

ThomX presentation: Injection feedback

Alexandre Moutardier

IJCLab, Orsay, France

December 10th, 2020

Goal

Goal: Create feedback for injection into the ThomX ring.

- Codes use :
 - ▶ Analytic transfer matrix calculation using MatLab
 - ▶ Propagation calculation using MadX
 - ▶ Calculus of deviation needed to correct beam injection using python
- Simulations done:
 - ▶ Test of injection correction code
 - ▶ Fined injection solution by iterative method
 - ▶ Sensitivity of beam injection
- Projected map of loss

Codes

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

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

For a **Drift** of length L:

$$M_{drift} = \begin{pmatrix} XX & 0 & 0 \\ 0 & YY & 0 \\ 0 & 0 & ZZ \end{pmatrix}$$

$$XX = YY = \begin{pmatrix} 1 & L \\ 0 & 1 \end{pmatrix}$$

$$ZZ = \begin{pmatrix} 1 & L/\gamma^2 \\ 0 & 1 \end{pmatrix}$$

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

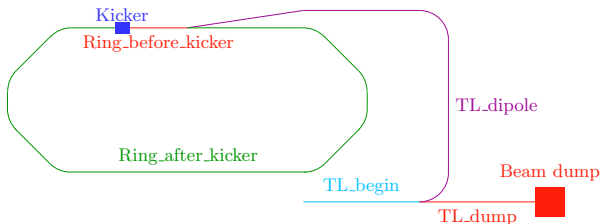
More matrix definition on [backup slide 29](#)

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.

- What is MadX ?
 - ▶ MadX : Methodical Accelerator Design version 10
 - ▶ Develop by CERN for the LHC
 - ▶ Code that :
 - ★ Simulate circular or linear accelerator
 - ★ Compute beam propagation
 - ★ Tracking singular particle
- Method used:
 - ▶ Based on classical linear transfer matrix calculation
 - ▶ Some non linear comportment added

ThomX representation on MadX

ThomX is split in several line combined in sequences.



```
TL_straight = TL_begin + TL_dump
TL = TL_begin + TL_dipole
Ring_with_kicker = Ring_before_kicker + kicker + Ring_after_kicker
Ring = Ring_before_kicker + drift at kicker place + Ring_after_kicker
TL_first_turn = TL + frame change + Ring_with_kicker
TL_n_turn = TL_first_turn + (n-1)*Ring
```

Figure: Representation of lattice definition of ThomX in MadX

What is the change of frame ?

Simulation of injection in MadX and analytics code

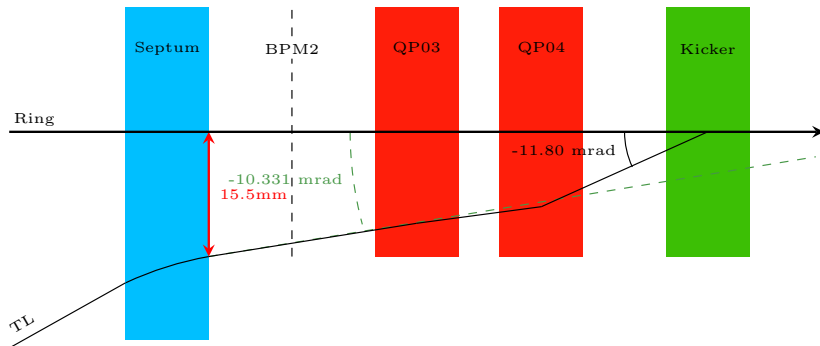


Figure: Diagram of beam injection on ThomX ring (not to scale)

Change of frame apply at the exit of the septum:

- $x_{TL} \rightarrow x_{Ring} + 15.5\text{mm}$
- $px_{TL} \rightarrow px_{Ring} - 10.331\text{mrad}$

Permit to take into account off axis travel in quadrupole at injection.

Warning:

from end of septum ($s = 14.6\text{ m}$) to end of kicker ($s = 16.4\text{ m}$) plot are in the frame of the ring !!!

