CLASSE
Cornell Laboratory for Accelerator-based ScienceS & Education

The Center for BRIGHT BEAMS
A National Science Foundation Science & Technology Center

Brookhaven National Laboratory

U.S. DEPARTMENT OF ENERGY

Machine Learning Applications for Improving Accelerator Operations at the AGS and AGS Booster

Lucy Lin

Advisor: Georg Hoffstaetter

CLASSE Accelerator Physics Journal Club

April 27, 2023

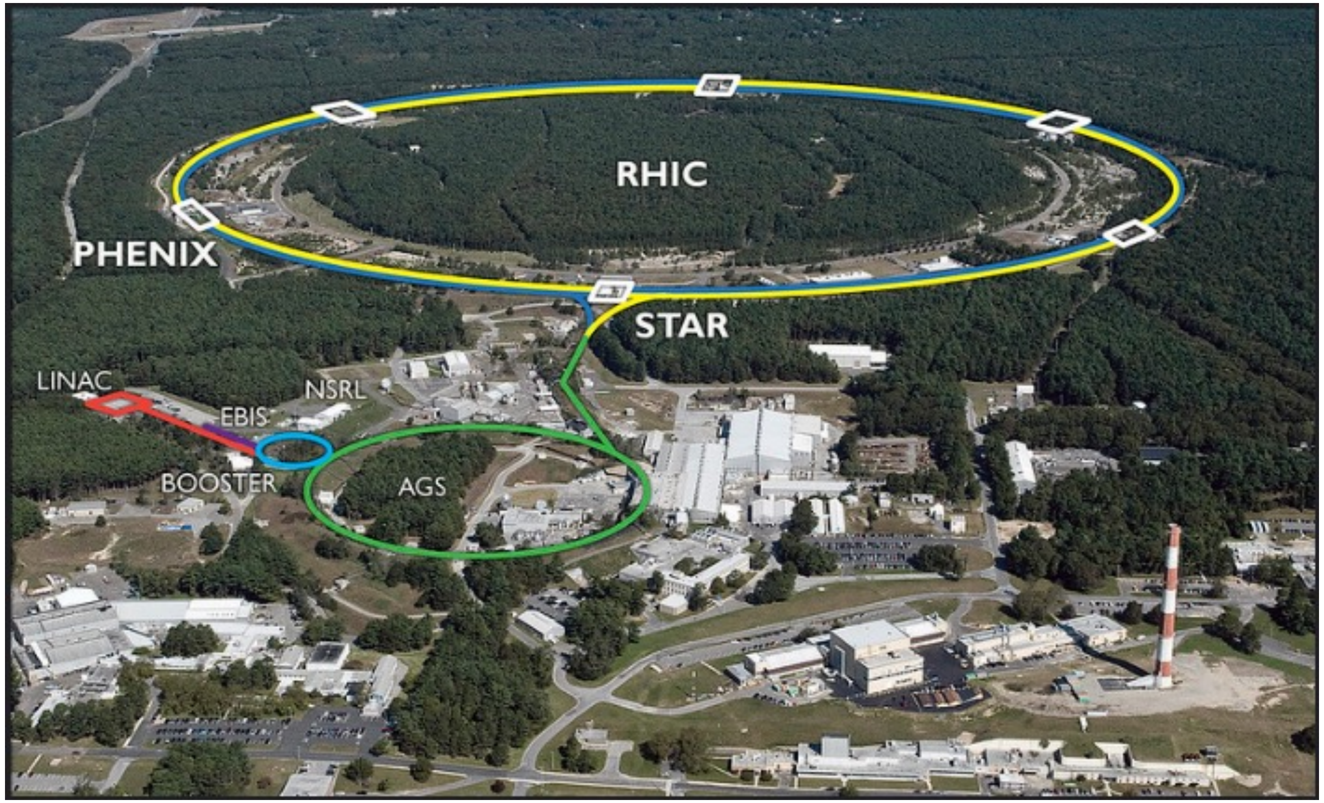
@BrookhavenLab

Summary

- Simulation Studies and Machine Learning Applications for Orbit and Optics Correction at the Alternating Gradient Synchrotron

- Beam-based Quadrupole Transfer Function Measurement with Neural Network at Alternating Gradient Synchrotron Booster

Relativistic Heavy Ion Collider (RHIC)



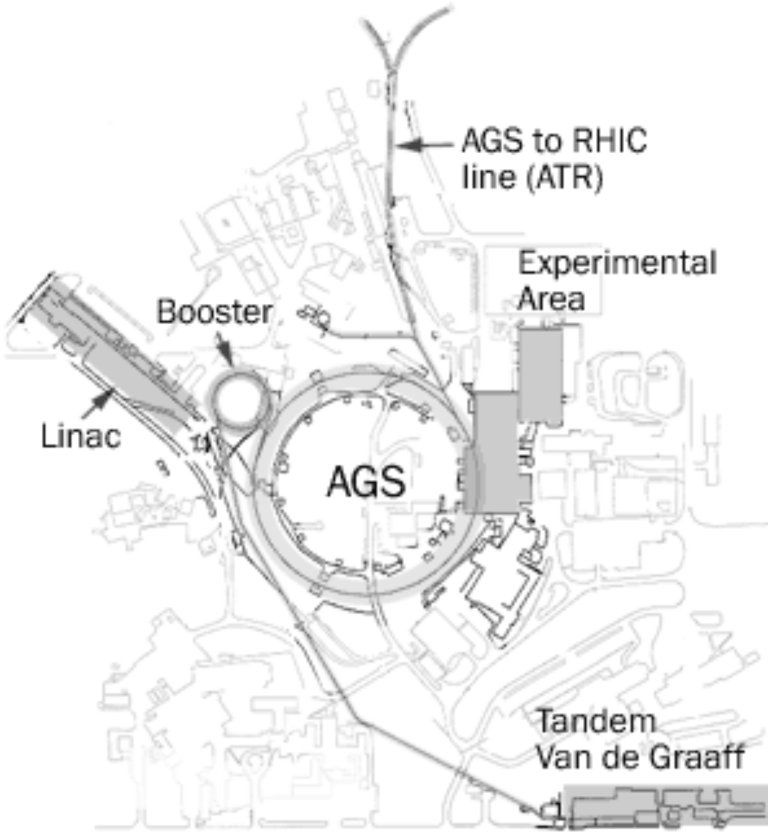
- Only operating heavy-ion colliders in the US, only spin-polarized proton collider ever built
- Two 3.8 km counter-rotating rings (Yellow & Blue) with superconducting magnets
- Six interaction regions (IR) where two rings cross

Simulation Studies and Machine Learning Applications for Orbit and Optics Correction at the Alternating Gradient Synchrotron

Brookhaven National Laboratory

CLASSE
Cornell Laboratory for Accelerator-based ScienceS & Education

Brightness control at the Alternating Gradient Synchrotron (AGS)



- Alternating gradient / strong focusing principle: achieve strong vertical and horizontal focusing of charged particle beam at the same time
- Accelerates proton to 33 GeV in 1960
- 12 super-periods (A to L), 240 main magnets, 810 m circumference
- Now serves as injector for Relativistic Heavy Ion Collider (RHIC)

Brookhaven National Laboratory
CLASSE Cornell Laboratory for Accelerator-based ScienceS and Education

Motivation: support for EIC Cooler

- Electron cooling for the EIC requires small incoming emittances from the AGS
- Necessary pre-cooler at RHIC injection energy (AGS extraction energy)
- Current AGS lacks systematic tuning routine, mostly hand tuned by operators
- Algorithm to better control beam in AGS will be helpful for future EIC cooler

Orbit Response Matrix (ORM)

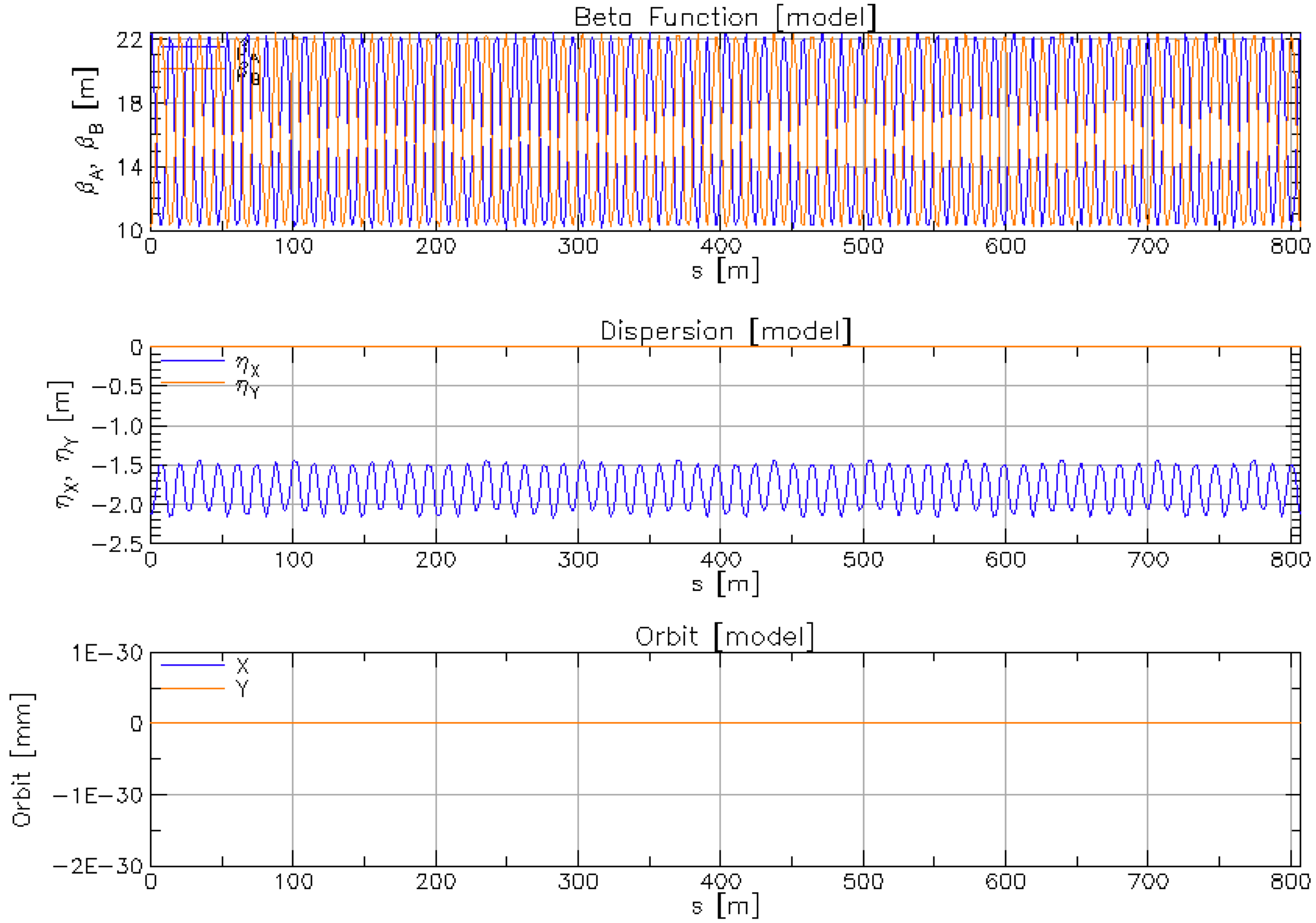
- Mapping $\underline{R}$ between closed orbit measurements and corrector settings
- 72 pick-up electrodes (PUE), 48 horizontal and vertical corrector pairs
- Linear orbit response to corrector change: calculate $\underline{R}$ matrix by changing each corrector pair separately
- Corrector current $I \rightarrow$ angle θ by calibration factor
- Traditional orbit correction: $\Delta\vec{\theta} = \underline{R}^{-1}\,\Delta\vec{y}$

$$\begin{pmatrix} \Delta\vec{x} \\ \Delta\vec{y} \end{pmatrix} = \underline{R} \begin{pmatrix} \Delta\vec{\theta}_x \\ \Delta\vec{\theta}_y \end{pmatrix}$$

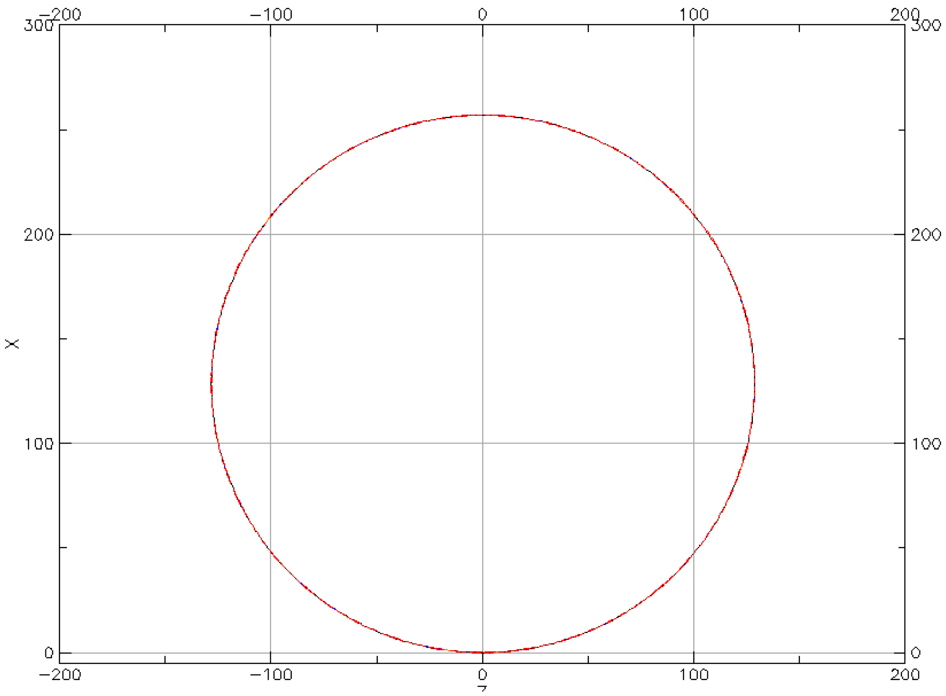
$$\frac{\Delta x_i}{\Delta\theta_j} = R_{ij}$$

MAD-X to BMAD translation

- Successfully translated bare machine to BMAD: ramping in progress
- Can use Python interface (PyTao) to run simulations much easier



Floor plan



BMAD and PyTao: best tool for ML

- Python interface: enable running simulations in Python scripts and Jupyter notebooks
- Get data with different control parameters in one go with for loops, without need to modify original lattice files
- Freedom to save data in any preferred form (i.e. combine into one huge data array and save in one file for easy fetch in the future)

Use ORM to identify machine errors

- Actual machine with errors (e.g. quadrupole gradient errors, corrector calibration errors, etc.) produce different $\underline{R}_{measured}$ from model/reference machine $\underline{R}_{model}$

$$\Delta R_{ij} = R_{ij}^{model} - R_{ij}^{measured}$$

- Considering all possible sources of errors as a vector $\vec{v}$, build response error model $\underline{J}_{model}$

$$\begin{pmatrix} \Delta R_{11} \\ \Delta R_{12} \\ \dots \\ \Delta R_{n(m-1)} \\ \Delta R_{nm} \end{pmatrix} = \underline{J}_{model} \begin{pmatrix} \Delta \nu_1 \\ \Delta \nu_2 \\ \dots \\ \Delta \nu_{N-1} \\ \Delta \nu_N \end{pmatrix}$$

- Reconstruct any $\vec{v}$ given known $\Delta\vec{R}$ and $\underline{J}_{model}$

