BPM buttons

non linear beam response meets machine learning

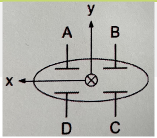
Nicola Selbach

2024/12/11&12

Table of Contents

1. Problem of non linear BPM response
2. Boundary Element Solver
3. Machine Learning approach
4. Experimental Application
5. Outlook

Introduction



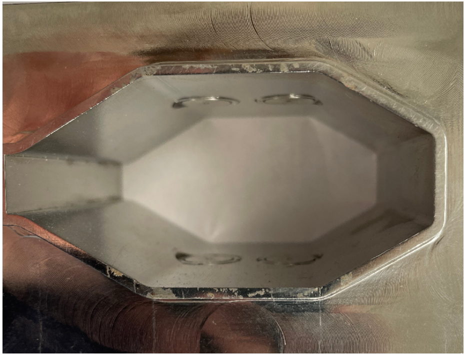


Figure: standard BPM implemented in BESSY II

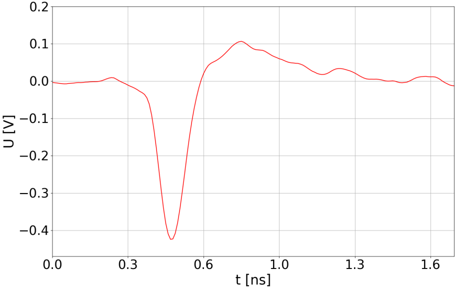
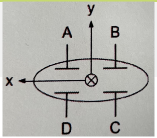


Figure: voltages over time, single bunch signal

Motivation



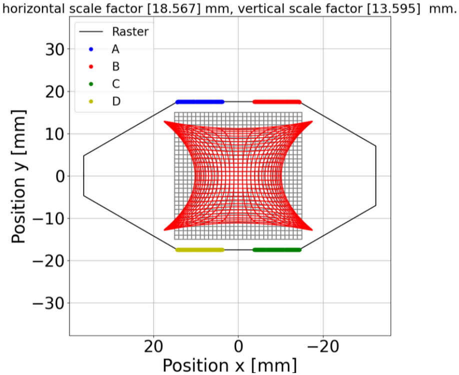


Figure: non-linearity in geometry of BESSY II

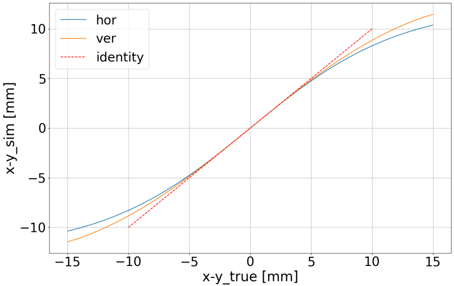


Figure: sensitivity and beam displacement

1. Problem of non linear BPM response
2. Boundary Element Solver
3. Machine Learning approach
4. Experimental Application
5. Outlook

Simulation

Principle

- parametrize **geometry** of BPMs & buttons
- simulate **beam positions** in geometry through raster
- calculate **induced charge** for beam positions on chamber wall of geometry
  $\Longrightarrow$ boundary element method (**BEM**) [1]
- calculate beam position **backwards** from induced charges on ABCD

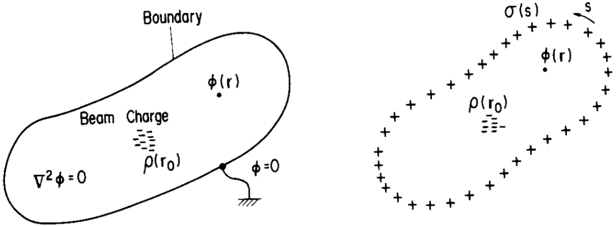
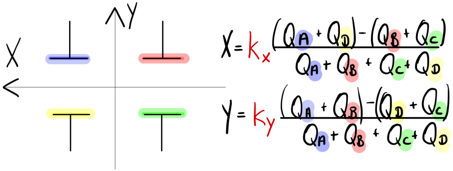


Figure: Reference [1]

Code Conversion



$$X = k_x \frac{(Q_A + Q_D) - (Q_B + Q_C)}{Q_A + Q_B + Q_C + Q_D}$$

$$Y = k_y \frac{(Q_A + Q_B) - (Q_D + Q_C)}{Q_A + Q_B + Q_C + Q_D}$$

Original Matlab code [2] $\Longrightarrow$ Python code

What's included?

