ThomX presentation: Injection correction

Alexandre Moutardier

IJCLab, Orsay, France

October 1st, 2020

Goal

- Goal: Prediction of correction needed for injection into the ThomX ring.
- Advancements presented here:
  - Analytic transfer matrix calculation using MatLab
  - Calculus of deviation needed to correct beam injection using python
  - Test python code with MadX

Analytics code on MatLab

- Why MatLab ?
  - Use for formal calculation
- Goal:
  - Computation of a vector to transfer the beam from one part of ThomX to another
- Method used:
  - Classical linear transfer matrix calculation

$$\begin{pmatrix} x \\ px \\ y \\ py \\ z \\ pz \end{pmatrix}_2 = M \times \begin{pmatrix} x \\ px \\ y \\ py \\ z \\ pz \end{pmatrix}_1$$

Where M is a 6x6 matrix that depends on elements between position 1 and 2.

Some comparison have being done between my code and MadX to validate it.
To the first order MadX and my code agreed but I found some minor differences.

Computation of deviation

- Why Python ?
  - To be implemented in ThomX cc and coupled with Taurus
- Goal:
  - Use position of beam measured on RI-C1/DG/BPM.02 and RI-C1/DG/BPM.03, constrain x,y and px to find correction of steerers TL/AE/STR.03, TL/AE/STR.04 and of kicker RI-C1/PE/KIC.01 needed to inject correctly the beam on the ring
    The previous analytic software is used for that.
- Method:
  - extract $(x,px,y,py)_{BPM2}$ from $(x,y)_{BPM2}$ and $(x,y)_{BPM3}$ using analytic calculation from BPM2 to BPM3
  - extract $(x,px,y,py)_{BPM2,wanted}$ and $kick_{wanted}$ in kicker from $(x,y)_{BPM2,wanted}$, $(x,y)_{BPM3,wanted}$ and $px_{BPM3,wanted} = 0$ using analytic calculation from BPM2 to BPM3
  - compute $Dev_{3,wanted} = (Dev_{x3,wanted}, Dev_{y3,wanted})$ and $Dev_{4,wanted}$ such that:

$$M_{fromBPM2,toSTR3}(Dev_{3,w}, Dev_{4,w}) \times \begin{pmatrix} x \\ px \\ y \\ py \\ 0 \\ 0 \end{pmatrix}_{BPM2,w} = M_{fromBPM2,toSTR3}(Dev_3, Dev_4) \times \begin{pmatrix} x \\ px \\ y \\ py \\ 0 \\ 0 \end{pmatrix}_{BPM2}$$

Where w is for wanted

See backup slide for more details

Coherence of deviation computation

- Protocol:
  - Simulate random particle: $(x, px, y, py, z, pz)_{init}$
  - Propagate it to evaluate $(x, y)_{BPM2}$ and $(x, y)_{BPM3}$
  - Simulate random $(x, y)_{BPM2,wanted}$ and $(x, y)_{BPM3,wanted}$
  - Calculate deviation and kick to go from $(x, y)_{BPM2}$ and $(x, y)_{BPM3}$ to $(x, y)_{BPM2,wanted}$, $(x, y)_{BPM3,wanted}$ and $px_{BPM3,wanted} = 0$
  - Propagate particle using deviation and compare $(x, y)_{BPM2,wanted}$ and $(x, y)_{BPM3,wanted}$ with $(x, y)_{BPM2,dev}$ and $(x, y)_{BPM3,dev}$

- Using analytic calculation to compute propagation
  - Error $< 5 \times 10^{-14}$ everywhere

- Using MadX tracking module to compute propagation
  Maximum differences between wanted value (imposed at (0,0),(0,0)) and calculate one over 100 random particles:
  - At BPM 2 in x : 1.7 $\times 10^{-04}$ mm
  - At BPM 2 in y : 3.3$\times 10^{-05}$ mm

  - At BPM 3 in x : 0.057 mm
  - At BPM 3 in y : 0.011 mm
  - At BPM 3 in px : 0.064% of P_0

Test of deviation computation with MadX

- Protocol:
  - Simulate random particle within predicted beam at the end of the accelerating section: $(x, px, y, py, z, pz)_{init}$
  - Propagate it on MadX to evaluate $(x, y)_{BPM2}$ and $(x, y)_{BPM3}$
  - Use $(x, y)_{BPM2,wanted} = (0, 0) = (x, y)_{BPM3,wanted}$ in the frame of the reference particle
  - Calculate deviation to go from $(x, y)_{BPM2}$ and $(x, y)_{BPM3}$ to $(x, y)_{BPM2,wanted}$ and $(x, y)_{BPM3,wanted}$
  - Propagate particle using deviation and look at beam propagation within the ring
    - Ether by ploting propagation
    - Or by using an estimator

- Ideal position in the frame of the ring after correction:
  - on the BPM 2

$$\begin{cases} \mathsf{X}_{BPM2} & = & 7.86mm \\ \mathsf{PX}_{BPM2} & = & -0.0103 \times P_0 \\ \mathsf{Y}_{BPM2} & = & 0 \\ \mathsf{PY}_{BPM2} & = & 0 \end{cases}$$

  - Everywhere in the ring after RI-C1/AE/DP01

$$\begin{cases} \mathsf{X} & = & 0 \\ \mathsf{PX} & = & 0 \\ \mathsf{Y} & = & 0 \\ \mathsf{PY} & = & 0 \end{cases}$$