Reconstruct errors using SVD

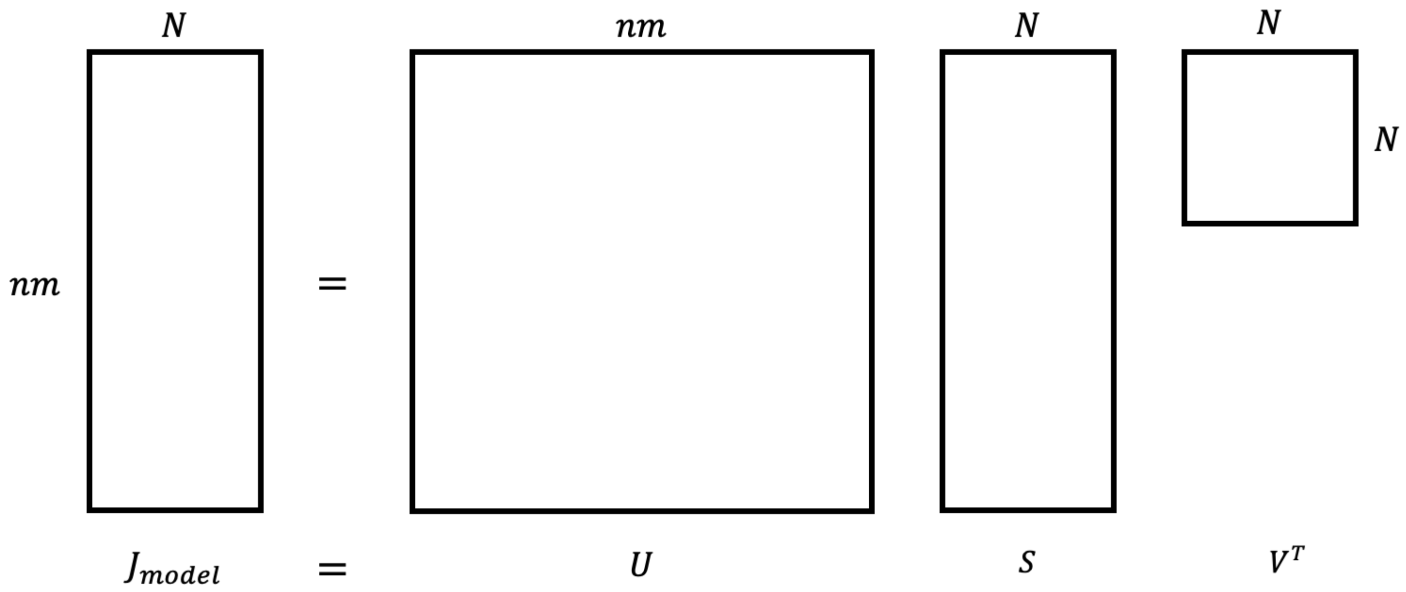
- Traditional tuning routine: perform singular value decomposition (SVD) directly on $\underline{R}$
- Machine error detection: perform SVD on $\underline{J}_{model}$
- Solve for $\Delta\vec{v}$ using $\Delta\vec{R} = \underline{J}_{model}\,\Delta\vec{v}$, where $\underline{J}_{model}$ is not a square matrix

$$J_{model} = USV^T$$

$$n = N_{corr}, m = N_{BPM}$$

$$\Delta\vec{R}: (48 \times 72, 1)$$

$$\underline{J}_{model}: (3456, N_{error})$$



Test case: quadrupole strength error

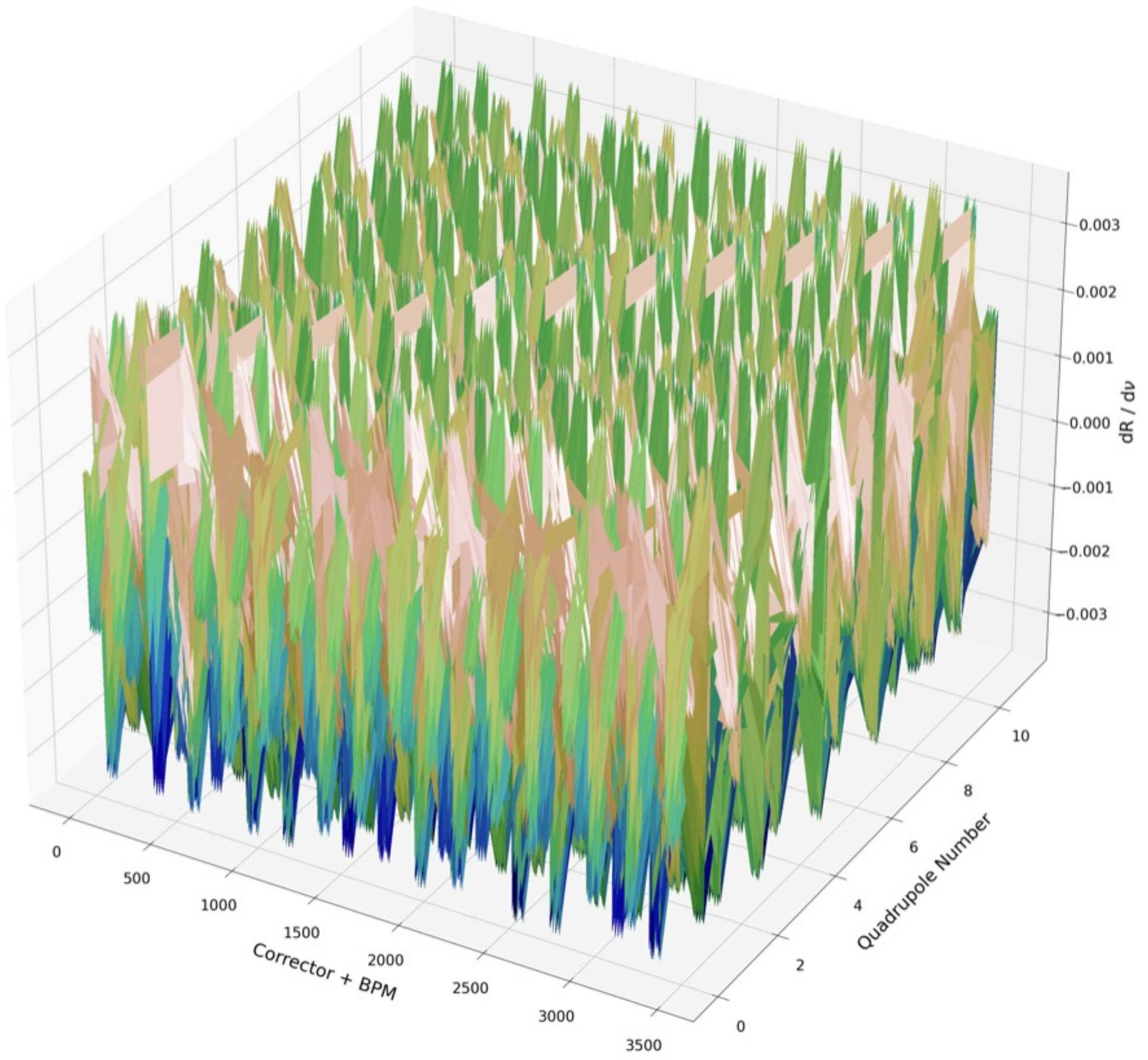
- 24 quadrupoles (12 horizontal, 12 vertical), 1 in each super-period
- Linear orbit response to quadrupole kick change: calculate $\Delta\vec{R} = \underline{R}_{measured} - \underline{R}_{ref}$ by changing each quadrupole separately $\rightarrow J_{ijk} = \frac{\Delta R_{ij}}{\Delta \upsilon_k}$
- Quad kick defined with one variable KQH/KQV in MAD-X → variables in BMAD allow separate change of quad kicks

```
tao.cmd('show var quads.x')
```

```
['  Variable                   Slave Parameters         Meas        Model       Design  Useit_opt',
 '  quads.x[1]                 QH_F17[K1]         0.0000E+00  -6.5349E-05  -6.5349E-05        T',
 '  quads.x[2]                 QH_G17[K1]         0.0000E+00  -6.5349E-05  -6.5349E-05        T',
 '  quads.x[3]                 QH_H17[K1]         0.0000E+00  -6.5349E-05  -6.5349E-05        T',
 '  quads.x[4]                 QH_I17[K1]         0.0000E+00  -6.5349E-05  -6.5349E-05        T',
 '  quads.x[5]                 QH_J17[K1]         0.0000E+00  -6.5349E-05  -6.5349E-05        T',
 '  quads.x[6]                 QH_K17[K1]         0.0000E+00  -6.5349E-05  -6.5349E-05        T',
 '  quads.x[7]                 QH_L17[K1]         0.0000E+00  -6.5349E-05  -6.5349E-05        T',
 '  quads.x[8]                 QH_A17[K1]         0.0000E+00  -6.5349E-05  -6.5349E-05        T',
 '  quads.x[9]                 QH_B17[K1]         0.0000E+00  -6.5349E-05  -6.5349E-05        T',
 '  quads.x[10]                QH_C17[K1]         0.0000E+00  -6.5349E-05  -6.5349E-05        T',
 '  quads.x[11]                QH_D17[K1]         0.0000E+00  -6.5349E-05  -6.5349E-05        T',
 '  quads.x[12]                QH_E17[K1]         0.0000E+00  -6.5349E-05  -6.5349E-05        T',
 '  Variable                   Slave Parameters         Meas        Model       Design  Useit_opt']
```

Test case $\underline{J}_{model}$ matrix (horizontal)

- Calculated using $\Delta\nu = 4$ A in power supply current for each quadrupole ($\pm 1\%$ in k1 value)
- Agreement with MAD-X model (redefined every quad individually) was obtained



Reconstruct errors using SVD

- $\underline{U}$ and $\underline{V}$ are square orthogonal matrices: $UU^T = VV^T = I$
- $\underline{S}$ is an $nm \times N$ matrix whose first N diagonal elements are singular values σ of $\underline{J}_{model}$

$$S = \begin{pmatrix} S_N \\ 0 \end{pmatrix} \in \mathbb{R}^{nm \times N}, \quad S_N := diag(\sigma_1, \ldots, \sigma_N, 0, \ldots, 0) \in \mathbb{R}^{N \times N}$$

- $\underline{S}^+$ is pseudoinverse of $\underline{S}$ whose first N diagonal elements are $\frac{1}{\sigma}$

$$S^+ = \begin{pmatrix} S_N^+ \\ 0 \end{pmatrix} \in \mathbb{R}^{N \times nm}, \quad S_N^+ := diag(\frac{1}{\sigma_1}, \ldots, \frac{1}{\sigma_N}, 0, \ldots, 0) \in \mathbb{R}^{N \times N}$$

$$\begin{pmatrix} \Delta\nu_1 \\ \Delta\nu_2 \\ \cdots \\ \Delta\nu_{N-1} \\ \Delta\nu_N \end{pmatrix} = J^+_{model} \begin{pmatrix} \Delta R_{11} \\ \Delta R_{12} \\ \cdots \\ \Delta R_{n(m-1)} \\ \Delta R_{nm} \end{pmatrix} = V S^+ U^T \begin{pmatrix} \Delta R_{11} \\ \Delta R_{12} \\ \cdots \\ \Delta R_{n(m-1)} \\ \Delta R_{nm} \end{pmatrix}$$

Test case: reconstruct errors with J_model

- Reconstructed error = quadrupole power supply current

Case 1: One quadrupole 1% (4A) error

```
# Quad A17 +4 Amp
np.dot(V, np.dot(S_inv, np.dot(UT, dr1)))
```

```
array([ 4.04152292e+00, -4.15488269e-05,  2.17313140e-05,  6.45374239e-05,
        4.03913733e-05,  3.09693635e-05,  2.76558248e-05, -4.31669566e-05,
       -1.36249941e-05,  4.91338661e-05, -6.14294896e-05,  3.19703471e-05])
```

Case 2: Two quadrupoles 0.5% (2A), 0.18% (0.7A) error

```
# Quad C17 +2 Amp, H17 +0.7 Amp
np.dot(V, np.dot(S_inv, np.dot(UT, dr2)))
```

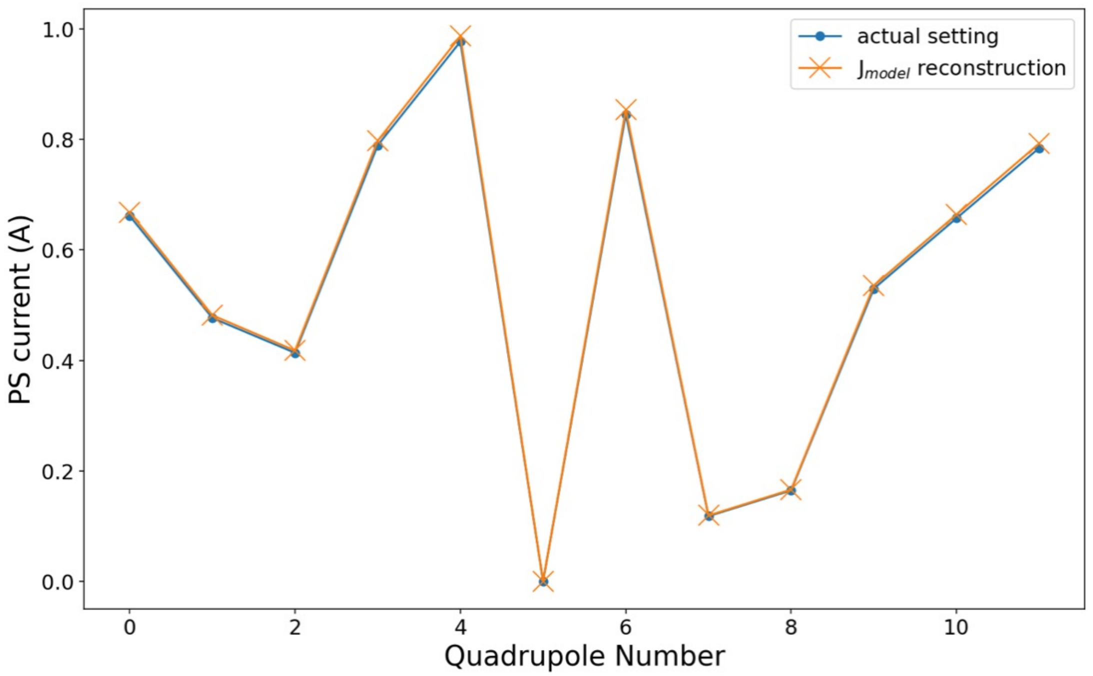
```
array([ 3.50482558e-05, -5.54479409e-05,  2.02147800e+00,  7.69381741e-05,
        5.06832047e-05,  4.13148646e-05,  4.02598848e-05,  7.07636616e-01,
       -2.78341654e-05,  4.27531143e-05, -6.90270247e-05,  2.50657000e-05])
```

Case 3: Three quadrupoles 0.75% (3A), 0.02% (0.08A), 0.25% (1A) error

```
# Quad B17 +3 Amp, F17 +0.08 Amp, J17 +1 Amp
np.dot(V, np.dot(S_inv, np.dot(UT, dr3)))
```

