Start-to-end beam dynamics simulations in the RCS chain with XSuite

D. Amorim, E. Kvikne, E. Métral
Thanks to F. Batsch, J. S. Berg, F. Boattini, L. Bottura, X. Buffat, C. Carli, A. Chancé, H. Damerau, A. Grudiev, I. Karpov, T. Pieloni, D. Schulte, K. Skoufaris

HEMAC meeting
2023-12-12

Funded by the European Union (EU). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the EU or European Research Executive Agency (REA). Neither the EU nor the REA can be held responsible for them.

Swiss Accelerator Research and Technology

This work was performed under the auspices and with support from the Swiss Accelerator Research and Technology (CHART) program (www.chart.ch).

Contents

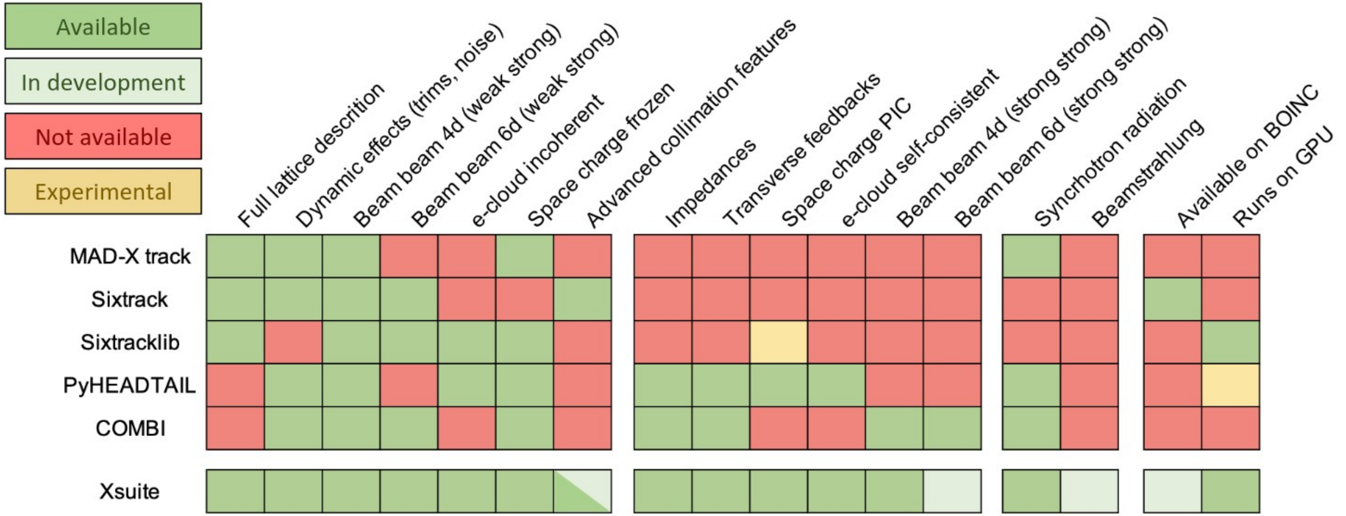
- XSuite presentation
- Simulation setup
- Example of start-to-end simulations and future work

XSuite presentation

Project launched to **rationalize and modernize software for multiparticle simulations**

