

SixTrack loss rate simulations and automatic exploration framework

M. Schenk for the EPFL team
EPFL & CERN, Switzerland

Acknowledgements

R. Bruce, R. De Maria, M. Giovannozzi, G. Iadarola, A. Mereghetti, D. Mirarchi, T. Persson, S. Redaelli

Objectives

Complement / compare LHC beam loss model based on experimental data (SDSC) with SixTrack simulations

- **Goals**
 1. **Loss rate dependency and sensitivity to machine knobs:**
How do chromaticity, octupoles, tunes, etc. affect long-term (10^6+ turns) particle loss?
 2. **H / V loss distribution and asymmetry**
 3. **Surrogate model on SixTrack simulation data:** *automatic exploration of parameter space*
- **Type of studies**
 - Similar to **dynamic aperture** studies
 - **Aperture:** only primary collimators
 - Results of **qualitative nature**
- **Framework**
 - [PySixDesk](#) / [SixTrack](#)
 - CERN HTCondor cluster, moving to [BOINC](#) eventually

Contents

Part I: SixTrack simulations

- Simulation set-up
- Post-processing & analysis
- Some test results
- Thoughts and future plans

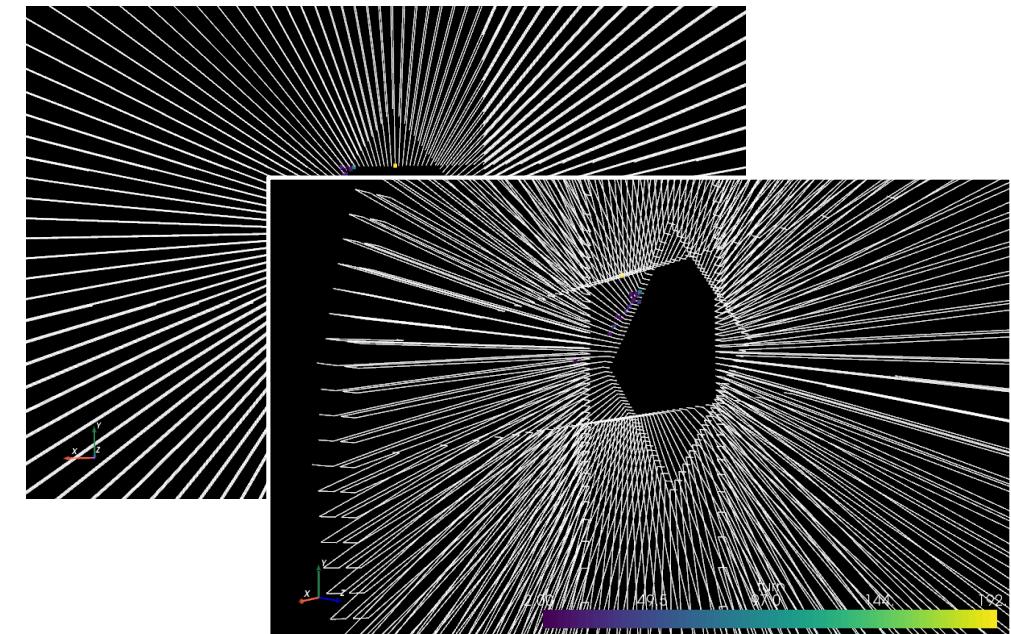
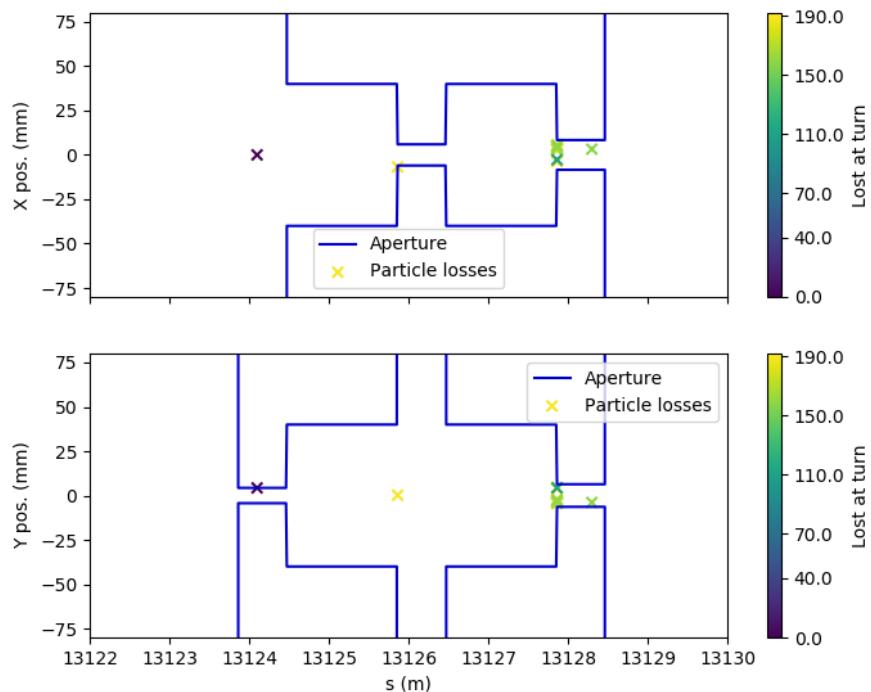
Part II: Automatic parameter exploration

- Objective
- SMT package and EGO algorithm
- Interface with PySixDesk
- Test in 2D parameter space
- Other ideas, issues, what's next?

Simulation set-up

Aperture model: primary collimators

- **Black absorber model:** expect losses only on primary collimators since our model neglects scattering
- **Full aperture model not required for our purposes:** use only the main LHC collimators (V, H, S)
- Half gaps set in our model **depend on beta functions** at elements
(Collimator gaps from [N. Fuster-Martinez, Evian 2019](#))

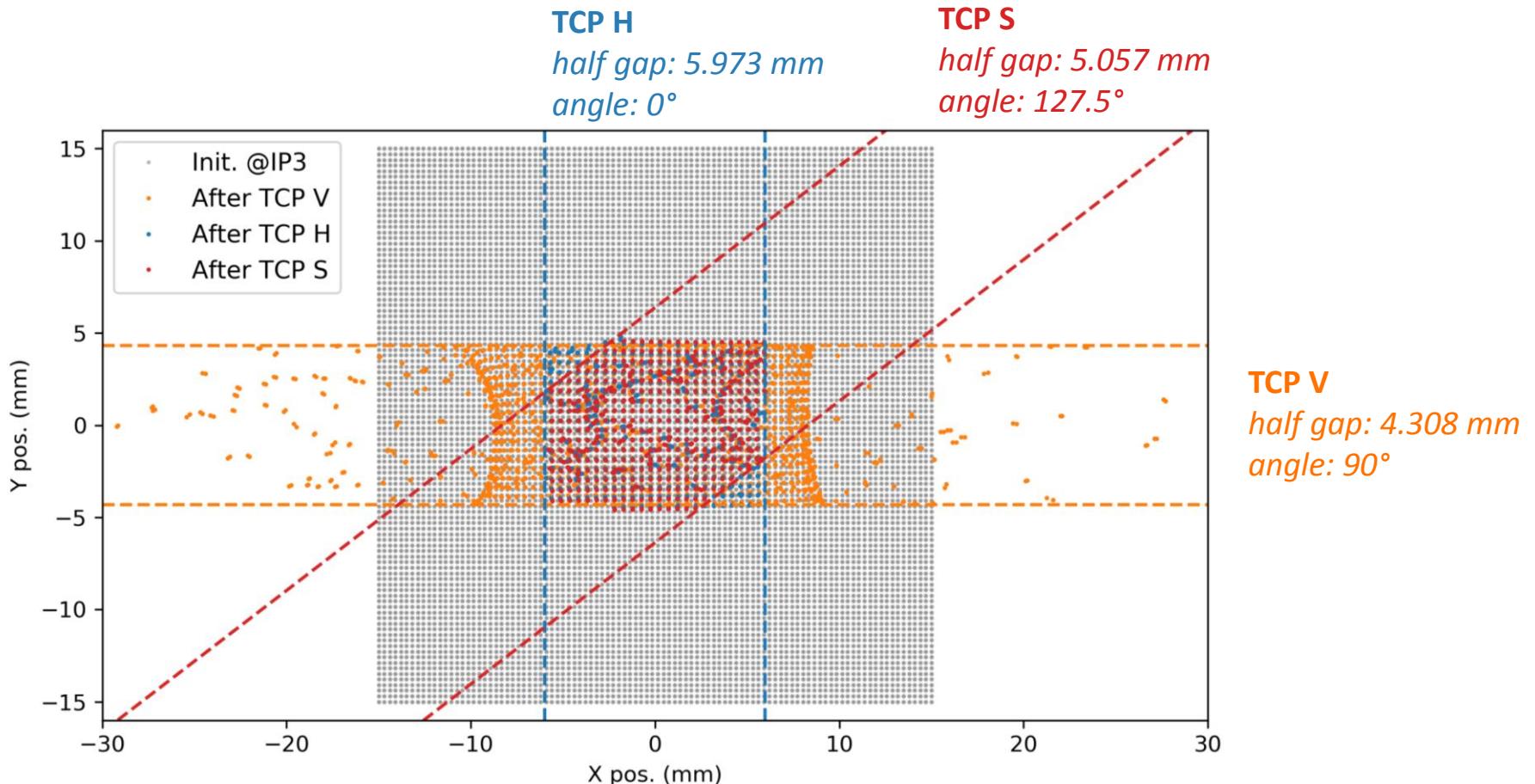


[Loic's 3D aperture / loss visualizer](#) 😊

Simulation set-up

Aperture validation

- Track square beam through machine to **verify gaps and skew collimator angles**

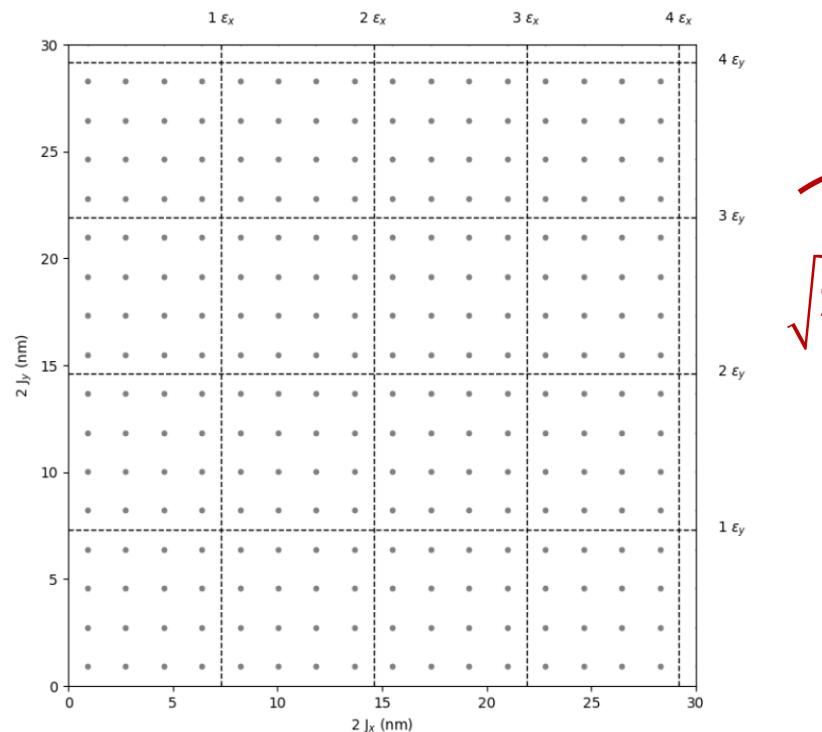


Gaps correctly applied and tilt angle correctly propagated from MAD-X to SixTrack