Recall of injection representation in MadX and analytics code

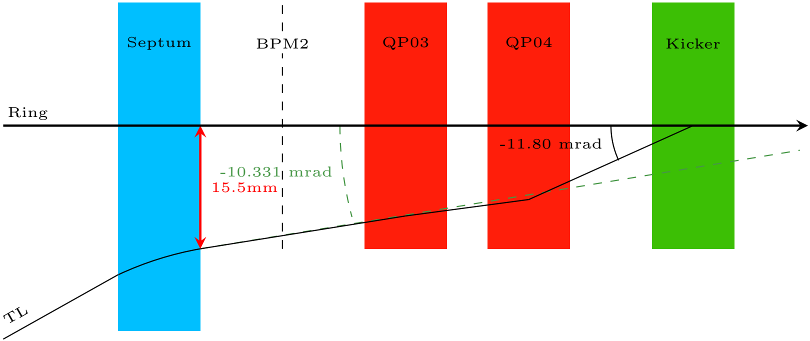


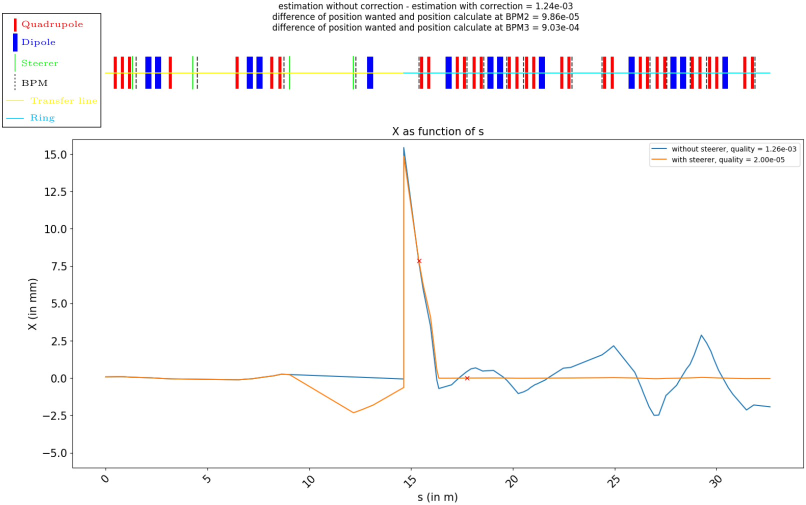
Figure: Diagram of beam injection on ThomX ring (not to scale)

Change of frame apply at the exit of the septum:

- $\mathbf{x \longrightarrow x + 15.5mm}$
- $\mathbf{px \longrightarrow px - 10.331mrad}$

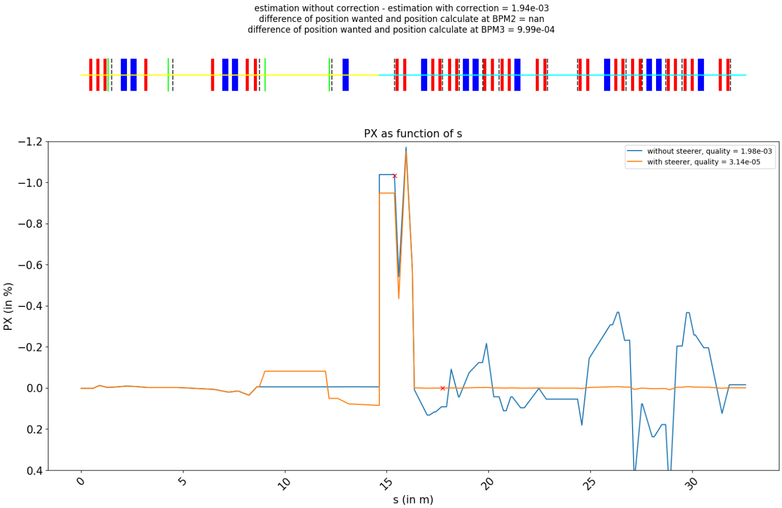
Warning:
from end of septum (s = 14.6 m) to end of kicker (s = 16.4 m) plot are in the frame of the ring !!!

In X plane



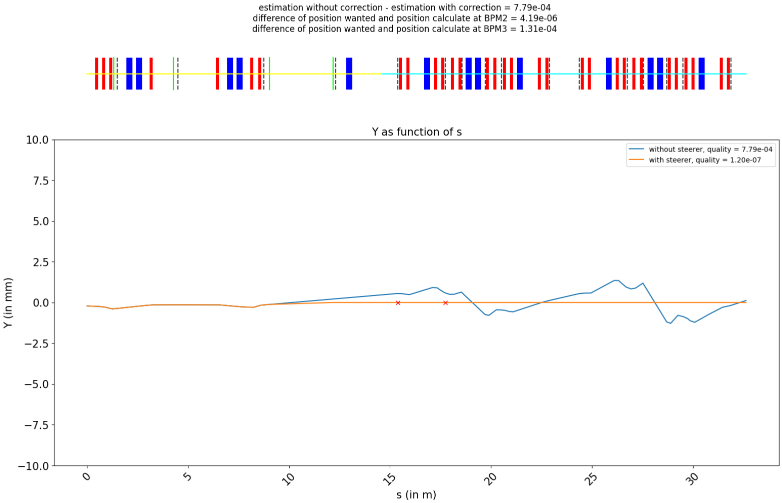
- Blue ligne :
  - Without any deviation
  - Oscillation on the ring grows up
  - Beam may be losses after some turn
- Orange line :
  - With calculate deviation
  - No oscillation
  - Perfect injection