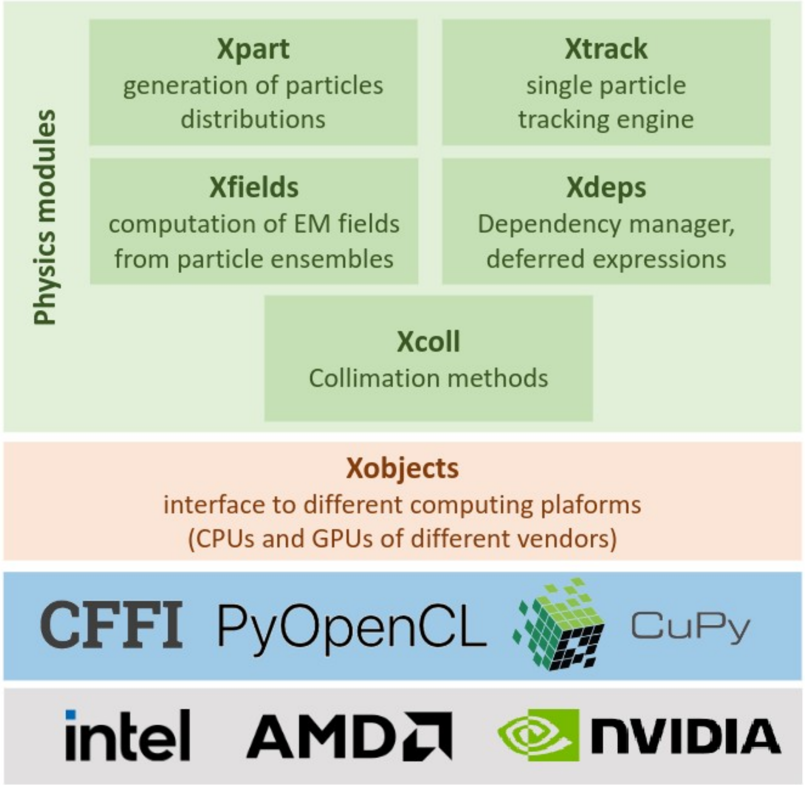
→ Moved **from a heterogenous range of programs** each with limited capabilities to an **integrated modular toolkit** (Xsuite)

- Covering with a single toolkit of **injectors, LHC, HL-LHC and design studies** (e.g. PBC, FCC hh & ee)
- Exploitation of **modern computing platforms** (e.g. GPUs) for a wide range of applications
- Strong **simplification** of development and maintenance process (removes several duplications)



G. Iadarola et al.,
Xsuite: An integrated beam physics simulation framework
CEI section meeting
03/11/2022

XSuite presentation



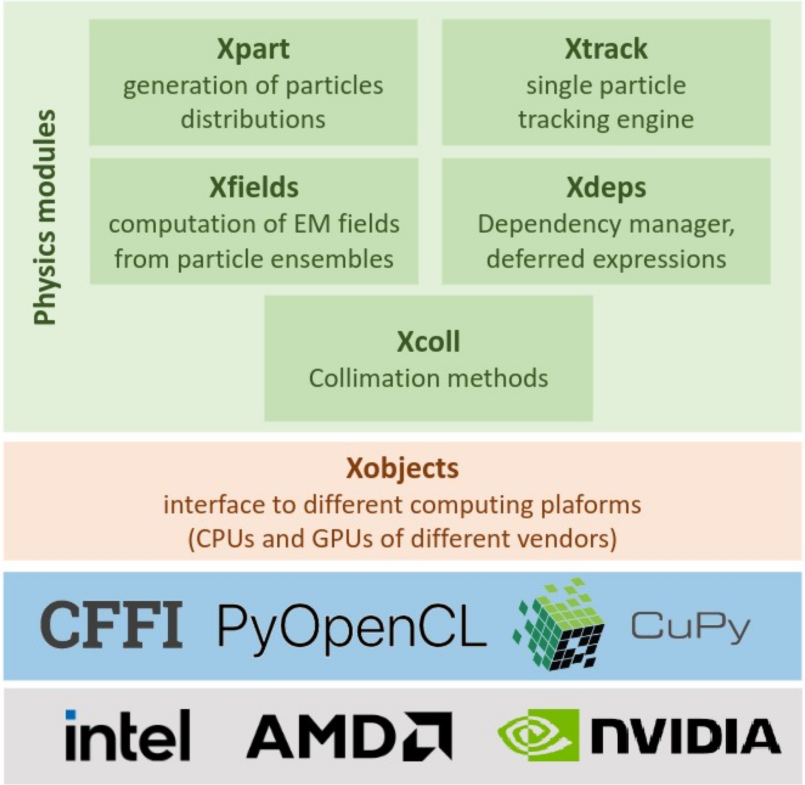
High level methods and objects, building blocks for the physics simulations