Simulation set-up

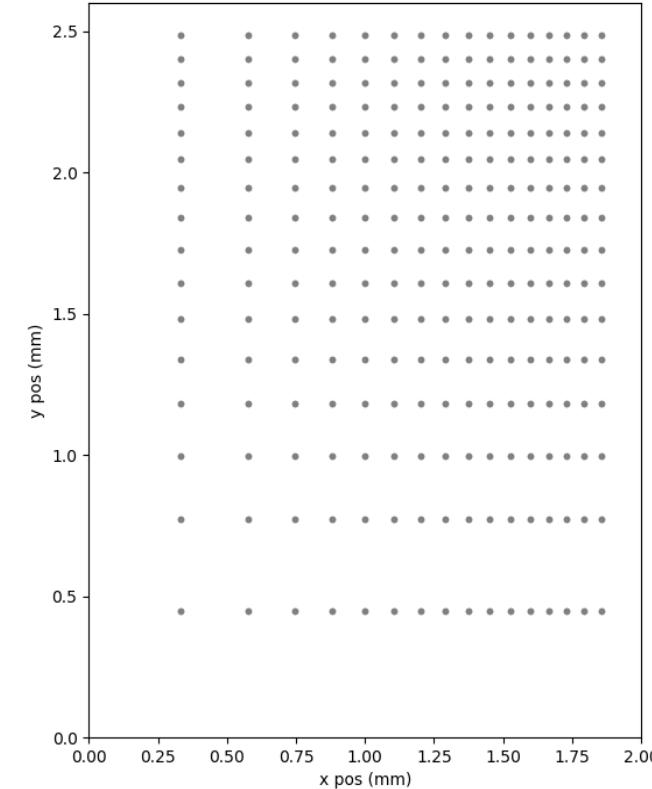
Particle initialization

- Initial conditions: uniform polar or rectangular grid in transverse action space (J_x, J_y)
- Assign weights in post-processing: Gaussian distributions require tracking many “irrelevant” core particles
- Example: steps of 0.25 in normalised amplitude
with $\varepsilon_{norm} = 3.5 \mu\text{m rad at } 450 \text{ GeV}$, i.e. $\varepsilon_{norm} / \beta\gamma \approx 7.3 \text{ nm}$, and $2J_{x,y} = \varepsilon_{geo}$



Action space

6



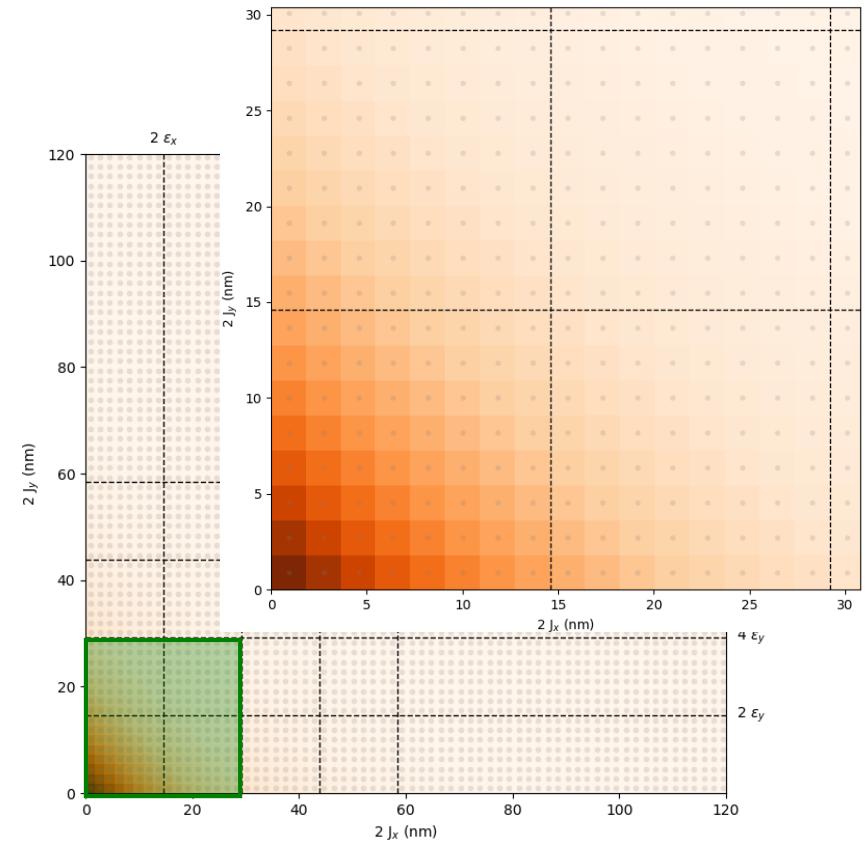
Coord. space

03.07.2020

Post-processing and analysis

Loss analysis: Gaussian weights

- **(J_x, J_y) space divided into square cells**, s.t. every cell contains 1 particle (*pair*)
- $c_{i,j}(t)$: flag particle in cell (i, j) **dead** ($= 1$) **or alive** ($= 0$) after t turns
- Calculate weights $w_{i,j}$ in action space for beam distribution we would like to mimic
- Illustrated by color



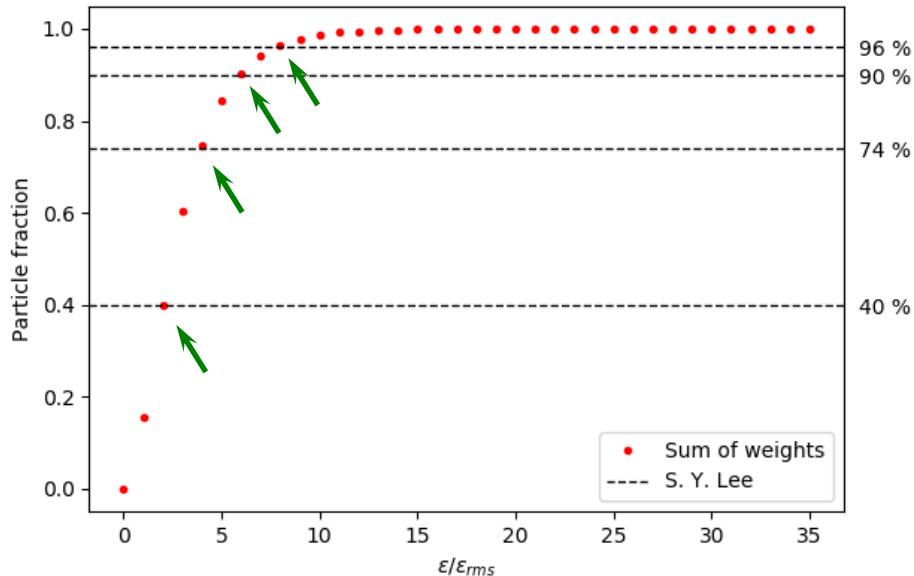
Example: round Gaussian beam

$$\begin{aligned} N_{\text{lost}}(t) &= \frac{N_{\text{total}}}{\varepsilon_{rms}} \sum_{i,j} c_{i,j}(t) \int_{J_{x,i}}^{J_{x,i+1}} \int_{J_{y,j}}^{J_{y,j+1}} e^{-J_x/\varepsilon_{rms}} e^{-J_y/\varepsilon_{rms}} dJ_x dJ_y = \\ &= N_{\text{total}} \sum_{i,j} c_{i,j}(t) [e^{-J_{x,i}/\varepsilon_{rms}} - e^{-J_{x,i+1}/\varepsilon_{rms}}] [e^{-J_{y,j}/\varepsilon_{rms}} - e^{-J_{y,j+1}/\varepsilon_{rms}}] \end{aligned}$$

$= w_{i,j}$

Post-processing and analysis

Gaussian weights: validation

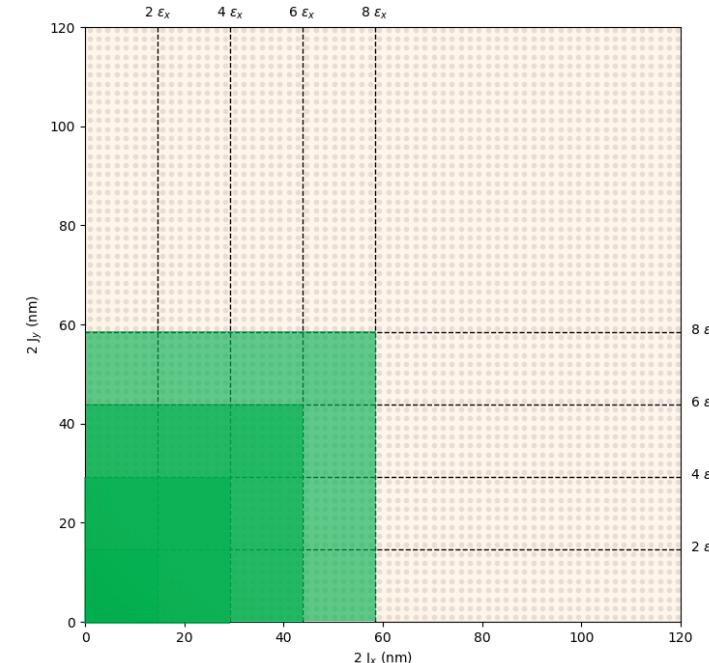


S. Y. Lee, Accelerator Physics, 4th edition, pg. 59

Table 2.1: Percentage of particles in the confined phase-space volume

ϵ/ϵ_{rms}	2	4	6	8
Percentage in 1D [%]	63	86	95	98
Percentage in 2D [%]	40	74	90	96