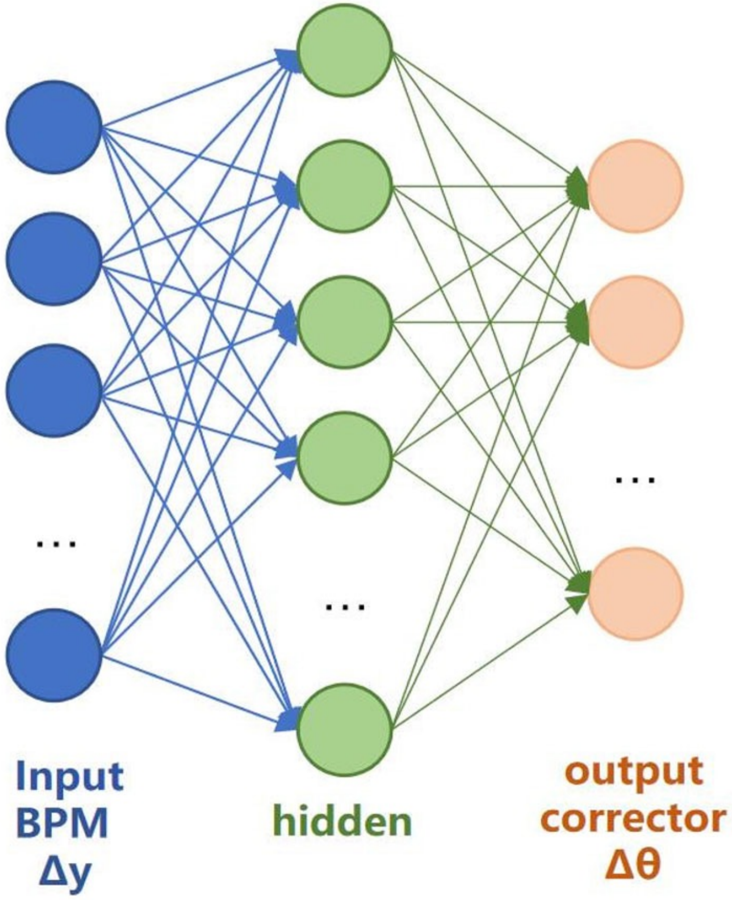
```
array([ 6.97595445e-05,  3.03074518e+00, -1.42673230e-05,  8.18292016e-06,
        6.05175589e-05,  8.07700864e-02,  4.40237777e-05, -8.92267806e-05,
       -4.99647748e-05,  1.01013295e+00, -2.99336376e-05, -2.01460387e-04])
```

Case 4: All quadrupoles random error within 0.25%



Satisfactory reconstruction results

Neural Network for real-time ORM

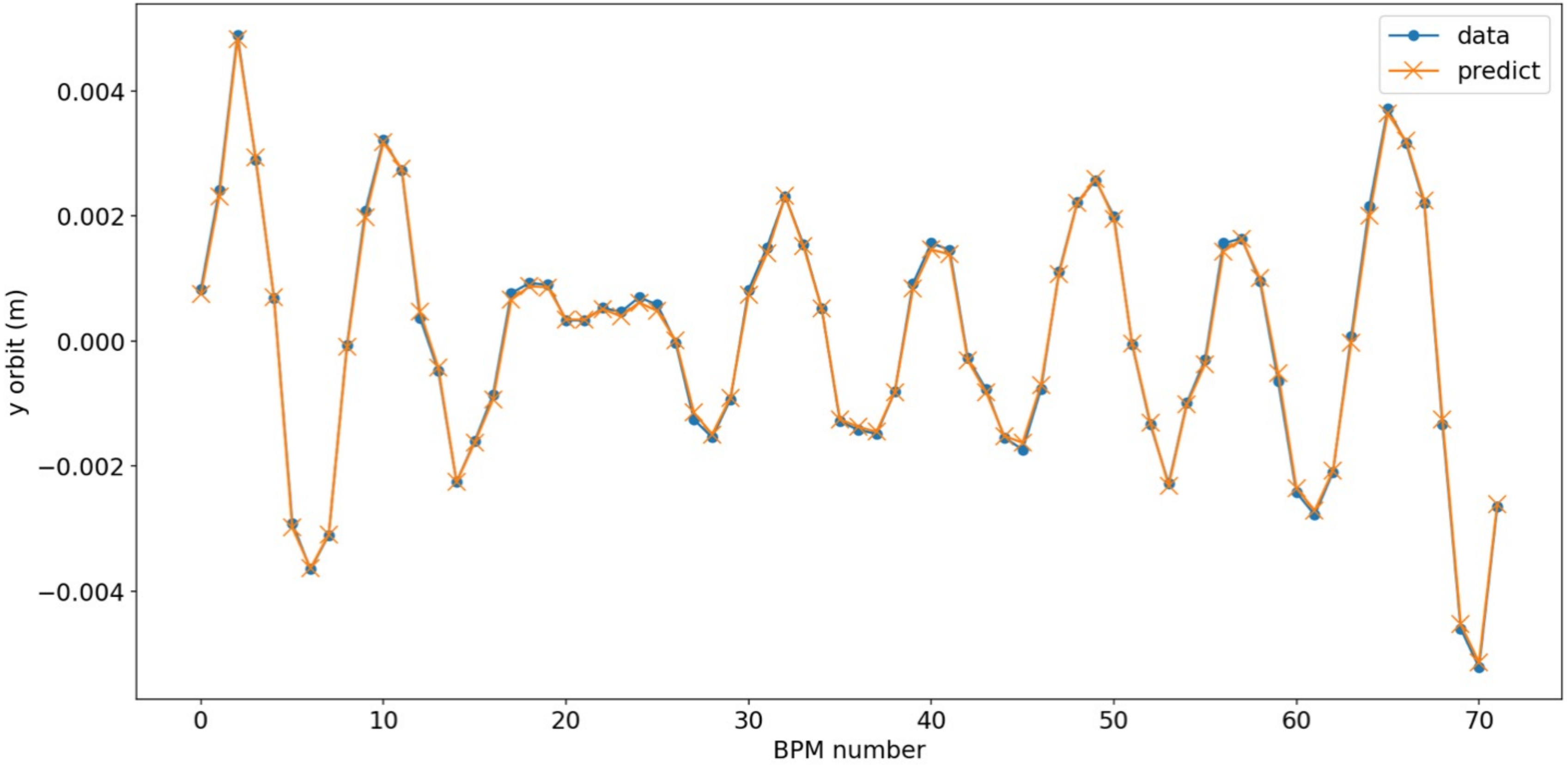


- Need dedicated machine time to measure ORM $\underline{R}_{measured}$: at least 30 min
- Pre-measured $\underline{R}_{measured}$ gets less accurate with time → orbit drift / brightness drop
- Update ORM with real-time data: build neural network model for $\underline{R}_{measured}$ or $\underline{R}_{measured}^{-1}$
- Can be used to calculate $\Delta\vec{R}$ for machine error reconstruction

ORM NN model: training results

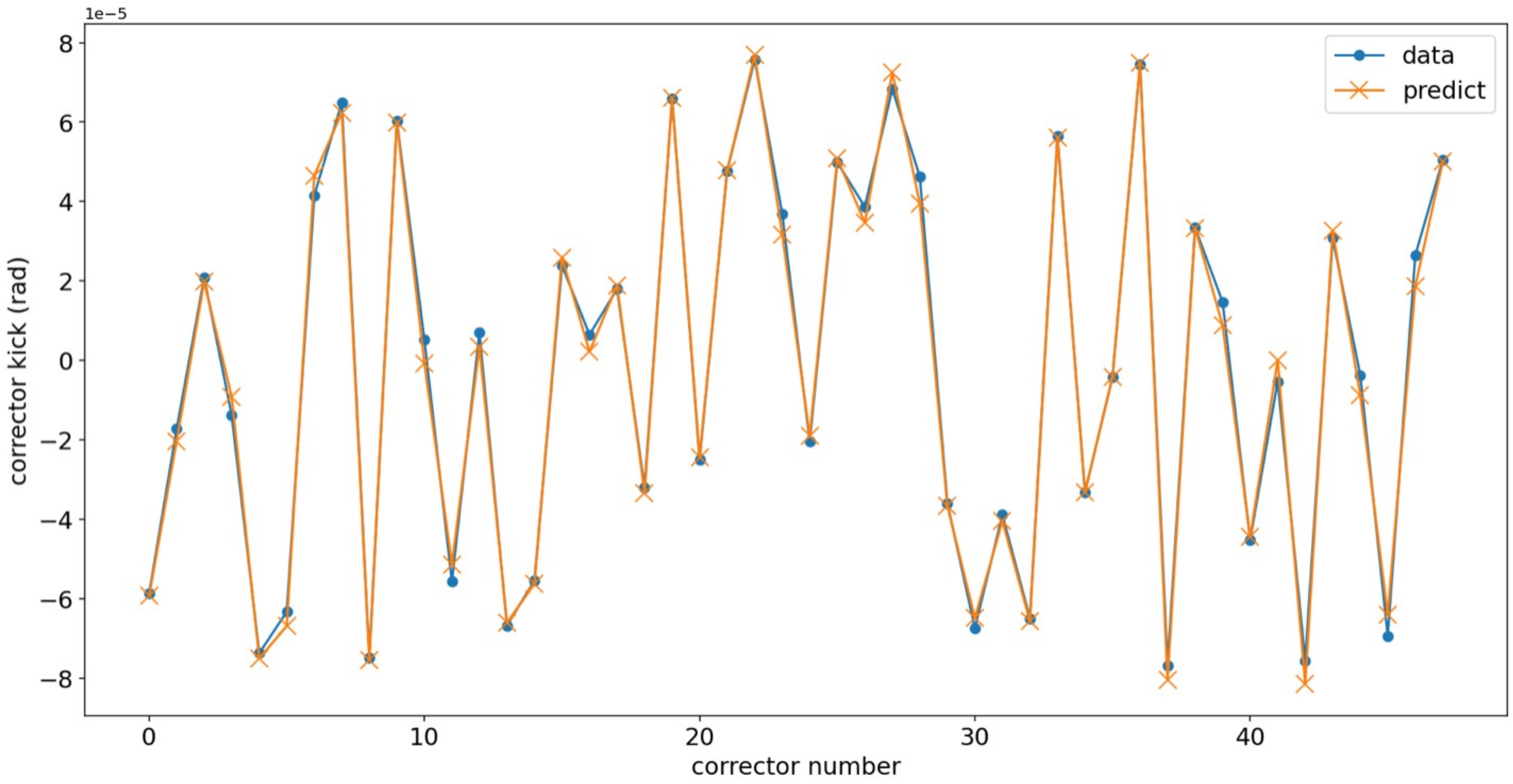
- Input 48 vertical corrector kick → Output 72 y orbit measured at BPM
- FFNN with one hidden layer and Tanh activation
- Trained on 800 data pairs, tested on 200 data pairs: R^2 score = 0.998

$$R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \overline{y}_i)^2}$$



Inverse ORM NN model: training results

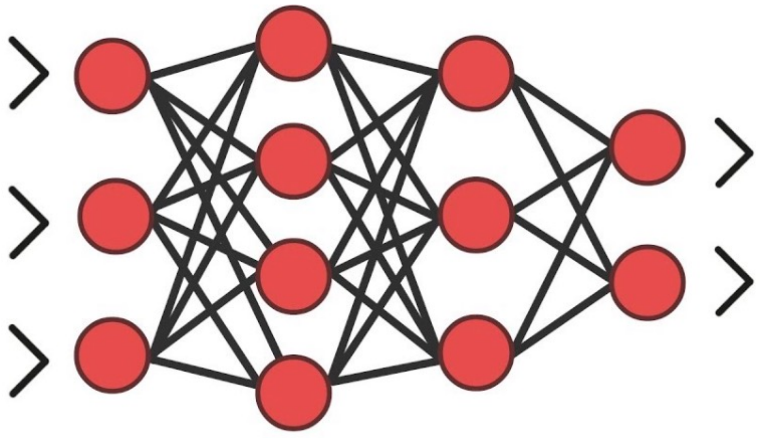
- Input 72 y orbit measured at BPM → Output 48 vertical corrector kick
- FFNN with one hidden layer and Tanh activation
- Trained on 800 data pairs, tested on 200 data pairs: R^2 score = 0.993



Sensitivity studies for ORM

- Scan through some common sources of error to see how much ORM changes
- Find relevant parameters to include for building error-detecting model
- <u>**Goal**</u>: establish a neural network that identify error source given a measured ORM

$$\begin{pmatrix} \Delta\nu_1 \\ \Delta\nu_2 \\ \cdots \\ \Delta\nu_{N-1} \\ \Delta\nu_N \end{pmatrix} = J^{+}_{model} \begin{pmatrix} \Delta R_{11} \\ \Delta R_{12} \\ \cdots \\ \Delta R_{n(m-1)} \\ \Delta R_{nm} \end{pmatrix}$$



Sensitivity studies: error sources

- Sources or error and ranges come from past survey data

- Criteria to quantify & visualize sensitivity:
  - RMS of double-plane ORM matrix
  - Beta-beating (vertical & horizontal)

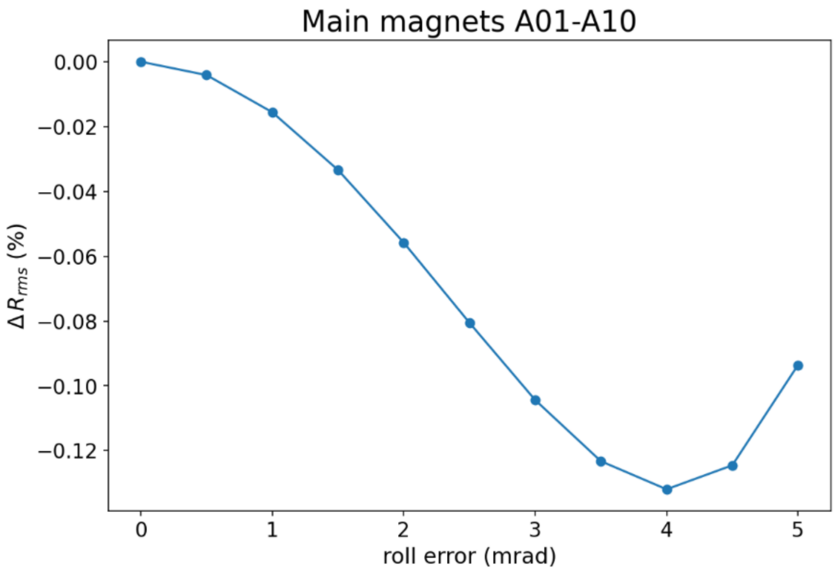
$$\frac{\Delta\beta}{\beta} = \frac{\beta_{measured} - \beta_{model}}{\beta_{model}}$$

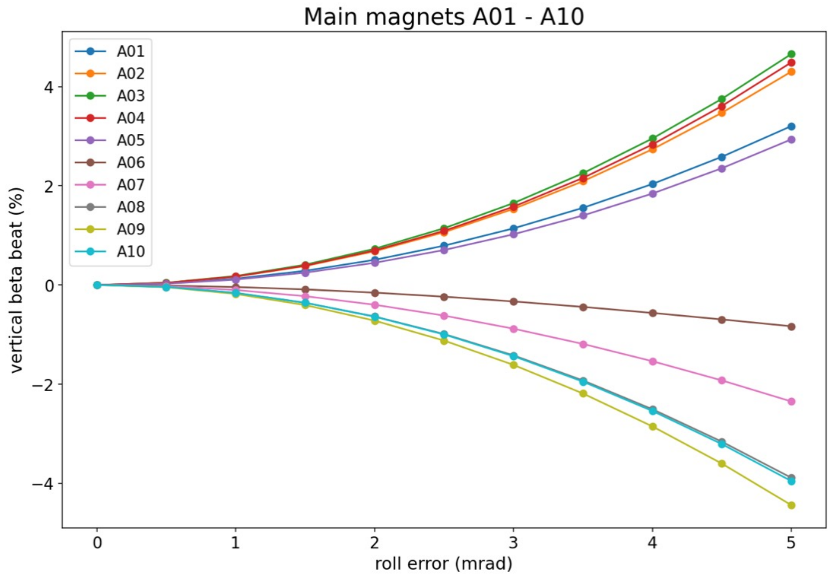
Name	Unit	Range
Main magnet roll error	mrad	[-0.5, 0.5]
Main magnet gradient error	m^{-2}	$\pm 0.1\%$
Quadrupole gradient error	m^{-2}	$\pm 0.2\%$
Sextupole offset error	mm	[-8, 8]
Snake magnet roll error	mrad	[-1.5, 1.5]

Main magnet roll error

- 240 main magnets, 20 magnets (01 to 20) in each super-period (A to L)
- Combined function magnets: dipole (Rbend) with non-zero k1, k2
- Scan range: ± 5 mrad with strong systematic super-periodicity (01 to 10 rolls one way, 11 to 20 rolls another way)

