>>> PTC - PyORBIT SIS18 Benchmark
>>> An Iterative Journey

Name: Haroon Rafique BE-ABP-HSI[†]

Date: November 27, 2018

[†]haroon.rafique@cern.ch

>>> Contents

Motivation

>>> What's the SIS18 Benchmark?

* Created by Guiliano Franchetti to confirm space charge induced trapping of particles in a bunch during long term storage.

>>> Why should I care?

* We rely on PTC-PyORBIT for space charge simulations, so it's good to check that it is reliable.

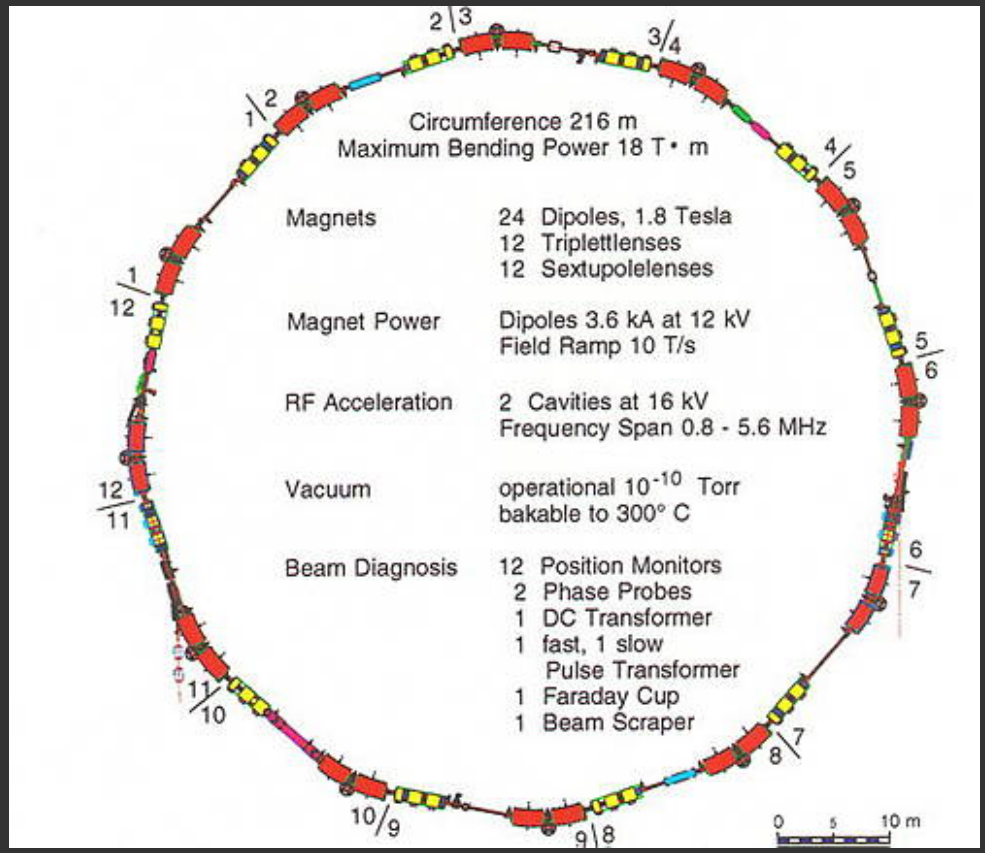
>>> Motivation

* To verify that PTC-PyORBIT behaves as expected when considering physical effects that should be present for space charge studies.
* To understand the necessary ingredients for a space charge simulation with PTC-PyORBIT.
* To provide those that wish to use PTC-PyORBIT with nicely commented code examples.

SIS18

>>> The Accelerator

This benchmark uses the SIS18 (German ShwerIonenSynchrotron) lattice. In the benchmark lattice only one sextupole is present.



The reference lattice can be found here, and the usable MAD-X lattice here.

Parameters

>>> Beam parameters (steps 1 - 6)

Parameter	Symbol	Value	Unit
Sextupole Strength (when used)	K_2	0.2	m^{-2}
Maximum Tuneshift	ΔQ_x	0.1	-
Intensity	N_p	$2.95 \cdot 10^9$	$particles$
Horizontal Emittance	ϵ_x	$4.91 \cdot 10^{-7}$	$m\ rad$
Vertical Emittance	ϵ_y	$3.635 \cdot 10^{-7}$	$m\ rad$
Kinetik Energy	E_k	11.4	$\frac{MeV}{u}$
Gamma Transition	γ_t	5	-
Momentum Spread	$\frac{dp}{p}$	$2.5 \cdot 10^{-4}$	-
Lorentz Gamma	γ	1.012149995	-
Lorentz Beta	β	0.15448	-
Bunch Length	τ	3472.7	ns
Synchrotron Tune	Q_{synch}	$\frac{1}{15000}$	-

Benchmark Information

Steps 1 - 6 Parameters

>>> Beam parameters (steps 7 - 9)

Parameter	Symbol	Value	Unit
Sextupole Strength (when used)	K_2	0.2	m^{-2}
Maximum Tuneshift	ΔQ_x	0.1	-
Intensity	N_p	$1.95 \cdot 10^9$	$particles$
Horizontal Emittance	ϵ_x	$4.91 \cdot 10^{-7}$	$m\ rad$
Vertical Emittance	ϵ_y	$3.635 \cdot 10^{-7}$	$m\ rad$
Kinetik Energy	E_k	11.4	$\frac{MeV}{u}$
Gamma Transition	γ_t	5	-
Momentum Spread	$\frac{dp}{p}$	$2.5 \cdot 10^{-4}$	-
Lorentz Gamma	γ	1.012149995	-
Lorentz Beta	β	0.15448	-
Bunch Length	τ	231.51	ns
Synchrotron Tune	Q_{synch}	$\frac{1}{1000}$	-

Benchmark Information

Steps 7 - 9 Parameters

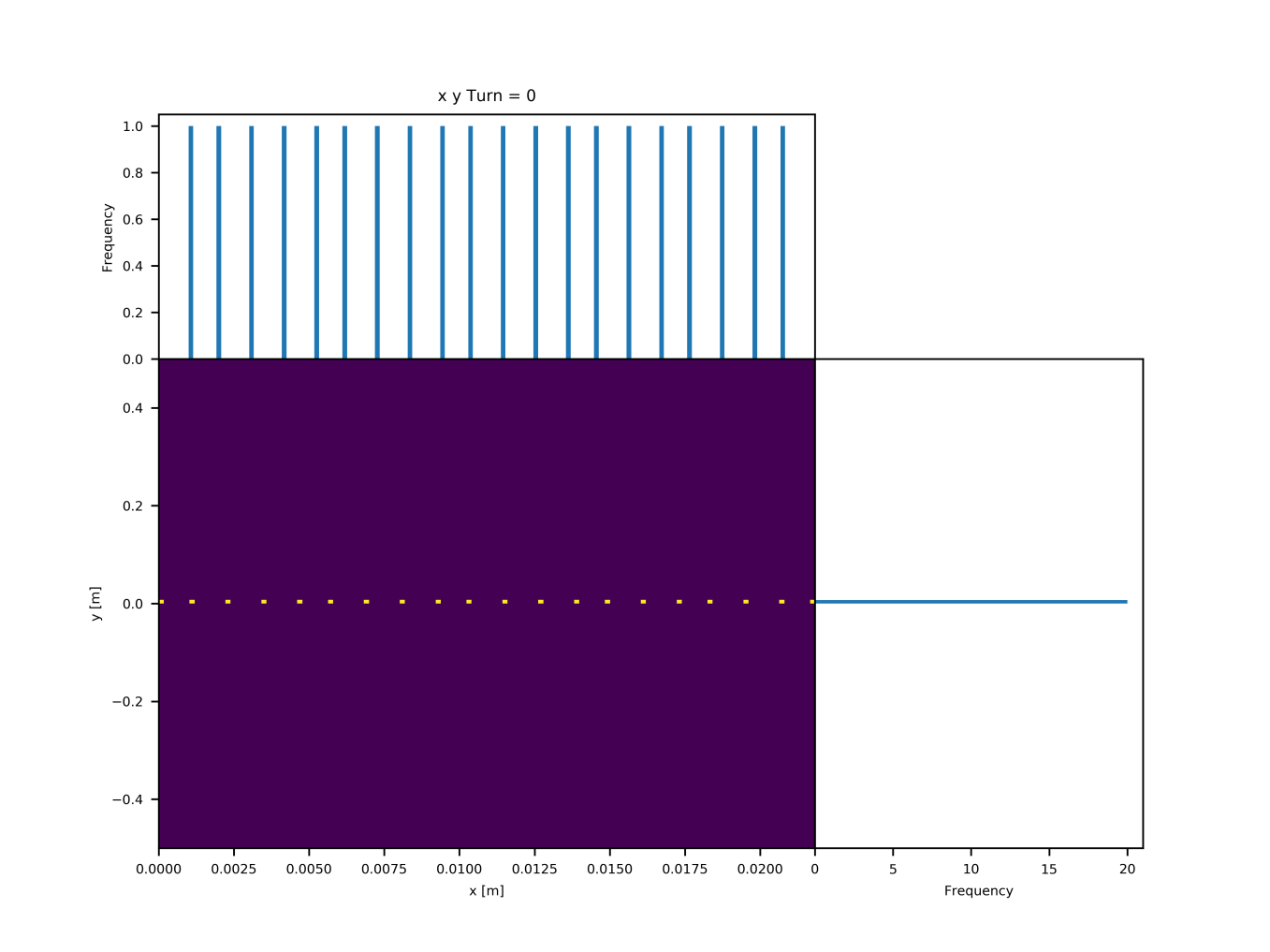
Code Examples

Poincaré distn:

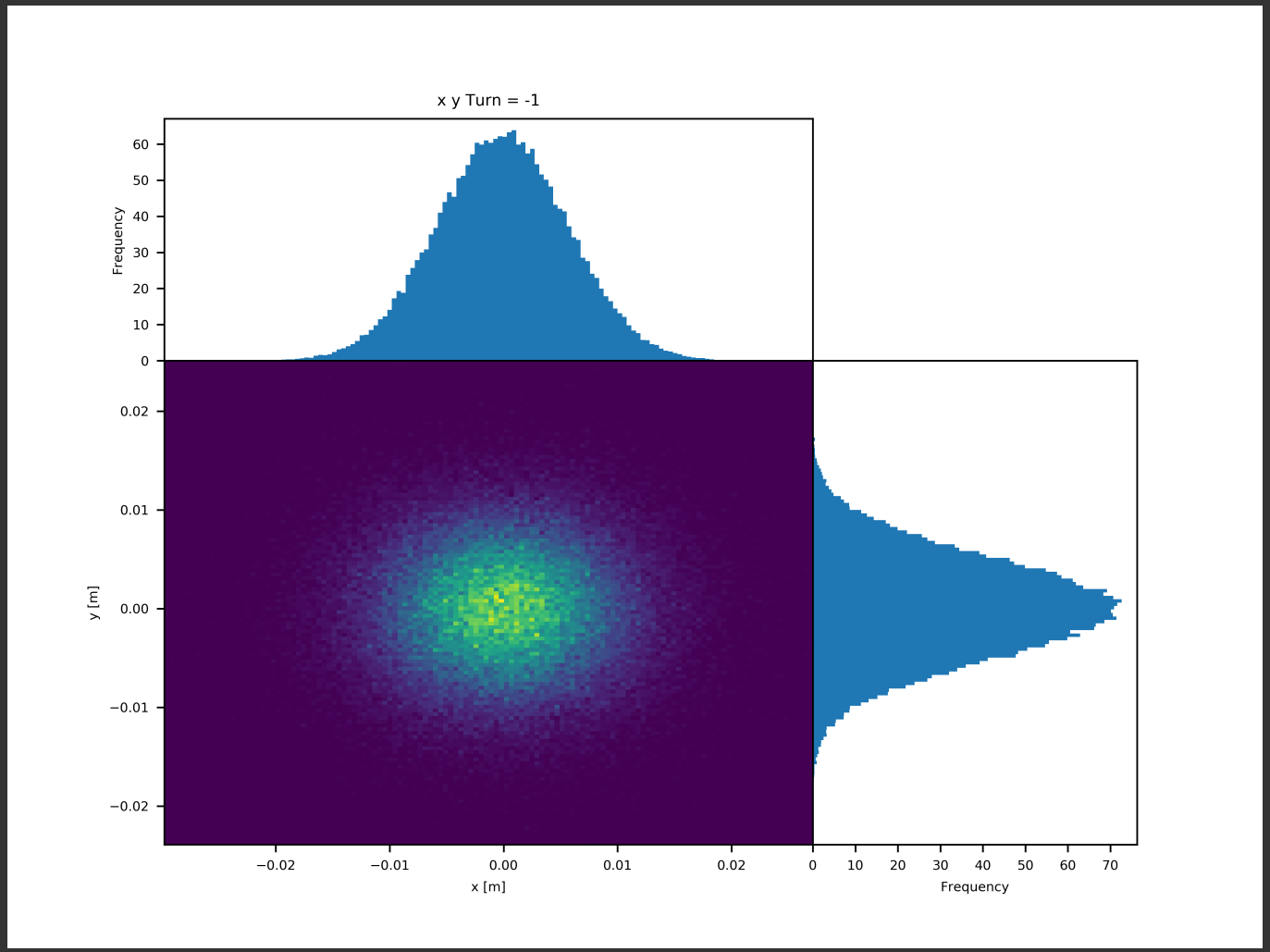
$$x_i = \frac{i\sigma_n}{N}\sqrt{\frac{\beta_{x0}\epsilon_x}{\beta\gamma}} \tag{1}$$

A simple approach - select a maximum amplitude in sigma σ_n, equally spaced (from 0) over the number of particles N, thus i is an integer iterator where we loop over i particles. Then the particle position is the square root of the beta function at the location of bunch creation and emittance (the local beam size) $\beta_{x0}\epsilon x$, normalised with the Lorentz $\beta\gamma$.

>>> Particle Distributions: Poincaré



>>> Particle Distributions:3D Gaussian



>>> Linear Restoring Force

A linear restoring force was used to imitate the synchrotron motion:
The function is simple:

$$dE_i = dE_i + F \cdot z_i$$

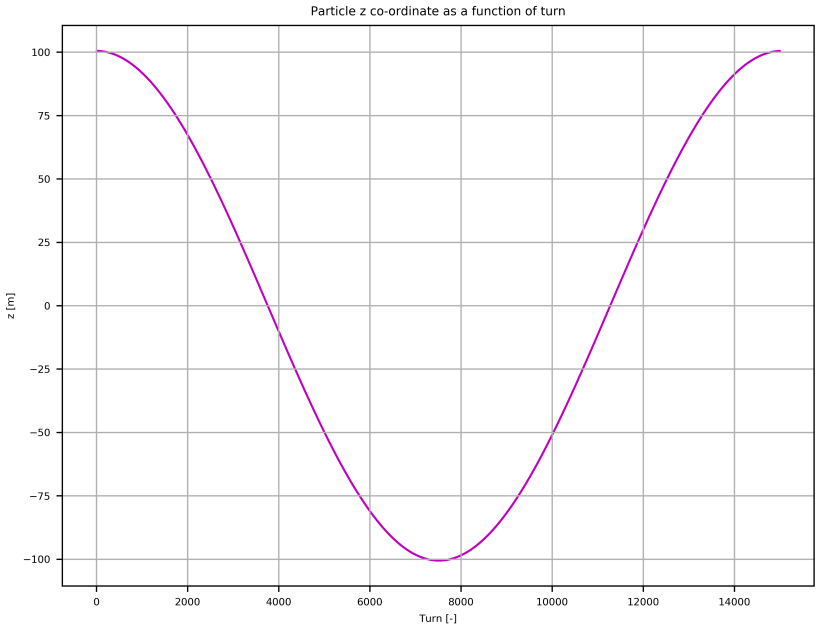
$dE_i =$ energy offset of particle i
z_i = z co-ordinate of particle i
F = restoring force
F may be found analytically using an expression for the synchrotron tune, or empirically.

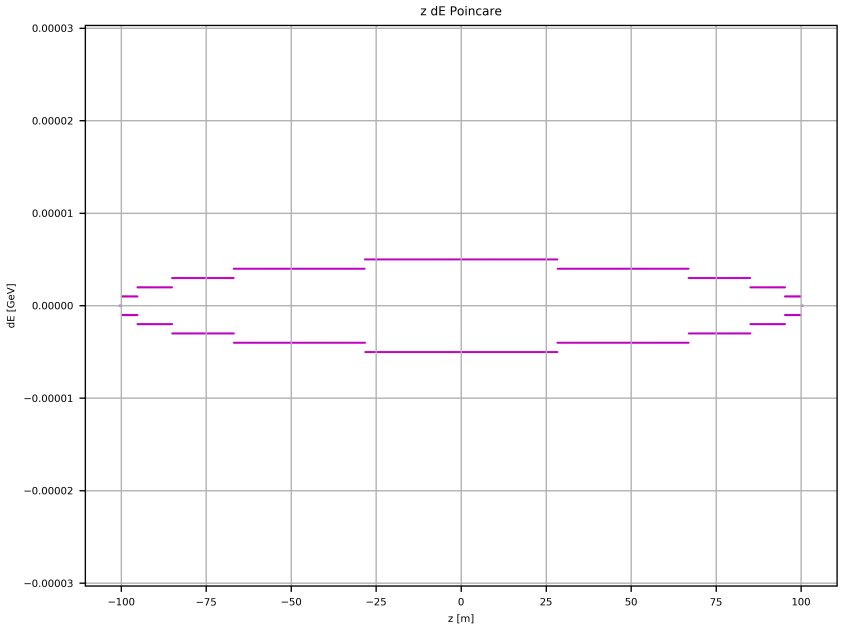
>>> Linear Restoring Force

GitHub, Inc. (US) | https://github.com/HaroonRafique/PTC_PyORBIT_SIS18_Benchmark/blob/master/Step6/lib/pyOrbit_LinearRestoringForce.py

```
6   import math
7   import orbit_mpi
8
9   from orbit_mpi import mpi_datatype, mpi_op
10  from bunch import Bunch
11
12  def LinearRestoringForce(bunch, force):
13
14          rank = 0
15          numprocs = 1
16
17          mpi_init = orbit_mpi.MPI_Initialized()
18          comm = orbit_mpi.mpi_comm.MPI_COMM_WORLD
19
20          if(mpi_init):
21                  rank = orbit_mpi.MPI_Comm_rank(comm)
22                  numprocs = orbit_mpi.MPI_Comm_size(comm)
23
24          nparts_arr_local = []
25          for i in range(numprocs):
26                  nparts_arr_local.append(0)
27
28          nparts_arr_local[rank] = bunch.getSize()
29          data_type = mpi_datatype.MPI_INT
30          op = mpi_op.MPI_SUM
31
32          nparts_arr = orbit_mpi.MPI_Allreduce(nparts_arr_local,data_type,op,comm)
33
34          for i in range(bunch.getSize()):
35                  en = bunch.dE(i)
36
37                  en = en + bunch.z(i) * force
38
39                  bunch.dE(i,en)
40
41          return
```

