Preliminary results for PyAT-based loco code

Elaf Musa - PhD student (DESY)

Supervisors: Ilya Agapov, Tessa Charles

October 6, 2022

Table of contents

Corrector-to-BPM response matrix

- An accelerator with m-BPMS and n-correctors produces an m x n dimensional response matrix.

$$C_{mn} = \frac{\sqrt{\beta_m \beta_n}}{2\sin(\pi\nu)} \cos(\pi\nu - \phi(s) + \phi(s_0))$$

- The aim of the orbit correction is to find a set of corrector kicks θ that satisfy the following relation:

$$\Delta x + C\Delta\theta = 0$$

- We derived the Corrector-to-BPM response matrix in PyAT and used it to obtain the proper correctors strengths used in orbits corrections.
- First tests in PETRA III and FCC-ee closed orbits were performed

LOCO(Linear Optics From Closed Orbits)

The parameters in the used model is varied by LOCO to minimize the χ^2: deviation between the model and measured orbit response matrices C_{model} and $C_{measured}$.

$$\chi^2 = \sum_{ij}(\Delta C^{ij} - \sum_k \frac{\partial C^{ij}}{\partial g_k}\Delta g_k)^2$$

$$\Delta g_k = (\frac{\partial C^{ij}}{\partial g_k}^T \frac{\partial C^{ij}}{\partial g_k})^{-1}(\frac{\partial C^{ij}}{\partial g_k}^T \Delta C^{ij})$$

- LOCO was previously implemented in other codes e.g. Matlab and Elegant.
- We derived PyAT implementation of the linear optics corrections (LOCO).
- First tests for the implemented code was done for the FCCee_t_v22 lattice (work in progress) .

Implementing LOCO in PYAT

```python
def ORM_x(dkick, ring, BPMs_random_noise, used_correctors_Names=None):
    cxx = []
    cxy = []

    for i in range(len(used_correctors_Names)):

        cor_index = get_refpts(ring, used_correctors_Names[i])
        cor_index = cor_index[0]

        ring[cor_index].KickAngle[0] = dkick

        closed_orbitx, closed_orbity = closed_orbit_bpm_r(ring, BPMs_random_noise)
        cxx.append(closed_orbitx)
        cxy.append(closed_orbity)

        ring[cor_index].KickAngle[0] = 0.00

    Cxx = np.squeeze(cxx) / dkick
    Cxy = np.squeeze(cxy) / dkick

    return Cxx, Cxy
```

Implementing LOCO in PYAT

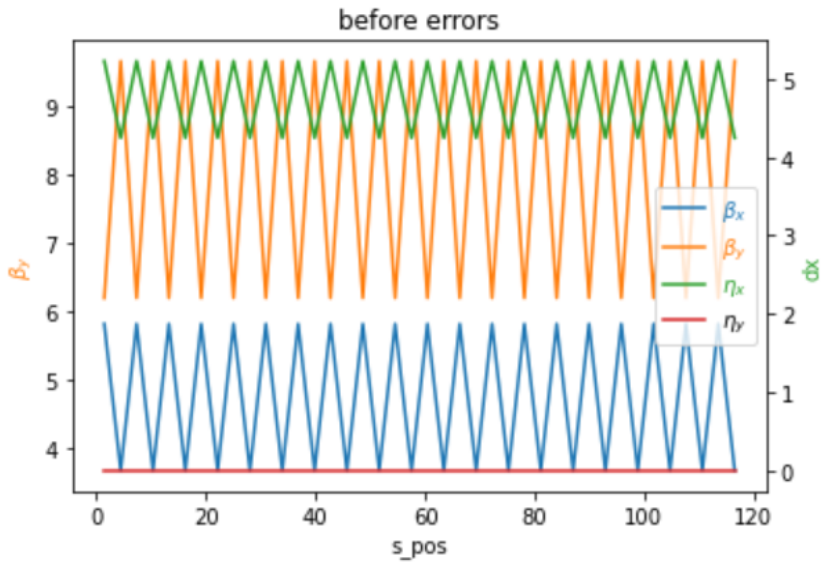
- Choosing families and subfamilies from quadrupoles in the lattice.
- Choosing orbit correctors equally distributed around the lattice.
- Calculating the quadrpoles families response matrices
- Applying LOCO correction loops

```
def QsensitivityMatrices(ring, qfamilyIndices, dk, BPMs_random_noise, used_correctors_Names):

    strength_before = []
    for i in qfamilyIndices:

        strength_before.append(ring[i].K)
        a = ring[i].K
        ring[i].K = a + dk

    qxx, qxy = ORM_x(dk, ring, BPMs_random_noise, used_correctors_Names)
    qyy, qyx = ORM_y(dk, ring, BPMs_random_noise, used_correctors_Names)

    for i in range(len(qfamilyIndices)):

        ring[qfamilyIndices[i]].K = strength_before[i]


    return  qxx, qxy, qyy, qyx
```