Magnet	$\Delta \underline{R}^{rms}$ (%)	$\Delta \beta_x$ (%)	$\Delta \beta_y$ (%)
01 - 10	[-0.13, 0]	[-2.5, 4.5]	[-4.5, 4.7]
11 - 20	[-0.1, 0.52]	[-5.7, 5.6]	[-8.5, 9.3]





Main magnet gradient error

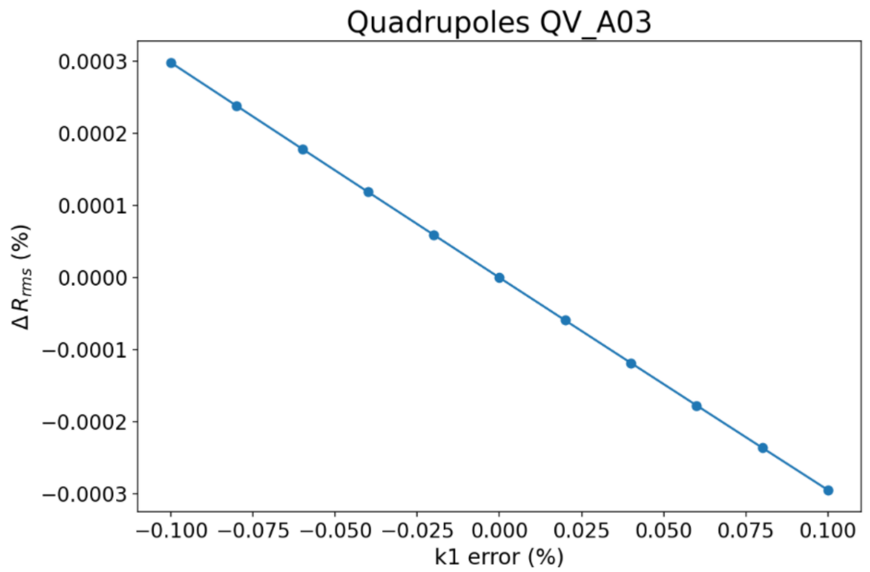
- 240 main magnets, 20 magnets (01 to 20) in each super-period (A to L), six families: AD, AF, BD, BF, CD, CF; BF*2 + CD*2 + AF*2 + CD*2 + BF*2 + BD*2 + CF*2 + AD*2 + CF*2 + BD*2
- Two different lengths: A and C 94 in, B 79 in
- Scan range: $\pm 0.1\%$ in k1 values

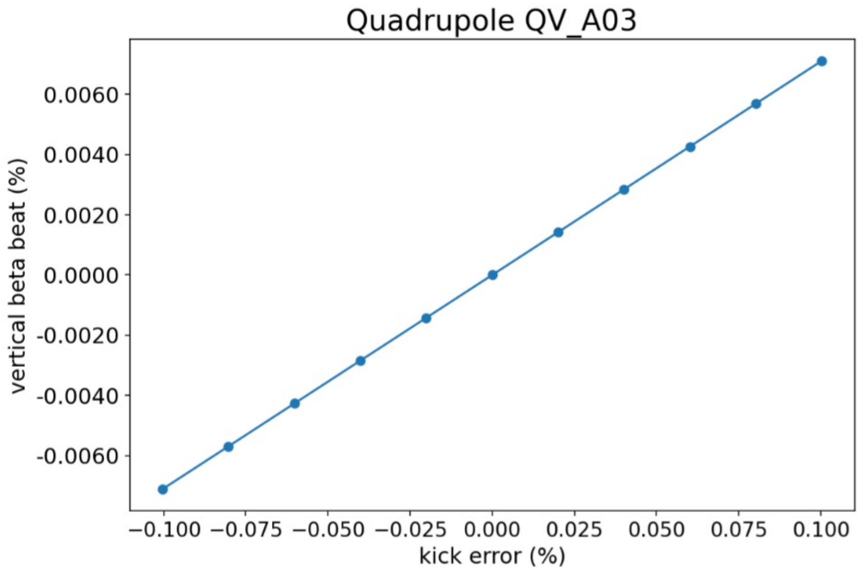
Family	$\Delta \underline{\boldsymbol{R}}^{rms}$ (%)	$\Delta \boldsymbol{\beta}_x$ (%)	$\Delta \boldsymbol{\beta}_y$ (%)
AD	[-1.6, 1.8]	$\pm$ 0.08	$\pm$ 0.1
AF	[-0.01, 0.11]	$\pm$ 0.12	$\pm$ 0.09
BD	[-2.34, 2.87]	$\pm$ 0.06	$\pm$ 0.1
BF	[-0.14, 0.46]	$\pm$ 0.1	$\pm$ 0.06
CD	[-2.11, 2.72]	$\pm$ 0.23	$\pm$ 0.29
CF	[-0.73, 1.18]	$\pm$ 0.34	$\pm$ 0.23

Quadrupole kick error

- 24 quadrupole magnets (12 horizontal, 12 vertical), one (17 for QH, 03 for QV) in each super-period
- Scan range: $\pm 0.1\%$ in k1 values

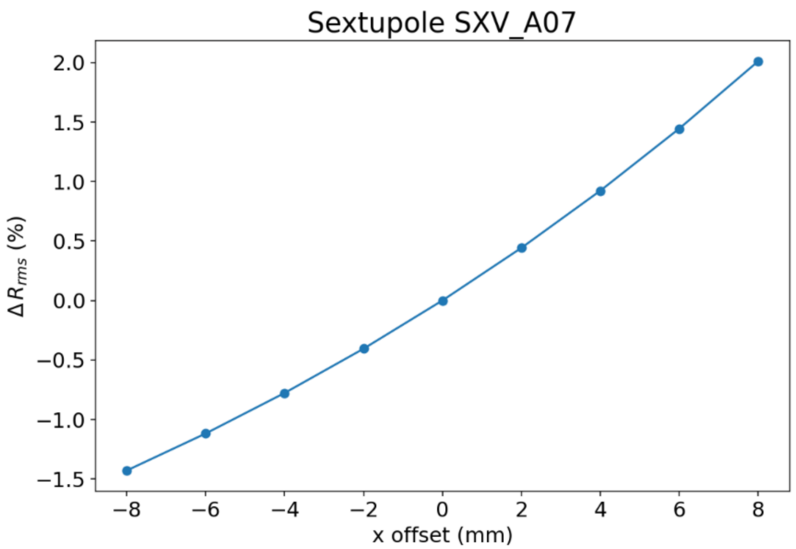
Magnet	$\Delta \underline{R}^{rms}$ (%)	$\Delta \boldsymbol{\beta}_x$ (%)	$\Delta \boldsymbol{\beta}_y$ (%)
QH	± 0.0048	± 0.0015	± 0.007
QV	± 0.00037	± 0.0049	± 0.0044



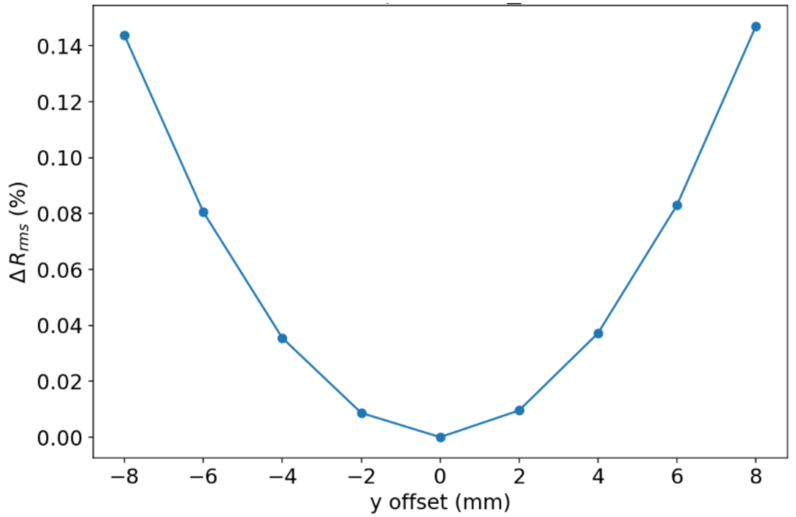


Sextupole offset error

- 28 sextupole magnets (14 horizontal, 14 vertical), 2 chromaticity sextupoles (13 for SXH, 07 for SXV) per super-period
- Scan range: ± 8 mm in x, y offset

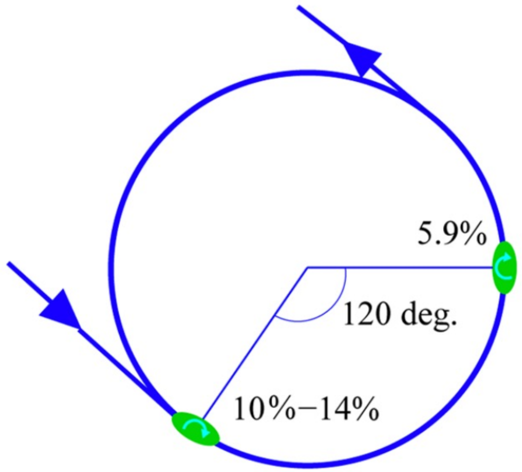


Source	$\Delta \underline{R}^{rms}$ (%)	$\Delta \beta_x$ (%)	$\Delta \beta_y$ (%)
SXH x-off	[-0.39, 0.6]	[-1.04, 1.05]	[-1.29, 1.55]
SXV x-off	[-1.4, 2]	[-0.9, 0.8]	[-2.46, 3.04]
SXH y-off	[0, 0.11]	[-0.017, 0.005]	[0, 0.07]
SXV y-off	[0, 0.15]	[-0.005, 0.025]	[0. 0.14]

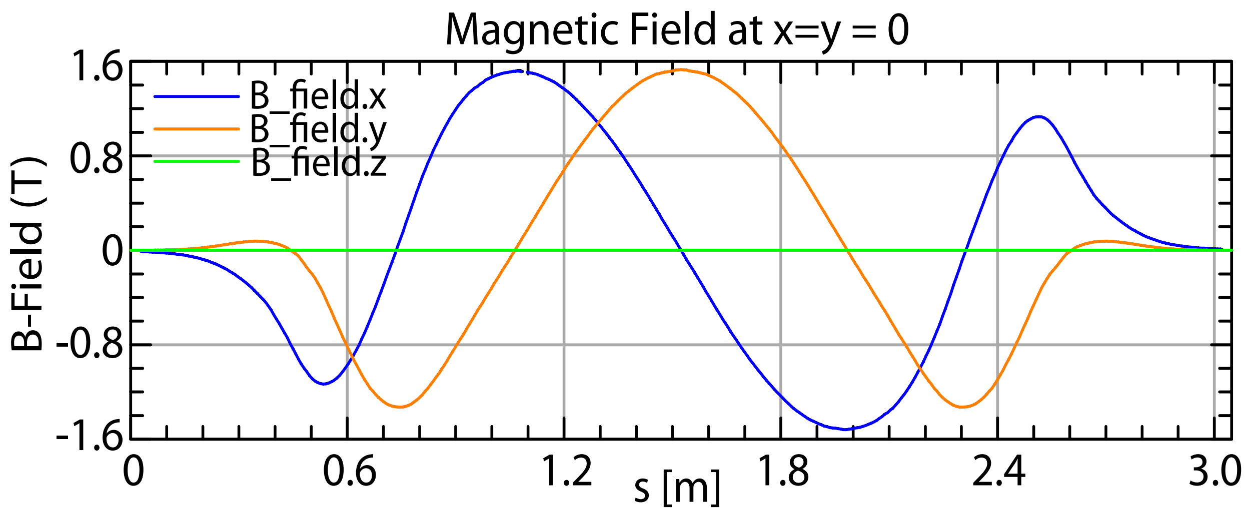


Siberian Snakes with generalized gradient

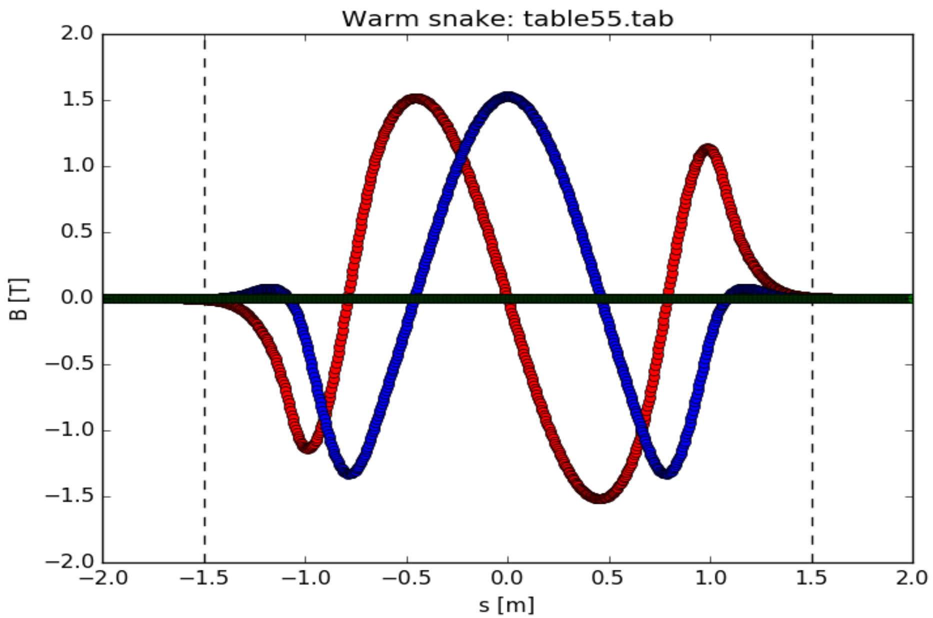
- 2 partial Siberian snakes (helical dipoles) to overcome depolarizing spin resonances
- Acceleration of 1.5e11 protons/bunch to 24 GeV with 65% polarization was achieved using 5.9% and 10% helical partial snakes
- Reproduce snakes in Bmad using generalized gradient



Generalized gradient

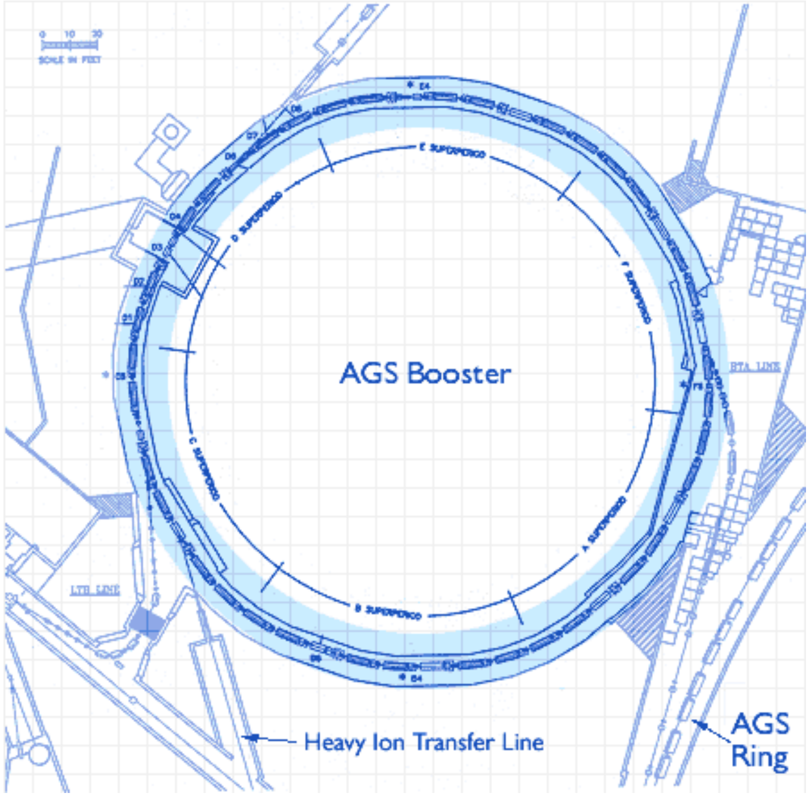


Grid fieldmap



Beam-based Quadrupole Transfer Function Measurement with Neural Network at Alternating Gradient Synchrotron (AGS) Booster