>>> Linear Restoring Force





>>> Particle Output Dictionary

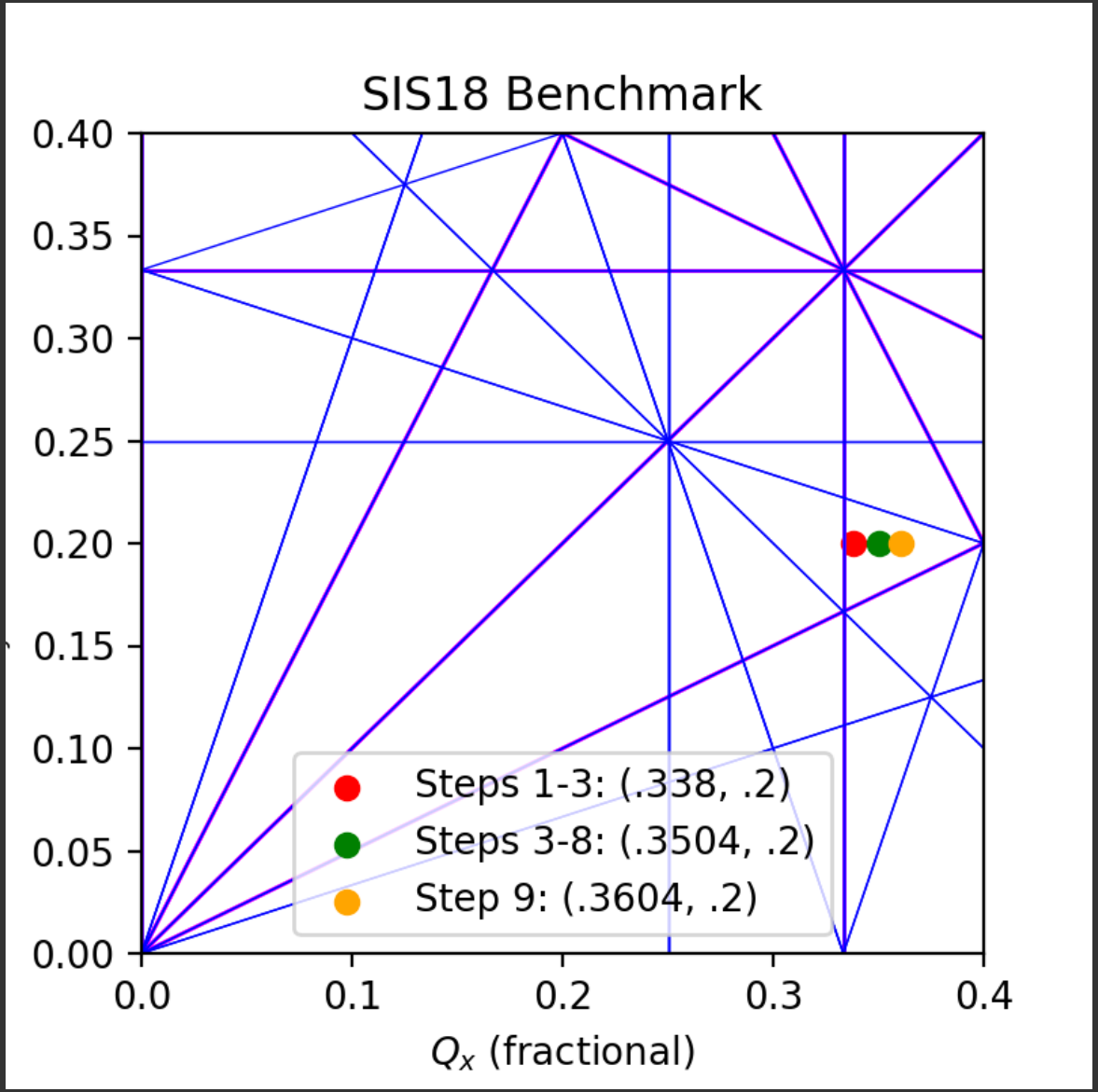
The particle output dictionary was created to output individual particle data into a single file.

GitHub, Inc. (US) | https://github.com/HaroonRafique/PTC_PyORBIT_SIS18_Benchmark/blob/master/Step6/pyOrbit.py

```
222
223  #---------------------------------------------------------------------
224  # Define particle output dictionary - used to plot trapping
225  #---------------------------------------------------------------------
226  particle_output = Particle_output_dictionary()
227
228  # Automatically adds particle 0, add all others manually as below
229  # ~ for i in range(1, p['n_macroparticles']):
230        # ~ particle_output.AddNewParticle(i)
231
232  # Update for turn -1 (pre tracking)
233  particle_output.update(bunch, -1)
234
235  #-----------------------------------------------------
236  # Do some turns and dump particle information
237  #-----------------------------------------------------
238  print '\nnow start tracking...'
239
240  for turn in range(p['turns_max']):
241        Lattice.trackBunch(bunch, paramsDict)
242         LinearRestoringForce(bunch, s['RestoringForce'])
243
244        bunchtwissanalysis.analyzeBunch(bunch)  # analyze twiss and emittance
245
246        if turn in p['turns_print']:
247                saveBunchAsMatfile(bunch, "output/mainbunch_%s"%(str(turn).zfill(6)))
248                saveBunchAsMatfile(lostbunch, "lost/lostbunch_%s"%(str(turn).zfill(6)))
249
250        output.save_to_matfile('output')
251        output.update()
252        particle_output.update(bunch, turn)
253
254        # For last turn output particle dictionary
255        if turn == (p['turns_max']-1):
256                for i in range(0, p['n_macroparticles']):
```