- Compare calculated Gaussian weights to literature
- All in good agreement



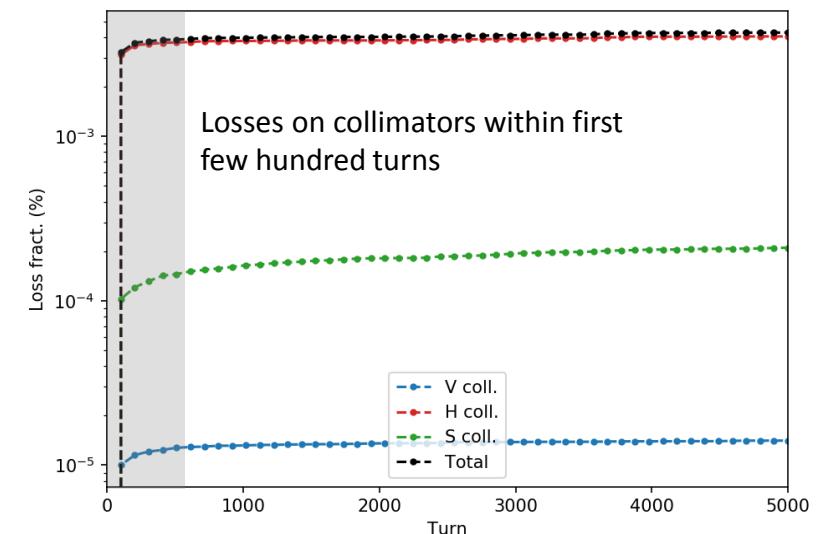
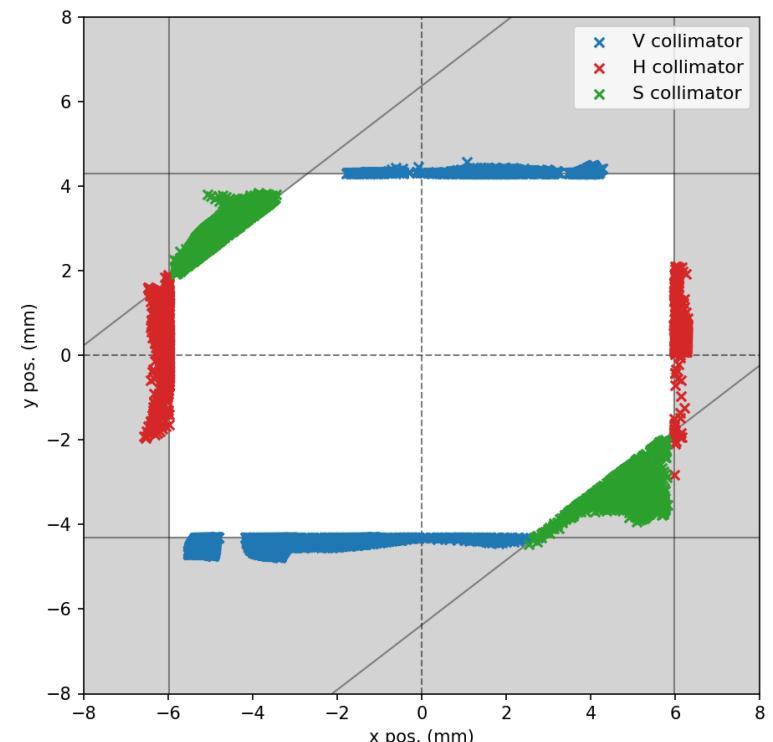
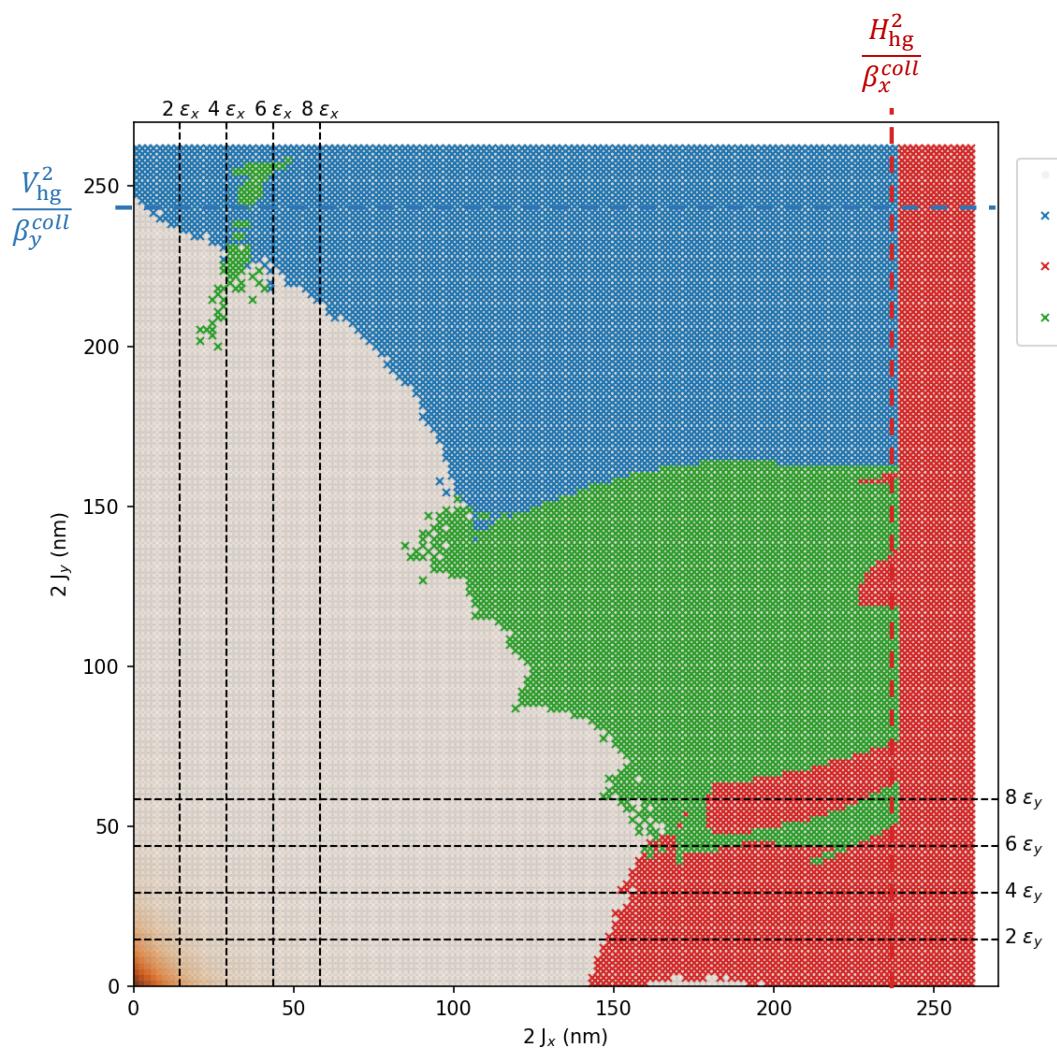
Some test results

Setup for short runs (5000 turns)

- **Goal:** verify that setup works
- **Machine parameters**
 - LHC @ 450 GeV
 - $Q_x = 62.28 / Q_y = 60.31, Q'_{x,y} = 15$
 - $\epsilon_{\text{norm}} = 3.5 \mu\text{m rad}$
 - Three cases: $I_{\text{oct}} = -40 \text{ A}$
- **Initialize particles on rectangular grid**
 - Up to $2J_{x,y} = 36 \epsilon_{\text{rms}}$, i.e. 6σ (*beyond collimator limits*)
("missing beam": $1 - [1 - e^{-36/2}]^2 = 3.1 \times 10^{-8}$)
 - Amplitude steps of 0.25
 - Total of $2 \times (36 / 0.25)^2 = 41'472$ particles (*incl. twin*)
- **5000 turns**
- Just one error seed
- Running on HTCondor