Alternating Gradient Synchrotron (AGS) Booster



- Pre-accelerate particles entering the AGS ring
- Accepts heavy ions from EBIS or protons from 200 MeV Linac
- Serves as heavy ion source for NASA Space Radiation Laboratory (NSRL)
- 6 super-periods (A to F), 72 main magnets

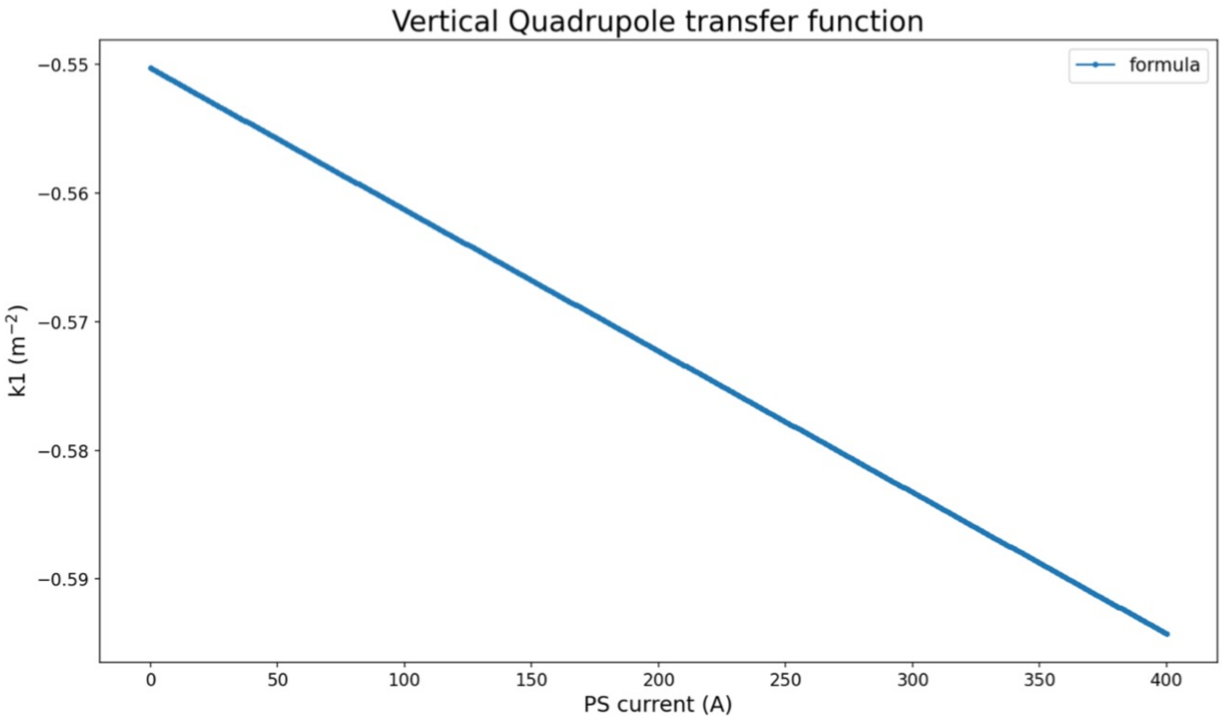
Quadrupoles in Booster

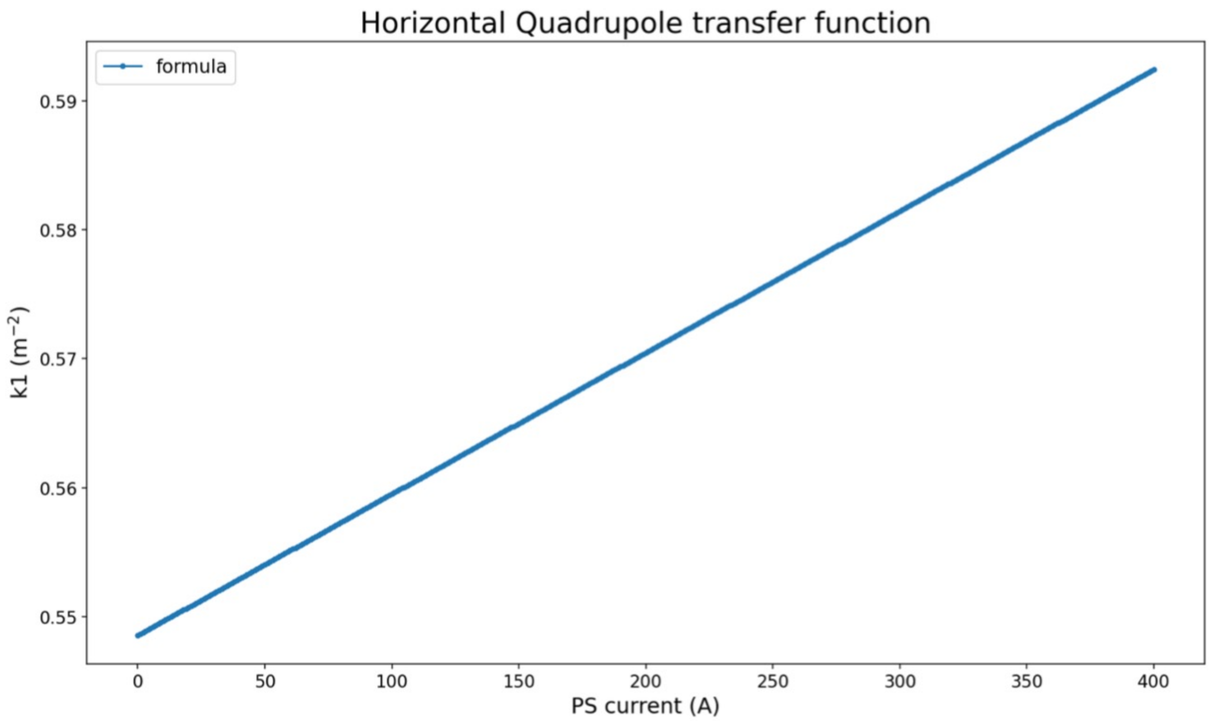
- 24 horizontal (short) + vertical (long) pairs, 48 in total
- Wired in series in each plane (IQHC/IQVC), also in series with dipoles (IDIPO)
- Extra term to compensate for back EMF due to $\dot{B}$, stop band correctors (QVSTR/QHSTR) to avoid tune resonances
- Transfer function coefficients matched to 5th order, different for horizontal and vertical

```
klvc0 = 0.002099
klvc1 = 9.257E-4
klvc2 = 1.164E-8
klvc3 = 1.046E-11
klvc4 = 4.057E-15
klvc5 = 5.75E-19

iqv = idipo + ckc * (iqvc + bdot*iqvbd) + ckc2*(qvstr1)

b1lv = klvc0 + iqv * klvc1 + iqv**2 * klvc2 - iqv**3 * klvc3 + iqv**4 * klvc4 - iqv**5 * klvc5

k1v = - (1 - 0.000041942 * (bdot/bdipo(idipo))) *1.0030*b1lv / (brho(idipo)*lenqv)
```

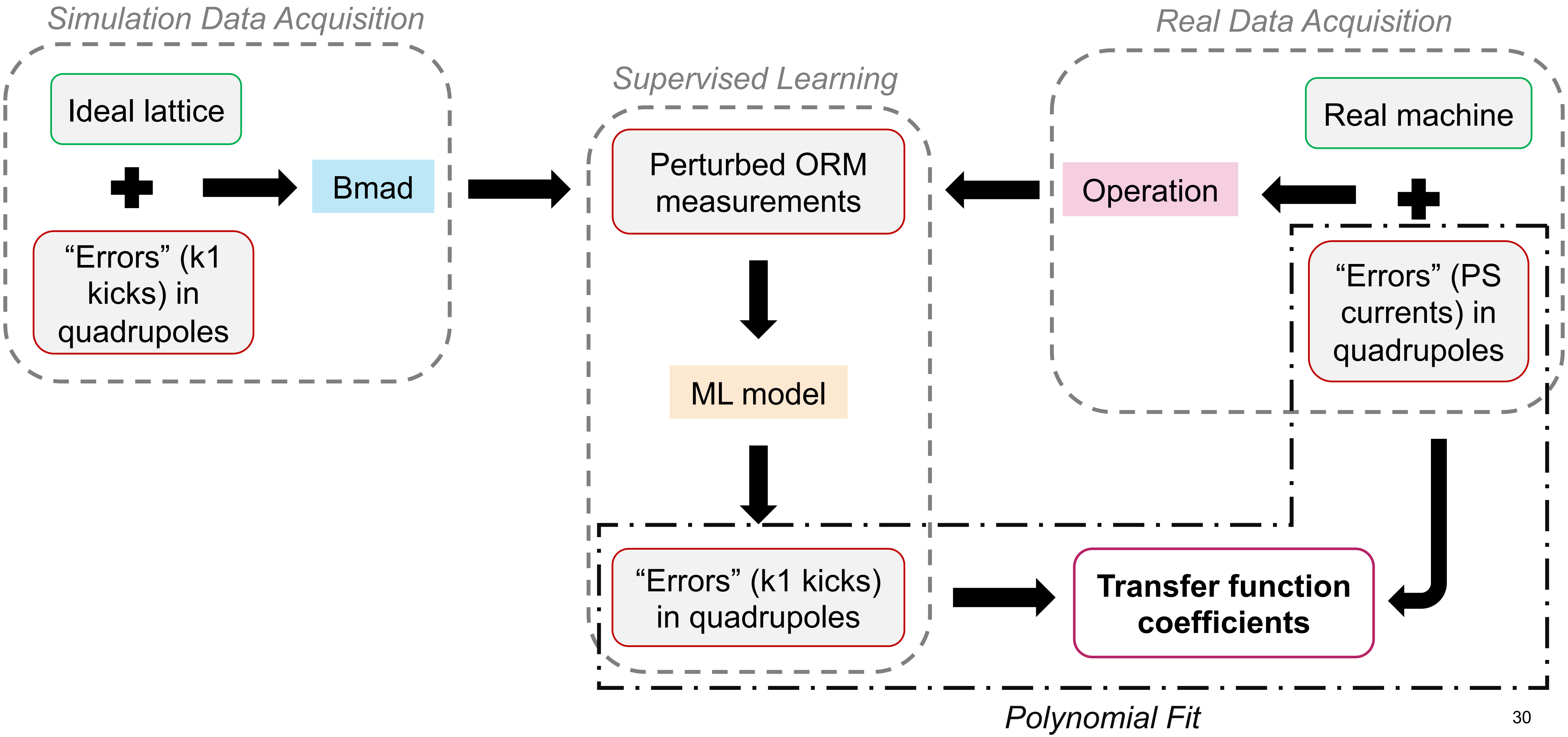
Quadrupole transfer functions

- Plan to take data on flat porch in Booster cycle: constant dipole current (IDIPO), constant B ($\dot{B} = 0$), no extra stop band correction (QVSTR/QHSTR = 0)
- Only variable is quadrupole power supply current (IQHC/IQVC)



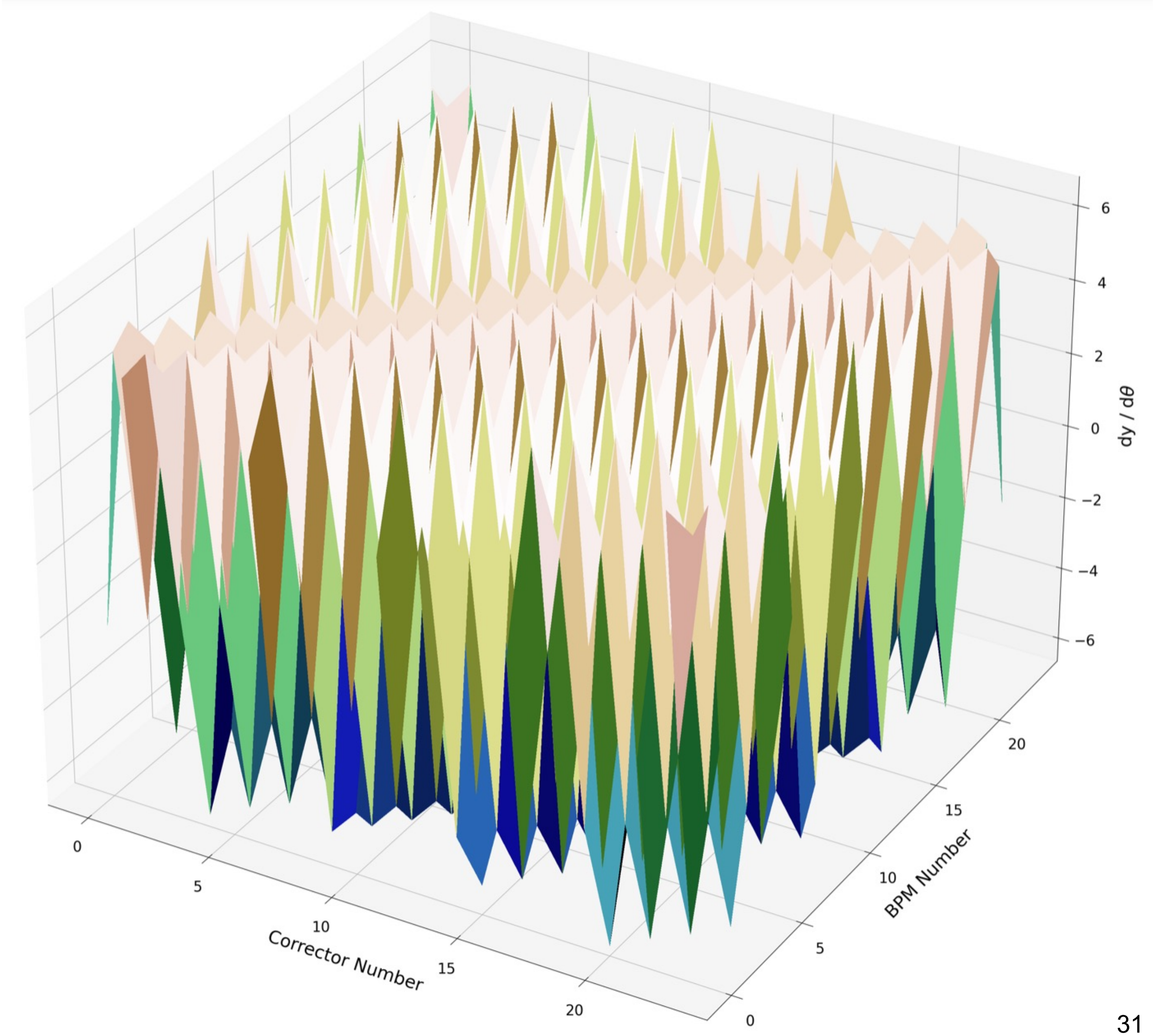


AGS Booster: transfer function coefficients



Correctors & BPMs in Booster

- 24 horizontal + vertical pairs, 48 in total
- Each corrector followed by a pick-up electrode (PUE), 48 PUEs in total
- ORM R in each plane will have dimension (24,24)