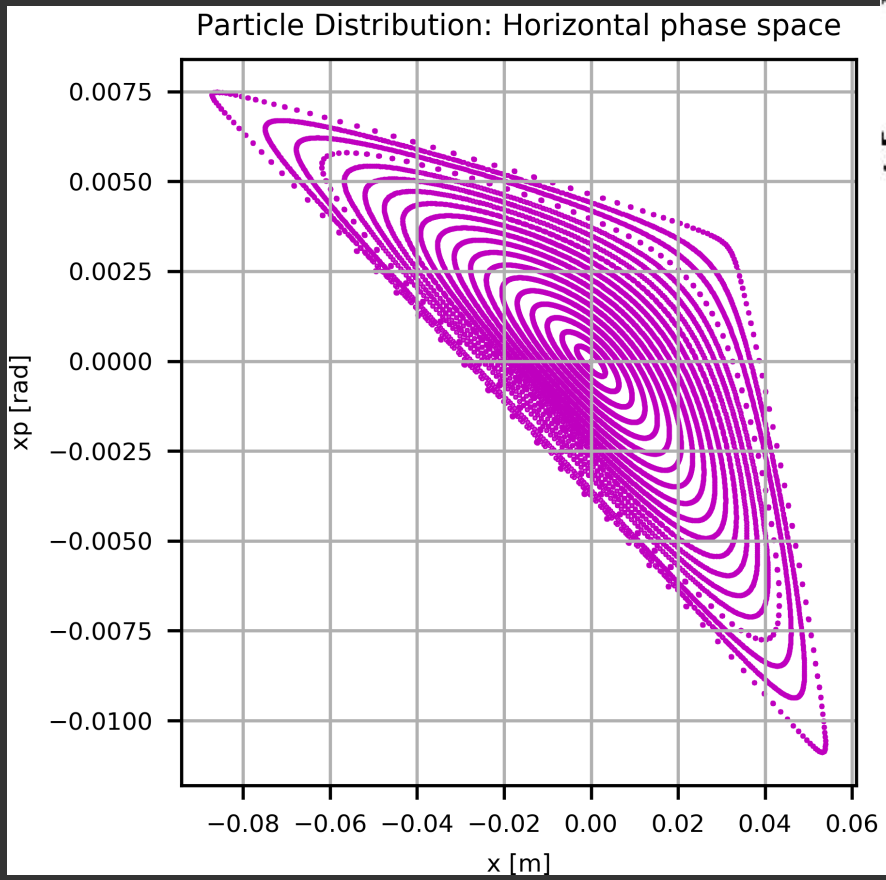
SIS18 Benchmark

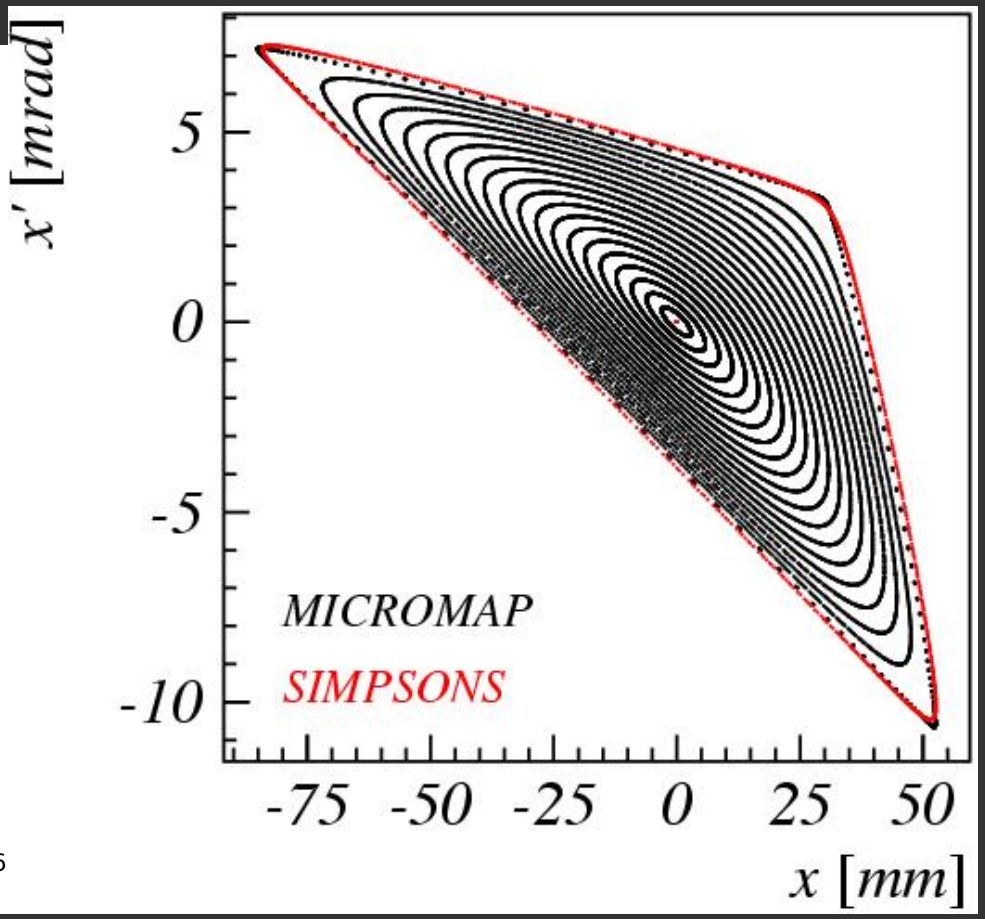
>>> Tune Diagram



>>> Step1: Phase space $(Q_x, Q_y) = (3.2, 4.338)$ $Q_s = \frac{1}{15000}$

This is our starting point. Sextupole on, no space charge - particles are stable up to 6.42 σ_x.

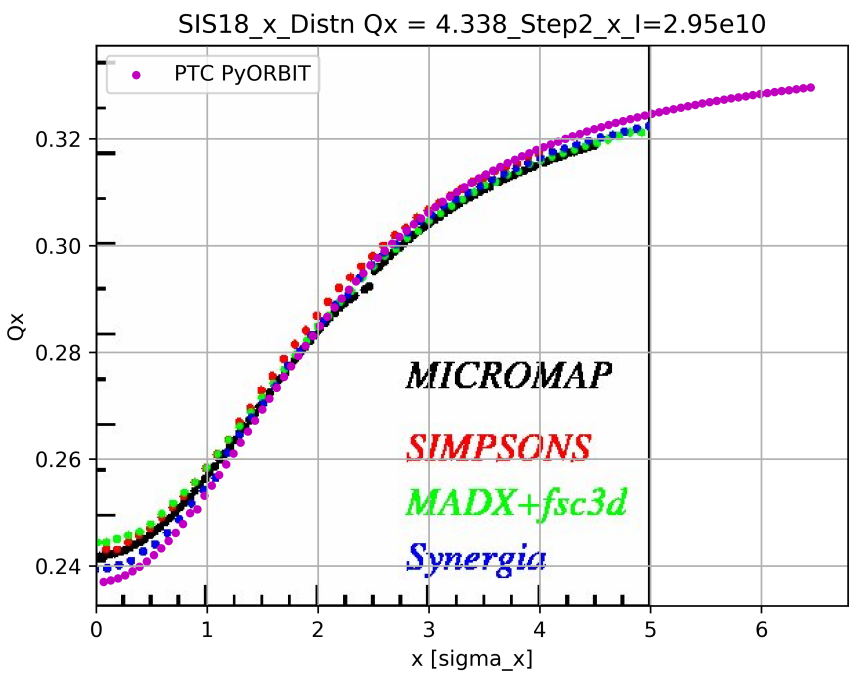


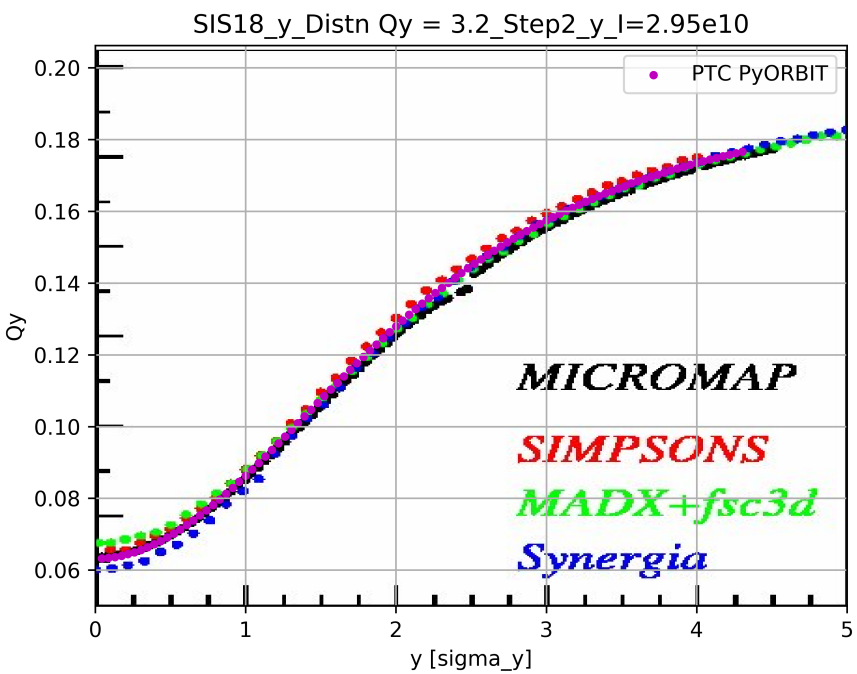


>>> Step2: Tunes with no sextupole

$$(Q_x, Q_y) = (3.2, 4.338) \quad Q_s = \frac{1}{15000}$$

This is the fractional tunespread as a function of particle amplitude $x = \sqrt{\beta_x \gamma_x} x_0$ for the same lattice.