Simple lattice example

Parameters:

- 60 quadrpoles QF,QD,QS, 40 orbit correctors
- 5.e-3 rms gradient errors
- 1.e-3 m tilt errors
- BPMs noise = 0.0
- Sextupoles turned off
- 5 corrections loops



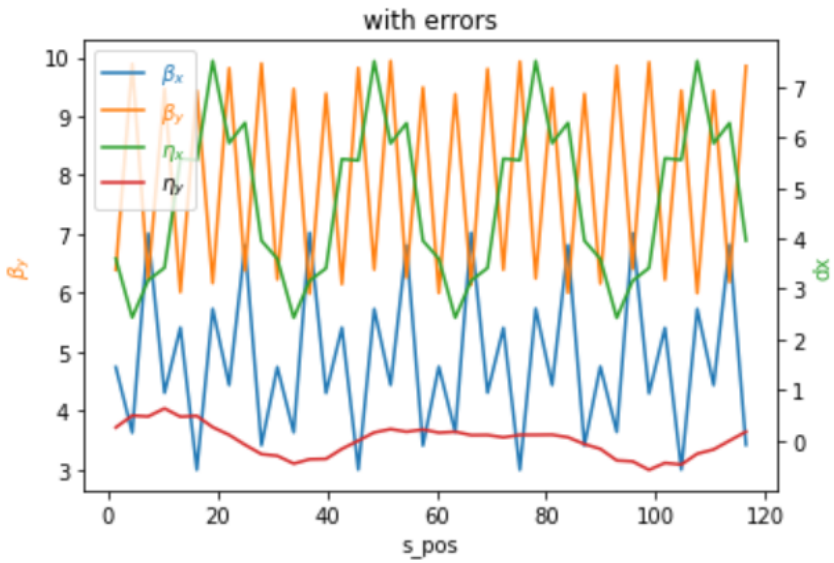
Results

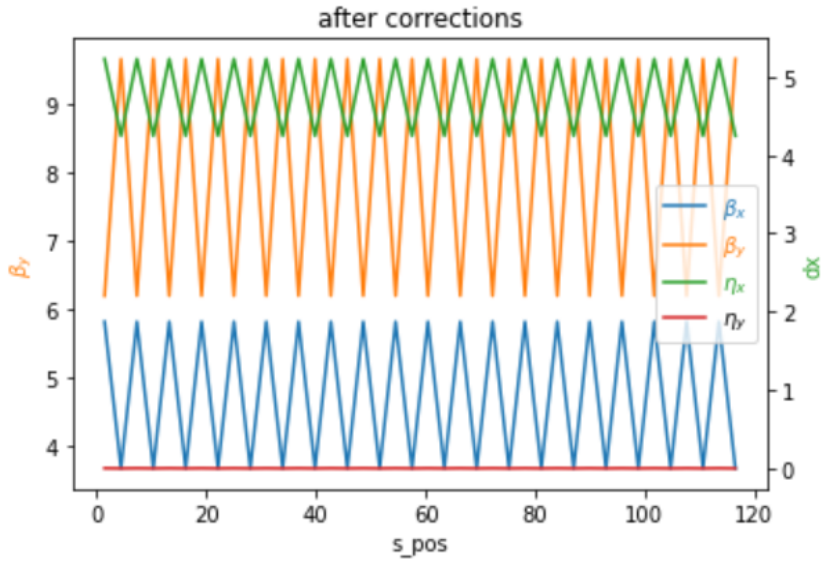
RMS Beta beating with errors:

x:14.9% y: 2.3%

RMS Beta beating after correction:

x:4.0e-06% y: 1.0e-4%





Results

Dispersion correction:

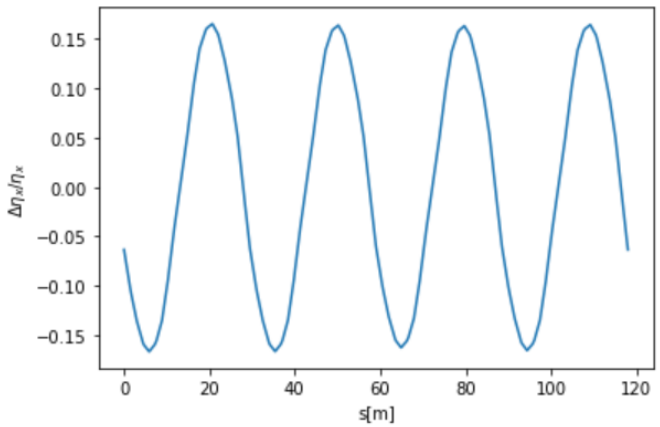


Figure: $\frac{\Delta\eta_x}{\eta_x}$ with errors

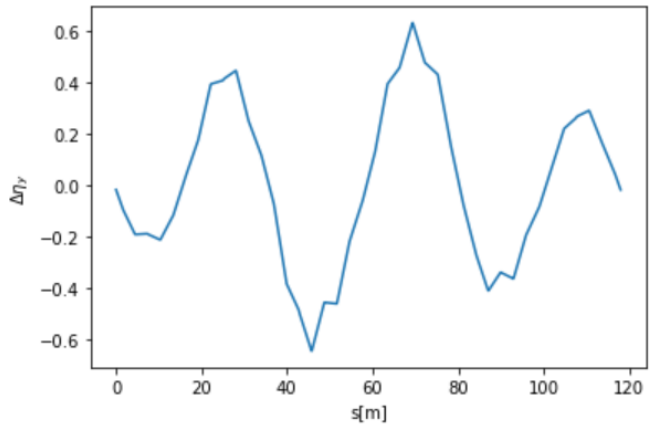


Figure: $\Delta\eta_y$ with errors

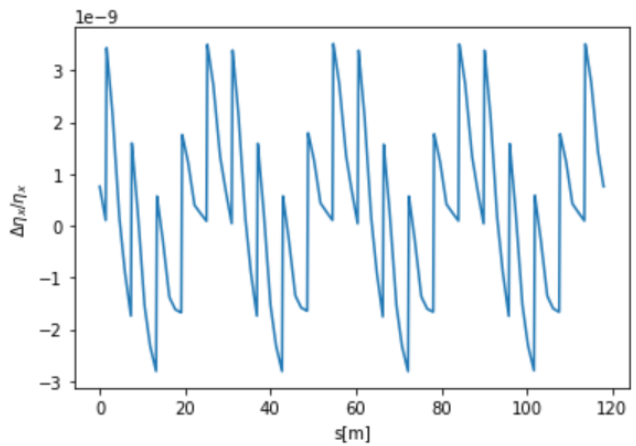


Figure: $\frac{\Delta\eta_x}{\eta_x}$ after correction

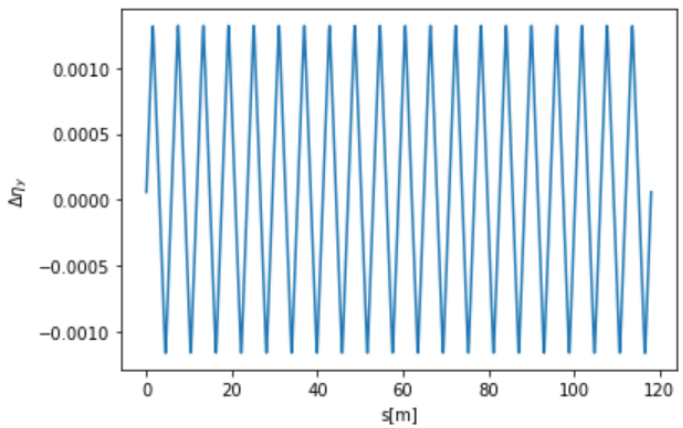
