Qualifying SixTrackLib :-)

Adrian Oeftiger

Accelerator Physics Meeting
6 June 2019

Setting up SixTrackLib: Strategy I

Goal: Setting up new fast tracker (ultimately with collective effects) based on the hardware (GPU) accelerated tracking code SixTrackLib.

Intermediate steps (*only single-particle tracking*):

- I. SIS-18 and SIS-100 thin lattices from MAD-X into SixTrackLib (STL)
  - ✓ STL: include MAD-X multipole errors in lattice
  - ✗ STL: closed orbit offset and tilt errors missing
  - ✗ STL: dipole edges and local apertures are being implemented
- ✓ II. check for same tunes between MAD-X TWISS and STL tracking
- ✓ III. quadrupole SIS-100 lattice: compare thin lens tracking, MAD-X vs. STL
  - $\longrightarrow$ off- and on-coupling resonance
  - $\Longrightarrow$ effect of coupling induced by exact drift (momentum conservation)
- ✓ IV. SIS-100 lattice with non-linear errors: compare tune scan

Setting up SixTrackLib: Strategy II

Steps *towards multi-particle tracking*:

- split the lattice into many segments
- ✗ inject space charge (SC) nodes in between these tracking sequences
  - ✗ PyHEADTAIL frozen model
  - ✗ PyHEADTAIL self-consistent PIC model
  - ✗ (to be implemented) STL frozen model
- ✗ SIS-18 space charge benchmark
- ✗ SIS-100 tune scan with space charge (+compare to Vera's MAD-X results)

I. Lattice from MAD-X into SixTrackLib

How to set up SixTrackLib (STL)? $\Longrightarrow$ python interface!

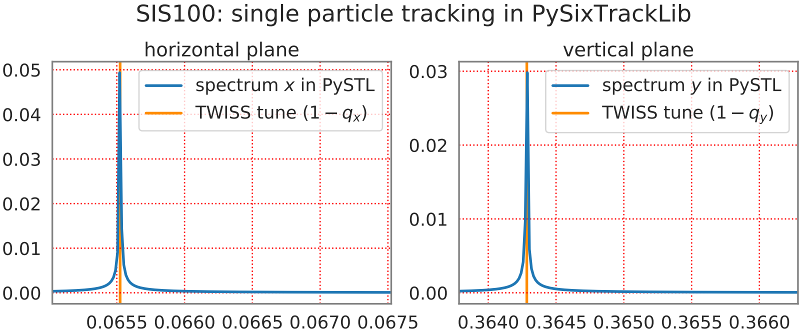
1. load lattice in MAD-X via `cpymad`[1] python library
   - $\longrightarrow$ for the moment, remove `dipedges` (dipole fringe fields)
   - $\longrightarrow$ load magnet error table and add multipole errors to magnet `knl`, `ksl`
2. pass MAD-X sequence to `pysixtracklib`[2] python library
3. define STL beam
4. define STL tracking job (which device to run on, could be usual CPU or a GPU)
5. profit from hardware acceleration :-)

Examples in jupyter notebooks:

- SIS-18 in `cpymad` – MAD-X tracking ↗
- SIS-18 in `PySixTrackLib` incl. MAD-X `makethin` ↗
- SIS-100 tracking with errors in `PySixTrackLib` ↗

[1] `https://pypi.org/project/cpymad/`
[2] shipped with SixTrackLib

Compare FFT spectrum of particle trajectory in STL with MAD-X TWISS tune for SIS-100 lattice (thin lens, dipole edges removed):

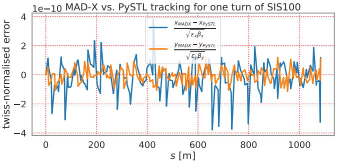


$\Longrightarrow$ thin lens tracking in STL accurately reproduces proper tunes!

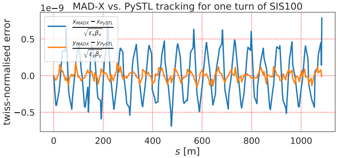
III. MAD-X vs. STL Tracking Comparison

Tracking the SIS-100 clean lattice (no errors, quadrupoles only):
jupyter notebook for 3 working points ↗

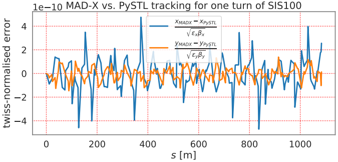
Figure: Benchmark element-by-element MAD-X vs. STL: $\epsilon = \mathcal{O}(10^{-10})$