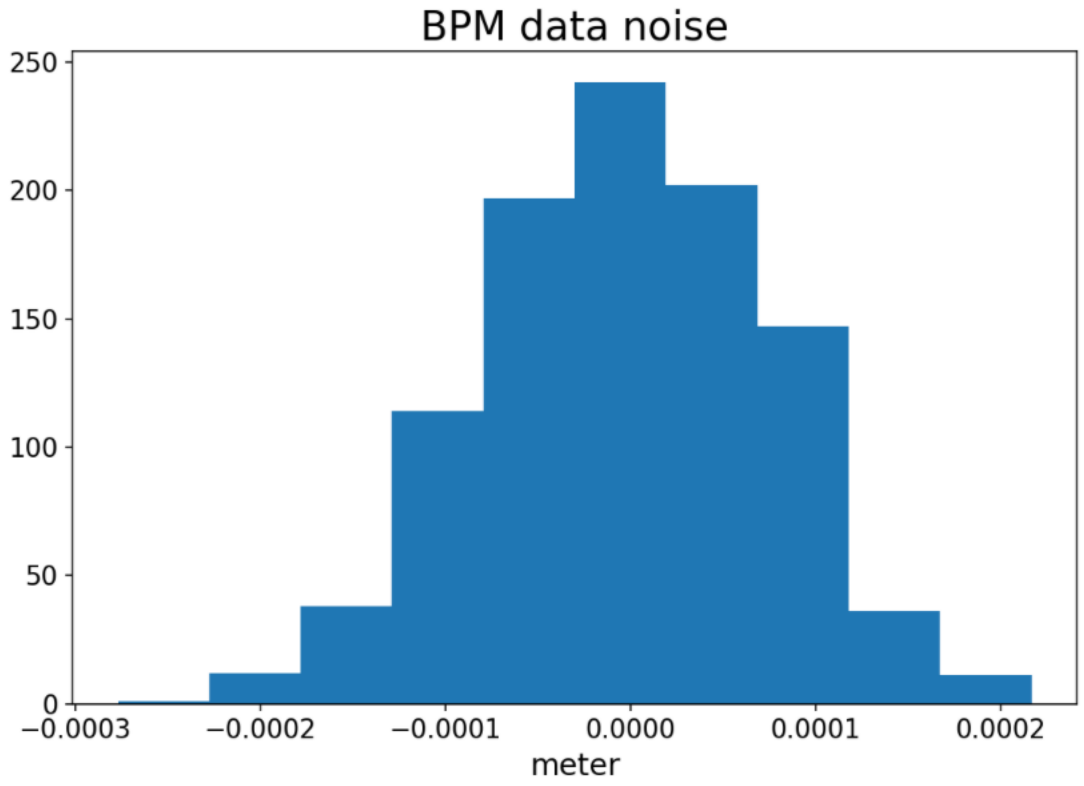


Quadrupole current scan for model training

- Get model ORM R_{model} from lattice with all quadrupole currents set to zero
- Add k1 values to quadrupoles for PS currents within a range, get corresponding ORM R_{meas}
- Good range: IDIPO = 1540 A, quad PS current 0 – 200 A, corrector current $\pm$10 A, which leads to orbit distortion of 5 to 6 mm at maximum
- Training dataset: $dR = (R_{meas} - R_{model}).\mathrm{flatten}()$ as input with shape (N, 576), quads k1 value as output with shape (N, 1) since they are wired in series
- Test ML model after training by doing a sequential PS current scan from 0 to 200 A, check whether the predicted k1 values fit the known polynomial pattern

Add noise to data

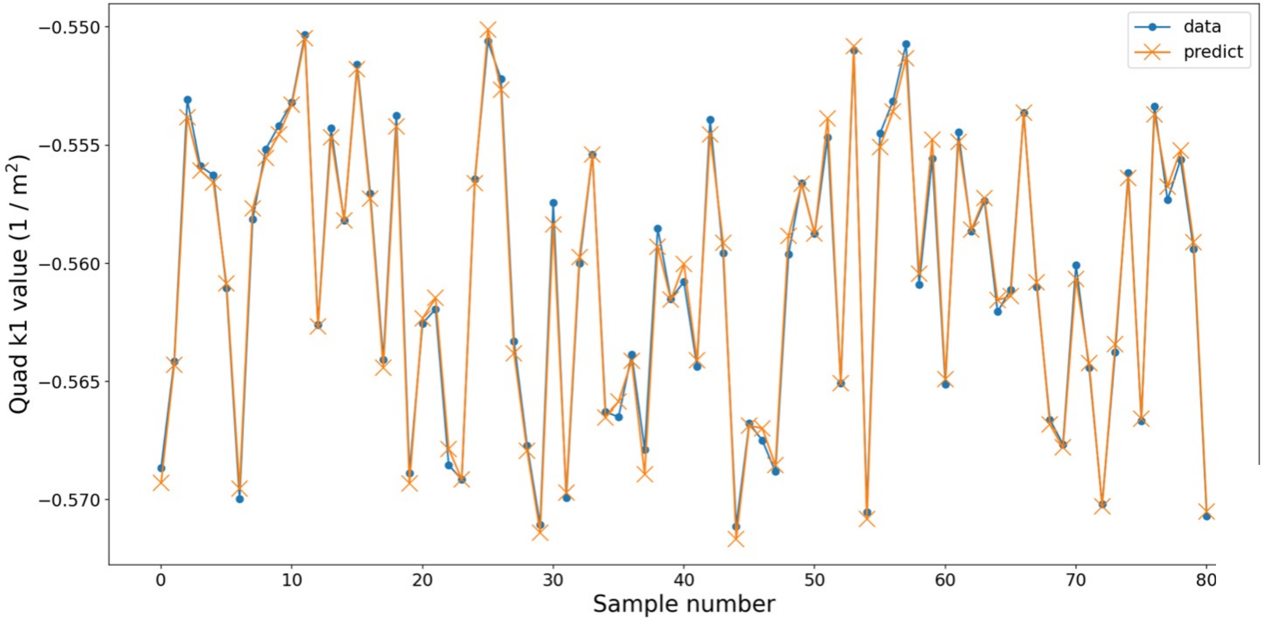
- Gaussian BPM noise: unit width ($\sigma = 1$), centered at zero ($\mu = 0$), amplitude $A = 80\mu m$

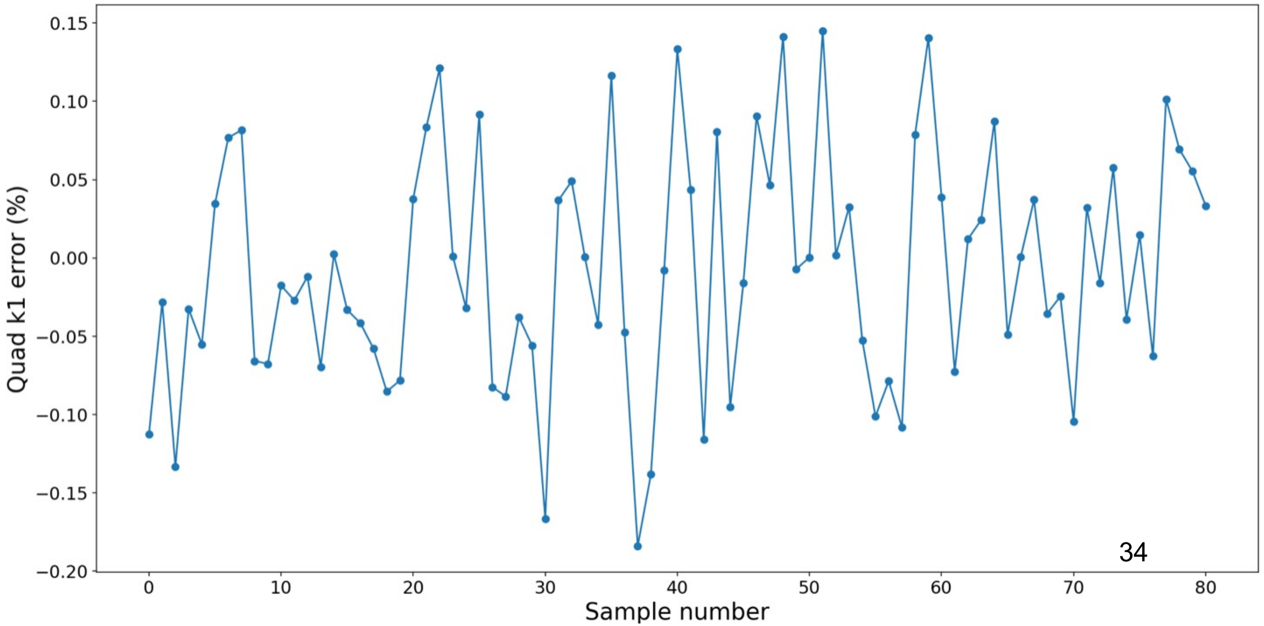


NN model for dR to k1

- One hidden layer NN with ELU activation

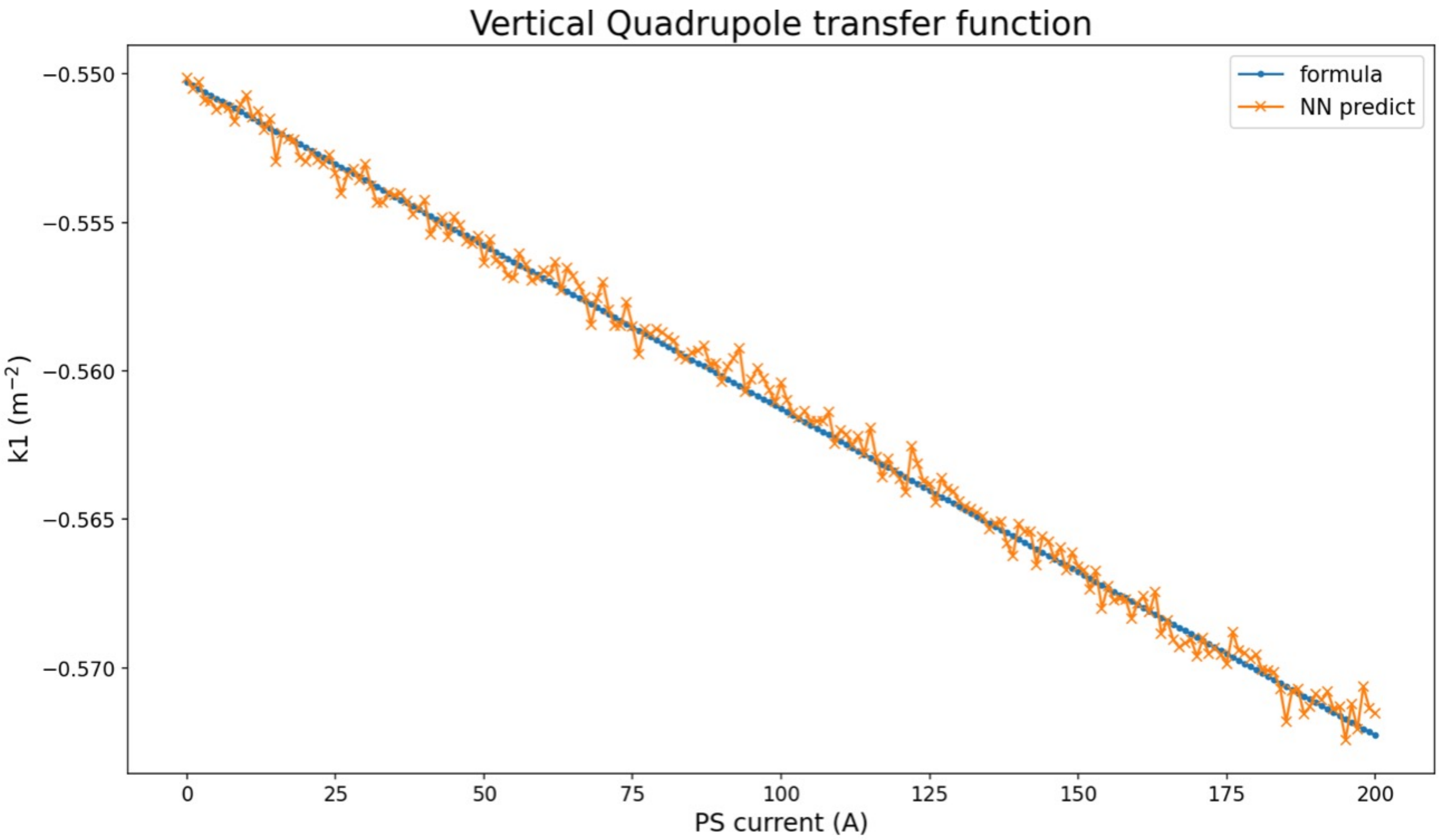
$R^2 = 0.995$





NN model for dR to k1

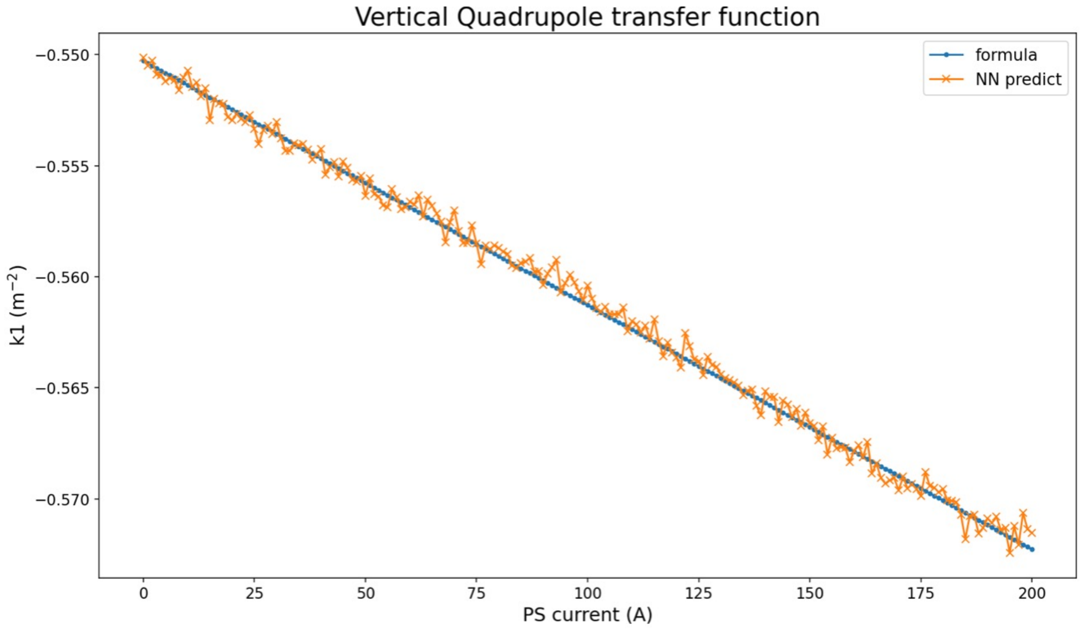
- One hidden layer NN with ELU activation



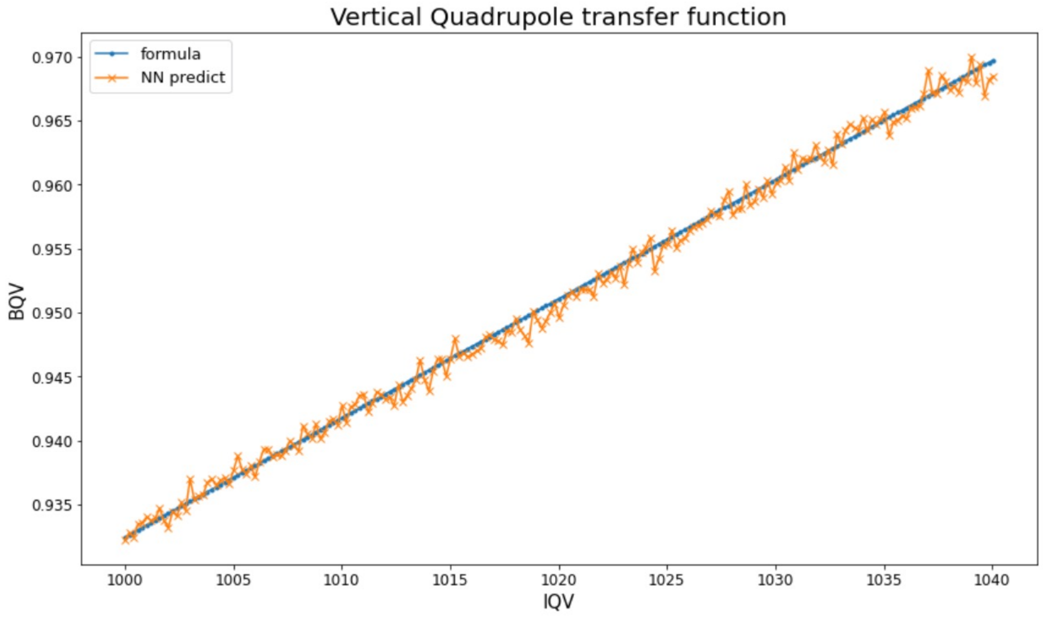
Polynomial fit for transfer function

```
klvc0 = 0.002099
klvc1 = 9.257E-4
klvc2 = 1.164E-8
klvc3 = 1.046E-11
klvc4 = 4.057E-15
klvc5 = 5.75E-19

iqv = idipo + ckc * (iqvc + bdot*iqvbd) + ckc2*(qvstr1)

b1lv = klvc0 + iqv * klvc1 + iqv**2 * klvc2 - iqv**3 * klvc3 + iqv**4 * klvc4 - iqv**5 * klvc5

k1v = - (1 - 0.000041942 * (bdot/bdipo(idipo))) *1.0030*b1lv / (brho(idipo)*lenqv)
```