In PX plane kicker modification



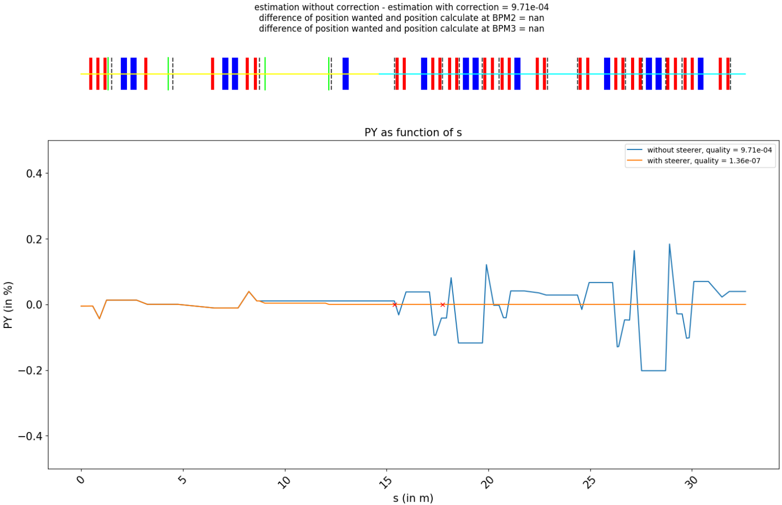
- Blue ligne :
  - Without any deviation
  - Oscillation on the ring grows up
- Orange line :
  - With calculate deviation
  - Nearly no oscillation

In Y plane



- Blue line :
  - Without any deviation
  - Oscillation on the ring grows up
  - Beam losses likely after some turns
- Orange line :
  - With computed deviation
  - No visible oscillation
  - Perfect trajectory

In PY plane



- Blue line :
  - Without any deviation
  - Oscillation on the ring grows up
- Orange line :
  - With computation deviation
  - No visible oscillation
  - Perfect trajectory

Analyse of results

In previous case :

- Correction computation permits a better injection
- No oscillations in the ring hence less beam losses

Other cases are less relevant.
A beam injection's quality factor is used to distinguish cases :

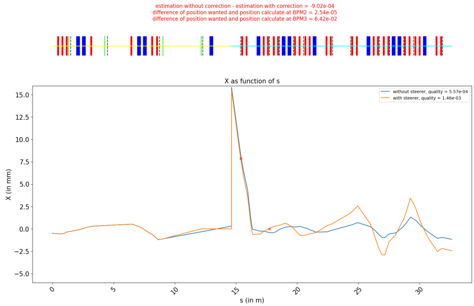
$$Q = \sqrt{\frac{\sum_{el=\text{RI-C1/AE/DP.01}}^{\text{end of ring}} i_{el}^2}{nb_{el}}}$$

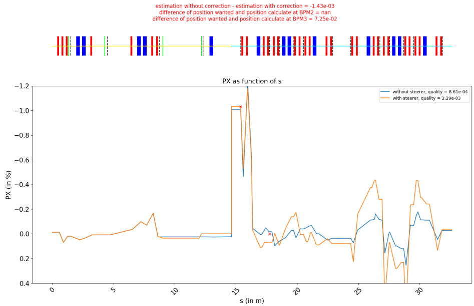
Where i_{el} is the value of i at the exit of the element el, i = x,px,y,py and nb_{el} is the number of element considered

There 2 cases :

- $Q_{init} - Q_{corrected} > 0$:
  - Improvement of the injection
  - around 88% of cases
- $Q_{init} - Q_{corrected} < 0$:
  - Deterioration of the injection
  - around 12% of cases

Example of beam injection deterioration





Up : X along ThomX
Down : PX along ThomX

In both plots :

- Blue line :
  - Small oscillation
  - Beam may be stored
- Orange line :
  - Larger oscillation
  - Beam may be lost

In the **Y** plane this problem of injection **never occur**.

Still under investigation.

Other approach

Protocol :

- Simulation of a random particle
- Propagation within ThomX
- Calculation of deviation
- Application of 10% of the deviation differences
- Repeated form second point 100 times

Value of x and y at BPM2 and BPM3 in the frame of the reference particle for each iterations

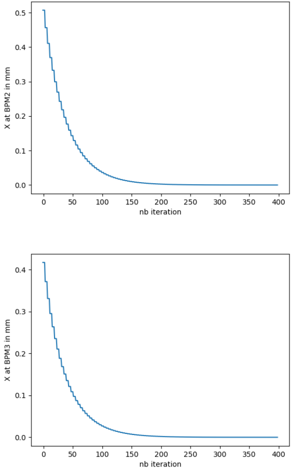


Figure: X (in mm) at BPM2 (up) and BPM3 for each iterations

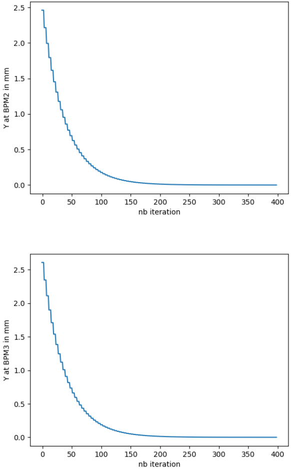


Figure: Y (in mm) at BPM2 (up) and BPM3 for each iterations

Conclusion

- An analytic transfer matrix code is implemented on MatLab.
- A Python code is implemented to compute correction needed to correctly inject beam on the ring. This code used analytics results.
- Injection in effectively better is 88% of the cases.

Next Steep :

- Understand why there are so much wrong cases.
- Try to apply the correction in several.
- Implemented correction code at ThomX cc and coupled it with Taurus to create a feedback.