(a) on-coupling $Q_x = Q_y = 18.88$



(b) off-coupling $Q_x = 18.84,\ Q_y = 18.73$

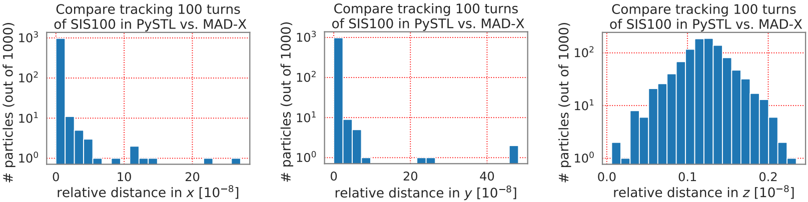


(c) different integers $Q_x = 19.84,\ Q_y = 17.73$

III. MAD-X vs. STL Tracking Comparison

Tracking the SIS-100 clean lattice (no errors, quadrupoles only):
jupyter notebook for 3 working points ↗

Figure: Benchmark 10000 turns MAD-X vs. STL:
transverse $\epsilon_{x,y} = \mathcal{O}(10^{-7})$ and longitudinal $\epsilon_z = \mathcal{O}(10^{-9})$



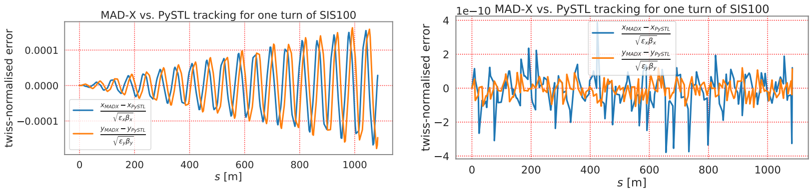
(a) horizontal plane (b) vertical plane (c) longitudinal plane

⟹ single-particle physics in MAD-X and SixTrackLib are equivalent (up to numerical errors)

Coupling Resonance: Emittance Exchange

For equal tunes in a thin-lens quadrupole lattice find emittance exchange:
jupyter notebook ↗

Figure: Comparing MAD-X and SixTackLib for quadrupole-only lattice!



(a) using truncated drift in STL

(b) using exact drift in STL

⟶ MAD-X uses exact drift (full expression with the 3-momentum norm)

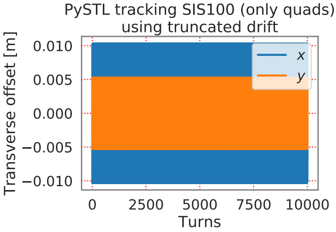
⟶ MAD-X TWISS gives zero coupling ($r11 = r12 = r21 = r22 = 0$)

⟹ higher-order coupling through exact drift!

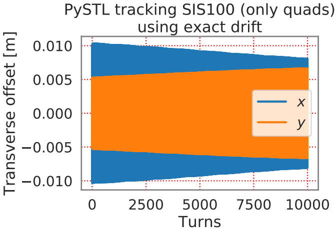
Coupling Resonance: Emittance Exchange

For equal tunes in a thin-lens quadrupole lattice find emittance exchange: jupyter notebook ↗

Figure: SixTrackLib tracking of 10000 turns comparing both drifts



(a) using truncated drift in STL



(b) using exact drift in STL

$\Longrightarrow$ this is a direct measure of how much the *paraxial approximation* is broken for SIS-100 (under this approximation $p_z \approx p_0$ and the truncated drift is identical to the exact drift)

Short Discussion: Drift Space and Coupling

Exact Hamiltonian for a drift space region reads

$$\mathcal{H}(x, p_x, y, p_y, z, p_z; s) = p_z - \sqrt{(1+\delta)^2 - \frac{p_x^2 - p_y^2}{p_0}} \quad . \tag{1}$$

where

x, y: transverse displacement,

p_x, p_y: canon. conj. transverse momenta,

z: longitudinal offset,

p_z: canon. conj. longitudinal momentum,

$\delta = \frac{p_z - p_0}{p_0}$: momentum deviation

$p_0 = \beta\gamma mc$: total momentum

Short Discussion: Drift Space and Coupling

Exact Hamiltonian for a drift space region reads

$$\mathcal{H}(x, p_x, y, p_y, z, p_z; s) = p_z - \sqrt{(1+\delta)^2 - \frac{p_x^2 - p_y^2}{p_0}} \quad . \tag{1}$$