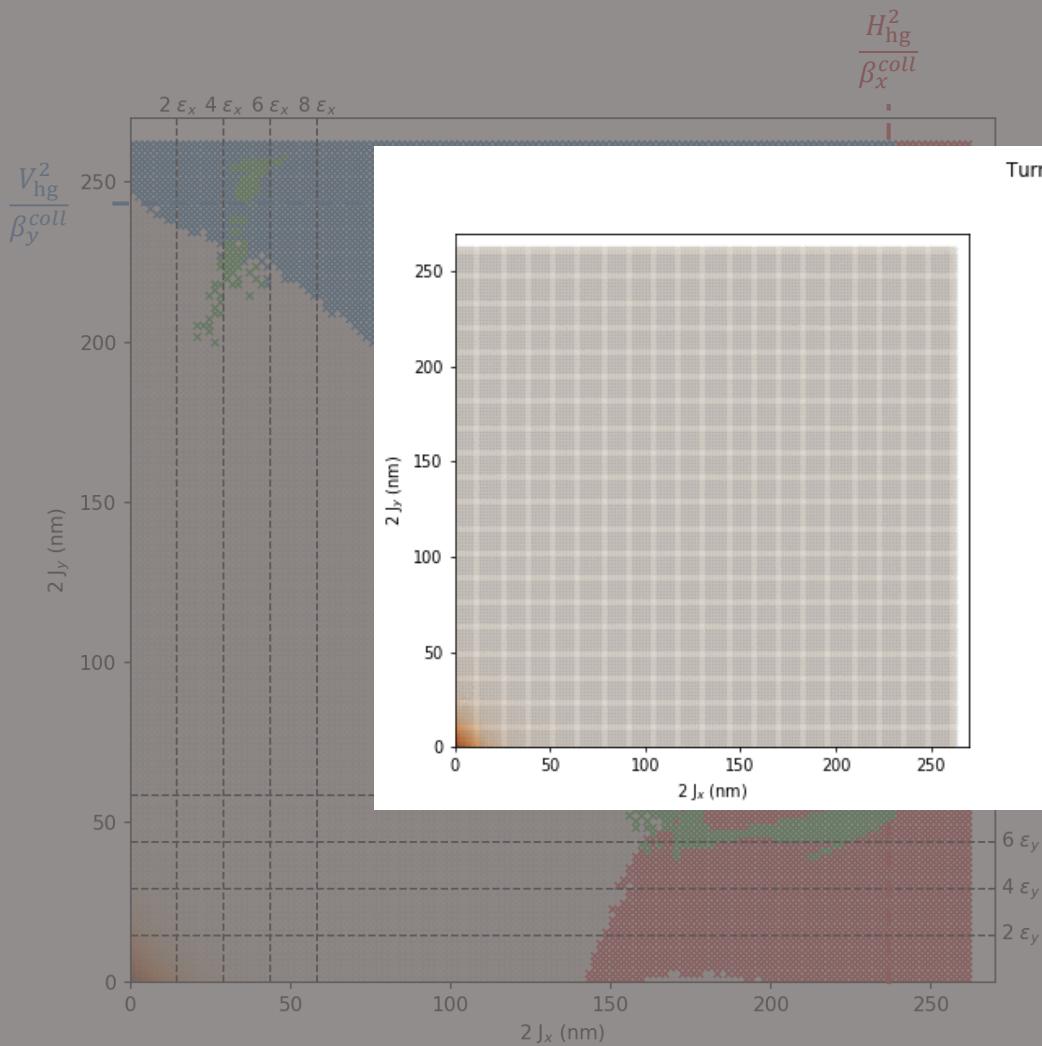
Some test results

$$I_{oct} = -40 A$$



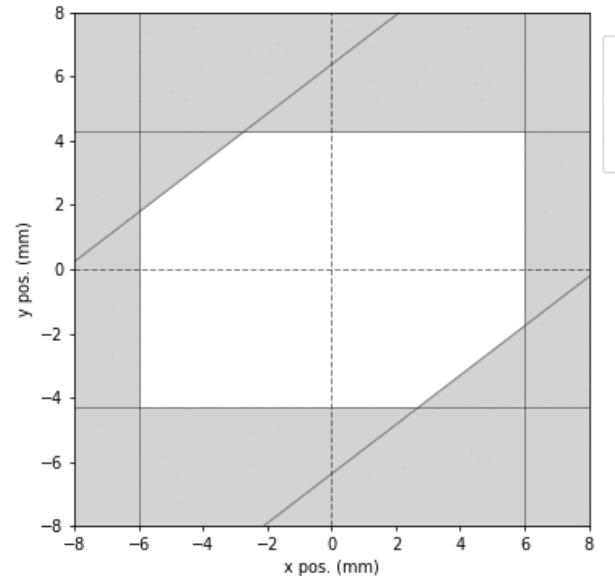
Some test results

$I_{oct} = -40 A$

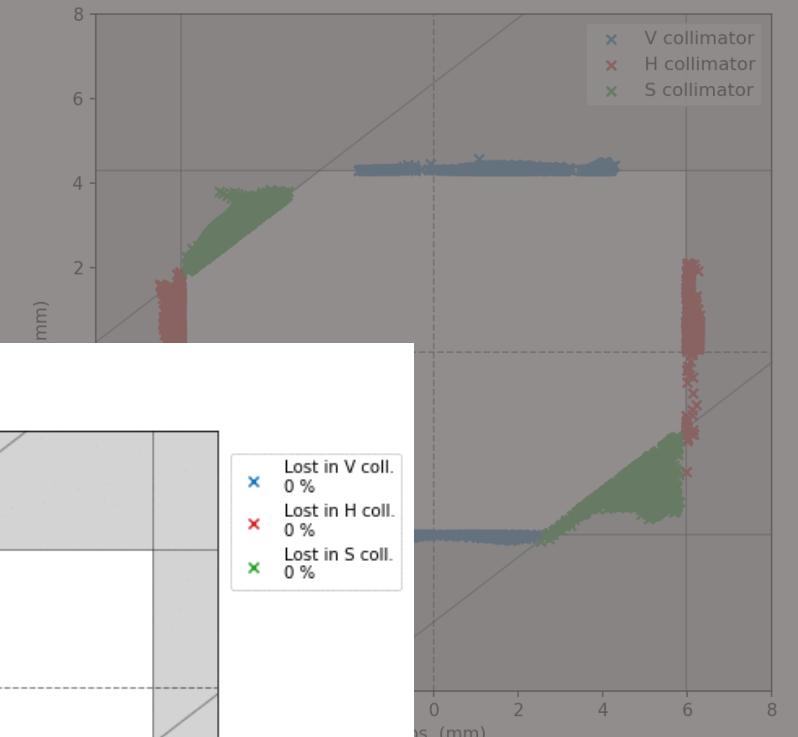


$$\frac{H_{hg}^2}{\beta_x^{coll}}$$

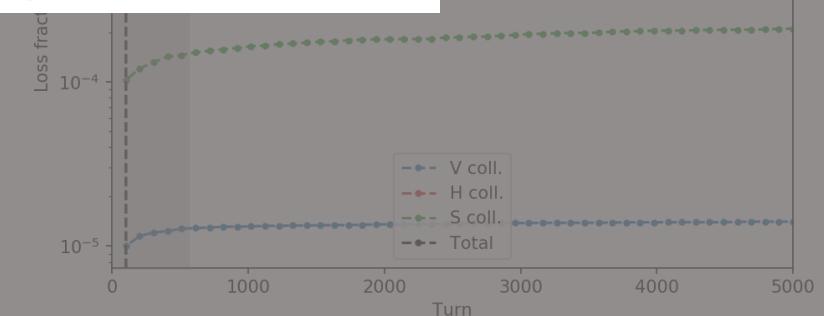
Turn 00000



- Lost in V coll. 0 %
- Lost in H coll. 0 %
- Lost in S coll. 0 %



ors within first



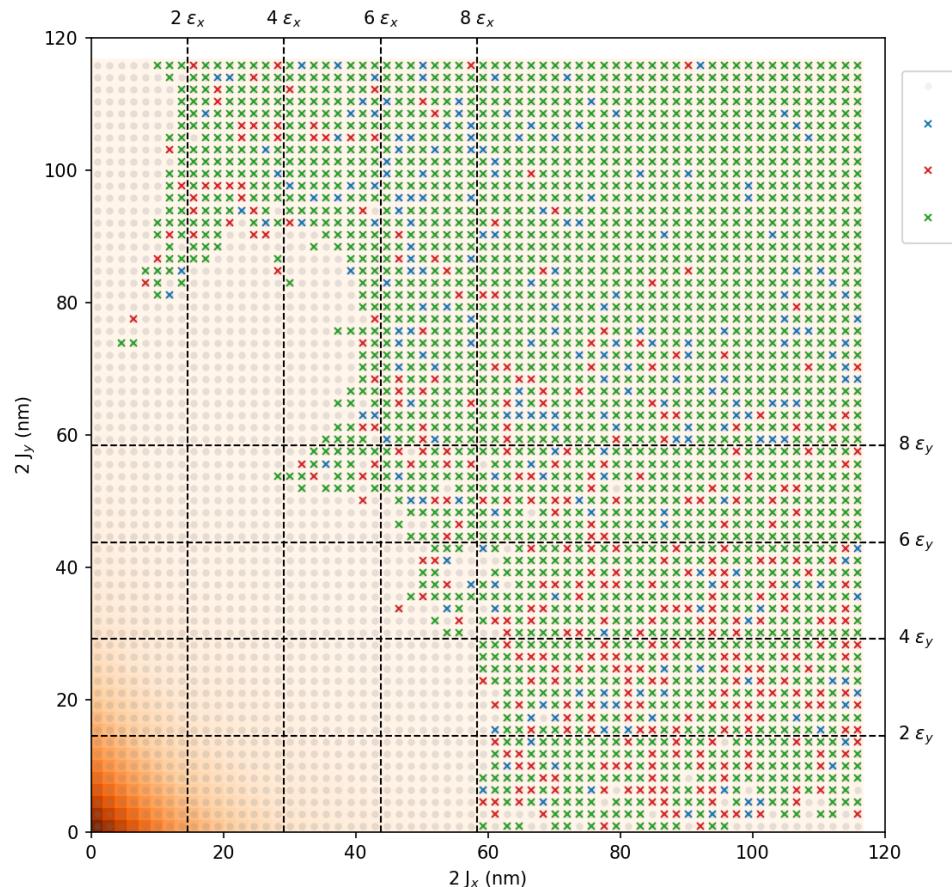
Some test results

Setup for long runs (10^6 turns)

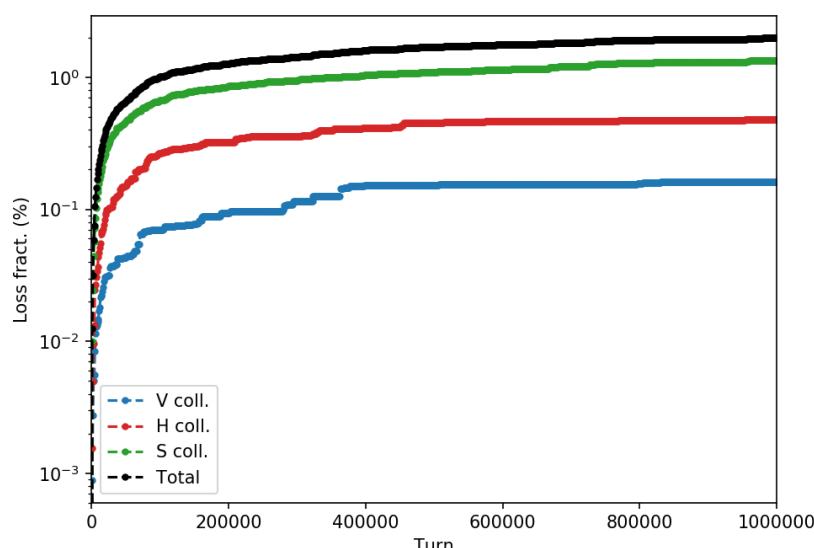
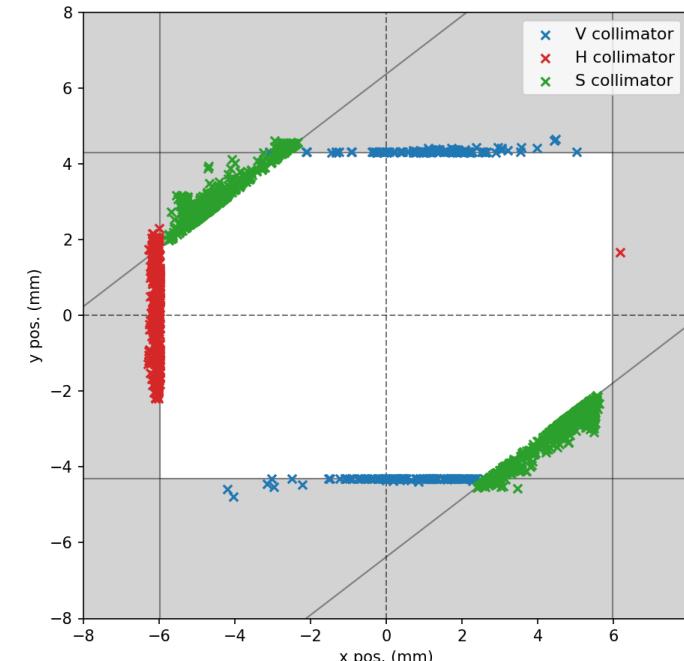
- **Goal:** verify “slow” losses driven by non-linearities
- **Same machine parameters**
 - LHC @ 450 GeV
 - $Q_x = 62.28 / Q_y = 60.31, Q'_{x,y} = 15$
 - $\epsilon_{\text{norm}} = 3.5 \mu\text{m rad}$
 - Three cases: $I_{\text{oct}} = \{-100 \text{ A}, -40 \text{ A}, -20 \text{ A}\}$
- **Initialize particles on rectangular grid**
 - Up to $2J_{x,y} = 16 \epsilon_{\text{rms}}$, i.e. “missing beam”: $1 - [1 - e^{-16/2}]^2 = 6.7 \times 10^{-4}$
 - Amplitude steps of 0.25
 - Total of $2 \times (16 / 0.25)^2 = 8'192$ particles (incl. twin)
- **10^6 turns**
- Just one error seed
- Running on HTCondor

Some more test results

Example 1: $I_{oct} = -100 A$...