Thanks

Backup

Classical transfer matrices

- **Drift** of length L:

$$XX = YY = \begin{pmatrix} 1 & L \\ 0 & 1 \end{pmatrix} \qquad ZZ = \begin{pmatrix} 1 & L/\gamma^2 \\ 0 & 1 \end{pmatrix} \qquad M_{drift} = \begin{pmatrix} XX & 0 & 0 \\ 0 & YY & 0 \\ 0 & 0 & ZZ \end{pmatrix}$$

- **Quadrupole** of length L and strength k:

$$F = \begin{pmatrix} \cos(\sqrt{k} \times L) & \frac{1}{\sqrt{k}} \sin(\sqrt{k} \times L) \\ -\sqrt{k} \times \sin(\sqrt{k} \times L) & \cos(\sqrt{k} \times L) \end{pmatrix} \qquad D = \begin{pmatrix} \cosh(\sqrt{k} \times L) & \frac{1}{\sqrt{k}} \sinh(\sqrt{k} \times L) \\ \sqrt{k} \times \sinh(\sqrt{k} \times L) & \cosh(\sqrt{k} \times L) \end{pmatrix}$$

if k > 0:

$$M_{quad} = \begin{pmatrix} F & 0 & 0 \\ 0 & D & 0 \\ 0 & 0 & ZZ \end{pmatrix}$$

if k < 0:

$$M_{quad} = \begin{pmatrix} D & 0 & 0 \\ 0 & F & 0 \\ 0 & 0 & ZZ \end{pmatrix}$$

- **Bending magnet** or **septum**:
  Based on: TRACE 3-D Documentation, K. R. Crandall and D. P. Rusthoi, Third Edition (LA-UR-97-886), May 1997, Los Alamos National Laboratory, Page 14
  `https://laacg.lanl.gov/laacg/services/traceman.pdf`

- **Change of frame**:

$$x_2 = x_1 + \delta x \qquad y_2 = y_1 + \delta y \qquad z_2 = z_1 + \delta z$$

$$px_2 = px_1 + \delta px \qquad py_2 = py_1 + \delta py \qquad pz_2 = pz_1 + \delta pz$$

- **Kicker** with kick in px:

Change of frame:

$$px_2 = px_1 + \frac{kick}{2}$$

Propagation within the kicker

$$M = M_{drift}(L_{kicker})$$

Change of frame:

$$px_2 = px_1 + \frac{kick}{2}$$

- **Steerer** with kick of Dev$_x$ in px and Dev$_y$ in py:

Propagation within half of the steerer:

$$M = M_{drift}(\frac{L_{str}}{2})$$

Change of frame:

$$px_2 = px_1 + Dev_x$$

$$py_2 = py_1 + Dev_y$$

Propagation within half of the steerer:

$$M = M_{drift}(\frac{L_{str}}{2})$$

In my presentation steerer have no length, hence it is simply a change of frame on px and py plane.

- **Screen**, **BPM**: specific element that does not have any effect on the beam

- **Start**, **end**: specific element to defined the beginning and the end of the line

Base on 2 classes:

acc_el

Attributes

- type_el: string depending on element type (quad,bend...)
- name: specific name (cf ThomX nomenclature)
- at: position of beginning of the element
- length: length of the element
- save_calc: properties to save calculation after the element
- variables: list of other useful parameter (depending on element type)

accelerator

- name: Name of the accelerator
- energy: Energy of the electron considered (50 MeV on my presentation)
- list_el: list of object of type acc_el that characterise a line of an accelerator

Backup

Analytics code on MatLab: structure

acc_el

- acc_el(type_el, name, position, length, variables, save_calc): constructor
- eq(acc_el,acc_el2): Comparator
- is_quasi_equal,acc_el,acc_el2): Comparator with tolerance of 10^{-14} on position and length of the element and without name comparison
- matrix(acc_el, gamma_square, exact_val): return the transfer matrix of the corresponding element

accelerator

Method

- accelerator(name,energy,list_el): Constructor
- add(acceletator, acc_el): add acc_el at the end of the list of element
- add_at_begining(acceletator, acc_el): add acc_el at the beginning of the list of element
- take_part_of_line(accelerator, name_start_line,name_end_line): return an accelerator that begin at the beginning of name_start_line and end at the end of name_end_line
- take_invert_of_line(accelerator): invert a line to permit calculation of inverse propagation
- calcul_propagation(accelerator, exact_val, name_file): Compute propagation from the first element of list_el to the last one. Output are save on the file name_file and calculation may use ever exact value or approximate one (only small differences has been seen)

All parameter of acc_el may be used as symbolic parameter.

The general solution is then:

$$\begin{cases} x_2 & = & f_x(x_1, px_1, y_1, py_1, z_1, pz_1, \text{symbolic parameters}) \\ px_2 & = & f_{px}(x_1, px_1, y_1, py_1, z_1, pz_1, \text{symbolic parameters}) \\ y_2 & = & f_y(x_1, px_1, y_1, py_1, z_1, pz_1, \text{symbolic parameters}) \\ py_2 & = & f_{py}(x_1, px_1, y_1, py_1, z_1, pz_1, \text{symbolic parameters}) \\ z_2 & = & f_z(x_1, px_1, y_1, py_1, z_1, pz_1, \text{symbolic parameters}) \\ pz_2 & = & f_{pz}(x_1, px_1, y_1, py_1, z_1, pz_1, \text{symbolic parameters}) \end{cases}$$

Where f_{i}, for i = x,px,y,py,z,pz, is a linear function of its variable.