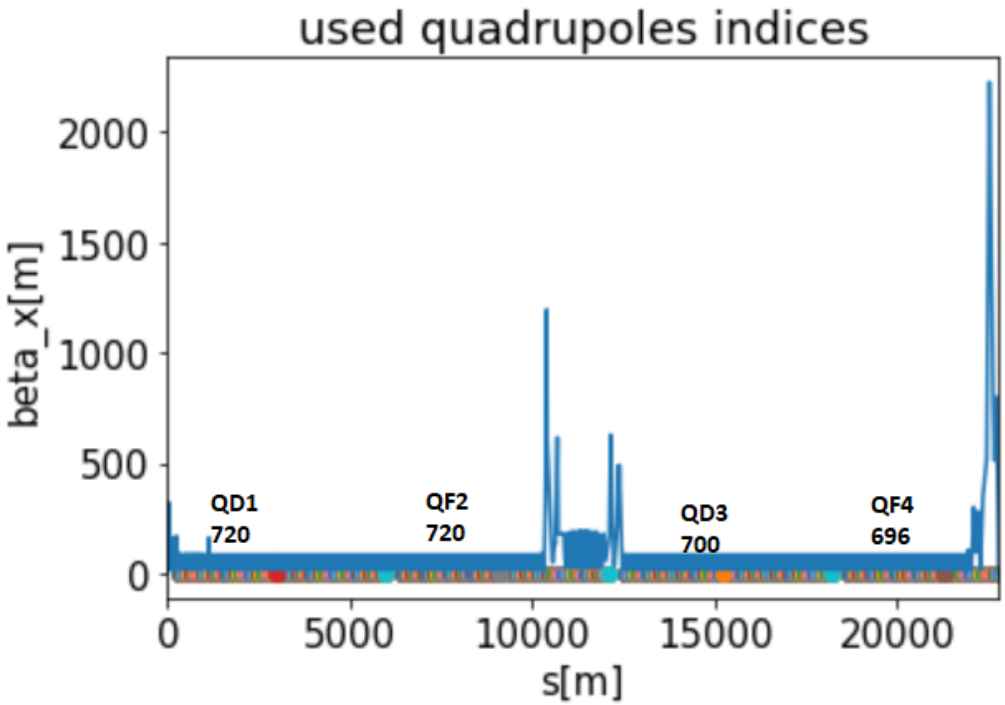


Figure: $\Delta\eta_y$ after correction

First tests: FCCee lattice

FCCee_t_v22 sequence file was converted to PyAT using the xsequense tool. (https://github.com/fscarlier/xsequence)
Number of correctors and BPMs were introduced to the FCCee_t_v22 lattice using PyAT