Original data: iqvc -> k1v



Fit data: iqv -> b1lv



Polynomial fit: numpy.polyfit

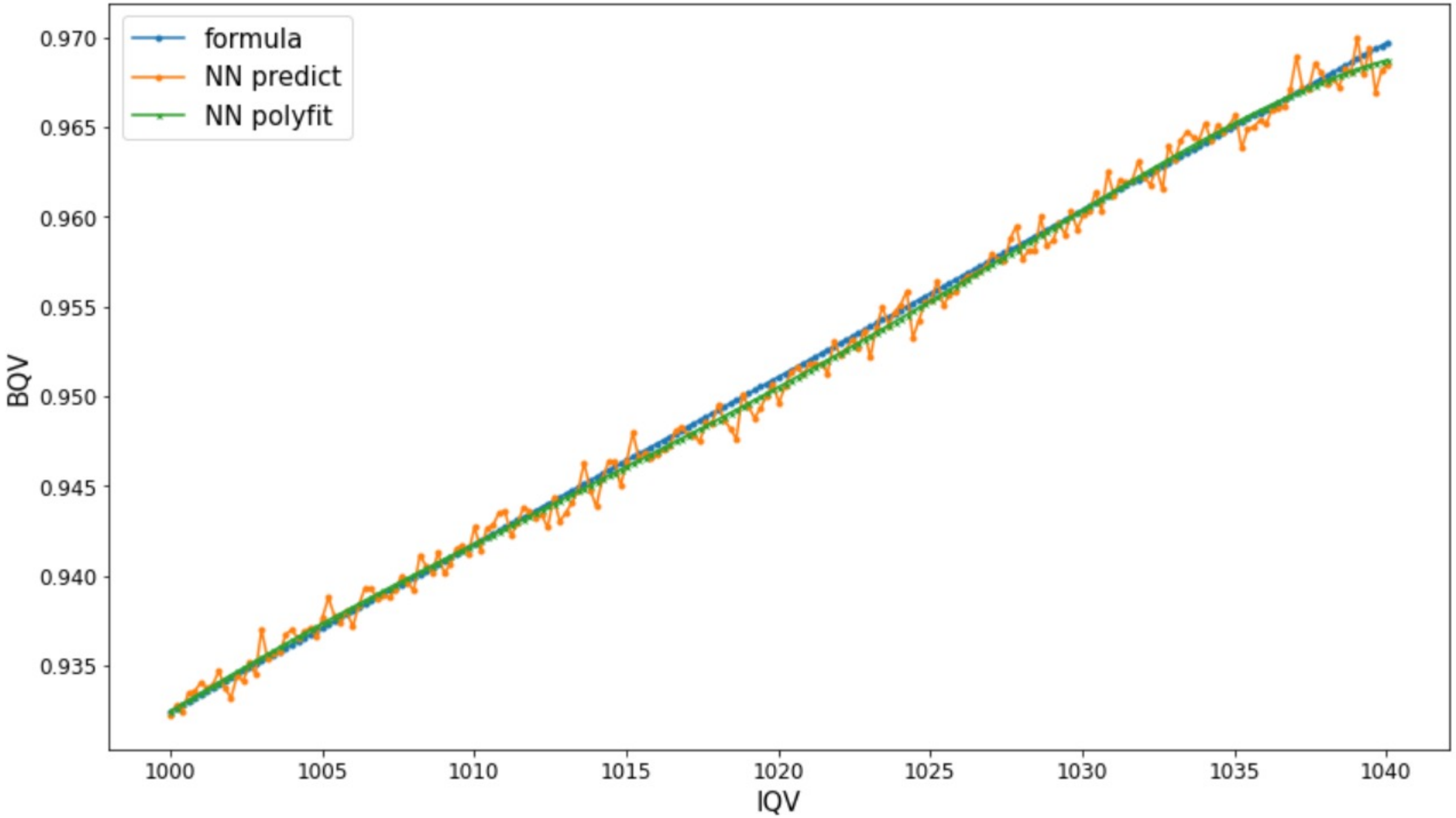
```
klvc0 = 0.002099
klvc1 = 9.257E-4
klvc2 = 1.164E-8
klvc3 = 1.046E-11
klvc4 = 4.057E-15
klvc5 = 5.75E-19
```

```
xx = np.arange(0,201,1)
coefs = np.polyfit(xx * ckc + 1000, b_formula, 5)
poly = np.poly1d(coefs)
coefs
```

```
array([-5.74944942e-19,  4.05671872e-15, -1.04594252e-11,  1.16394128e-08,
        9.25700300e-04,  2.09893872e-03])
```

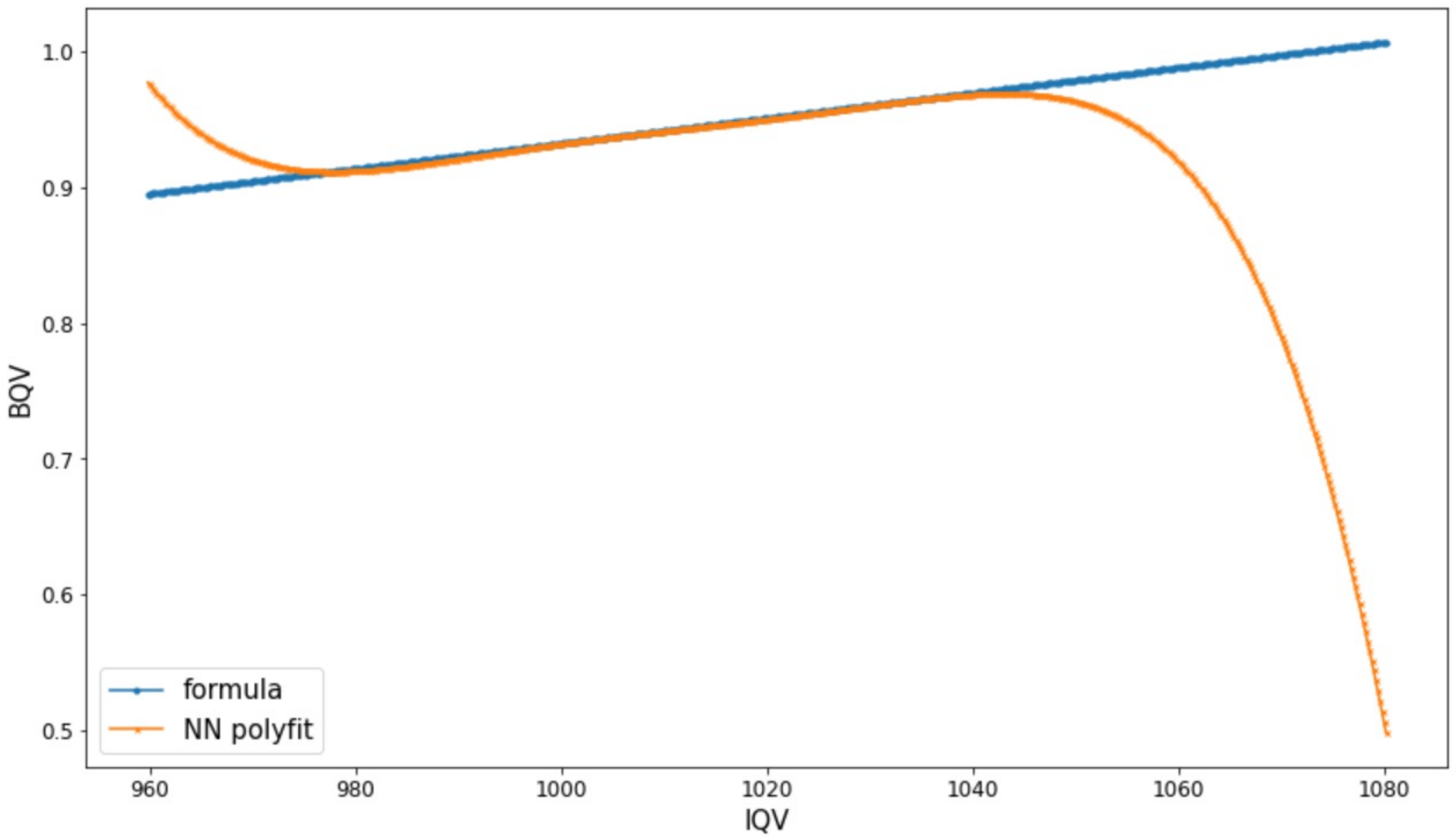
```
coefs = np.polyfit(xx * ckc + 1000, b_preds, 5)
poly = np.poly1d(coefs)
coefs
```

```
array([-4.10936456e-10,  2.07730310e-06, -4.19988525e-03,  4.24519014e+00,
       -2.14525361e+03,  4.33583006e+05])
```



Polynomial fit: numpy.polyfit

- Fit is very bad once extend the input range



Polynomial fit: scipy.optimize.curve_fit

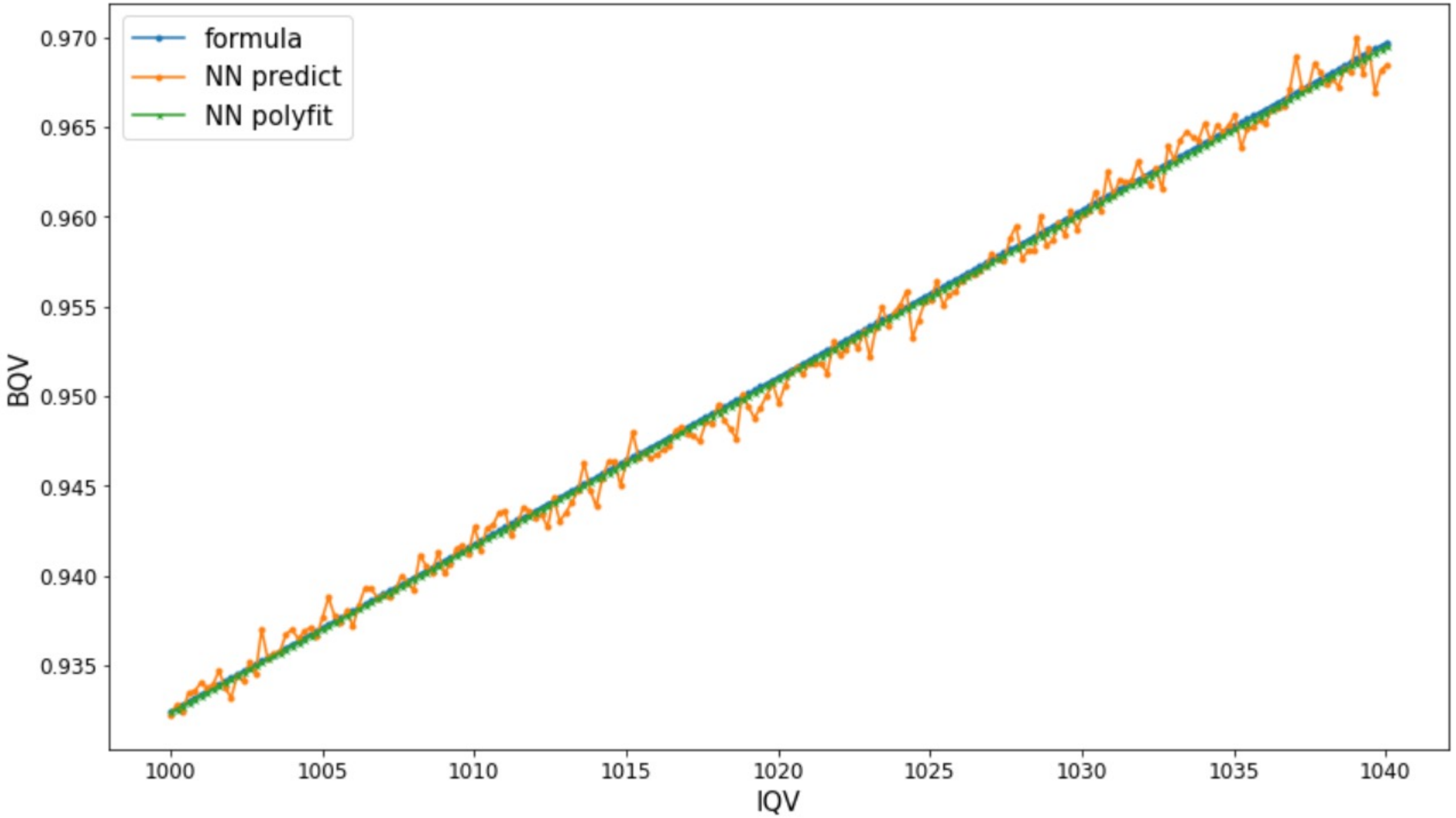
```
klvc0 = 0.002099
klvc1 = 9.257E-4
klvc2 = 1.164E-8
klvc3 = 1.046E-11
klvc4 = 4.057E-15
klvc5 = 5.75E-19
```

- Can set value ranges on all the fit coefficients, default is - inf to inf

```
def func(x, a, b, c, d, e, f):
    return a + b * x + c * x ** 2 + d * x ** 3 + e * x ** 4 + f * x ** 5

upbound = [0.005, 2e-3, np.inf, np.inf, np.inf, np.inf]

popt_cons, _ = curve_fit(func, xx * ckc + 1000, b_preds, \
                         bounds=([0, 0, 0, 0, 0, 0], upbound))
```