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 33](#) for more details of calculation

Simulations

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 to see if the injection is better

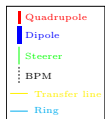
To evaluate the efficiency of the injection, an estimator is used :

$$E_v = \sqrt{\frac{\sum_{Ring} v_{el}^2}{nb_{el}}}$$

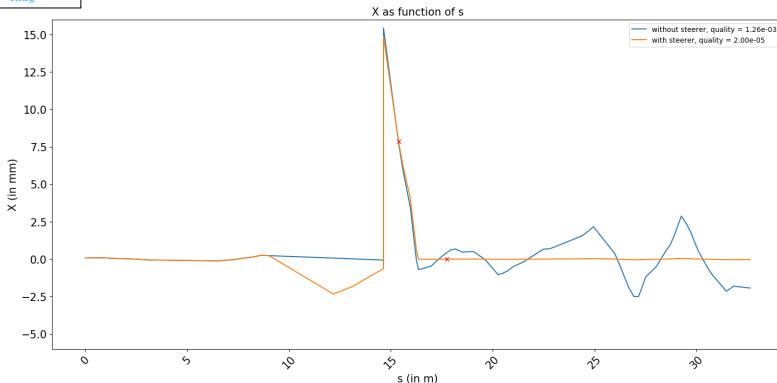
Where v_{el} is the value of v at the exit of the element el , $v = x, px, y, py$ and nb_{el} is the number of element considered.

Beginning of the ring is exclude since element RI-C1/AE/DP.01.

Example of improvement of beam injection in X plane



estimation without correction - estimation with correction = $1.24\text{e-}03$
 difference of position wanted and position calculate at BPM2 = $9.86\text{e-}05$
 difference of position wanted and position calculate at BPM3 = $9.03\text{e-}04$



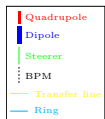
- Blue ligne :

- ▶ Without any deviation
- ▶ Oscillation on the ring grows up
- ▶ Beam may be losses after some turn
- ▶ $E_x = 1.3 \text{ mm}$

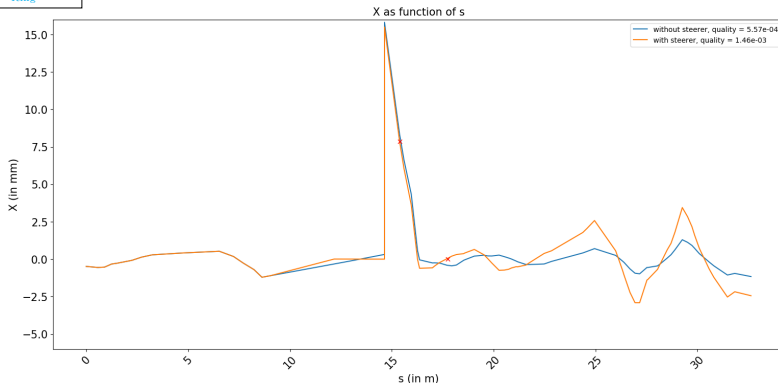
- Orange line :

- ▶ With calculate deviation
- ▶ No oscillation
- ▶ Perfect injection
- ▶ $E_x = 2 \times 10^{-2} \text{ mm}$

Example of deterioration of beam injection in X plane



estimation without correction - estimation with correction = -9.02×10^{-4}
 difference of position wanted and position calculate at BPM2 = 2.54×10^{-5}
 difference of position wanted and position calculate at BPM3 = 6.42×10^{-2}



- Blue ligne :

- ▶ Without any deviation
- ▶ Oscillation on the ring grows up slowly
- ▶ Beam may be losses after some turn
- ▶ $E_x = 0.56 \text{ mm}$

- Orange line :

- ▶ With calculate deviation
- ▶ Larger oscillation
- ▶ Worse injection
- ▶ $E_x = 1.5 \text{ mm}$

Conclusion of direct application of deviation

Plot in other plan are presented in [backup slide 40](#).

After tenth of thousand of simulations :

- In 88% of cases : Improvement of the injection in x plan
- In 12% of cases : Deterioration of the injection in x plan
- In 100% of cases : Improvement of injection in y plan

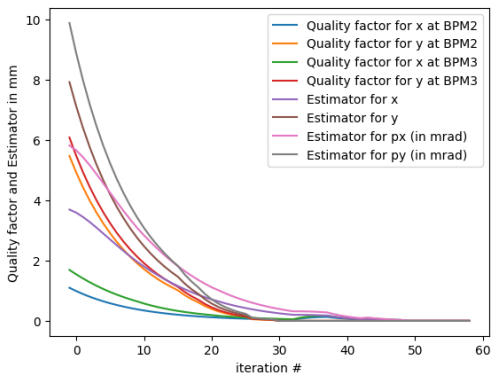
In y plan, there always an improvement that lead to no oscillation of the beam.

To improve efficiency in x plan, an iterative method has been develop.