- Total number of quadrupoles 3324: 100 quadrpoles families

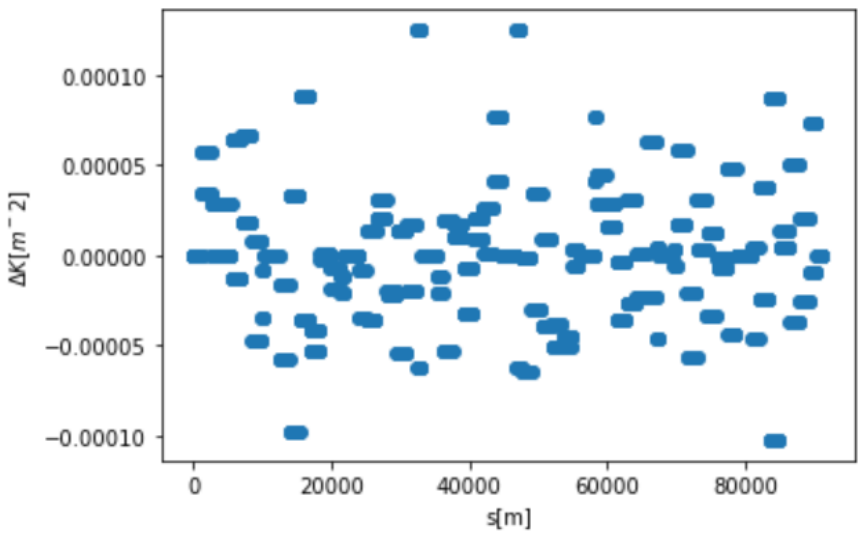


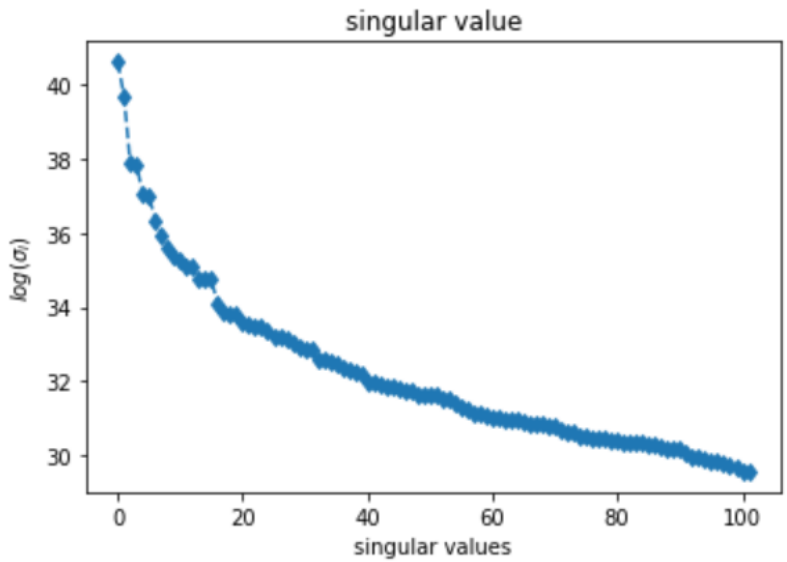
- Total number of correctors 3324: using 4 correctors
- Total number of BPMs 3324: using all BPMS

First tests: FCCee lattice

Parameters:

- 5.e-3 rms gradient errors (fixed through the subfamilies)
- BPMs noise = 0.0
- Sextupoles turned off
- 5 corrections loops





Results

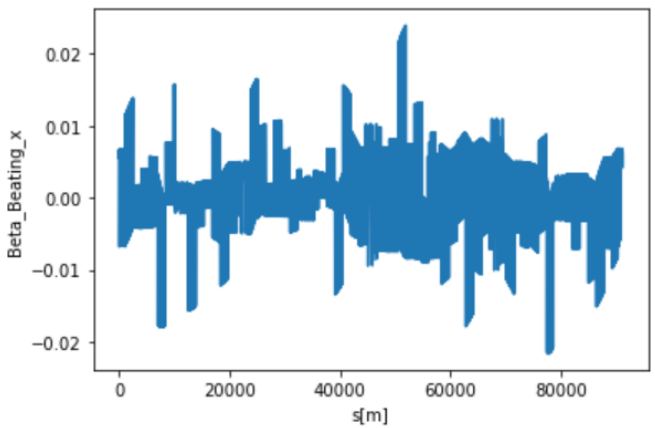
Execution tim ~ 35 min

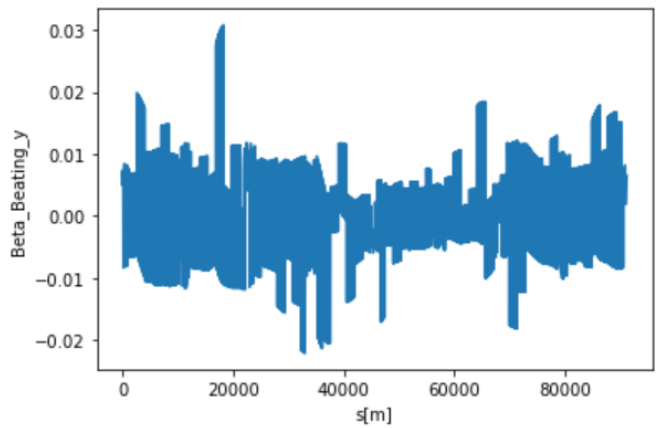
RMS Beta beating with errors:

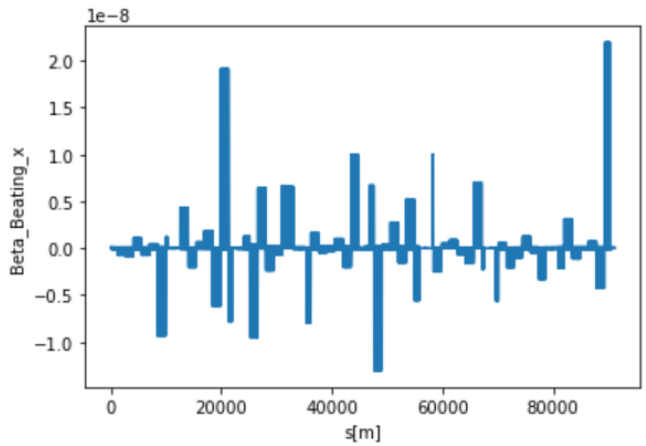
x:0.59% y: 0.75%

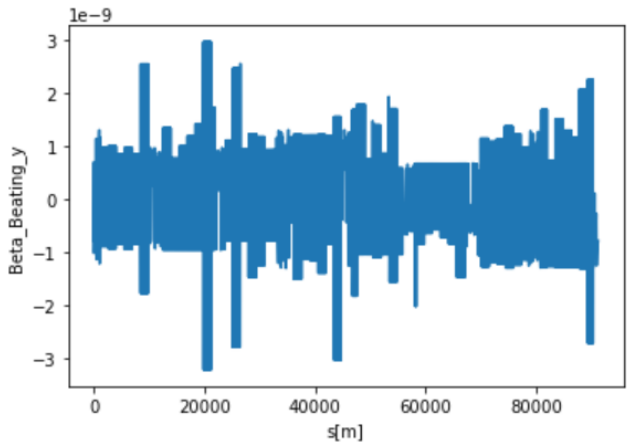
RMS Beta beating after correction:

x:2.6e-07% y: 9.9e-08%





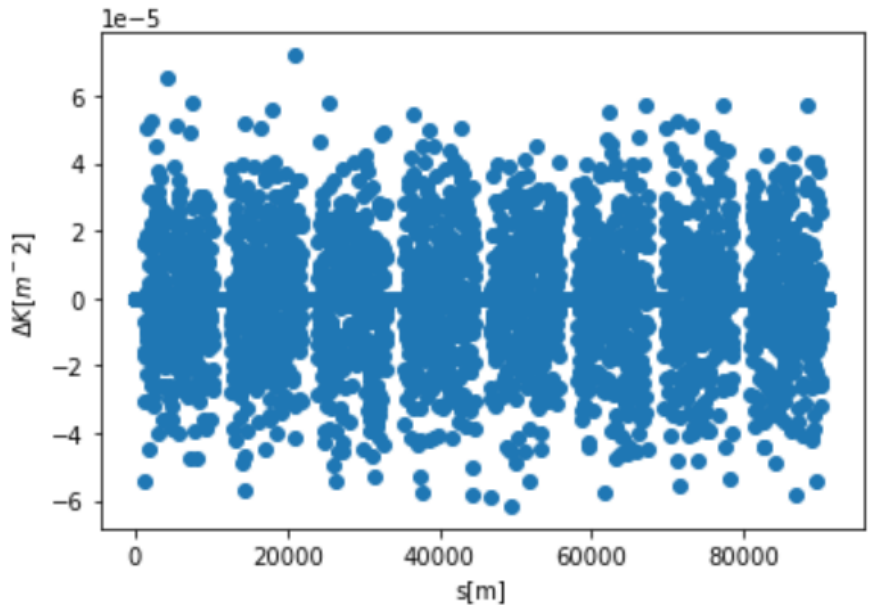




Real case example

Parameters:

- Increasing the number of quadrpoles and correctors families
- Random gradient errors
- BPMs noise = 0.0
- Sextupoles turned off
- Work is still in progress.



Real case example for FCCee

Parameters:

- 400 quadrpoles families, 10 orbit correctors
- 1.e-3 rms gradient errors(random)
- BPMs noise = 0.0
- Sextupoles turned off
- 5 corrections loops

Result:

Execution tim $\sim$ 9 hours

RMS Beta beating with errors:

x:6.04% y: 9.19%

RMS Beta beating after correction:

x:3.34% y: 2.8%

Summary

- Deriving PyAT implementation of closed orbit and linear optics corrections github.com/iagapov/pyat-loco
- Applying PyAT-based loco and orbit correction implementation in correcting the orbits and optics for simple lattice and FCCee_tt lattice

Outlook

- Choosing the proper numbers of quadruples families and orbit correctors for real case corrections (work is still in progress).
- Improving the PyAT-based loco code to reduce the consumed time
- Coupling and dispersion correction for FCCee lattice
- Including BPMs noise

Thank You