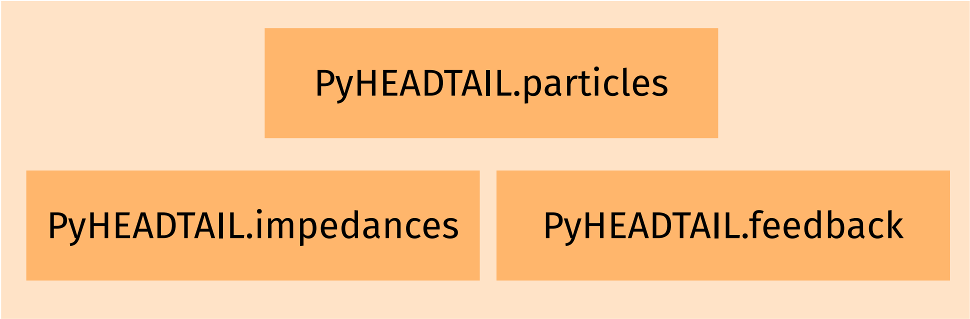
External libraries
(lower-level, interface with hardware)

Hardware

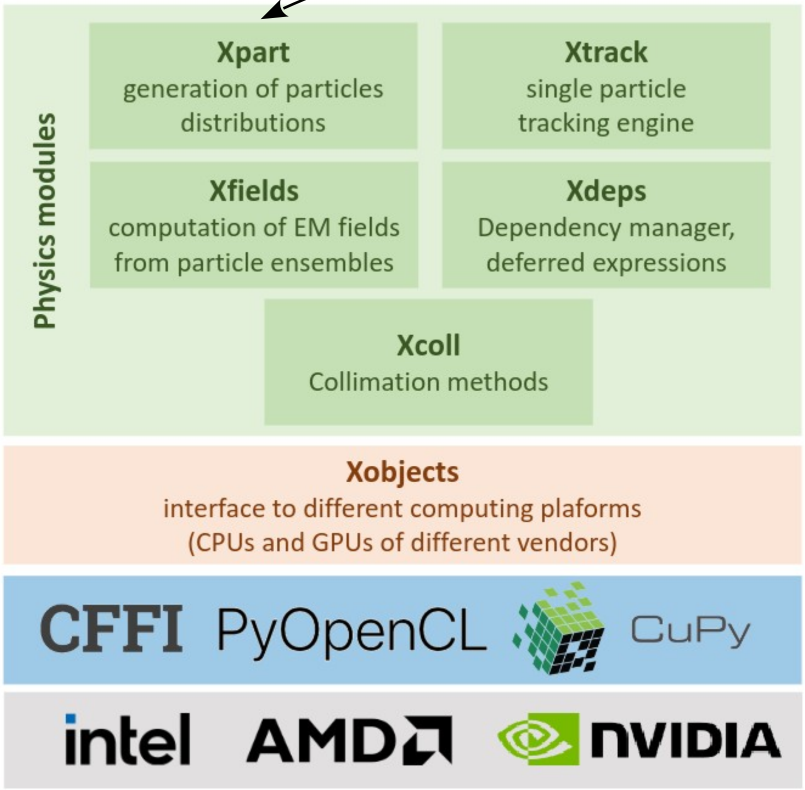
G. Iadarola et al., *ibid*

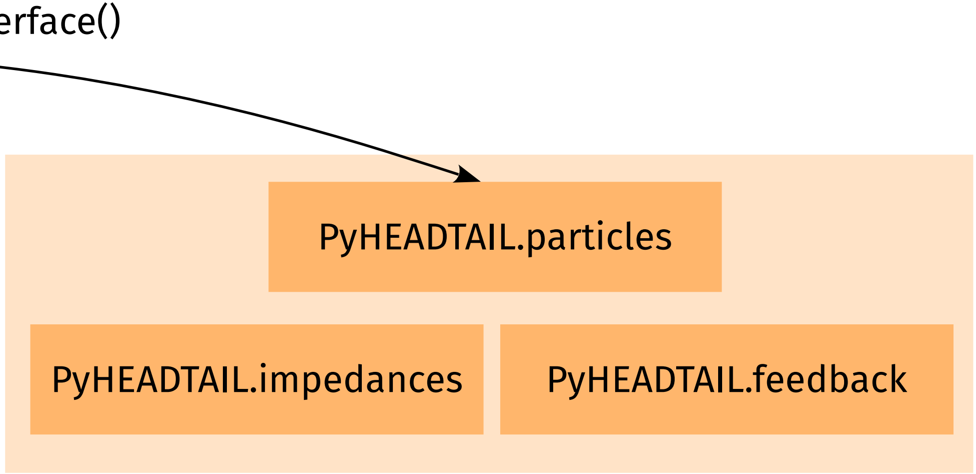
XSuite presentation





XSuite presentation





- enable_pyheadtail_interface() translates the particle coordinates from XSuite to PyHEADTAIL and vice-versa

- Particle distributions are generated with Xpart generators
- Longitudinal and transverse tracking are performed with Xtrack objects
- Impedance and transverse damper effects are computed with PyHEADTAIL objects, then coordinates are translated to XSuite

Contents

- XSuite presentation
- Simulation setup
- Example of start-to-end simulations and future work

RCS parameters and beam dynamics scripts

- Scripts and input data are collected in Gitlab repository https://gitlab.cern.ch/muon-collider-bd/muc-impedance

Scripts and notebooks related to the 10 TeV collider

Python package with modules for machine parameters

Scripts and notebooks for the different RCS

rcs-collective-e… muc-impedance / + History Compare

Name	Last commit
coll10tev	coll10tev: add wake model for copper on tungsten cha...
mucimpedanceparameters	Add pyproject.toml file to make mucrcsparameters pip i...
rcs1	rcs: add single TESLA cavity HOMs wake files
rcs2	rcs2: add the RCS 2 impedance model notebook
results	[RCS1] Add results for LL SRF cavities impedance model
.gitignore	Erik/monitor
LICENSE.md	Add license
README.md	Update the README.md
pyproject.toml	Add pyproject.toml file to make mucrcsparameters pip i...

RCS parameters and beam dynamics scripts