Iterative method : feedback for ThomX

Protocol of iterative method :

- 1) Simulation of a random particle
- 2) Propagation within TL_first_turn
- 3) Calculation of deviation
- 4) Application of some percentage of the deviation differences between used one and calculate one (for more information see [backup slide 44](#))
- 5) Repeated 60 times from 2)



$$\text{Quality factor} = |V_{\text{wanted}} - V_{\text{measured}}|$$

With $V = x$ or y at BPM2 or BPM3

Figure: Plot of quality factor and estimator for each iteration

Convergence test

Convergence defined by :

- $E_x < 10\mu\text{m}$
and
- $E_y < 10\mu\text{m}$

Evaluation of the beam aperture at position of an element :


- Random particles track from the beginning of the TL to the element
- If the particles is not loss, keep it

- Particle taken within the beam aperture at RI-C1/DG/BPM.03 :

- ▶ Number of particles : 205
 - ▶ Converge within 60 turns : 100%
- With :

- ★ Converge within 45 turns (slow convergence) : 45%
- ▶ Not converge : 0%
- ▶ Loss : 0%

If a particle (and by analogy the centroid of a beam) pass through RI-C1/DG/BPM.03, iterative code **always** permit to correct injection in less than 60 iterations.
Injection correction never leads to particle loss.

 Aperture decreases highly (divided by 2) between septum and second BPM of the ring hence passing through septum is not sufficient. Adaptation of the feedback in those cases are needed.

Conclusion of iterative method of injection correction

Less than 60 iterations are needed to find optimized deviation.

Some tests must be done with small fluctuations in particle position at the entrance of the transfer line to simulate fluctuation of centroid position from one beam to the next.

A way to treat case where particle is lost between septum and RI-C1/DG/BPM.03 must be implemented.

Modification of change of frame definition and kicker simulation

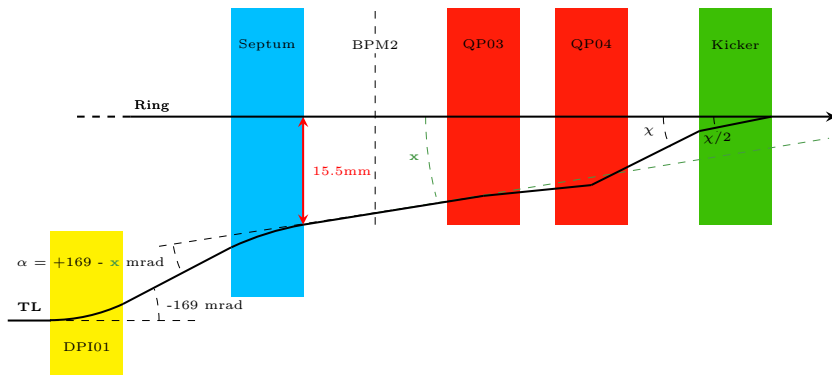


Figure: Diagram of beam injection on ThomX ring (not to scale)

Modification :

- px change of frame : dependent of septum rotation α
- Previous kicker : infinitesimal kicker of kick x at the entrance of the object
- New kicker : 2 infinitesimals half kickers of kick $\frac{x}{2}$ at the entrance and the exit of the object

Modification of change of frame definition and kicker simulation

Optimisation of injection must be redone :

- Propagate a center beam through the transfer line and the first turn of the ring
- Adapt α and χ since beam centroid follow the ring reference path

Nominal parameters found :

- Angle of the septum : $\alpha_0 = 0.159125\text{rad}$
- Kick of the kicker : $\chi_0 = 0.0119965\text{rad}$

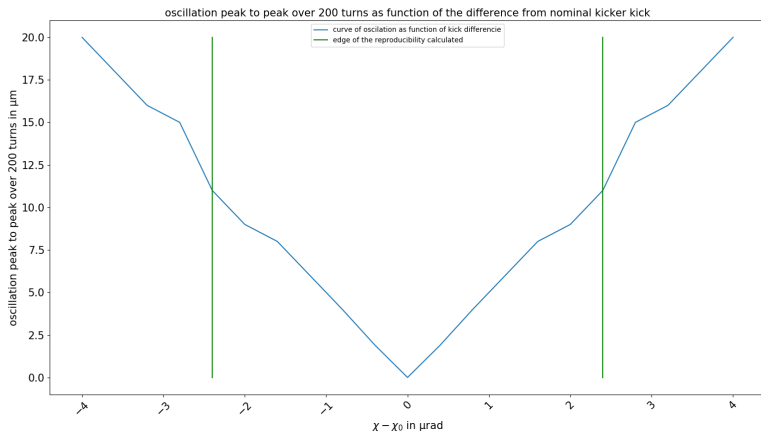
Beam characterization over 200 turn:

- Negligible oscillation of 5 nm
- Dispersion bounded

Remark :

- Feedback has been tested with non-optimum parameters and convergence is also achieved

Sensitivity of the beam injection to the kick of the kicker

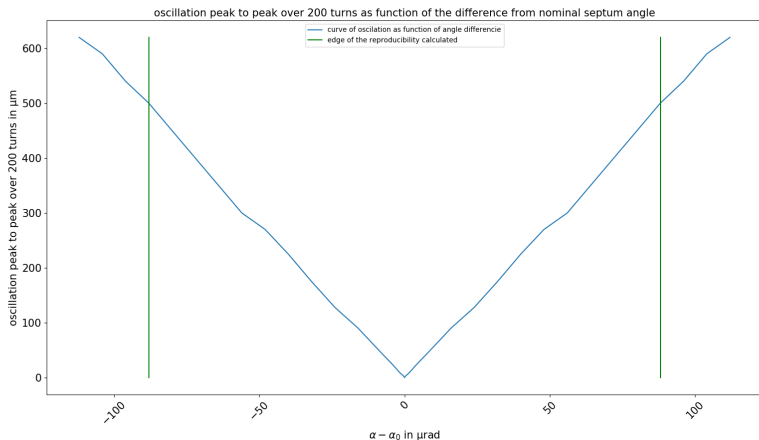


Measure done on the kicker :

- $\frac{\delta B}{B} = 2.0 \times 10^{-4}$; $\chi_0 = 11996.5 \mu\text{rad}$
With δB the standard deviation around wanted value

- $\frac{\delta B}{B} \approx \frac{\delta \chi}{\chi}$
- $\delta \chi = \frac{\delta B}{B} \times \chi = 2.4 \times 10^{-6} \text{ rad}$
- Kicker reproducibility induce oscillation of the order of $10 \mu\text{m}$

Sensitivity of the beam injection to the angle of the septum



Measure done on the septum :

- $\frac{\delta B}{B} = 5.5 \times 10^{-4}$; $\alpha_0 = 159125 \mu\text{rad}$
With δB the standard deviation around wanted value

- $\frac{\delta B}{B} \approx \frac{\delta \alpha}{\alpha}$
- $\delta \alpha = \frac{\delta B}{B} \times \alpha = 8.8 \times 10^{-5} \text{ rad}$
- Septum reproducibility induce oscillation of the order of 500 μm

Projected map of loss

Map of loss projected at the entrance of the transfer line

Beam parameters at exit of linac (according to Alexandre Loulergue calculation for TDR) :

- $\beta_x = 34.46\text{m}$
- $\beta_y = 33.94\text{m}$
- $\alpha_x = -4.24$
- $\alpha_y = -4.34$
- $\epsilon_x = 5 \times 10^{-8} \text{ mm mrad} = \epsilon_y$

Random particles simulated within the beam :

- $\beta_x = 34.46\text{m} \times 100$
- $\beta_y = 33.94\text{m} \times 100$
ie : beam 10 time larger
- same other parameters
- Number of particles : 10 000

Goal :

- Simulate the propagation of particles from the beginning of the transfer line to the end of the first ring turn
- use aperture definition to check loss of particle
- Plot particles position at the beginning of the transfer line according to place of loss of the particle