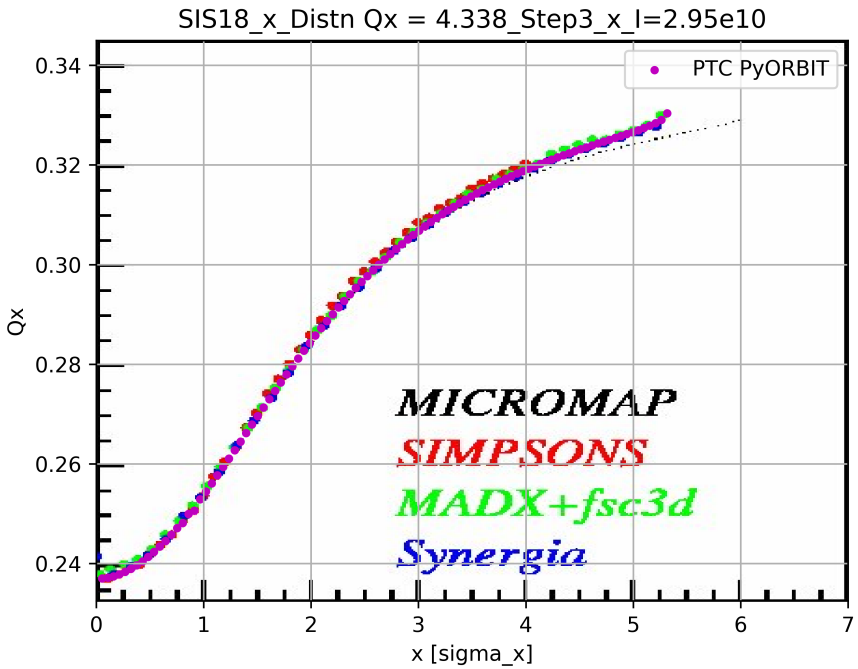


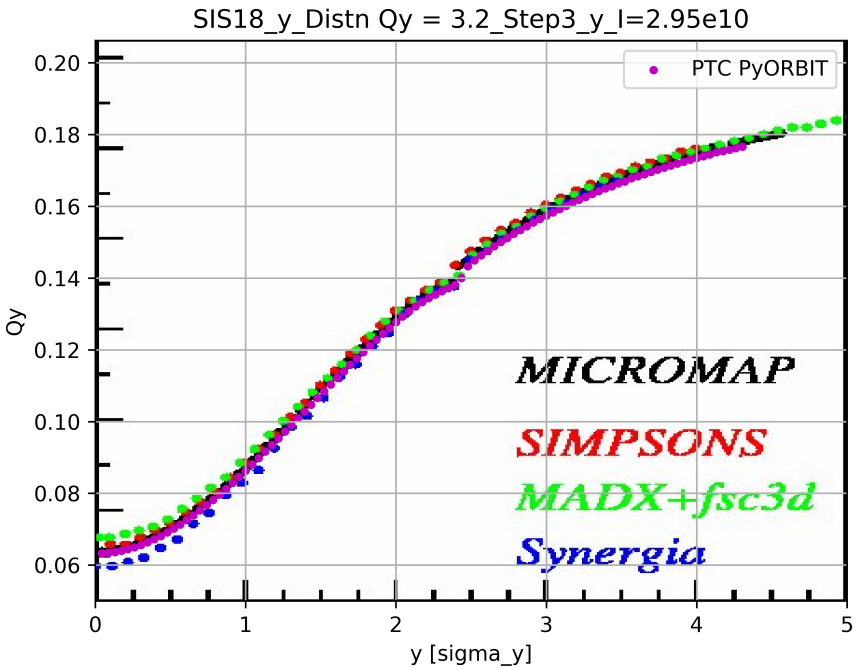


>>> Step3: Tunes with sextupole

$(Q_x, Q_y) = (3.2, 4.338) \quad Q_s = \frac{1}{15000}$

Now we power the sextupole, maintaining the tune. We note that the island is not visible here.

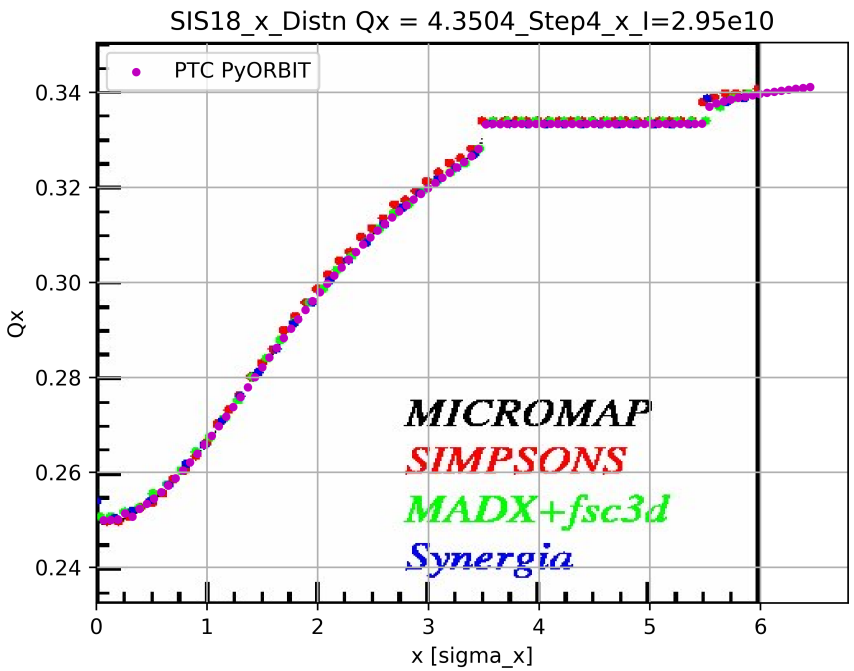


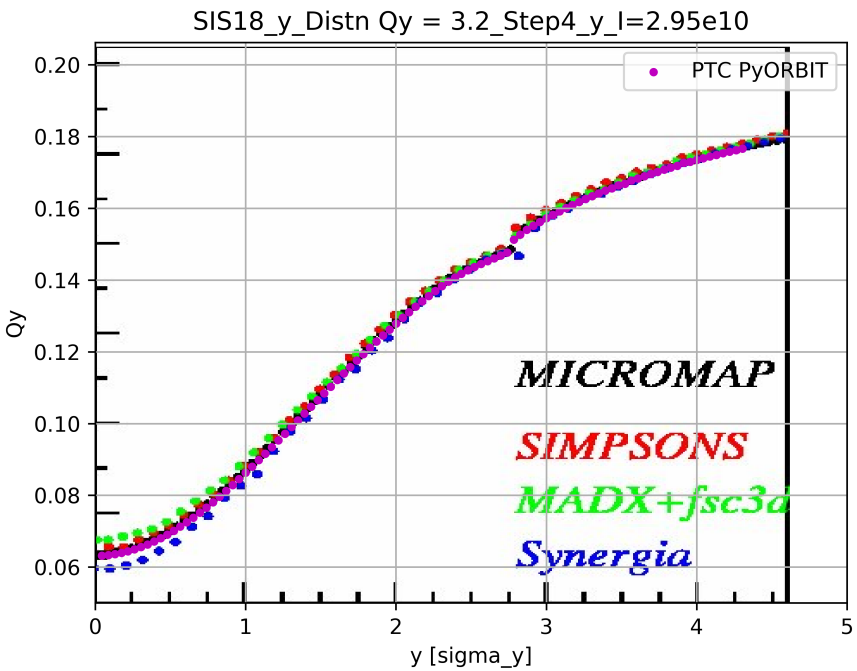


>>> Step4: Tunes with sextupole

$(Q_x, Q_y) = (3.2, 4.3504) \quad Q_s = \frac{1}{15000}$

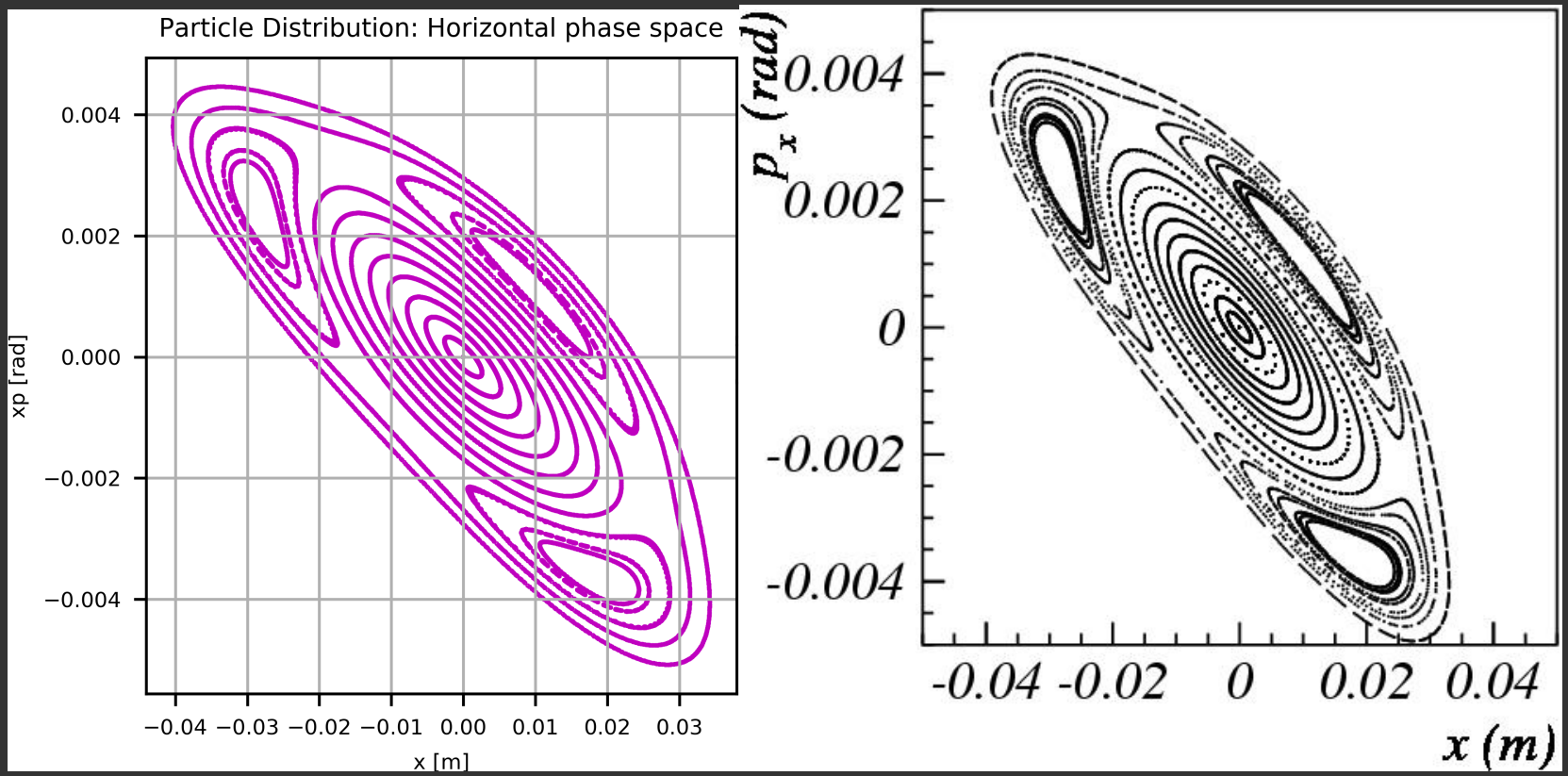
We move further away from the third order resonance in order to view the islands.