- Scripts and input data are collected in Gitlab repository https://gitlab.cern.ch/muon-collider-bd/muc-impedance
- The mucimpedanceparameters folder is a python package and must be pip installed
  - Requires recent versions of pip and setuptools (tested with versions 23.2 and 68.1)
  - Provides modules particle_parameters.py and synchrotron.py

RCS parameters and beam dynamics scripts

- The synchrotron.py module provides a Synchrotron class
- This class requires a parameter file as input, with the main machine parameters
- Config files are present for RCS 1, 2 and 3, values are based on IMCC parameter report/Fabian's table

muc-impedance / mucimpedanceparameters / machine_configuration / RCS / RCS1_RF_1300MHz_posmuon.yaml

RCS1_RF_1300MHz_posmuon.yaml 790 B

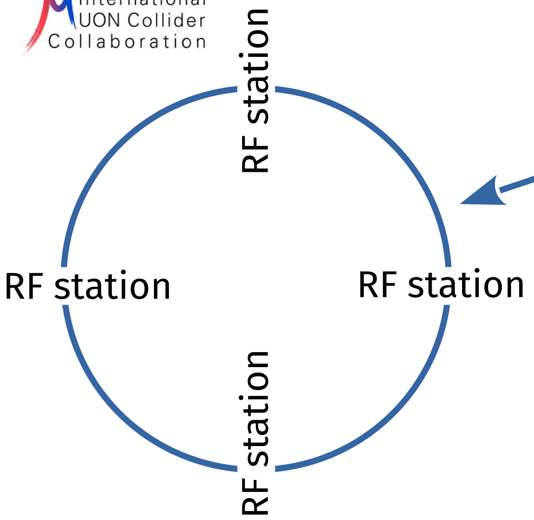

```yaml
# Paramters file for the RCS1 at injection energy
# Reference for values: F. Batsch HEMAC parameters
#
# Bunch length 1 sigmaz = 23.1mm/4 = 5.775 mm
# RF cavity phase is given in degrees
# emit_z is the product sigma_z * sigma_E in eV s
Ring Parameters:
    name: RCS1
    year: 2022
    state: injection
    circumference: 5990
Beam Parameters:
    particle_name: PosMuon
    E_kinetic: 63.0e+09
    harmonic: 25917
    RF_voltage: 20.87e+09
    sigmaz: 5.775e-3
    emit_z: 0.025
    alphap: 2.4e-3
    synchrotron_phase: 45
    energy_gain_per_turn: 14755.0e+06
    number_of_rf_stations: 32
    number_of_bunches: 1
    initial_bunch_intensity: 2.7e+12
    Qx_frac: 0.26
    Qy_frac: 0.26
    average_beta_x: 50
    average_beta_y: 50
    norm_emit_x: 25.0e-06
    norm_emit_y: 25.0e-06
```