Map of loss projected at the entrance of the transfer line

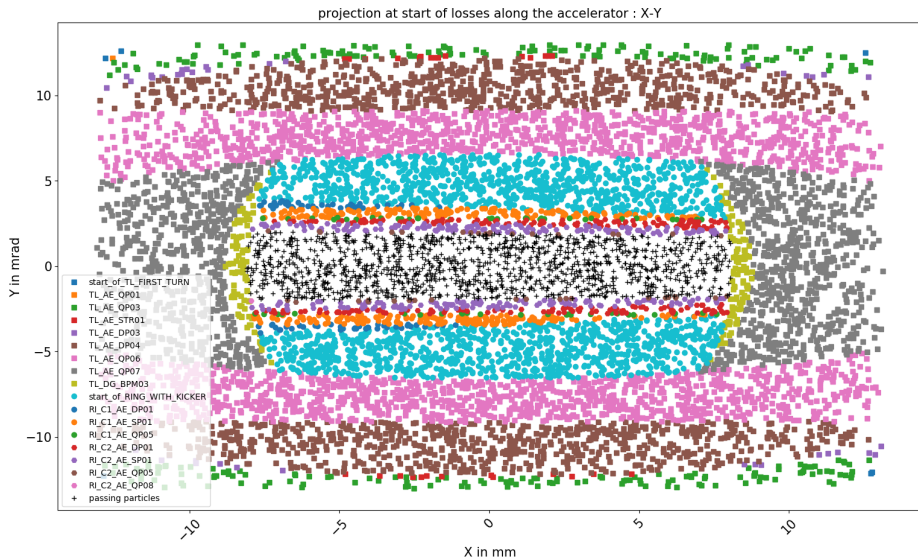


Figure: Plot of losses projected at the entrance of the transfer line

Map of loss projected at screen TL/DG/SST.01-SCR.01

Same plot may be done on each screen.