>>> Step5: Phase space with sextupole

$(Q_x, Q_y) = (3.2, 4.3504) \quad Q_s = \frac{1}{15000}$

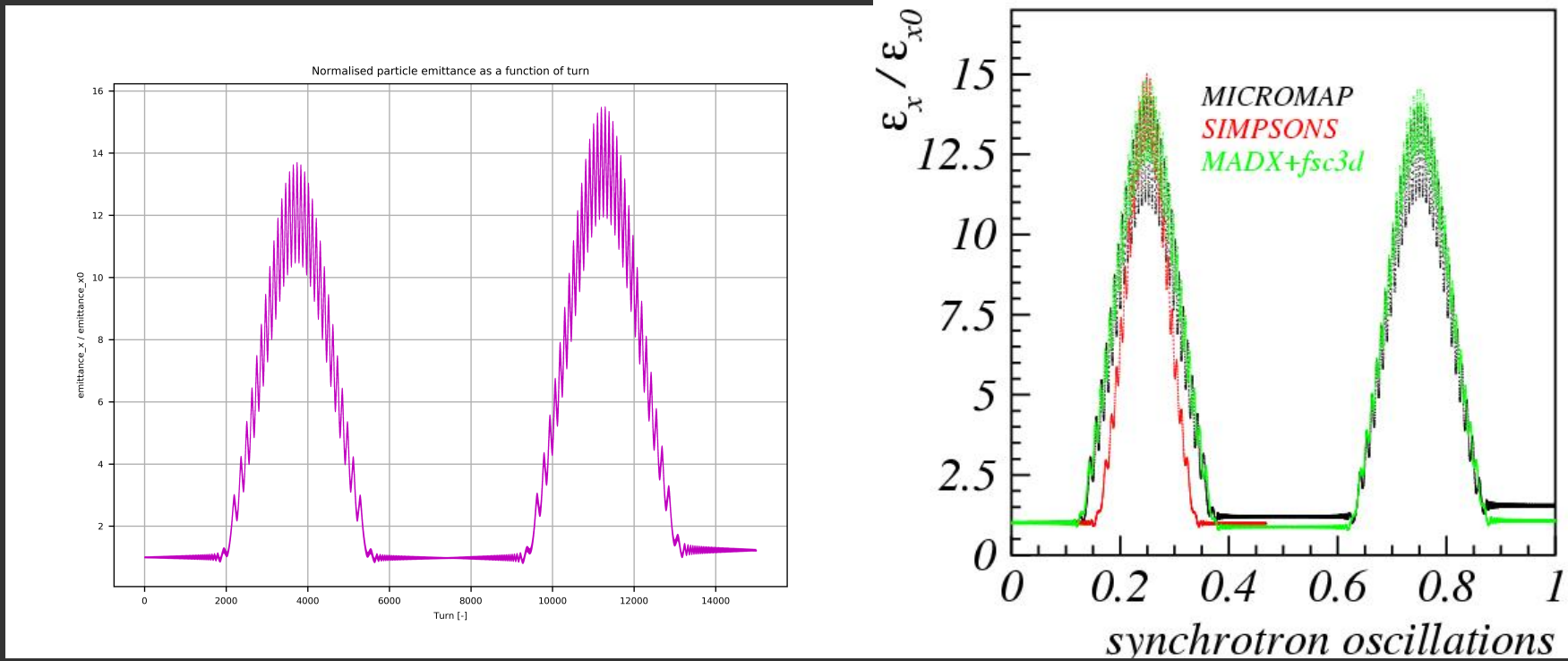


The amplitude of the island may be measured by the size of the flat region in the tune which crosses the resonance (previous slide). It is around 2 σ_x

>>> Step6: Slow trapping with sextupole

$(Q_x, Q_y) = (3.2, 4.3504) \quad Q_s = \frac{1}{15000}$

Trapping of a single particle for one synchrotron period - adiabatic crossing of the resonance.

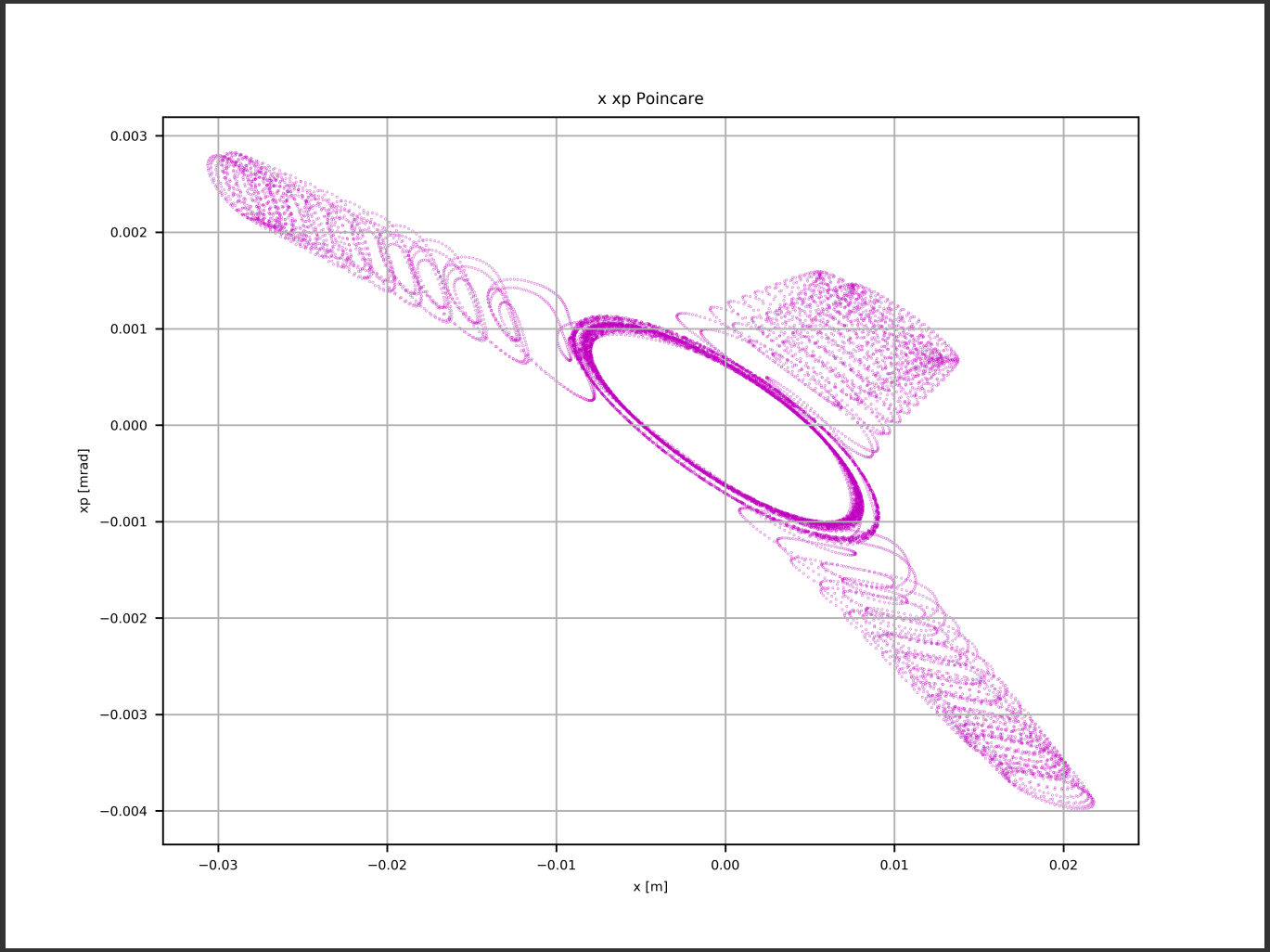


Single particle emittance $\epsilon_x = \beta_x p_x^2 + 2\alpha_x x p_x + \gamma_x x^2$ as a function of turn.

>>> Step6: Slow trapping with sextupole

$(Q_x, Q_y) = (3.2, 4.3504) \quad Q_s = \frac{1}{15000}$

Pretty cool Poincaré section...