- calculation of **induced charge**
  $\Longrightarrow$BEM method
- beam position raster & non linear solution
- Differences over Sum (**DoS**) method
  $\Longrightarrow$applied method to calculate XY-beam-position from ABCD-signals
- calculation of non linear **k-factor**
  $\Longrightarrow$ scale factor is change of beam position divided by change of x or y around centres, needs to be applied to DoS solution
- **Newton Raphson Algorithm**
  $\Longrightarrow$iteration method to find XY-position from ABCD signals !time consuming!

Code Conversion: Matlab to Python

Most important changes

- conversion to **SI** units
  $\Longrightarrow$mm $\longrightarrow$ m
- 18 **geometries** included
  $\Longrightarrow$e.g. round, rectangular, two buttoned
- **axis** transformation
  $\Longrightarrow$according to BESSY II convention

Small Bug

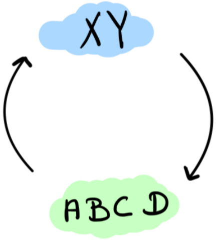
- BEM method $\longrightarrow$ calculation of induced charge
  $\Longrightarrow$matrix of Green's functions solving the 2D electrostatic Laplace equation:

$$G_{ii} = \int_{\Gamma_i} \ln\left(\frac{1}{|r_i - r_i|}\right) dr_i = s \cdot \left(1 - \ln\left(\frac{s}{2}\right)\right);\ s = segments\ width$$

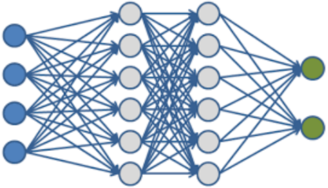
1. Problem of non linear BPM response
2. Boundary Element Solver
3. Machine Learning approach
4. Experimental Application
5. Outlook

Idea

- train **neural network** how to solve XY-beam-position to ABCD-button-signal
  $\Longrightarrow$ well known & easy calculation
- generate **training data** with simulation code
- goal: neuronal network learns **inverse** problem
  $\Longrightarrow$ the inversion we use is a good **approximation** for **small** beam displacements
  $\Longrightarrow$ we want a **fast** solution for small & large amplitudes



Some Numbers



Example model:

- time of training : *57265.4s* $\approx$ *7h* on CPU
- training data size : *5.2GB*
- model size : *4.6MB* (not minimized)
- model architecture : 4 layers : [1000, 700, 500, 300]
- CPU * times for ML prediction : $\approx$ *80ms* for measured dataset **

  $\Longrightarrow$ CPU * times for Newton Raphson : $\approx$ *7s* for measured dataset **

  * *13th Gen Intel(R) Core(TM) i7-1365U, 12 threads, 5.2 GHz*

  ** $\approx$ *7800 ABCD signals*

Outcomes

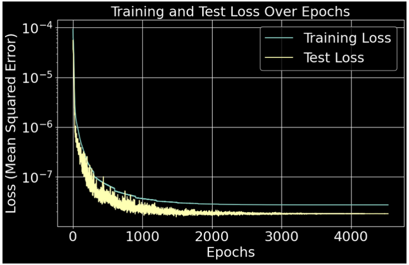
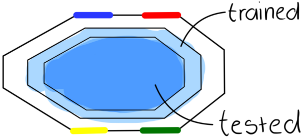