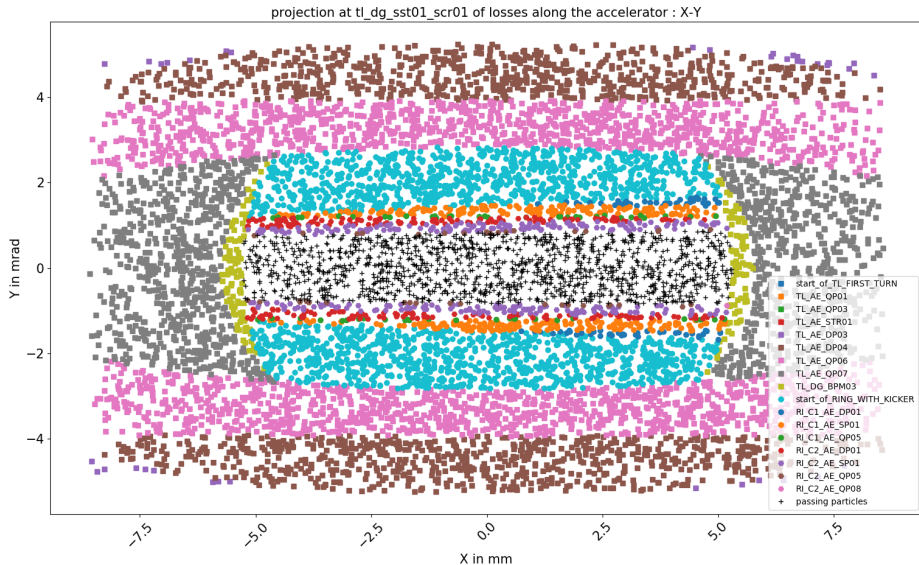


Figure: Plot of losses projected at the position of screen TL/DG/SST.01-SCR.01

Aperture projected at screen TL/DG/SST.01-SCR.01

projection at tl_dg_sst01_scr01 of losses along the accelerator : X-Y

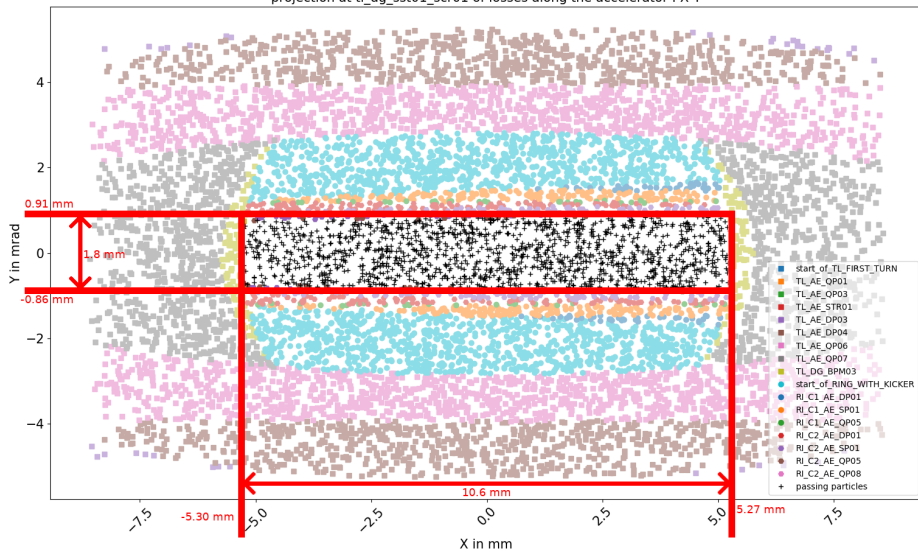


Figure: Plot of loss projected at position of screen TL/DG/SST.01-SCR.01

Projection of loss map on camera

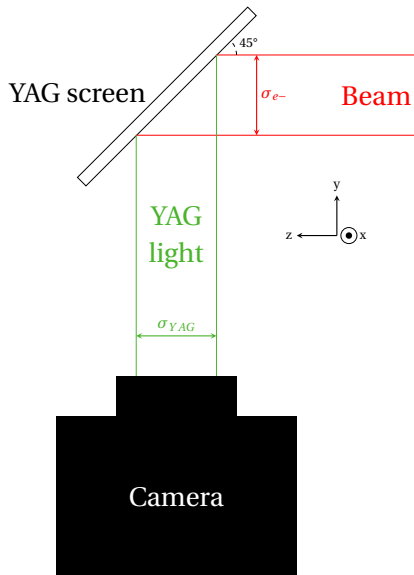


Figure: Sketch of SST stations

Beam observation is done by YAG screen and camera.

Beam interact with a screen at 45° and screen emit light perpendicular to beam propagation.

Geometry of setup permit to say :

- No modification of x plan :

$$\sigma_{screen,x} = \sigma_{beam,x}$$

- x plan is reflected :

$$\sigma_{screen,y} = \sigma_{beam,y}$$

Hence, if one know optical magnification of camera, it's possible to superimposed aperture on images to see directly if some part of the beam may be loss.

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.
- Direct calculation improved injection efficiency in 88% of the cases.
- Iterative method of feedback is efficient at 100%
- Reproducibility has to be check again once septum and kicker are commissioned and calculation as to be done more precisely
- Impact of septum reproducibility has to be investigated
- A way to show beam aperture on SST screen has been created

Next Step :

- Try iterative code with small random variation of position to simulate fluctuation at each shot
- Implement injection feedback the control system of ThomX and couple it with Taurus
- Evaluate aperture for each screen
- Implement aperture visualization on IHM_SST on Taurus

Thanks

Backup

Classical transfer matrices

- **Drift** of length L :

$$XX = YY = \begin{pmatrix} 1 & L \\ 0 & 1 \end{pmatrix}$$

$$ZZ = \begin{pmatrix} 1 & L/\gamma^2 \\ 0 & 1 \end{pmatrix}$$

$$M_{drift} = \begin{pmatrix} XX & 0 & 0 \\ 0 & YY & 0 \\ 0 & 0 & ZZ \end{pmatrix}$$

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

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

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

if $k > 0$:

if $k < 0$:

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

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

- **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>

Backup

Other elements type

- **Change of frame:**

$$x_2 = x_1 + \delta x$$

$$px_2 = px_1 + \delta px$$

$$y_2 = y_1 + \delta y$$

$$py_2 = py_1 + \delta py$$

$$z_2 = z_1 + \delta z$$

$$pz_2 = pz_1 + \delta pz$$

- **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:

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

Change of frame:

$$px_2 = px_1 + Dev_x$$

$$py_2 = py_1 + Dev_y$$

Propagation within half of the steerer:

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

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

Backup

Analytics code on MatLab: `structure`

Base on 2 classes:

`acc_el`

`accelerator`

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)
- `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

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(accelerator, acc_el)`: add `acc_el` at the end of the list of element
- `add_at_begining(accelerator, 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)

Backup

Results returned by analytics code

All parameter of `acc_el` may be used as symbolic parameter.

The general solution is then:

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

Where f_i , for $i = x, px, y, py, z, pz$, is a linear function of its variable.

Backup

Detailed calculus of $(x, p_x, y, p_x)_{BPM2}$ from $(x, y)_{BPM2}$ and $(x, y)_{BPM3}$

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:

- $Z_{BPM2} = 0$
- $PZ_{BPM2} = 0$
- $PX_{BPM2,w} = 0$
- $X_{BPM2,w}$, $Y_{BPM2,w}$, $X_{BPM3,w}$ and $Y_{BPM3,w}$ known (wanted values, 0 in the particle reference frame)
- Unknown :
 - ▶ $PX_{BPM2,w}$
 - ▶ $KICK_w$
 - ▶ $PY_{BPM2,w}$