With Hamilton's equations of motion, the transfer map through a drift space of length L becomes

$$x \mapsto x + x' L \qquad p_x \mapsto p_x$$
$$y \mapsto y + y' L \qquad p_y \mapsto p_y$$

with

exact expression	linearisation in p_x, p_y
$x' = \frac{p_x}{\sqrt{p_0^2(1+\delta)^2 - p_x^2 - p_y^2}}$	$x' = \frac{p_x}{p_0(1+\delta)}$
$y' = \frac{p_y}{\sqrt{p_0^2(1+\delta)^2 - p_x^2 - p_y^2}}$	$y' = \frac{p_y}{p_0(1+\delta)}$

see e.g. PhD thesis of Mattias Fjellström, section 2.5 in https://cds.cern.ch/record/1642385/files/CERN-THESIS-2013-248.pdf

Short Discussion: Drift Space and Coupling

Exact Hamiltonian for a drift space region reads

$$\mathcal{H}(x, p_x, y, p_y, z, p_z; s) = p_z - \sqrt{(1+\delta)^2 - \frac{p_x^2 - p_y^2}{p_0}} \quad . \qquad (1)$$

Coupling

exact expression ⇝ transverse coupling: $x \mapsto f(x, p_x, p_y, \delta)$

vs.

linearised expression ⇝ no transverse coupling: $x \mapsto f(x, p_x, \delta)$

with

exact expression	linearisation in p_x, p_y
$x' = \frac{p_x}{\sqrt{p_0^2(1+\delta)^2 - p_x^2 - p_y^2}}$	$x' = \frac{p_x}{p_0(1+\delta)}$
$y' = \frac{p_y}{\sqrt{p_0^2(1+\delta)^2 - p_x^2 - p_y^2}}$	$y' = \frac{p_y}{p_0(1+\delta)}$

see e.g. PhD thesis of Mattias Fjellström, section 2.5 in `https://cds.cern.ch/record/1642385/files/CERN-THESIS-2013-248.pdf`

IV. SIS-100 Tune Scan in SixTrackLib

The SIS-100 benchmark on **SixTrackLib tracking with errors (no space charge)** is available in the aoeftiger github repository ↗. Have a look at the evaluation notebook comparing the results to Vera's results:

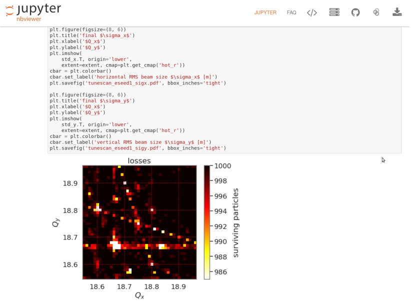
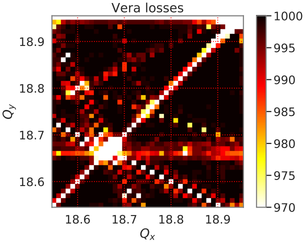


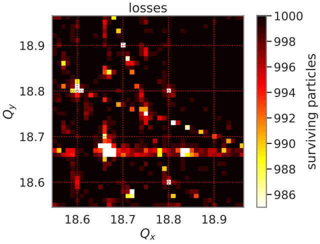
Figure: jupyter notebook for evaluation ↗

⟹ serial tune scan finishes in less than half a working day on NVIDIA V100 GPU cards (for 20000 turns and 1000 particles)

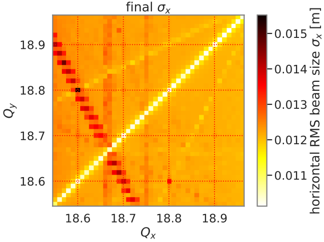
Tune Scan Results



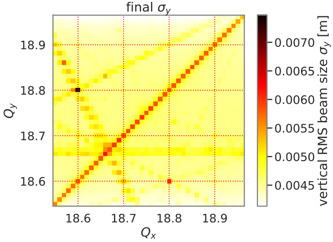
(a) Vera's loss results



(b) STL loss results (global aperture 1 m)



(c) STL horizontal beam growth



(d) STL vertical beam growth

Summary & Outlook

Status today:

✓ SixTrackLib models same physics as MAD-X (single-particle thin lens tracking)

✓ SixTrackLib allows to run on GPUs with $\mathcal{O}(1000)$ speed-up

✓ SixTrackLib is being installed on GSI 150 nodes AMD GPU cluster

$\Longrightarrow$ we are almost ready for production scale single-particle simulations!

Next steps:

- implement interface with PyHEADTAIL on GPU (space charge, impedances, ...)
- qualify new fast tracker with space charge: SIS-18 benchmark suite, SIS-100 MAD-X results
- investigate applicability of frozen SC models vs. self-consistent SC models w.r.t. resonance studies