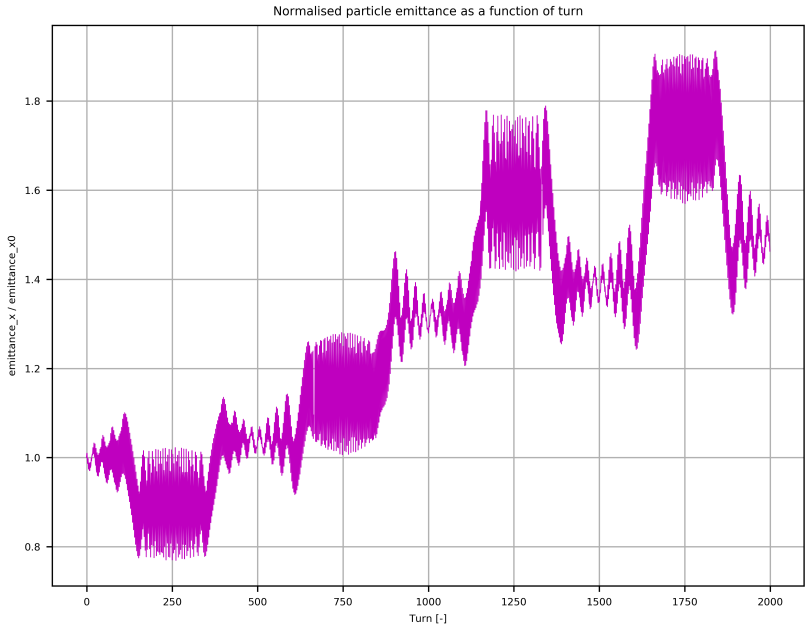


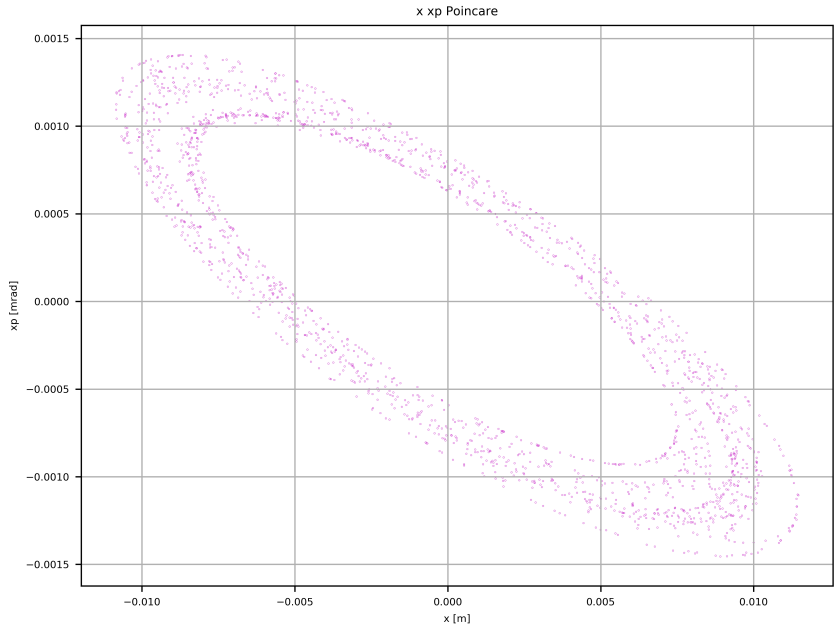
Single particle emittance $\epsilon_x = \beta_x p_x^2 + 2\alpha_x x p_x + \gamma_x x^2$ as a function of turn

>>> Step7: Fast trapping with sextupole

$$(Q_x, Q_y) = (3.2, 4.3504) \quad Q_s = \frac{1}{1000}$$

Single particle scattering - crossing of the resonance too fast to be adiabatic.

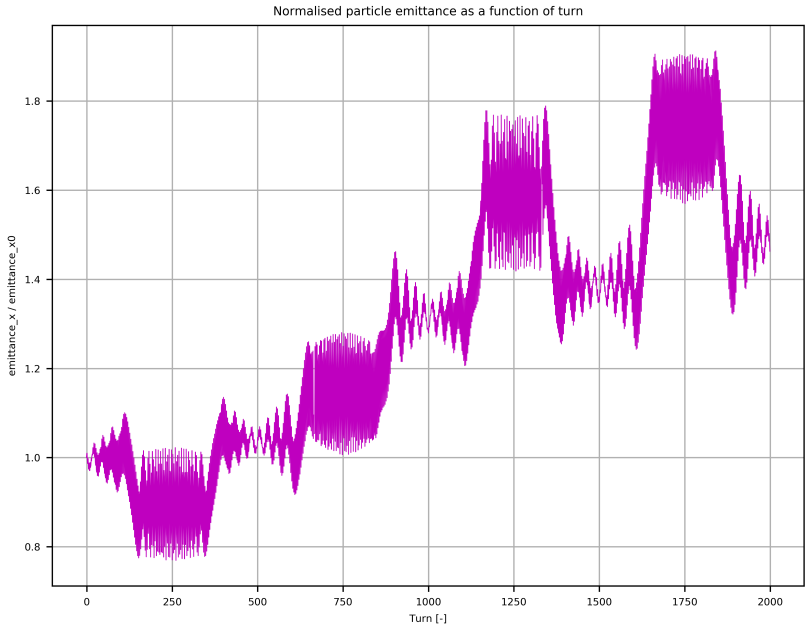


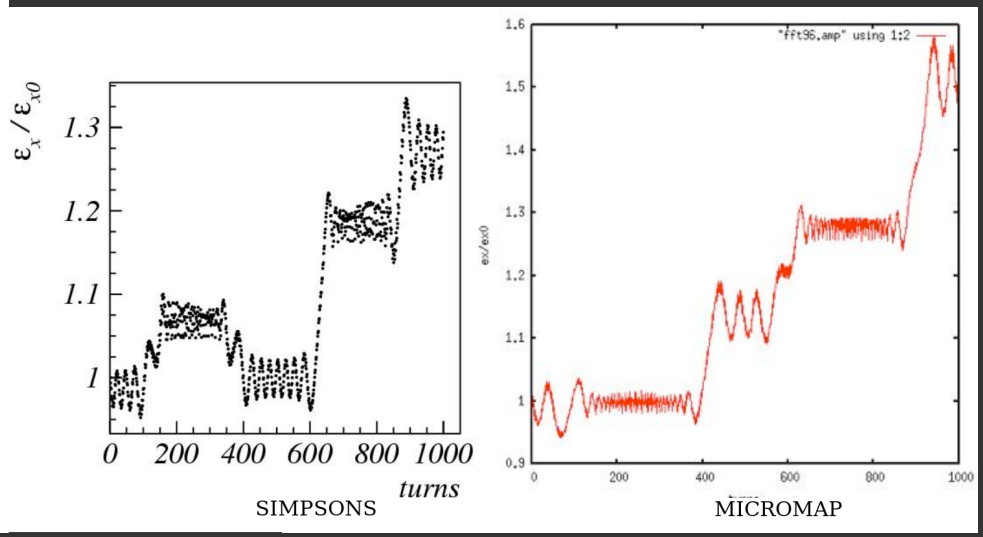


>>> Step7: Fast trapping with sextupole

$(Q_x, Q_y) = (3.2, 4.3504) \quad Q_s = \frac{1}{1000}$

Single particle scattering - crossing of the resonance too fast to be adiabatic.

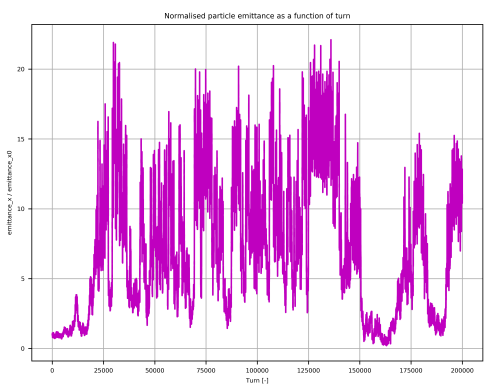


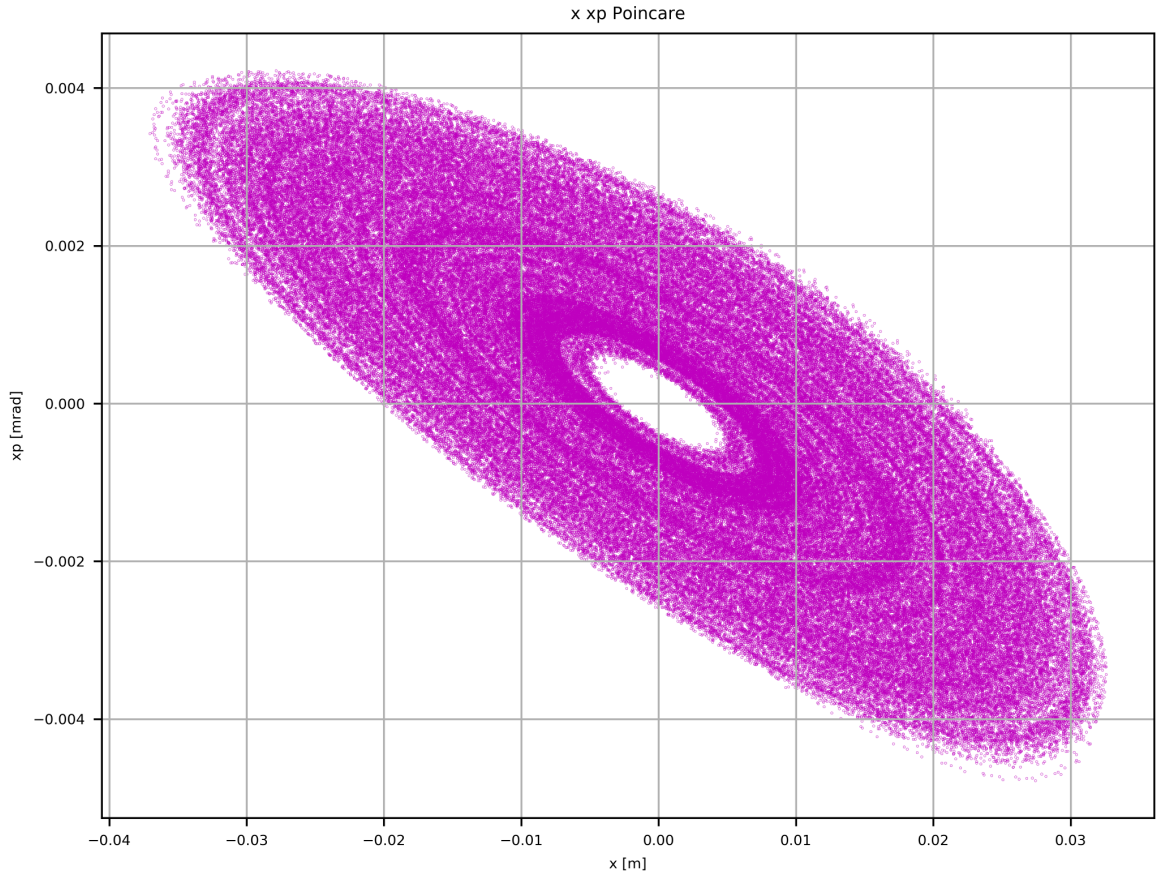


>>> Step8: Fast trapping with sextupole for many turns

$(Q_x, Q_y) = (3.2, 4.3504) \quad Q_s = \frac{1}{1000}$

Single particle scattering - over a longer number of synchrotron periods.