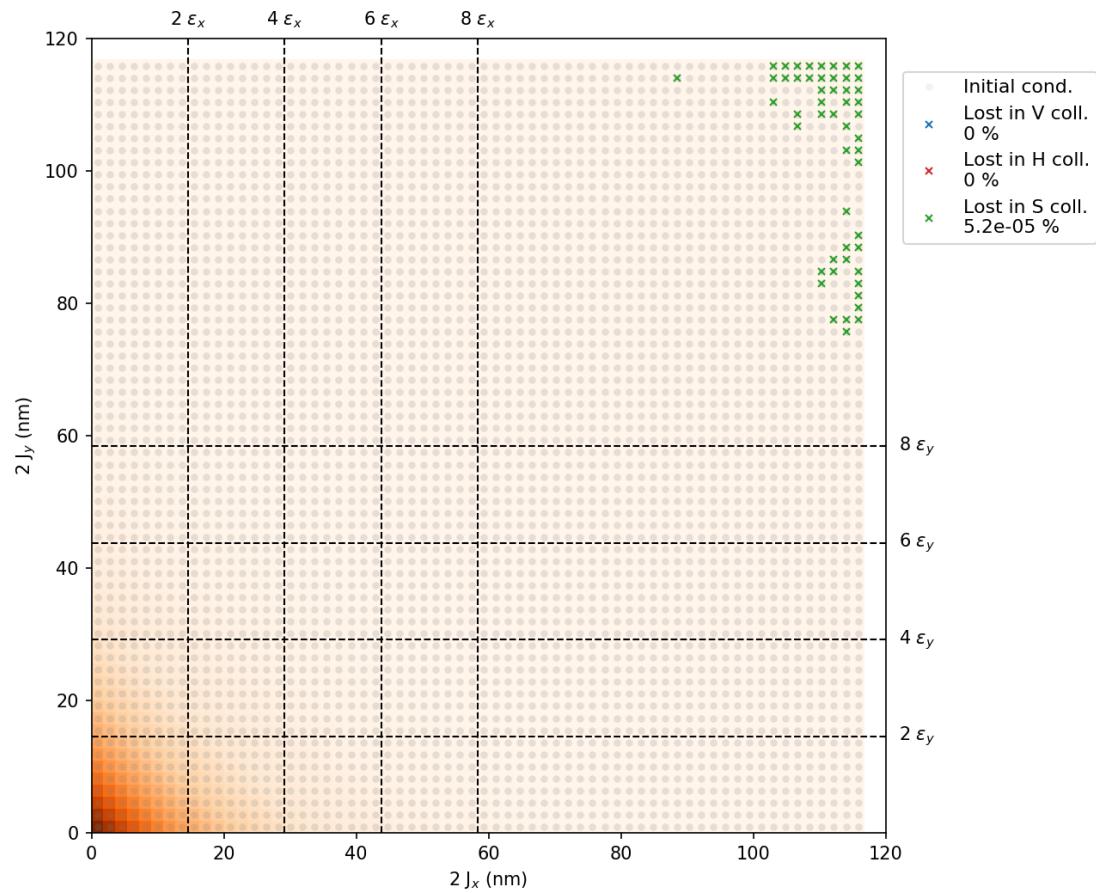


- $I_{oct} = -100 A$
- Looks interesting, but is **unrealistic scenario**

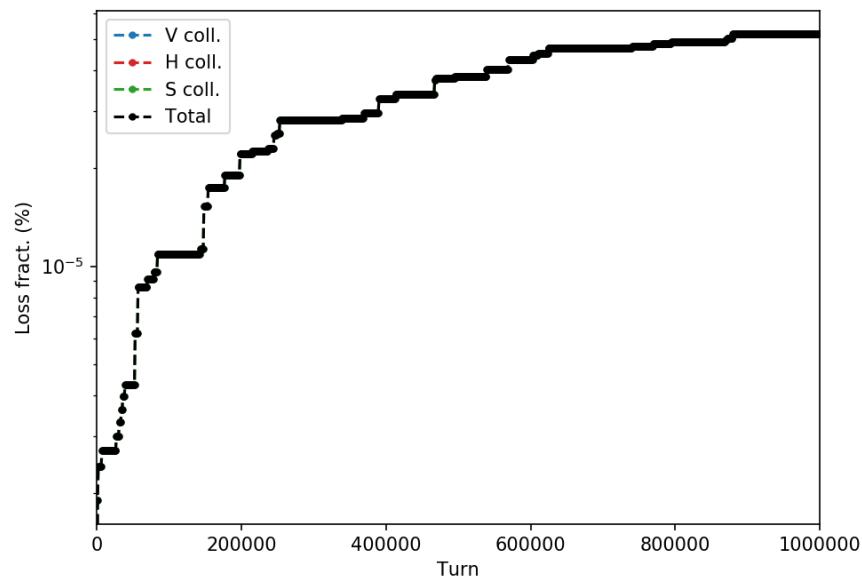
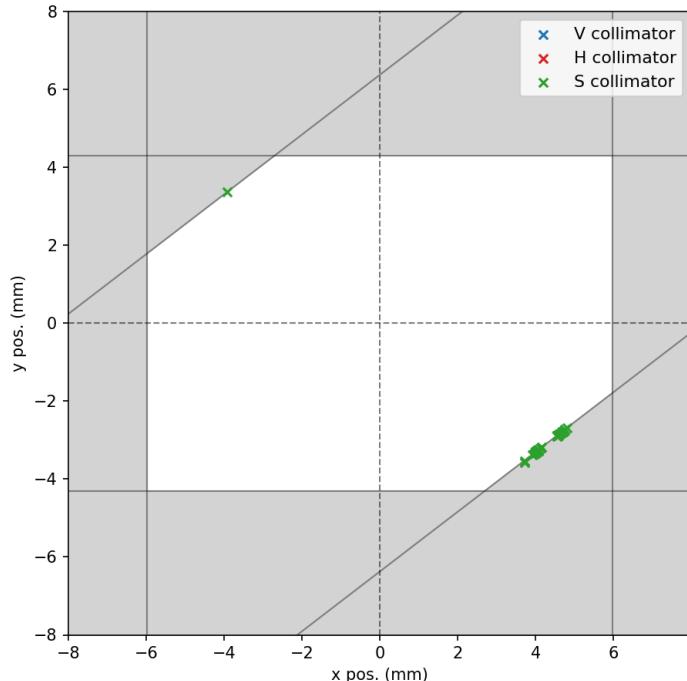


Some more test results

Example 2: $I_{oct} = -40 A$...

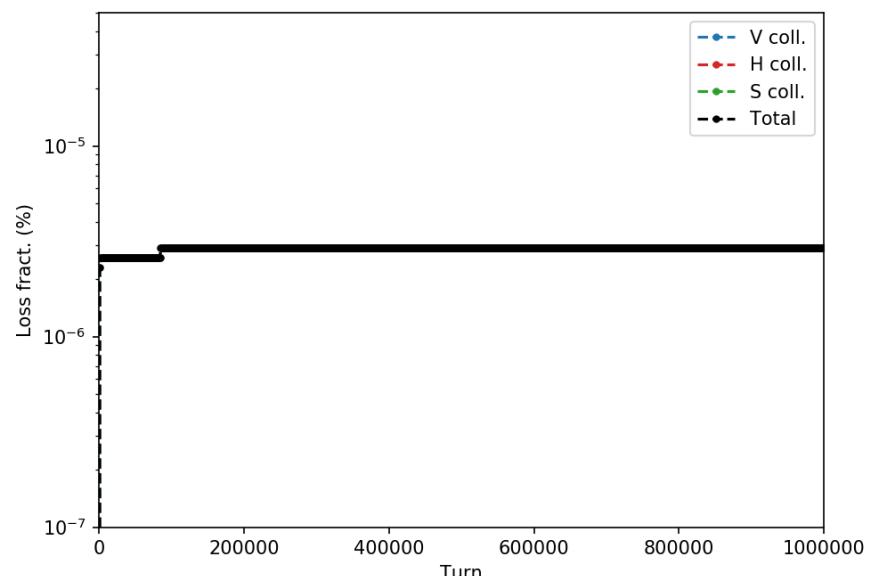
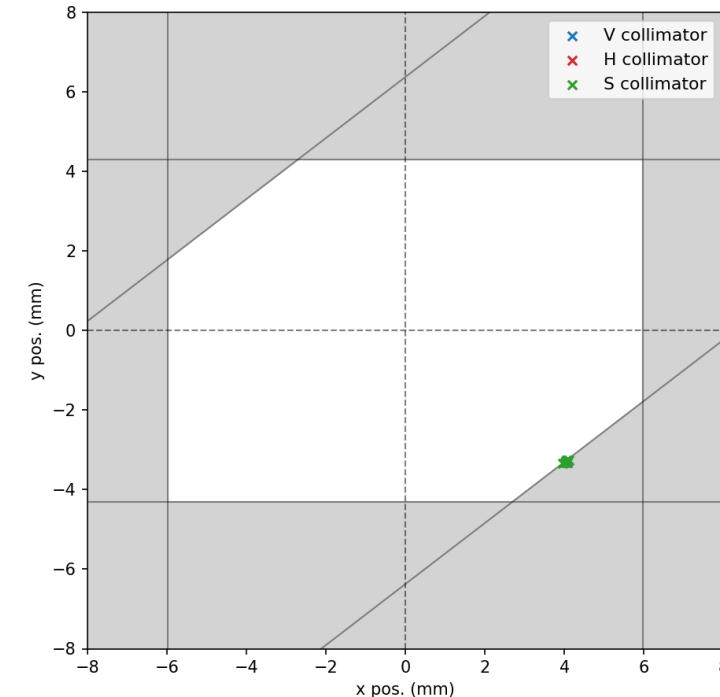
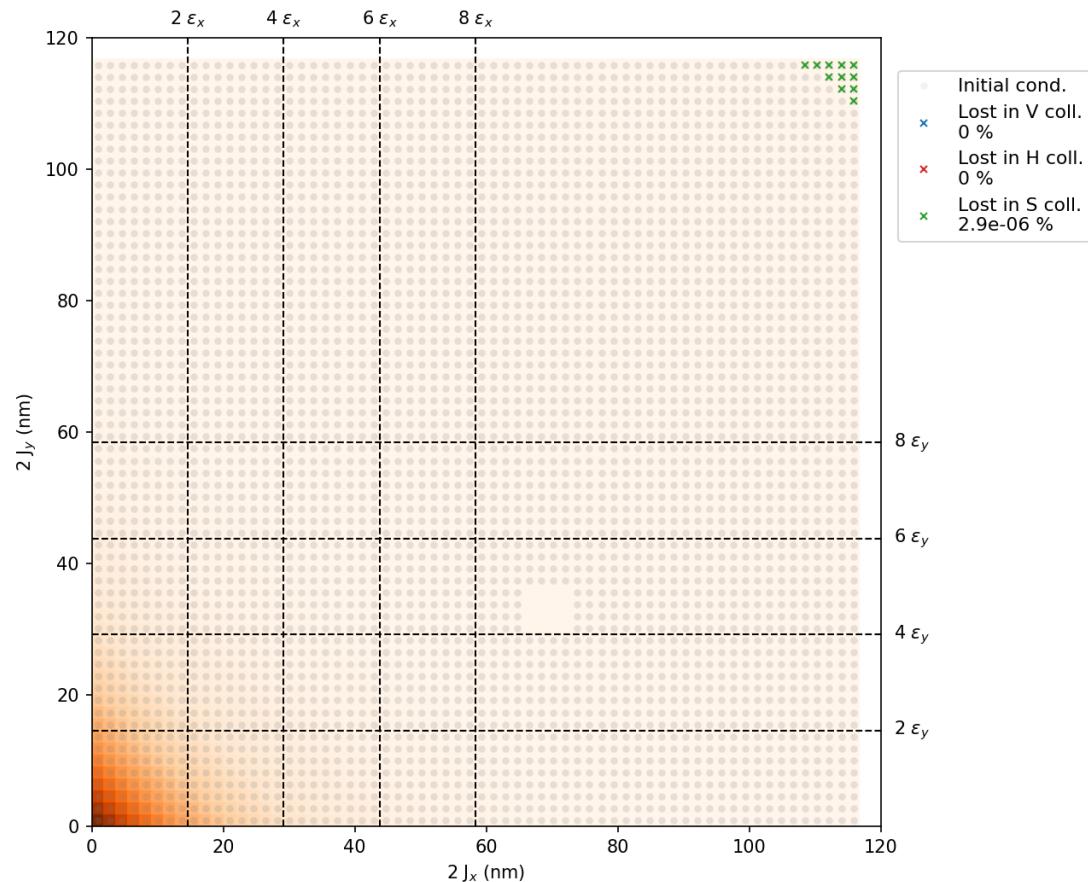