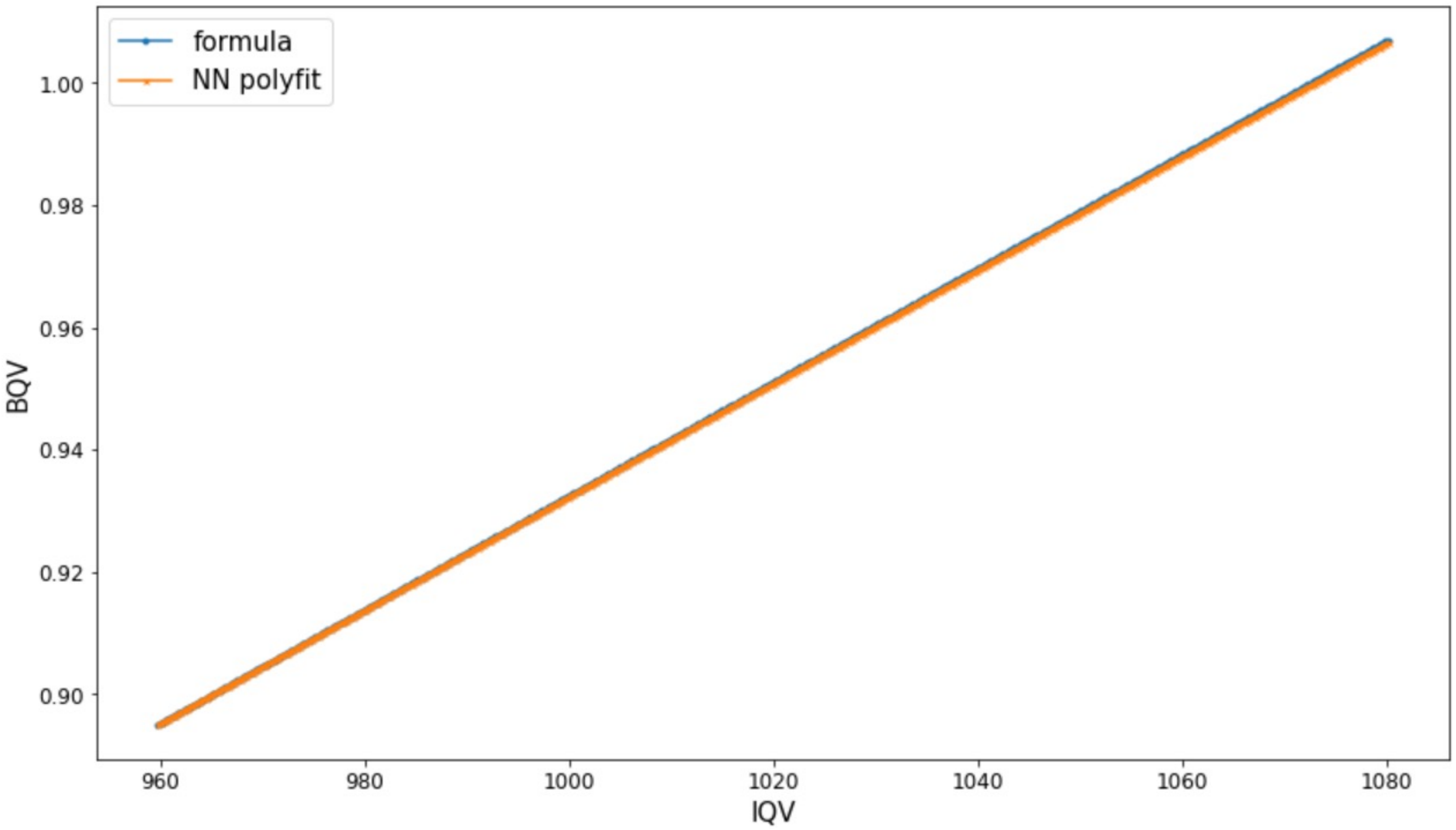
```
popt_cons
```

```
array([4.99997677e-03, 9.27382159e-04, 1.24499699e-13, 2.84106303e-19,
       1.26886886e-18, 8.46889710e-22])
```



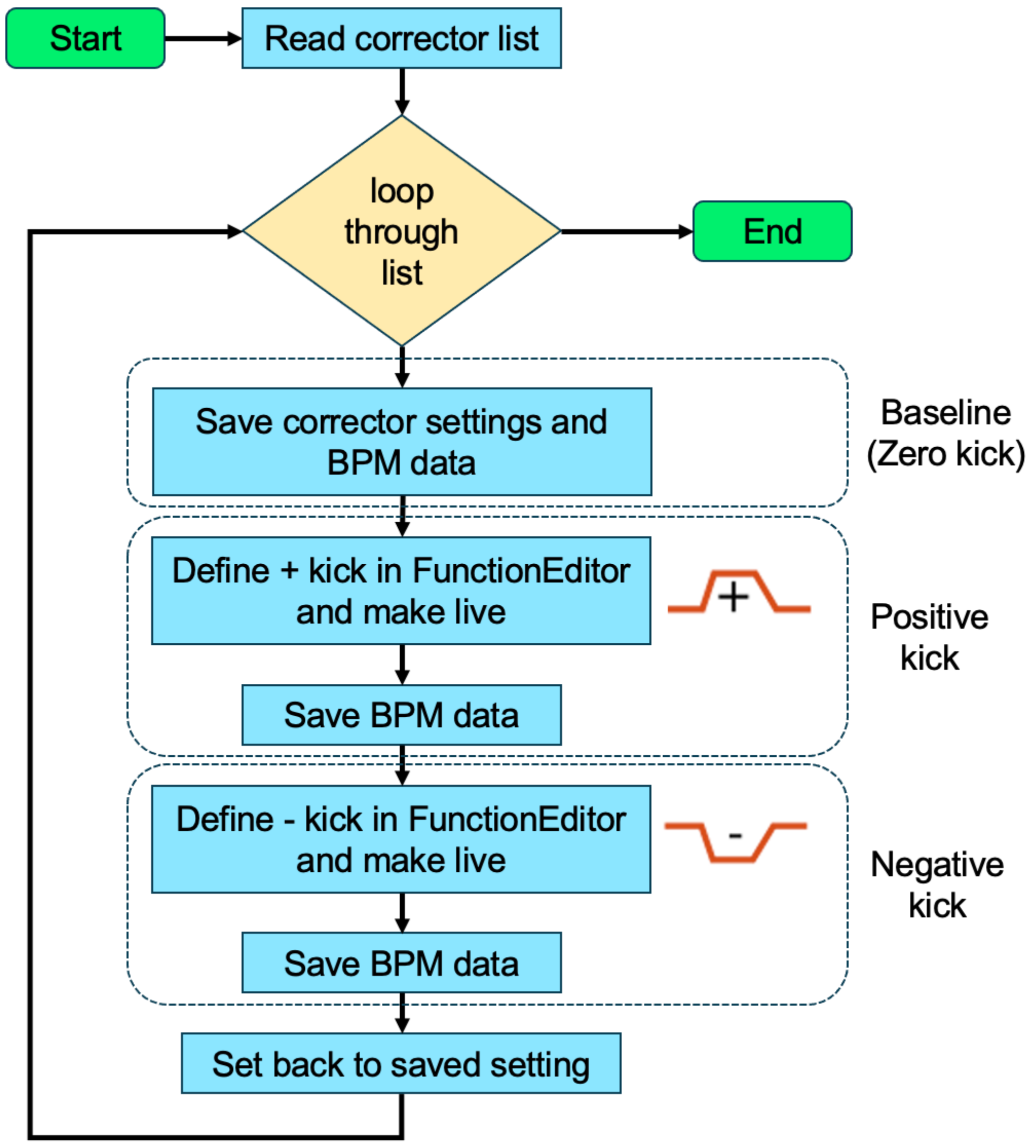
Polynomial fit: scipy.optimize.curve_fit

- Fit stays good once extend the input range
- Bounds for coefficients need to be carefully adjusted, otherwise doesn't fit properly



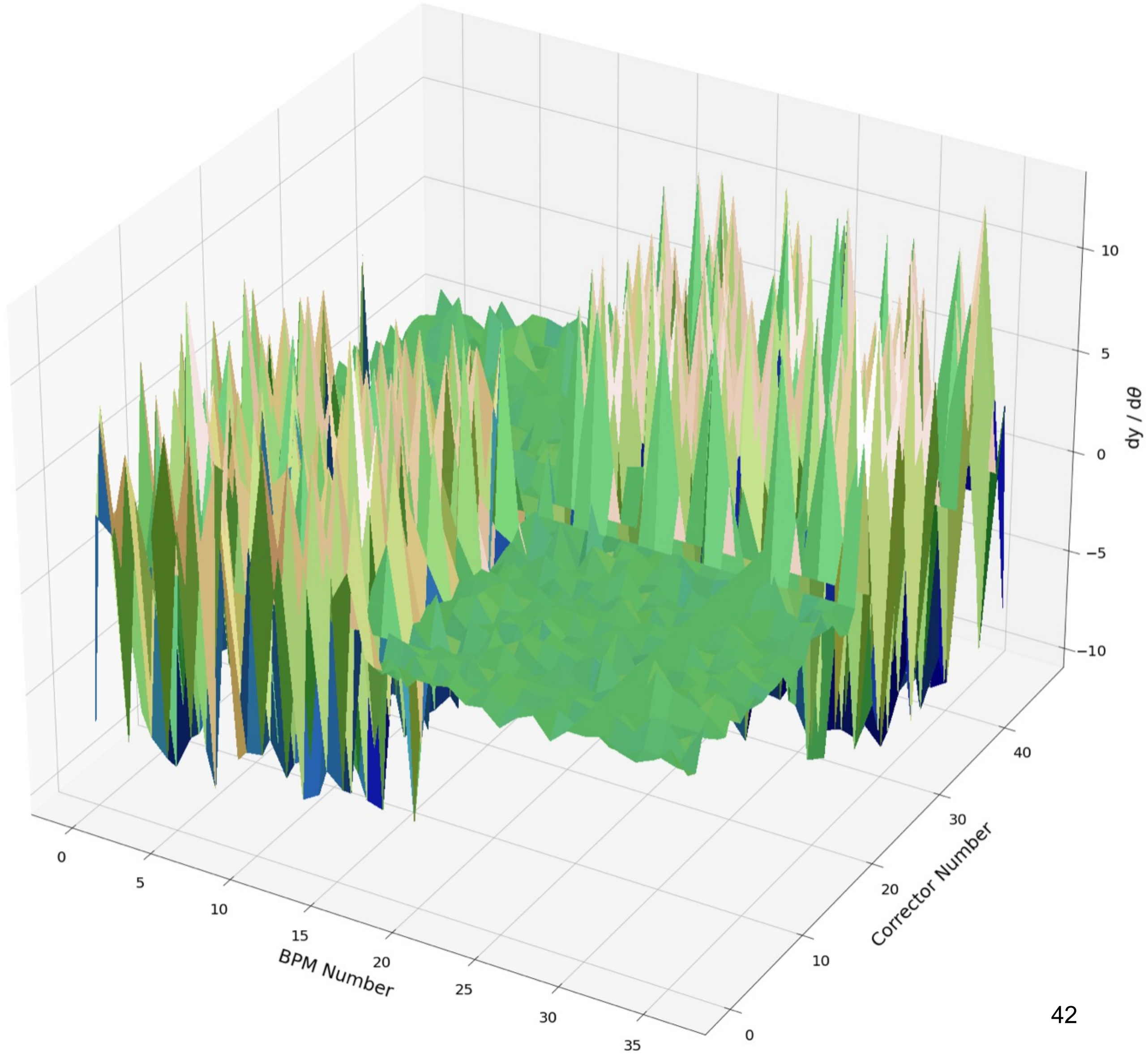
CAD script to get real ORM

- Script from Collider Accelerator Department (CAD) Controls Group
- FunctionEditor: send trapezoid-like time-dependent function to corrector power supplies
- Script sets three corrector settings: positive, zero, negative



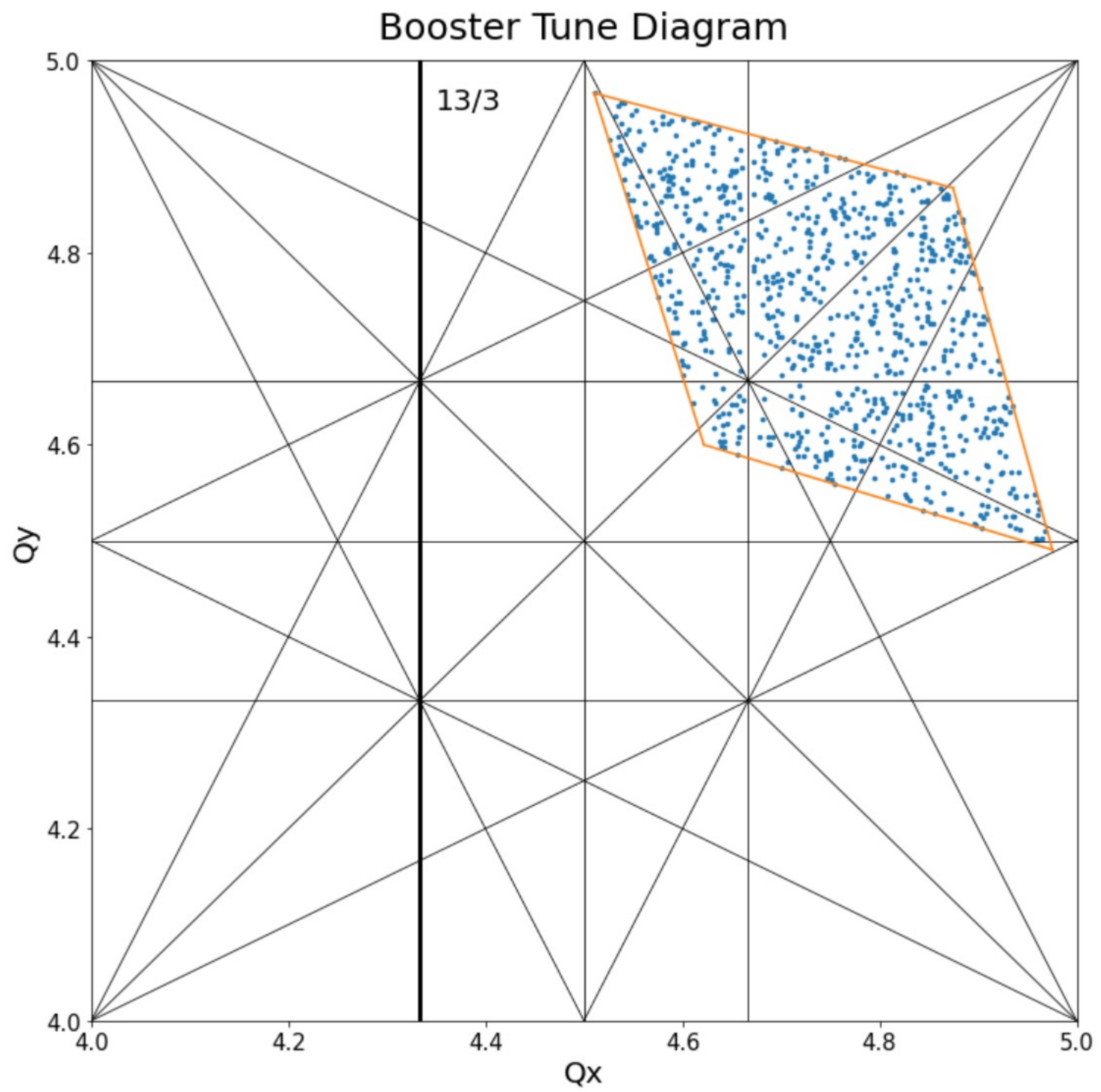
Real data: double plane ORM

- In model: 48 BPMs, 48 correctors
- In reality:
  - 47 correctors (no BD6-th)
  - 37 good BPMs (bad ones produce NaN values)
- Real ORM R has dimension (37,47)



Sample data in tune space

- Sample non-zero H & V quadrupole settings that don't hit a resonance
- Quadrupole PS current range 0 – 400 A
- Produce double-plane ORM in the same format as real data



Ongoing: NN model for double-plane data

- Get model double-plane ORM R_{model} from lattice with all quadrupole currents set to zero
- Find k1 value combos for horizontal and vertical quadrupoles that avoid resonances
- Add k1 combos to Bmad and get corresponding ORM R_{meas}
- Training dataset: $dR = (R_{meas} - R_{model}).\mathrm{flatten}()$ as input with shape (N, 1739), quads k1 value as output with shape (N, 2) for horizontal and vertical quads
- Problem to be solved: how to include uncertainty analysis in ML model training so the mapping is closest to reality

References

- [1] Alternating Gradient Synchrotron, https://www.bnl.gov/rhic/ags.php, Accessed on Sep. 6 2022.
- [2] Y. Bai et al., “Research on the slow orbit feedback of BEPCII using machine learning”, Rad. Det. Tech. Meth. 6, 179-186 (2022).
- [3] E. Fol, R. Tomás, and G. Franchetti, “Supervised learning-based reconstruction of magnet errors in circular accelerators”, Eur. Phys. J. Plus 136, 365 (2021).
- [4] M. Venturini, A.J. Dragt, “Accurate computation of transfer maps from magnetic field data”, Nucl. Instrum. Methods Phys. Res. A: Accel. Spectrom. Detect. Assoc. Equip. 427, 387-392 (1999).
- [5] K. Brown et al., “A high precision model of AGS Booster Tune Control”, in Proc. EPAC’02, Paris, France, June 2002, pp. 548-550.