Hence:

$$\begin{cases} X_{BPM3,w} &= a_{x,x} X_{BPM2,w} + a_{x,px} PX_{BPM2,w} + a_{x,kick} KICK_w \\ PX_{BPM3,w} &= a_{px,x} X_{BPM2,w} + a_{px,px} PX_{BPM2,w} + a_{px,kick} KICK_w \\ Y_{BPM3,w} &= a_{y,y} Y_{BPM2,w} + a_{y,py} PY_{BPM2,w} \end{cases} = 0$$

Finally :

$$\begin{cases} PX_{BPM2,w} &= \frac{a_{x,kick} \times X_{BPM3,w} - (a_{px,kick} \times a_{x,x} - a_{px,x} \times a_{x,kick}) \times X_{BPM2,w}}{a_{px,px} \times a_{x,kick} - a_{x,x} \times a_{px,kick}} \\ KICK_w &= \frac{a_{px,px} \times X_{BPM3,w} - (a_{px,px} \times a_{x,x} - a_{px,x} \times a_{x,px}) \times X_{BPM2,w}}{a_{x,x} \times a_{px,kick} - a_{px,px} \times a_{x,kick}} \\ PY_{BPM2,w} &= \frac{Y_{BPM3,w} - a_{y,y} Y_{BPM2,w}}{a_{y,py}} \end{cases}$$

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 = $Dev_{x3}, Dev_{y3}, Dev_{x4}, Dev_{y4}$ from BPM RI-C1/DG/BPM02 to TL/AE/STR03

Form of analytic output:

$$\left\{ \begin{array}{lclclcl} X_2 & = & a_{x,x}X & +a_{x,px}PX & +a_{x,z}Z & +a_{x,pz}PZ & +a_{x,devx3}Dev_{x3} & +a_{x,devx4}Dev_{x4} \\ PX_2 & = & a_{px,x}X & +a_{px,px}PX & +a_{px,z}Z & +a_{px,pz}PZ & +a_{px,devx3}Dev_{x3} & +a_{px,devx4}Dev_{x4} \\ Y_2 & = & a_{y,y}Y & +a_{y,py}PY & +a_{y,z}Z & +a_{y,pz}PZ & +a_{y,devy3}Dev_{y3} & +a_{y,devy4}Dev_{y4} \\ PY_2 & = & a_{py,y}Y & +a_{py,py}PY & +a_{py,z}Z & +a_{py,pz}PZ & +a_{py,devy3}Dev_{y3} & +a_{py,devy4}Dev_{y4} \\ Z_2 & = & a_{z,x}X & +a_{z,px}PX & +a_{z,z}Z & +a_{z,pz}PZ & +a_{z,devx3}Dev_{x3} & +a_{z,devx4}Dev_{x4} \\ PZ_2 & = & a_{y,pz}PZ & & & & & \end{array} \right.$$

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

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

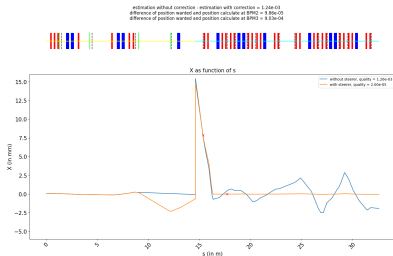
Finally:

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

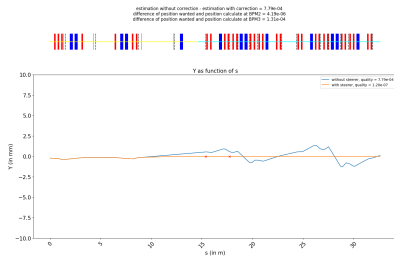
Where $\mathbf{U}_v = \mathbf{U}_{BPM2,v}$, with $\mathbf{U} = \mathbf{X}, \mathbf{Y}, \mathbf{PX}, \mathbf{PY}$ and $v = i, w$

Backup

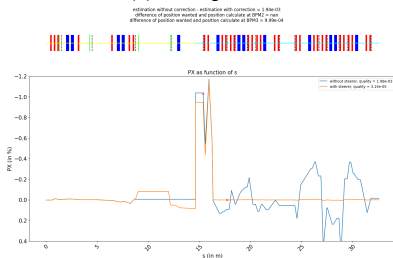
Example of improvement of beam injection in X plane