Hypothesis:

- Linear calculation
- X and Y plans uncoupled
- coupling between x and z because of bending magnet
- Symbolic parameters: Only the kick of the kicker in x and px planes from RI-C1/DG/BPM02 and RI-C1/DG/BPM03 of the transfer line
- No other change of frame, hence no constant value

Form of analytic output:

$$
\begin{cases}
\mathbf{X}_{BPM3} & = a_{x,x}\mathbf{X}_{BPM2} + a_{x,px}\mathbf{PX}_{BPM2} + a_{x,z}\mathbf{Z}_{BPM2} + a_{x,pz}\mathbf{PZ}_{BPM2} + a_{x,kick}\mathbf{KICK} \\
\mathbf{PX}_{BPM3} & = a_{px,x}\mathbf{X}_{BPM2} + a_{px,px}\mathbf{PX}_{BPM2} + a_{px,z}\mathbf{Z}_{BPM2} + a_{px,pz}\mathbf{PZ}_{BPM2} + a_{px,kick}\mathbf{KICK} \\
\mathbf{Y}_{BPM3} & = a_{y,y}\mathbf{Y}_{BPM2} + a_{y,py}\mathbf{PY}_{BPM2} + a_{y,z}\mathbf{Z}_{BPM2} + a_{y,pz}\mathbf{PZ}_{BPM2} \\
\mathbf{PY}_{BPM3} & = a_{py,y}\mathbf{Y}_{BPM2} + a_{py,py}\mathbf{PY}_{BPM2} + a_{py,z}\mathbf{Z}_{BPM2} + a_{py,pz}\mathbf{PZ}_{BPM2} \\
\mathbf{Z}_{BPM3} & = a_{z,x}\mathbf{X}_{BPM2} + a_{z,px}\mathbf{PX}_{BPM2} + a_{z,z}\mathbf{Z}_{BPM2} + a_{z,pz}\mathbf{PZ}_{BPM2} \\
\mathbf{PZ}_{BPM3} & = a_{y,pz}\mathbf{PZ}_{BPM2}
\end{cases}
$$

Backup

Simplification and resolution for initial values

Assumption:

- $Z_{BPM2} = 0$
- $PZ_{BPM2} = 0$
- $\mathbf{KICK}_{init}$ known (value use to track the particle)
- $\mathbf{X}_{BPM2,init}$,$\mathbf{Y}_{BPM2,init}$,$\mathbf{X}_{BPM3,init}$ and $\mathbf{Y}_{BPM3,init}$ known (computed or measured values)
- Unknown :
  - $\mathbf{PX}_{BPM2,init}$
  - $\mathbf{PY}_{BPM2,init}$

Hence:

$$\begin{cases} \mathbf{X}_{BPM3,init} = a_{x,x}\mathbf{X}_{BPM2,init} + a_{x,px}\mathbf{PX}_{BPM2,init} + a_{x,kick}\mathbf{KICK}_{init} \\ \mathbf{Y}_{BPM3,init} = a_{y,y}\mathbf{Y}_{BPM2,init} + a_{y,py}\mathbf{PY}_{BPM2,init} \end{cases}$$

Finally :

$$\begin{cases} \mathbf{PX}_{BPM2,init} = \frac{\mathbf{X}_{BPM3,init} - a_{x,x}\mathbf{X}_{BPM2,init} - a_{x,kick}\mathbf{KICK}_{init}}{a_{x,px}} \\ \mathbf{PY}_{BPM2,init} = \frac{\mathbf{Y}_{BPM3,init} - a_{y,y}\mathbf{Y}_{BPM2,init}}{a_{y,py}} \end{cases}$$

Backup

Simplification and resolution for wanted value

Assumption:

- $Z_{BPM2} = 0$
- $PZ_{BPM2} = 0$
- $PX_{BPM2,w} = 0$
- $\mathbf{X}_{BPM2,w}$, $\mathbf{Y}_{BPM2,w}$, $\mathbf{X}_{BPM3,w}$ and $\mathbf{Y}_{BPM3,w}$ known (wanted values, 0 in the particle reference frame)
- Unknown :
  - $\mathbf{PX}_{BPM2,w}$
  - $\mathbf{KICK}_w$
  - $\mathbf{PY}_{BPM2,w}$

Hence:

$$\begin{cases} \mathbf{X}_{BPM3,w} & = & a_{x,x}\mathbf{X}_{BPM2,w} & +a_{x,px}\mathbf{PX}_{BPM2,w} & +a_{x,kick}\mathbf{KICK}_w \\ \mathbf{PX}_{BPM3,w} & = & a_{px,x}\mathbf{X}_{BPM2,w} & +a_{px,px}\mathbf{PX}_{BPM2,w} & +a_{px,kick}\mathbf{KICK}_w & = & 0 \\ \mathbf{Y}_{BPM3,w} & = & a_{y,y}\mathbf{Y}_{BPM2,w} & +a_{y,py}\mathbf{PY}_{BPM2,w} \end{cases}$$

Finally :

$$\begin{cases} \mathbf{PX}_{BPM2,w} & = & \frac{a_{x,kick}\times\mathbf{X}_{BPM3,w}-(a_{px,kick}\times a_{x,x}-a_{px,x}\times a_{x,kick})\times\mathbf{X}_{BPM2,w}}{a_{px,px}\times a_{x,kick}-a_{x,x}\times a_{px,kick}} \\ \mathbf{KICK}_w & = & \frac{a_{px,px}\times\mathbf{X}_{BPM3,w}-(a_{px,px}\times a_{x,x}-a_{px,x}\times a_{x,px})\times\mathbf{X}_{BPM2,w}}{a_{x,x}\times a_{px,kick}-a_{px,px}\times a_{x,kick}} \\ \mathbf{PY}_{BPM2,w} & = & \frac{\mathbf{Y}_{BPM3,w}-a_{y,y}\mathbf{Y}_{BPM2,w}}{a_{y,py}} \end{cases}$$

Backup

Detailed calculus of Dev_x3 and Dev_x3

Hypothesis:

- Linear calculation
- X and Y plans uncoupled
- coupling between x and z because of bending magnet
- Symbolic parameters = $\mathbf{Dev}_{x3}$,$\mathbf{Dev}_{y3}$,$\mathbf{Dev}_{x4}$,$\mathbf{Dev}_{y4}$ from BPM RI-C1/DG/BPM02 to TL/AE/STR03

Form of analytic output:

$$\begin{cases} \mathbf{X}_2 & = a_{x,x}\mathbf{X} & +a_{x,px}\mathbf{PX} & +a_{x,z}\mathbf{Z} & +a_{x,pz}\mathbf{PZ} & +a_{x,devx3}\mathbf{Dev}_{x3} & +a_{x,devx4}\mathbf{Dev}_{x4} \\ \mathbf{PX}_2 & = a_{px,x}\mathbf{X} & +a_{px,px}\mathbf{PX} & +a_{px,z}\mathbf{Z} & +a_{px,pz}\mathbf{PZ} & +a_{px,devx3}\mathbf{Dev}_{x3} & +a_{px,devx4}\mathbf{Dev}_{x4} \\ \mathbf{Y}_2 & = a_{y,y}\mathbf{Y} & +a_{y,py}\mathbf{PY} & +a_{y,z}\mathbf{Z} & +a_{y,pz}\mathbf{PZ} & +a_{y,devy3}\mathbf{Dev}_{y3} & +a_{y,devy4}\mathbf{Dev}_{y4} \\ \mathbf{PY}_2 & = a_{py,y}\mathbf{Y} & +a_{py,py}\mathbf{PY} & +a_{py,z}\mathbf{Z} & +a_{py,pz}\mathbf{PZ} & +a_{py,devy3}\mathbf{Dev}_{y3} & +a_{py,devy4}\mathbf{Dev}_{y4} \\ \mathbf{Z}_2 & = a_{z,x}\mathbf{X} & +a_{z,px}\mathbf{PX} & +a_{z,z}\mathbf{Z} & +a_{z,pz}\mathbf{PZ} & +a_{z,devx3}\mathbf{Dev}_{x3} & +a_{z,devx4}\mathbf{Dev}_{x4} \\ \mathbf{PZ}_2 & = a_{y,pz}\mathbf{PZ} \end{cases}$$

Where 2 index symbolise steerer's position and non index symbolise BPM position

For each variable there is also a constant term because of change of frame at the end of the septum. This term is not considered there because it would be simplify when one do "initial system" = "wanted system".

Backup

Simplification

Assumption:

- $Z_{BPM2} = 0$
- $PZ_{BPM2} = 0$
- $a_{x,devx3} = 0 = a_{y,devy3}$ because calculus are stop at the end of the steerer TL/AE/STR03 simulated by no-length change of variable in px and py

There is 2 systems:

- Initial system:

$$\begin{cases} \mathbf{X}_{str3,i} & = a_{x,x}\mathbf{X}_{BPM2,i} & + a_{x,px}\mathbf{PX}_{BPM2,i} & & + a_{x,devx4}\mathbf{Dev}_{x4,i} \\ \mathbf{PX}_{str3,i} & = a_{px,x}\mathbf{X}_{BPM2,i} & + a_{px,px}\mathbf{PX}_{BPM2,i} & + a_{px,devx3}\mathbf{Dev}_{x3,i} & + a_{px,devx4}\mathbf{Dev}_{x4,i} \\ \mathbf{Y}_{str3,i} & = a_{y,y}\mathbf{Y}_{BPM2,i} & + a_{y,py}\mathbf{PY}_{BPM2,i} & & + a_{y,devy4}\mathbf{Dev}_{y4,i} \\ \mathbf{PY}_{str3,i} & = a_{py,y}\mathbf{Y}_{BPM2,i} & + a_{py,py}\mathbf{PY}_{BPM2,i} & + a_{py,devy3}\mathbf{Dev}_{y3,i} & + a_{py,devy4}\mathbf{Dev}_{y4,i} \end{cases}$$

- Wanted system:

$$\begin{cases} \mathbf{X}_{str3,w} & = b_{x,x}\mathbf{X}_{BPM2,w} & + b_{x,px}\mathbf{PX}_{BPM2,w} & & + b_{x,devx4}\mathbf{Dev}_{x4,w} \\ \mathbf{PX}_{str3,w} & = b_{px,x}\mathbf{X}_{BPM2,w} & + b_{px,px}\mathbf{PX}_{BPM2,w} & + b_{px,devx3}\mathbf{Dev}_{x3,w} & + b_{px,devx4}\mathbf{Dev}_{x4,w} \\ \mathbf{Y}_{str3,w} & = b_{y,y}\mathbf{Y}_{BPM2,w} & + b_{y,py}\mathbf{PY}_{BPM2,w} & & + b_{y,devy4}\mathbf{Dev}_{y4,w} \\ \mathbf{PY}_{str3,w} & = b_{py,y}\mathbf{Y}_{BPM2,w} & + b_{py,py}\mathbf{PY}_{BPM2,w} & + b_{py,devy3}\mathbf{Dev}_{y3,w} & + b_{py,devy4}\mathbf{Dev}_{y4,w} \end{cases}$$

Backup

Simplification and resolution

- Unknown:
  - $\mathbf{Dev}_{x3,w}$
  - $\mathbf{Dev}_{y3,w}$
  - $\mathbf{Dev}_{x4,w}$
  - $\mathbf{Dev}_{y4,w}$
- Known:
  - All other variables