Figure: behaviour of training and testing data during model development



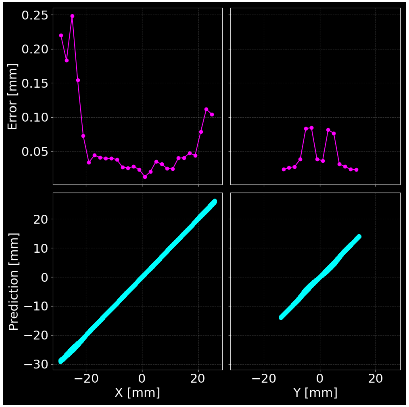


Figure: performance

Outcomes

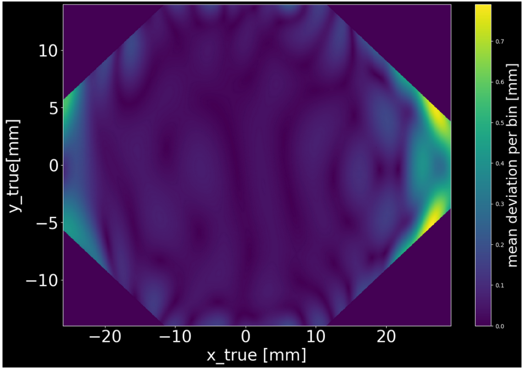


Figure: 2D model performance

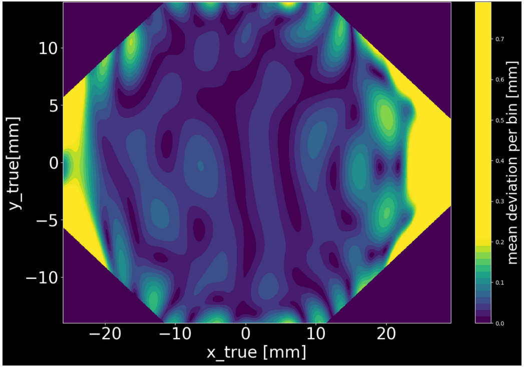
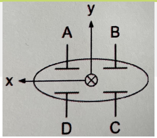


Figure: 2D performance (dicrete area)

1. Problem of non linear BPM response
2. Boundary Element Solver
3. Machine Learning approach
4. Experimental Application
5. Outlook

Measurement



measured data of huge beam position shifts by changing masterclock frequency
$\alpha \approx 7 \cdot 10^{-4}$, dispersion ≈ 0.5m

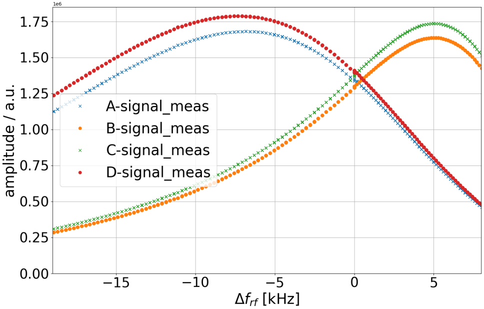


Figure: signals on BPM buttons

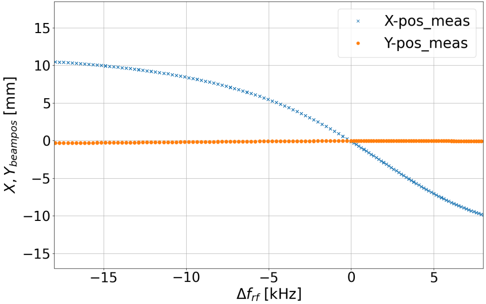
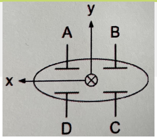


Figure: XY positions

Evaluate model



evaluation in BESSY II model [3] in comparison to measurement results:

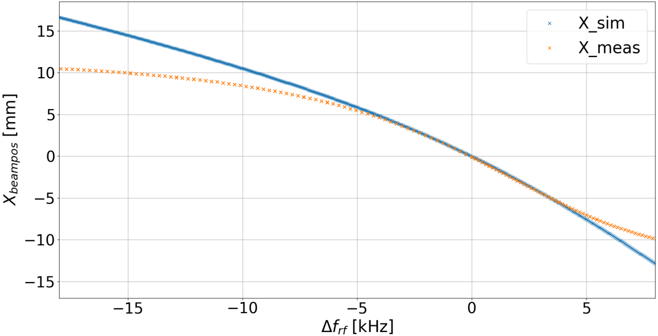
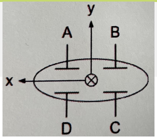