- $I_{oct} = -40 A$
- **Do observe losses until up to 10^6 turns**
- Encouraging, but $I_{oct} = -40 A$ for $3.5 \mu\text{m}$ rad beam at injection energy is **still very high**



Some more test results

Example 3: $I_{oct} = -20 A$...



- Losses are reduced: that's OK
- Loss occurs only within $\sim 10^5$ turns
- **"Sensitivity issue"?**

Thoughts and future plans

- **May be challenging to reach loss sensitivity** for (*close-to*) operational machine parameters
 - More turns required?
 - Effects missing: noise, other non-linearities (*most notably electron cloud*)
- **Electron cloud** is an important player in LHC at injection energy
 - Observed experimentally in MDs
 - *Strong non-linearity* and as such responsible for halo population / losses, and potentially H/V asymmetry
 - Not available in SixTrack, but work in progress by K. Paraschou (PhD, CERN) for [SixTrackLib](#)
- **Compare simulations to MD results** at injection w/o electron cloud
- **Gaussian weights**
 - Actual *particle distribution unknown*; there is *halo population*
 - Different weights can be applied in post-processing *IF physics model unchanged* (e.g. not possible when adding e-cloud effects)

Contents

Part I: SixTrack simulations

- Simulation set-up
- Post-processing & analysis
- Some test results
- Thoughts and future plans

Part II: Automatic parameter exploration

- Objective
- SMT package and EGO algorithm
- Interface with PySixDesk
- Test in 2D parameter space
- Other ideas, issues, what's next?

Automatic model

Objective

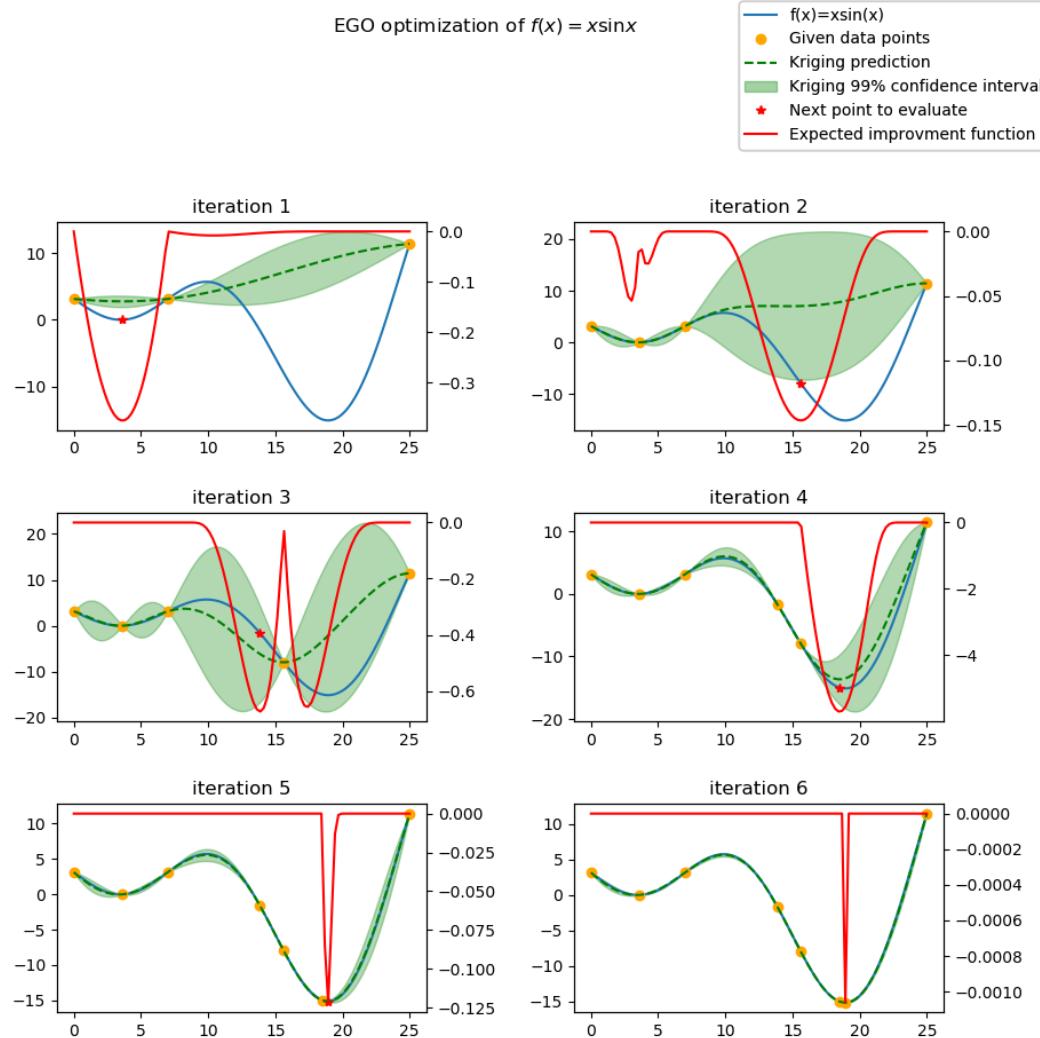
- **Running 10^6 + turn Sixtrack simulations is computationally expensive**
- Same or at least similar configurations often run more than once
- **Keypoints**
 - Build framework to have **Sixtrack run and explore automatically** when resources available (BOINC)
 - Collect results in global database to “preserve knowledge”
 - **Develop interpolation / ML model** from simulation data (*continuous process*)
- Instead of random exploration: **better ways to sample parameter space**
 - One suggestion shown here
 - Ideas are most welcome!

- Among many ... open-source Python package for optimization and modelling of black-box functions
- Mostly Bayesian optimization (BO), but not exclusively
- BO advantages
 - Estimate of uncertainty on predictions available
 - Acquisition function: **next sampling points**, e.g. (1) places of highest prediction uncertainty (*exploration*) or (2) Expected Improvement for global optimum search (*rather exploitation*), + others
 - Parallel implementations exist
 - => can submit q next sampling points, i.e. run SixTrack jobs in parallel on cluster
- BO disadvantages
 - Performance (*strongly*) affected by increasing number of samples / parameters
 - Typically, BO said to be efficient up to 20 dim. parameter space (*we may reach close to this limit*) although BO algorithms for high dim. problems exist
- Explore if it can work for our purpose

Automatic parameter exploration

Efficient Global Optimization (EGO)

- Bayesian optimization method
- Builds Kriging model
- Find global optimum (*within bounds*) of expensive black-box functions
- **How it works**
 - Start with some initial datapoints
 - Evaluate black-box function
 - Evaluate acquisition function
Here Expected Improvement (EI):
For “every possible point”, by how much can function value be expected to improve compared to current optimum
 - Sample at parameter value(s) where acq. fct. max.
- Here done in *series*, i.e. one sample at a time



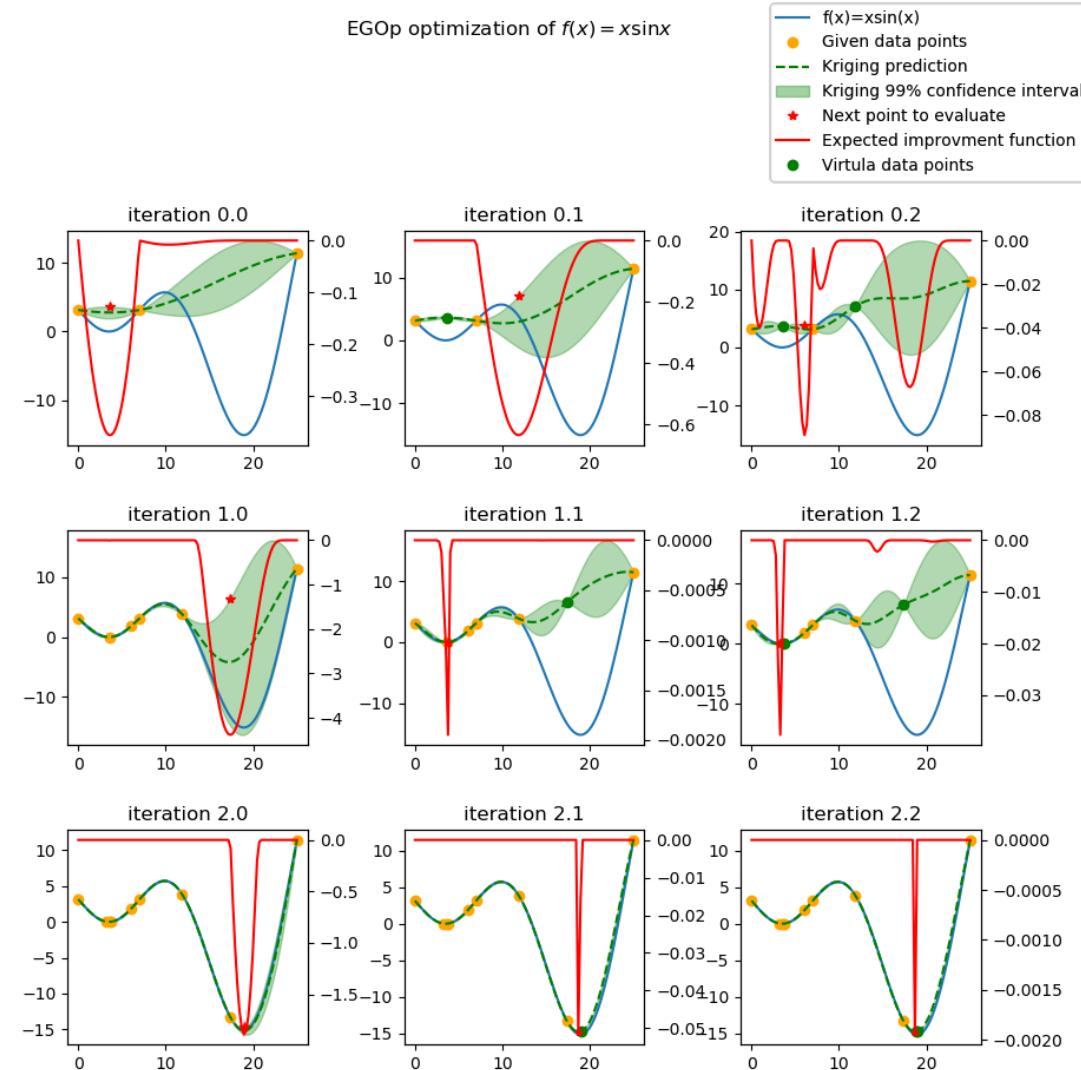
<https://smt.readthedocs.io/en/latest/>

Automatic parameter exploration

Efficient Global Optimization (EGO)