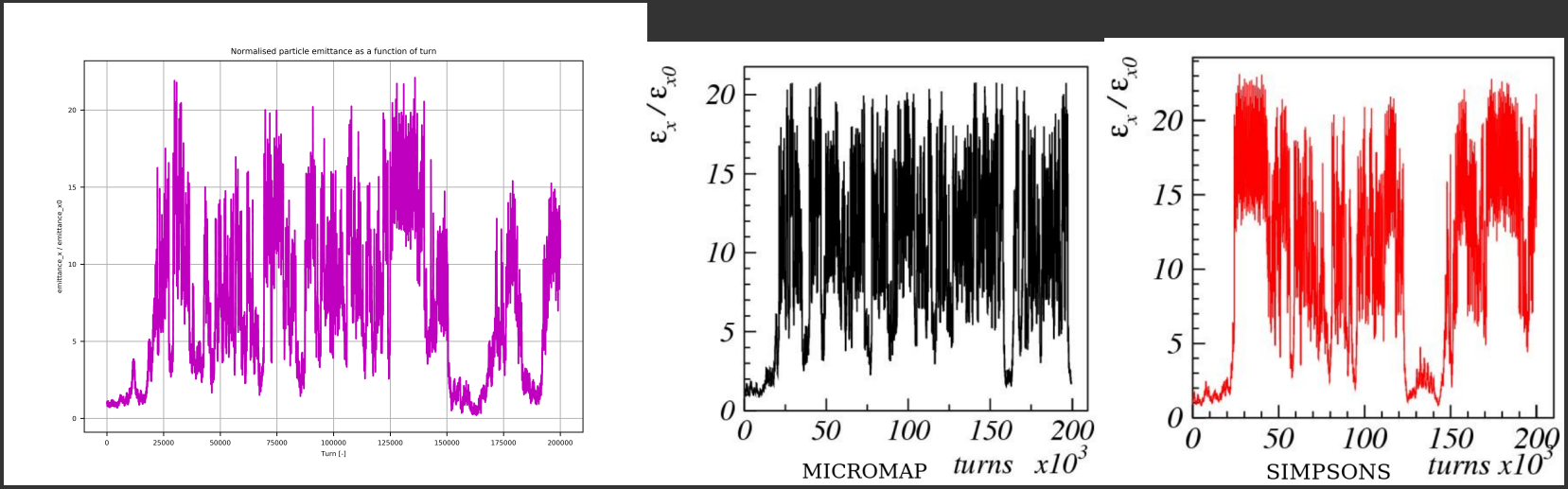




>>> Step8: Fast trapping with sextupole for many turns

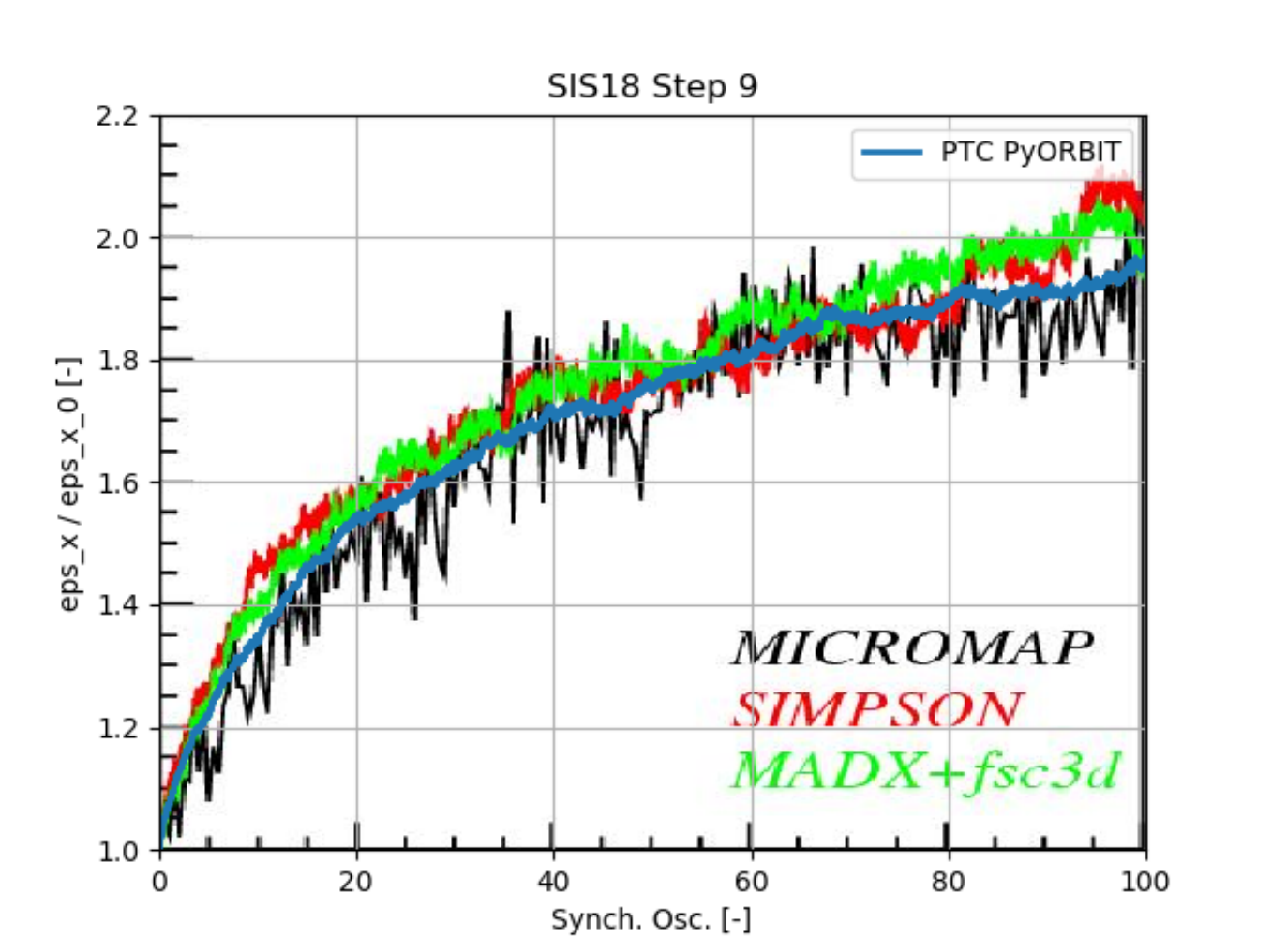
$(Q_x, Q_y) = (3.2, 4.3504) \quad Q_s = \frac{1}{1000}$

Single particle scattering - over a longer number of synchrotron periods.



>>> Step9: Bunch emittance evolution

$(Q_x, Q_y) = (3.2, 4.3604) \quad Q_s = \frac{1}{1000}$



Full bunch of 10^3 particles - emittance growth over 10^5 turns.

Github

>>> Github repository

Nearly everything required to run the benchmark in PTC-PyORBIT can be found at:
https://github.com/HaroonRafique/PTC_PyORBIT_SIS18_Benchmark
Requirements:

* Run on LXPlus
* PyORBIT on AFS (LIU Project Space) - modification of environment variables?
* HTCondor scripts to run long simulations (provided)

Conclusion

>>> Don't Forget

Important things required for PTC-PyORBIT to function as expected:

* 'time.ptc' input file to make sure PTC is in the correct units
* initial parameters must be defined as expected and propagated consistently

>>> Conclusions

* PTC-PyORBIT is in agreement with all other space charge codes that have completed this benchmark:

Person	Code	Laboratory
G. Franchetti	MICROMAP	GSI
S. Machida	SIMPSON	RAL
V. Kapin	MADX+fsc3d	ITEP
F. Schmidt	MADX+ORBITPTC	CERN
J. Amundson, E. Stern	Synergia	FERMILAB
J. Holmes	ORBIT	SNS

* It is imperative that the initial conditions are set up correctly.
* PTC-PyORBIT can be extended relatively easily to include new particle distributions, methods of handling output, etc.