mathbf{X'} = \frac{d}{dt} \begin{pmatrix} \phi \\ \phi' \\ y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} \phi' \\ -g y_1/L \\ y_2 \phi' \\ -y_1 \phi' \end{pmatrix} = P_1 \mathbf{X} + P_2 \mathbf{X}^2 \]

2) represent the unknown solution as a Taylor map:

\[ \mathbf{X} = W_1 \mathbf{X}_0 + W_2 \mathbf{X}_0^{[2]} + W_3 \mathbf{X}_0^{[3]} \]

3) combine (1) and (2) and derive new system for \( W_i \):

\[ W'_1 = P_1 W_1, \]

\[ W'_2 = P_1 W_2 + P_2 W_1^{[2]}, \]

\[ W'_3 = P_1 W_3 + 2P_2 W_1 \otimes W_2, \]
Translating ODE of the pendulum into TM-PNN

1) transform ODE of mathematical pendulum to polynomial form

\[ \varphi'' = -g \sin(\varphi)/L \]

\[ y_1 = \sin(\varphi), y_2 = \cos(\varphi) \]

\[ \mathbf{x}' = \frac{d}{dt} \begin{pmatrix} \varphi \\ \varphi' \\ y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} \varphi' \\ -g y_1/L \\ y_2 \varphi' \\ -y_1 \varphi' \end{pmatrix} = \mathbf{P}_1 \mathbf{x} + \mathbf{P}_2 \mathbf{x}^2 \]

2) represent the unknown solution as a Taylor map: \( \mathbf{x} = \mathbf{W}_1 \mathbf{x}_0 + \mathbf{W}_2 \mathbf{x}_0^2 + \mathbf{W}_3 \mathbf{x}_0^3 \)

3) combine (1) and (2) and derive new system for \( \mathbf{W}_i \):

\[ \mathbf{W}_1' = \mathbf{P}_1 \mathbf{W}_1, \]

\[ \mathbf{W}_2' = \mathbf{P}_1 \mathbf{W}_2 + \mathbf{P}_2 \mathbf{W}_1^2, \]

\[ \mathbf{W}_3' = \mathbf{P}_1 \mathbf{W}_3 + 2\mathbf{P}_2 \mathbf{W}_1 \otimes \mathbf{W}_2, \]

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

\[ \mathbf{W}_1 = \begin{pmatrix} 1 & 0.099 & -7.64E-06 & 0 \\ 0 & 1.000 & -1.54E-04 & 0 \end{pmatrix} \]

\[ \mathbf{W}_2 = \begin{pmatrix} 0 & 0 & 0 & 0 & -1.58E-05 & 0.611E-10 \\ 0 & 0 & 0 & 0 & -4.80E-04 & 0.247E-08 \end{pmatrix} \]

\[ \mathbf{W}_3 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 2.46E-05 & 0 & -2.28E-09 & 0.761E-10 & 5.87E-14 & 0 & -1.95E-14 \0 & 0 & 0 & 0 & 0 & 0 & 9.95E-04 & 0 & -1.15E-07 & 0.384E-08 & 3.56E-12 & 0 & -1.18E-12 \end{pmatrix} \]
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

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

![Graph showing epochs and training time](image.png)
Training with ODE-based initialized weights

Having an approximate knowledge 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

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

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.

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

\[
\begin{align*}
w_1^{11}w_1^{22} - w_1^{12}w_1^{21} - 1 &= 0, \\
w_1^{11}w_2^{23} - w_1^{13}w_2^{21} &= 0, \\
w_2^{12}w_2^{23} - w_2^{13}w_2^{22} &= 0,
\end{align*}
\]
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

PETRA IV cell

Elegant

NN in TensorFlow

different grid search and numerical maps
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.
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

Traditional RL agents

Observations (transmission rate and correctors)

random misalignments

Physics-Based Neural Networks for Particle Accelerators | Andrei Ivanov
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

Traditional RL agents

Number of layers, learning rate, ...

Observations (transmission rate and )

random misalignments
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

\[ t \text{ [m]} \]

\[ \text{quadrupole} \]

\[ \text{dipole} \]
RL for control enhanced by physics-based NN

Incorporate a priory knowledge in form of a trainable NN

real lattice with random misalignments

ideal lattice

correctors
RL for control enhanced by physics-based NN

Incorporate a priori knowledge in form of a trainable NN

RL agents with traditional NN

ideal lattice

misalignments

correctors

observations

real lattice with random misalignments
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 priori knowledge in form of trainable NN

RL agents with traditional NN

ideal lattice

hidden variables

correctors

observations

uncertainties

known model variations

real lattice with random misalignments + wrong magnet strengths

Physics-Based Neural Networks for Particle Accelerators | Andrei Ivanov
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

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       Andrei Ivanov
                                           andrei.ivanov@desy.de

www.desy.de