Parallel algorithm (qEI)

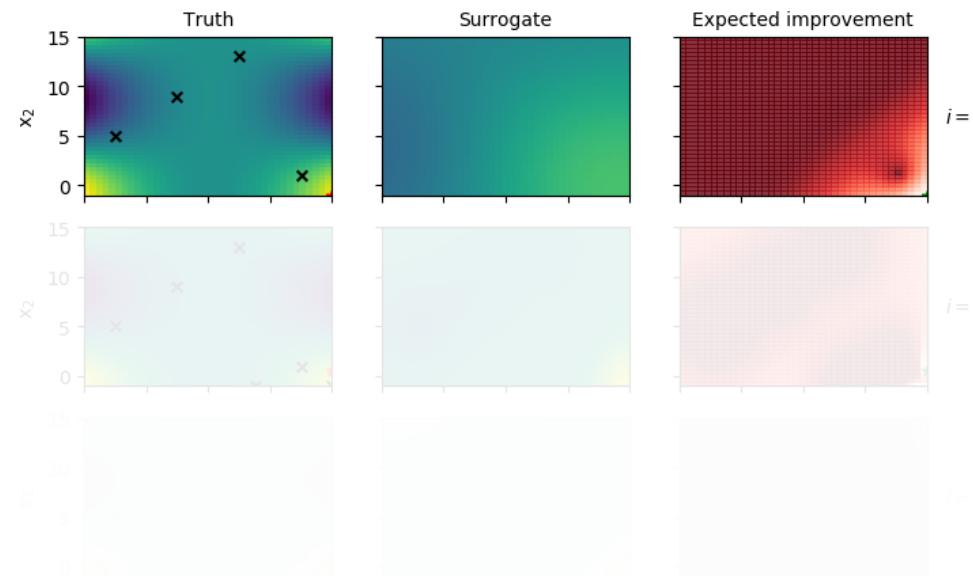
- At every iteration, instead of picking just one next sample, we **choose $q > 1$ next samples**
- How to find q sampling points
 - Get 1st sampling point according to acq. fct. (EI)
 - **Define “virtual” value based on current knowledge**
 - Compute acq. fct. (EI) based on model incl. virtual value to choose next sampling point
 - Iterate
- Different strategies to define virtual values
- Much more efficient than serial implementation *in terms of time* – not necessarily *in terms of number of sampling points*



<https://smt.readthedocs.io/en/latest/>

Automatic parameter exploration

Efficient Global Optimization (EGO): example in 2D



Automatic parameter exploration

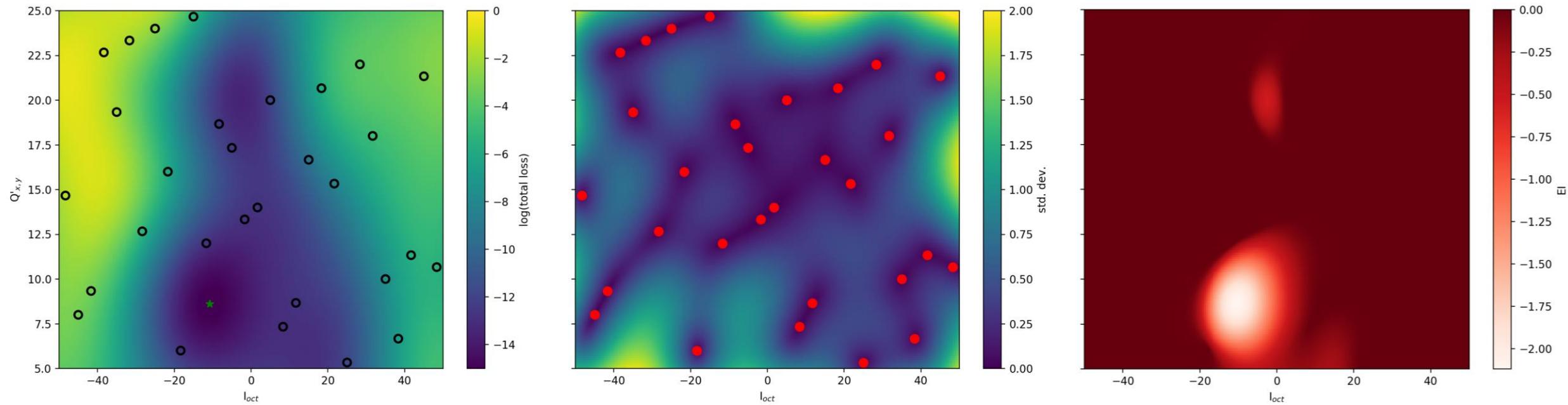
SMT & PySixdesk

- **Combined SMT with PySixdesk**
 1. Submit MAD-X jobs for parameter values suggested by acquisition function (*e.g. qEI*)
 2. Once finished (*few minutes*), collect MAD-X jobs and submit Sixtrack jobs
 3. Once Sixtrack is done (*can be hours – days*): collect jobs and analyse for total loss
 4. Update model with latest results and iterate
- **Semi-automatic, not yet entirely fail-safe:** needs some manual supervision – *work in progress*
- Tests running on HTCondor, but adaptable for BOINC once new executables available
- **Works conceptually**, but not much time yet to evaluate results
- **First test case**
 - Two parameters: $I_{oct}, Q'_{x,y}$
 - Objective: optimize $f(x) = \log(\text{"total loss after 50'000 turns"})$
 - Using parallel EGO (*qEI*) with $q = 20$

Automatic parameter exploration

Test in 2D parameter space

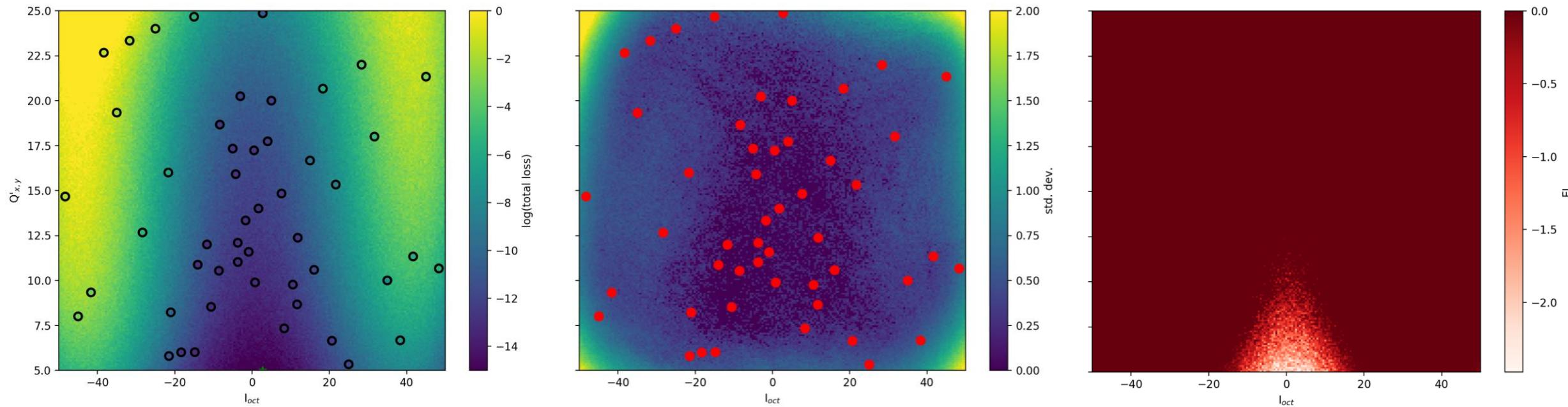
- **Iteration 0: 30 initial samples**
background: surrogate model, circles: data points (Sixtrack), star: current optimum



Automatic parameter exploration

Test in 2D parameter space

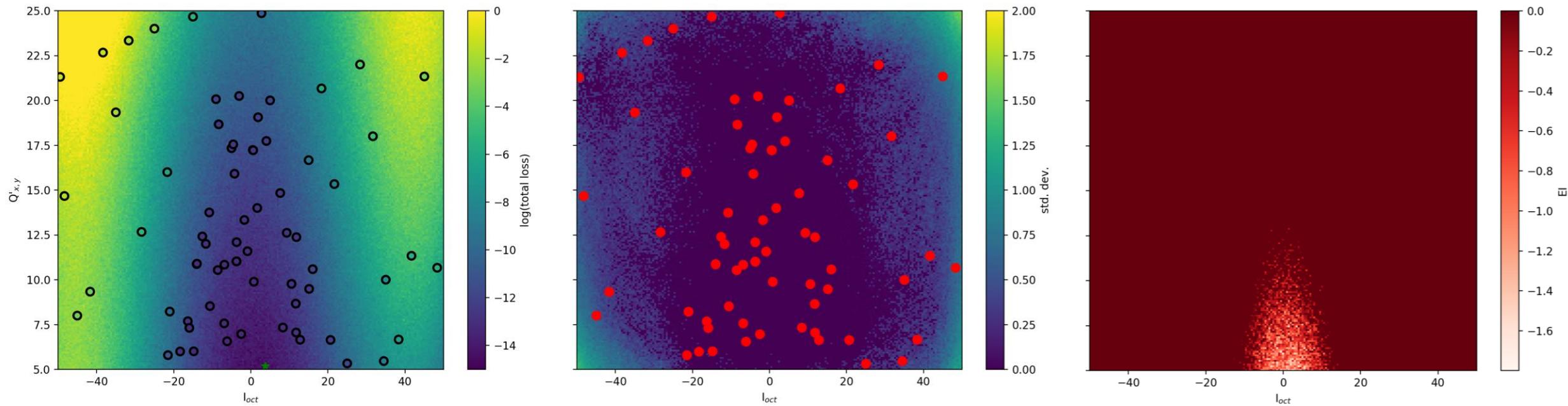
- **Iteration 1:** total 50 samples
background: surrogate model, circles: data points (Sixtrack), star: current optimum