To ensure physical coherence, $(X,PX,Y,PY)_{str3,i} = (X,PX,Y,PY)_{str3,w}$

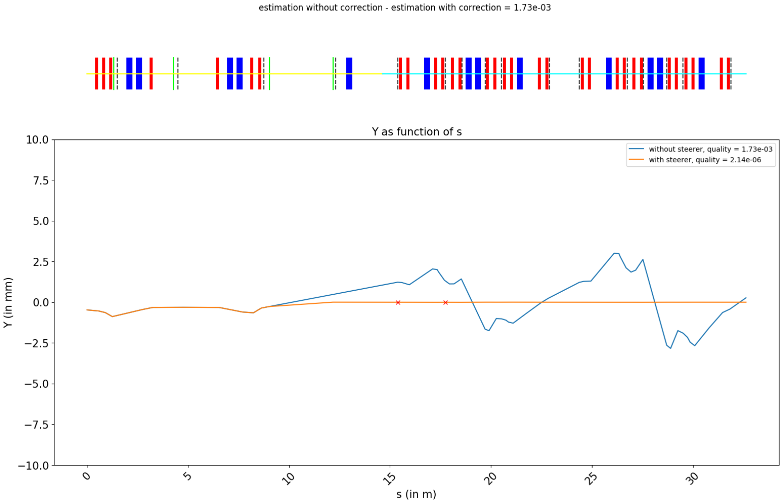
Finally:

$$\begin{cases} \mathbf{Dev}_{x4,w} = \dfrac{a_{x,x}\mathbf{X}_i + a_{x,px}\mathbf{PX}_i + a_{x,devx4}\mathbf{Dev}_{x4,i} - b_{x,x}\mathbf{X}_w - b_{x,px}\mathbf{PX}_w}{b_{x,devx4}} \\ \mathbf{Dev}_{y4,w} = \dfrac{a_{y,y}\mathbf{Y}_i + a_{y,py}\mathbf{PY}_i + a_{y,devy4}\mathbf{Dev}_{y4,i} - b_{y,y}\mathbf{Y}_w - b_{y,py}\mathbf{PY}_w}{a_{y,devy4}} \\ \mathbf{Dev}_{x3,w} = \dfrac{a_{px,x}\mathbf{X}_i + a_{px,px}\mathbf{PX}_i + a_{px,devx3}\mathbf{Dev}_{x3,i} + a_{px,devx4}\mathbf{Dev}_{x4,i} - b_{px,x}\mathbf{X}_w - b_{px,px}\mathbf{PX}_w - b_{px,devx4}\mathbf{Dev}_{x4}}{b_{px,devx3}} \\ \mathbf{Dev}_{y3,w} = \dfrac{a_{py,y}\mathbf{Y}_i + a_{py,py}\mathbf{PY}_i + a_{py,devy3}\mathbf{Dev}_{y3,i} + a_{py,devy4}\mathbf{Dev}_{y4,i} - b_{py,y}\mathbf{Y}_w - b_{py,py}\mathbf{PY}_w - b_{py,devy4}\mathbf{Dev}_{y4}}{b_{py,devy3}} \end{cases}$$

Where $\mathbf{U}_v = \mathbf{U}_{BPM2,v}$, with $\mathbf{U} = \mathbf{X}, \mathbf{Y}, \mathbf{PX}, \mathbf{PY}$ and $v = i, w$

Backup

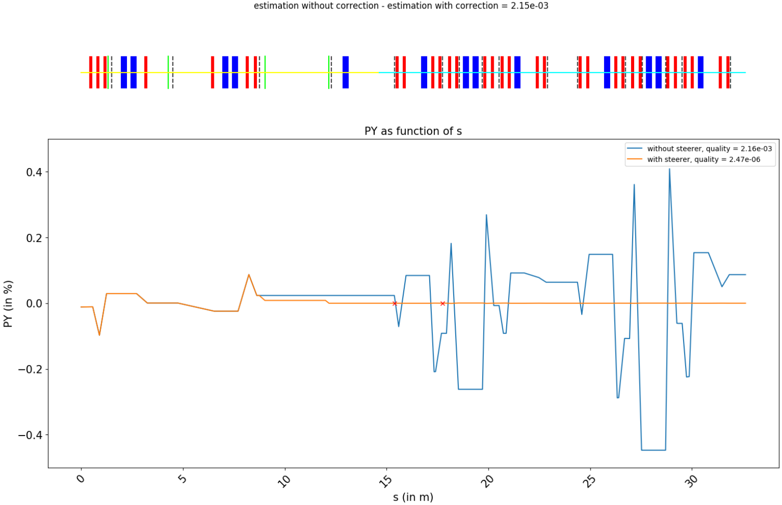
In Y plane without kicker modification



- Blue line :
  - Without any deviation
  - Oscillation on the ring that grows up
- Orange line :
  - With computed deviation
  - No visible oscillation

Backup

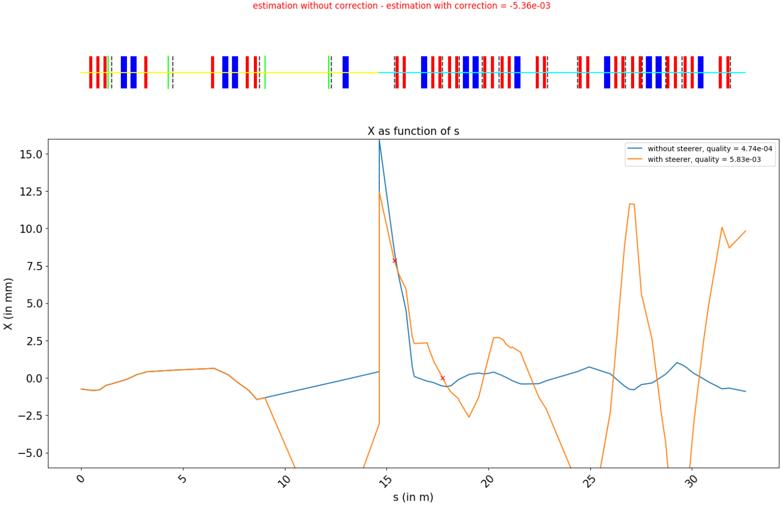
In PY plane without kicker modification



- Blue line :
  - Without any deviation
  - Oscillation on the ring grows up
- Orange line :
  - With computation deviation
  - No visible oscillation