Simulation setup

International MUON Collider Collaboration

- XSuite uses Line objects (part of Xtrack) to model a ring
  - A line can contain all kind of elements defined in Xtrack: bends, quadrupoles, multipoles, RF cavities, electron lenses…
  - For our studies, we use **LineSegmentMap** elements (analog to the TransverseMap and LongitudinalMap objects of PyHEADTAIL)

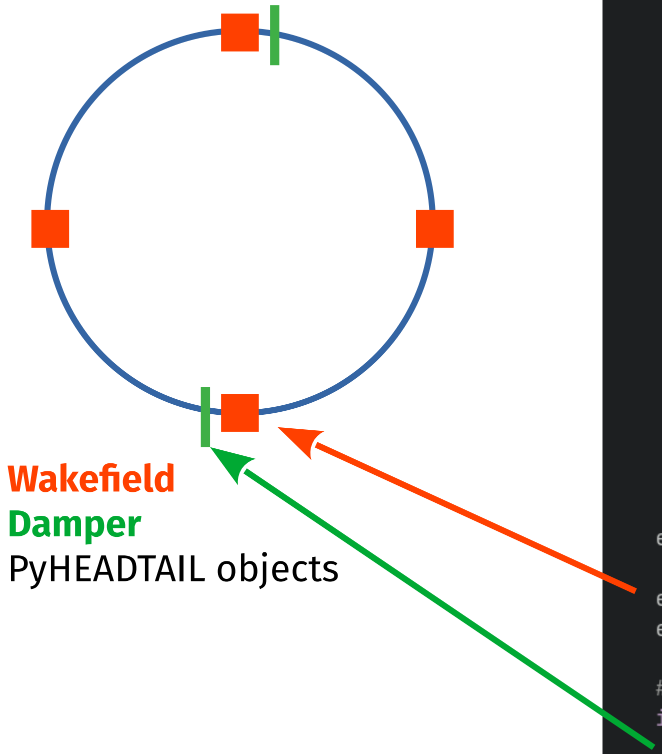
Simulation setup




```
for ii_rf_station in range(0, number_of_rf_stations):
    elements_list.append(xt.LineSegmentMap(length=accelerator_parameters.circumference/number_of_rf_stations,
                                           qx=average_Qx/number_of_rf_stations, qy=average_Qy/number_of_rf_stations,
                                           betx=beta_x, bety=beta_y, alfx=alpha_x, alfy=alpha_y,
                                           dx=0., dpx=0., dy=0., dpy=0.,
                                           x_ref=0.0, px_ref=0.0, y_ref=0.0, py_ref=0.0,
                                           longitudinal_mode=rf_longitudinal_mode,
                                           qs=None, bets=None,
                                           momentum_compaction_factor=momentum_compaction_factor,
                                           slippage_length=None,
                                           voltage_rf=rf_voltage/number_of_rf_stations,
                                           frequency_rf=rf_frequency, lag_rf=rf_lag_degrees,
                                           dqx=chroma_x, dqy=chroma_y,
                                           detx_x=0.0, detx_y=0.0, dety_y=0.0, dety_x=0.0,
                                           energy_increment=0,
                                           energy_ref_increment=energy_increment_per_turn/number_of_rf_stations,
                                           damping_rate_x = 0.0, damping_rate_y = 0.0, damping_rate_s = 0.0,
                                           equ_emit_x = 0.0, equ_emit_y = 0.0, equ_emit_s = 0.0,
                                           gauss_noise_ampl_x=0.0,gauss_noise_ampl_px=0.0,
                                           gauss_noise_ampl_y=0.0,gauss_noise_ampl_py=0.0,
                                           gauss_noise_ampl_zeta=0.0,gauss_noise_ampl_delta=0.0,))
    elements_names_list.append(f'arc_{ii_rf_station}_{ii_rf_station+1}')
```