Automatic parameter exploration

Test in 2D parameter space

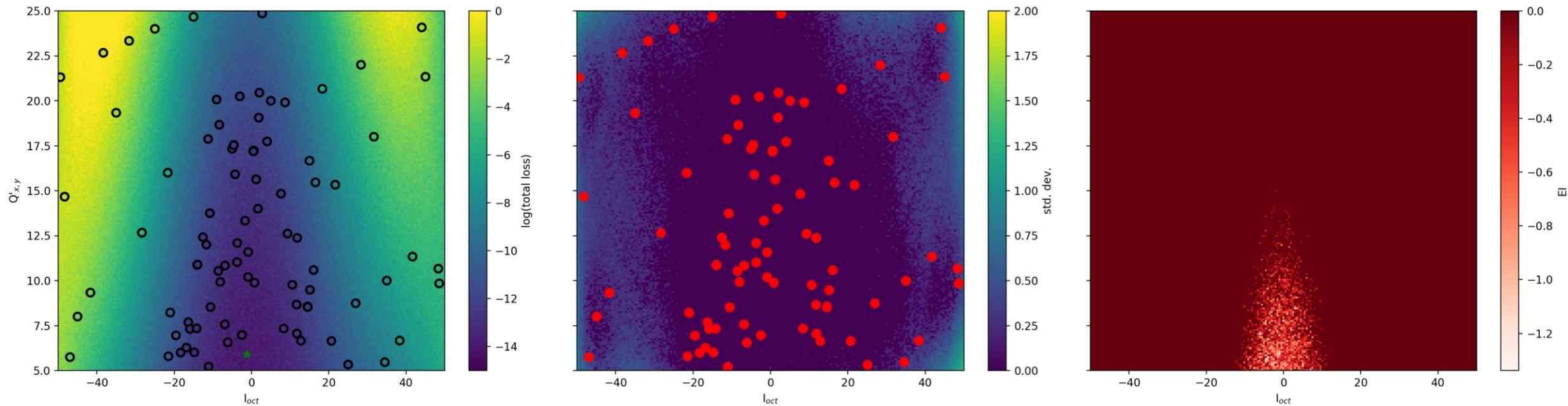
- **Iteration 2: total 70 samples**
background: surrogate model, circles: data points (Sixtrack), star: current optimum



Automatic parameter exploration

Test in 2D parameter space

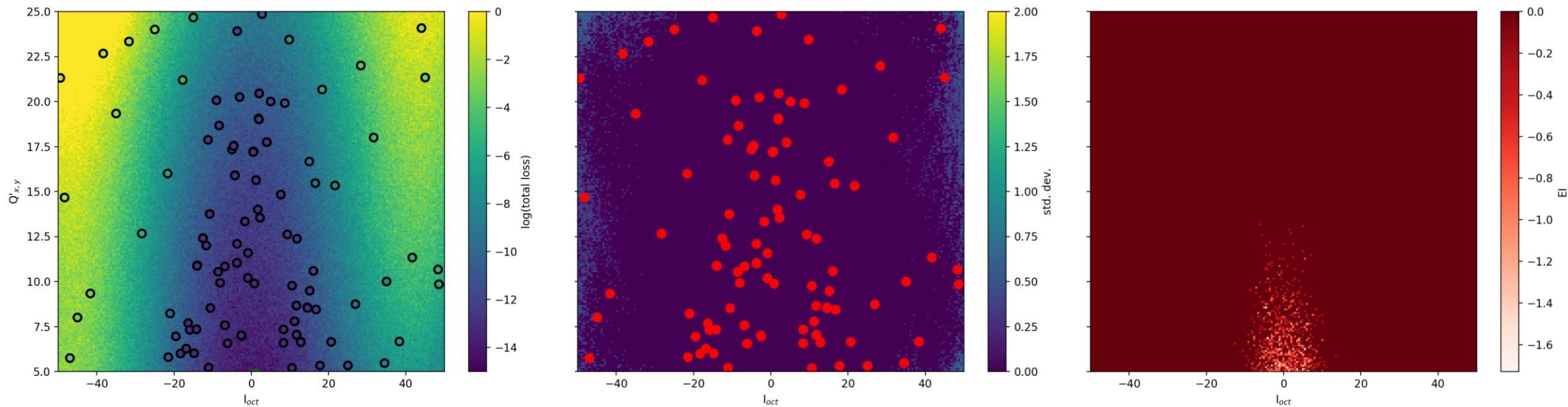
- **Iteration 3: total 90 samples**
background: surrogate model, circles: data points (Sixtrack), star: current optimum



Automatic parameter exploration

Test in 2D parameter space

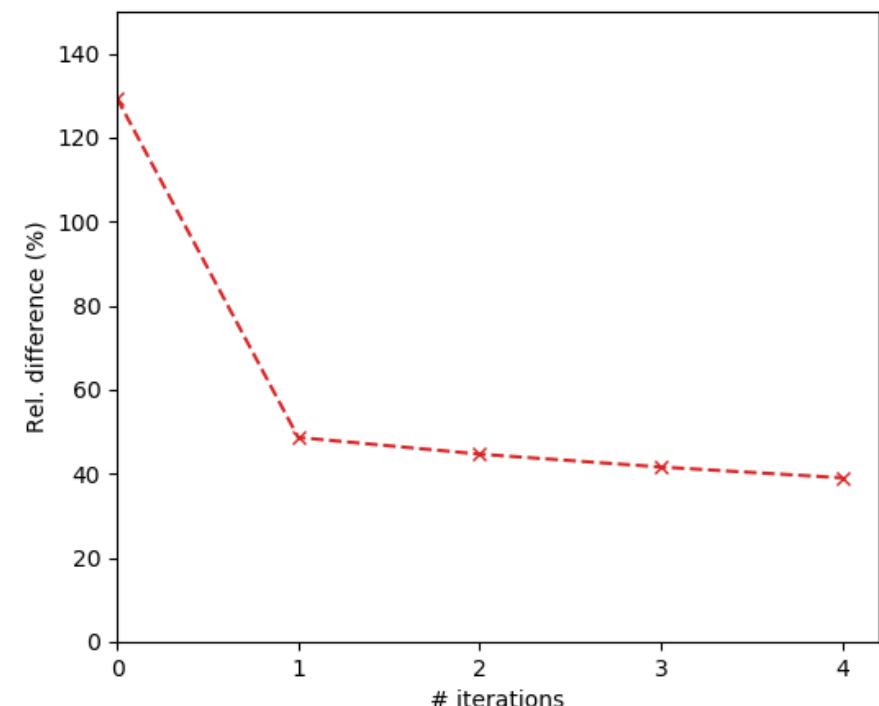
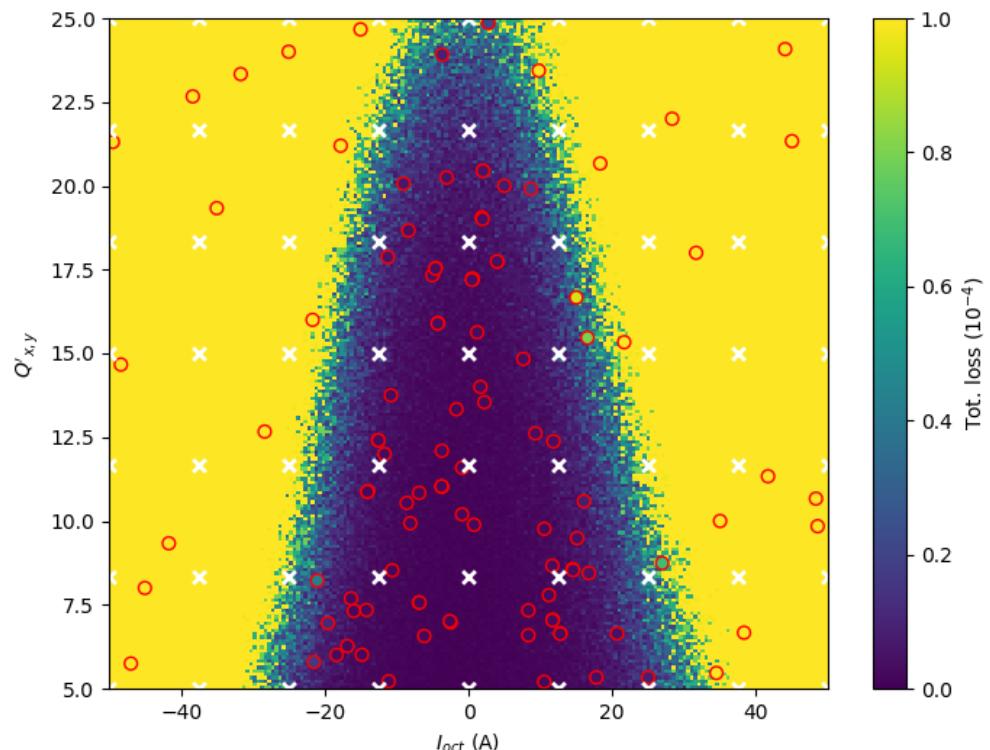
- **Iteration 4: total 110 samples**
background: surrogate model, circles: data points (Sixtrack), star: current optimum



Automatic parameter exploration

Test validation

- How accurate is surrogate model at this point with given samples (*red circles*)?
- Evaluate loss on grid (*white crosses*): compare model vs. Sixtrack simulations
- Relative difference of losses (not log!) drops with number of iterations but remains high (~40%)
- Possible reasons: localized validation; loss value 'quantized', heavily dependent on individual particles; EI finds optimum (different acquisition function?); more iterations; modeling method not suitable / badly tuned



Automatic parameter exploration

Ideas, issues, what's next?

Ideas / questions

- Better methods for **efficient sampling**?
- **Acquisition function:** go more **towards exploration**, e.g. using prediction uncertainty, and compare results
Better suited for parameter exploration if not just interested in global optimum(?)
- Use **BO just for efficient parameter space exploration** *assuming Gaussian weights and total loss.*
Then: apply any technique to build model, even on "lower-level" data
- **Lower-level data:** we know at **what turn particles are lost and where in (J_x, J_y) space** with relatively high granularity and independent of the beam distribution
- Can we **impose knowledge** about the function, such as $0 \leq f(x) \leq 1$ for fractional loss?
- **Improve job management:** robustness, autonomy, submission from local machine
- What data to save? (*e.g. desirable to continue a simulation*)

Potential issues

- How many parameters and samples are feasible? => **Scaling tests**
- Suited for all parameter dependencies? *E.g. resonance lines in tune space: $f(x)$ not smooth everywhere*