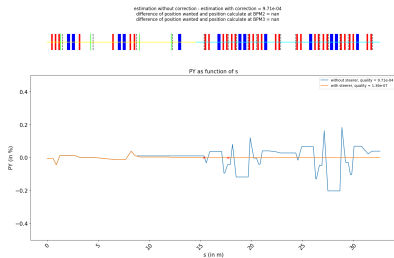
(a) X along the accelerator



(a) Y along the accelerator



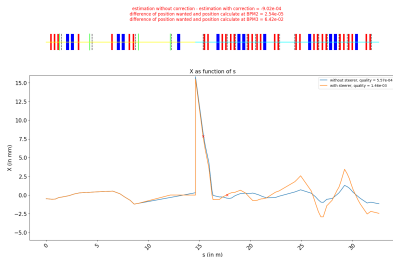
(b) PX along the accelerator



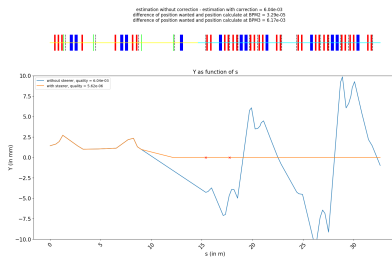
(b) PY along the accelerator

Backup

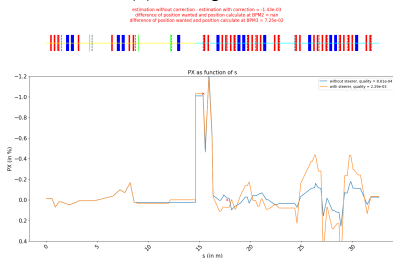
Example of deterioration of beam injection in X plane



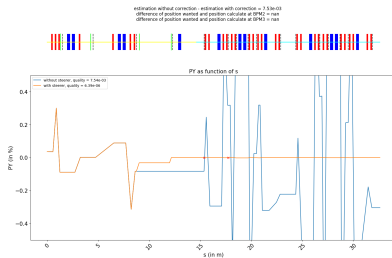
(a) X along the accelerator



(a) Y along the accelerator



(b) PX along the accelerator



(b) PY along the accelerator

Backup

Comparison of injection optimisation with and without modified kick of kicker

At first, only steerer are used to optimise injection.

Calculation of deviation may lead to large oscillation in x plan.

$PX_{RI-C1/DG/BPM.03} = 0$ permit to smooth the injection.

Need one degree of freedom : Kick of injection kicker.

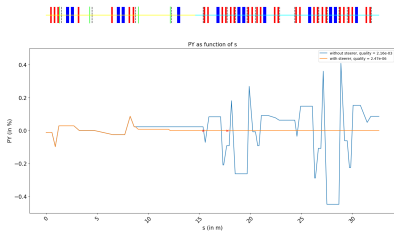
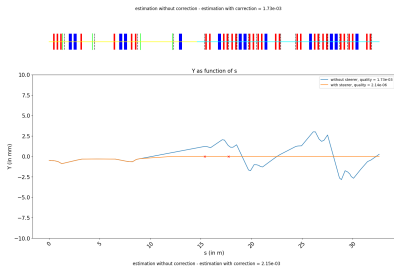
Results :

- In y-py plane
 - ▶ No modification of computation
 - ▶ Injection is improved
- In x-px plane
 - ▶ Less oscillations
 - ▶ Injection is improved

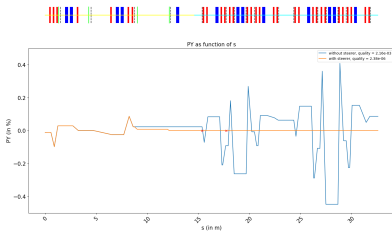
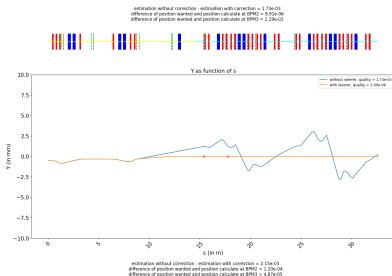
Backup

Comparison with and without kicker modification in y plane

Without kicker modification



With kicker modification



Computation stay the same.

Backup

Precision about point 4) of iterative protocol

Classical deviation $\approx 3 \times 10^{-3}$

At one step:

- If $|dev_u - dev_c| > 10^{-4}$:

$$dev_a = dev_u + \frac{dev_c - dev_u}{10}$$

- If $|dev_u - dev_c| > 10^{-3}$:

$$dev_a = dev_u + \frac{dev_c - dev_u}{5}$$

- Else :

$$dev_a = dev_c$$

With :

- dev_u the deviation use in last calculation
- dev_c the deviation calculated at this step
- dev_a the deviation apply at the next step