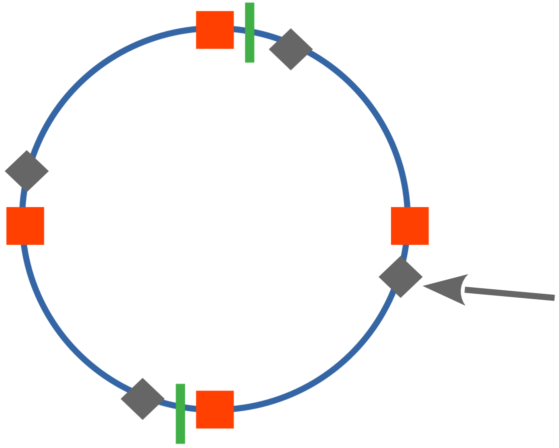
LineSegmentMap
Longitudinal map
(including acceleration)
+ Transverse map

Simulation setup




```python
for ii_rf_station in range(0, number_of_rf_stations):
    elements_list.append(xt.LineSegmentMap(length=accelerator_parameters.circumference/number_of_rf_stations,
                                           qx=average_Qx/number_of_rf_stations, qy=average_Qy/number_of_rf_stations,
                                           betx=beta_x, bety=beta_y, alfx=alpha_x, alfy=alpha_y,
                                           dx=0., dpx=0., dy=0., dpy=0.,
                                           x_ref=0.0, px_ref=0.0, y_ref=0.0, py_ref=0.0,
                                           longitudinal_mode=rf_longitudinal_mode,
                                           qs=None, bets=None,
                                           momentum_compaction_factor=momentum_compaction_factor,
                                           slippage_length=None,
                                           voltage_rf=rf_voltage/number_of_rf_stations,
                                           frequency_rf=rf_frequency, lag_rf=rf_lag_degrees,
                                           dqx=chroma_x, dqy=chroma_y,
                                           detx_x=0.0, detx_y=0.0, dety_y=0.0, dety_x=0.0,
                                           energy_increment=0,
                                           energy_ref_increment=energy_increment_per_turn/number_of_rf_stations,
                                           damping_rate_x = 0.0, damping_rate_y = 0.0, damping_rate_s = 0.0,
                                           equ_emit_x = 0.0, equ_emit_y = 0.0, equ_emit_s = 0.0,
                                           gauss_noise_ampl_x=0.0,gauss_noise_ampl_px=0.0,
                                           gauss_noise_ampl_y=0.0,gauss_noise_ampl_py=0.0,
                                           gauss_noise_ampl_zeta=0.0,gauss_noise_ampl_delta=0.0,))
    elements_names_list.append(f'arc_{ii_rf_station}_{ii_rf_station+1}')

    elements_list.append(wake_field)
    elements_names_list.append(f'wakefield_{ii_rf_station+1}')

    # Add the transverse damper at the given location in the ring
    if ii_rf_station in damper_location_index_list:
        elements_list.append(TransverseDamper(dampingrate_x=damper_strength,
                                              dampingrate_y=damper_strength))
        elements_names_list.append(f'damper_{ii_rf_station+1}')
```