Figure: X positions comparison

Comparison to Newton Raphson



application of measurement data on Newton Raphson algorithm:

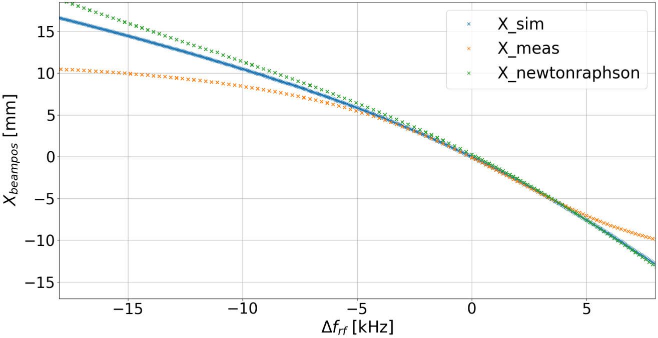
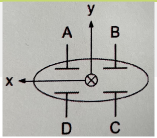


Figure: X positions comparison

Comparison to ML model



application of measurement data on ML model:

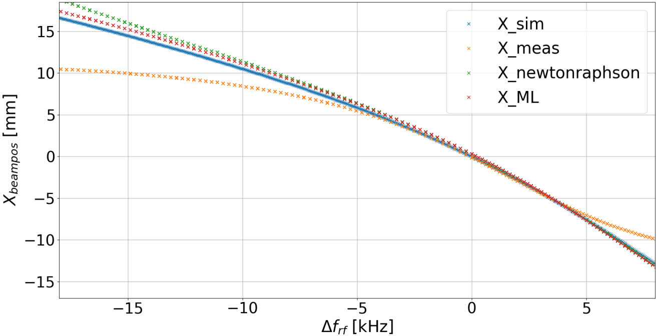
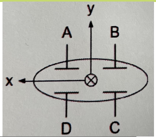


Figure: X positions comparison

1. Problem of non linear BPM response
2. Boundary Element Solver
3. Machine Learning approach
4. Experimental Application
5. Outlook

- convert python code into **python package**
  ⟹ loadable & easy to apply
- implementation of different **BPM button** geometries
- apply **beam current** to ML model
  ⟹ atm needs to be fixed by **norm factor** on ABCD
- apply ML model to **FPGA system**



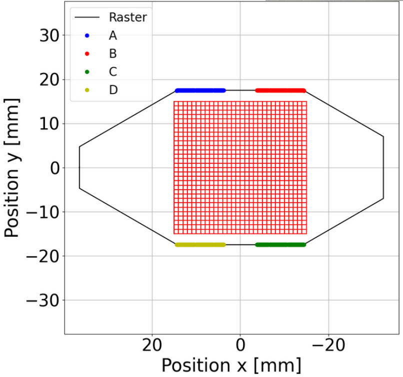


Figure: HZB, BESSY II [4]

References

[1] Shintake et al., " Sensitivity calculation of beam position monitor using boundary element method " , 1987, journal NIM-A, volume 254, pp. 146-150

[2] A. Olmos, F. Perez, and G. Rehm, “Matlab Code for BPM Button Geometry Computation”, in Proc. DIPAC’07, Venice, Italy, May 2007, paper TUPC19, pp. 186-188

[3] Pyat documentation: $https://atcollab.github.io/at/p/index.html$

[4] Dirk Laubner $https://www.helmholtz-berlin.de/forschung/quellen/bessy/index_de.html$

Thanks to:
PhD. Meghan Mc Atheer, Dr. Günther Rehm, Dr. Markus Ries,
Dr. Gregor Hartmann, Dr. Andreas Schälicke