Backup

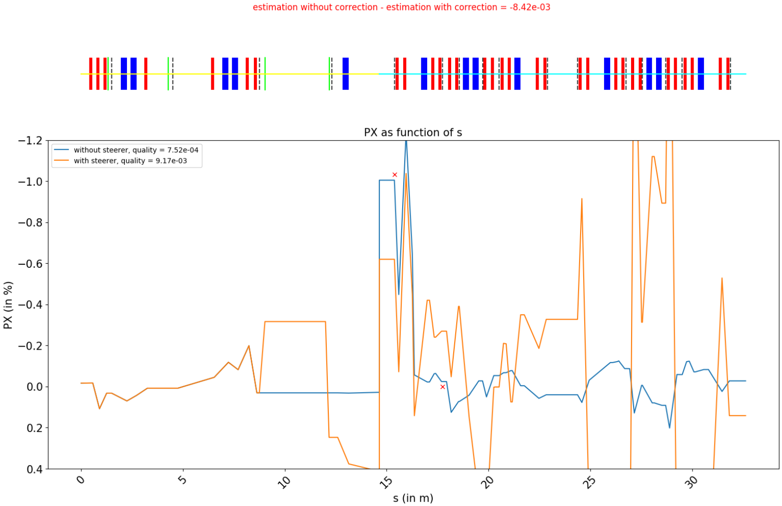
In X plane without kicker modification



- Blue line :
  - Without any deviation
  - Small oscillation on the ring
- Orange line :
  - With calculate deviation
  - Increase of oscillation

Backup

In PX plane kicker modification



- Blue line :
  - Without any deviation
  - Small oscillation
- Orange line :
  - With calculate deviation
  - Increase of the oscillation

Backup
Analyse of results

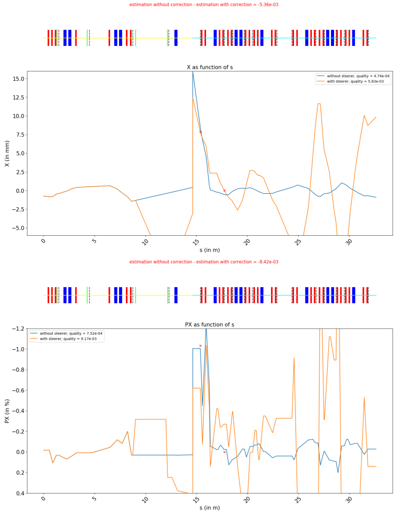
- In y-py plane
  - Particle is better injected on the ring
  - No modification of computation
- In x-px plane
  - Increase of oscillation
  - Beam losses likely after some turn
  - Need other correction
- Solution for x-px plane:
  - Force $PX_{BPM3} = 0$ to avoid oscillations
  - Need one more decrease of freedom.

  Solution apply : Adapt kick of the kicker

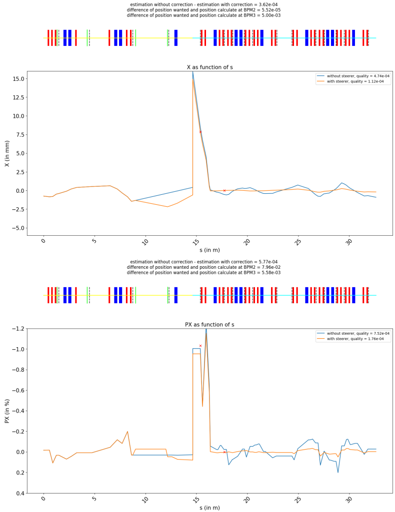
Backup

Comparison with and without kicker modification in x plane

Without kicker modification



With kicker modification

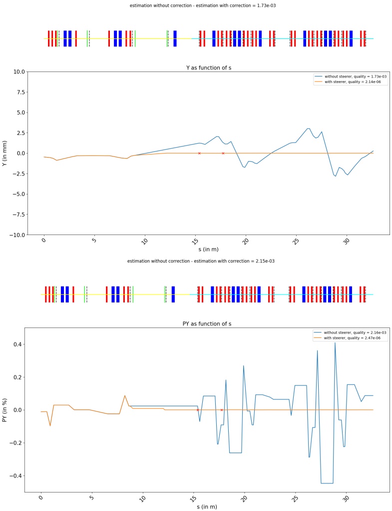


Improvement of storage.

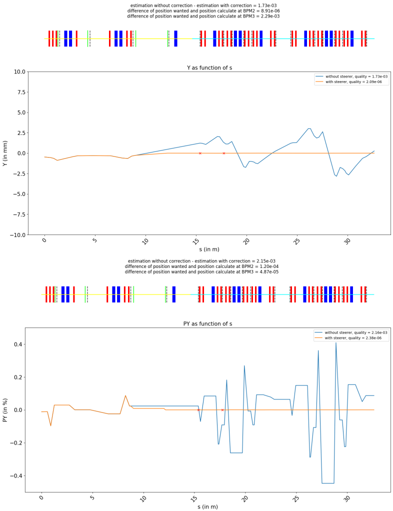
Backup

Comparison with and without kicker modification in y plane

Without kicker modification



With kicker modification



Computation stay the same.