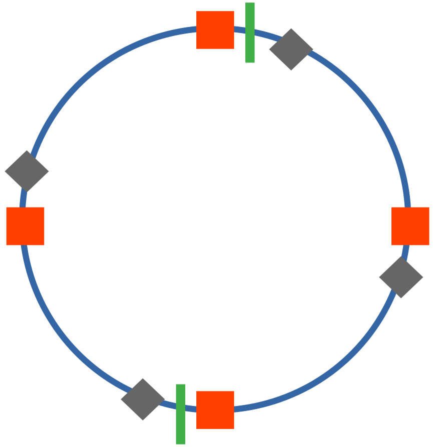
Simulation setup



ParticleMonitor
Longitudinal and Transverse apertures

```
# Add a particle monitor at each RF station
elements_list.append(xt.ParticlesMonitor(start_at_turn=n_turns_scan_cumsum[ii_rcs_to_study],
                                         stop_at_turn=n_turns_scan_cumsum[ii_rcs_to_study+1],
                                         num_particles=n_macroparticles_monitored))
elements_names_list.append(f'monitor_{ii_rf_station+1}')

# Add a Longitudinal aperture to remove uncaptured particles (in longitudinal)
elements_list.append(xt.LongitudinalLimitRect(min_zeta=-0.1, max_zeta=0.1))
elements_names_list.append(f'longitudinal_aperture_{ii_rf_station+1}')

# Add a Transverse aperture to remove unstable particles
elements_list.append(xt.LimitRect(min_x=-100e-3, max_x=100e-3, min_y=-100e-3, max_y=100e-3))
elements_names_list.append(f'transverse_rectangular_aperture_{ii_rf_station+1}')


line = xt.Line(elements=elements_list, element_names=elements_names_list)
```

Simulation setup



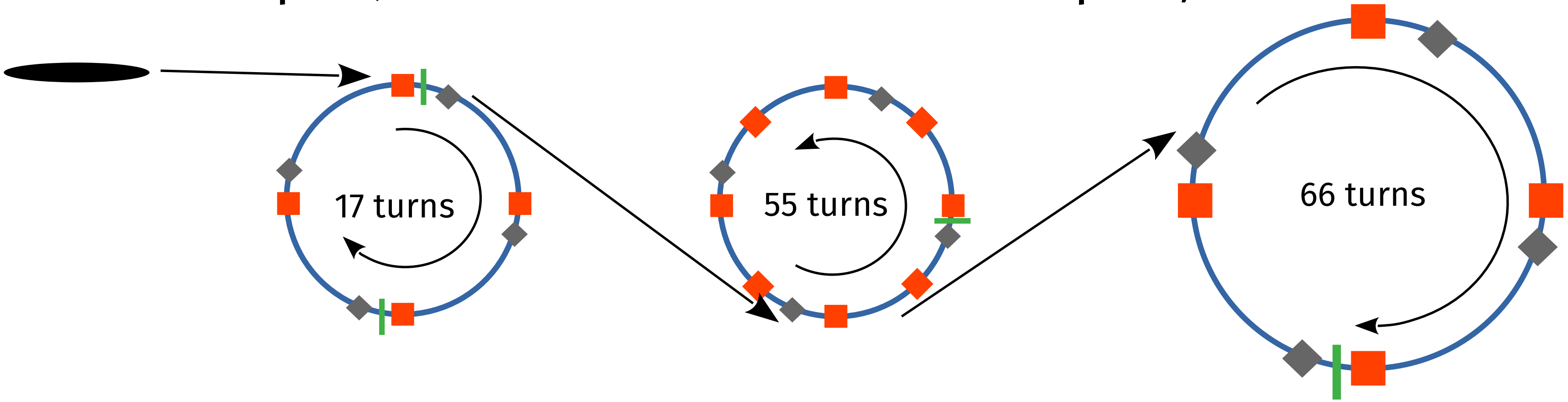
```
    # Add a particle monitor at each RF station
    elements_list.append(xt.ParticlesMonitor(start_at_turn=n_turns_scan_cumsum[ii_rcs_to_study],
                                             stop_at_turn=n_turns_scan_cumsum[ii_rcs_to_study+1],
                                             num_particles=n_macroparticles_monitored))
    elements_names_list.append(f'monitor_{ii_rf_station+1}')

    # Add a Longitudinal aperture to remove uncaptured particles (in longitudinal)
    elements_list.append(xt.LongitudinalLimitRect(min_zeta=-0.1, max_zeta=0.1))
    elements_names_list.append(f'longitudinal_aperture_{ii_rf_station+1}')

    # Add a Transverse aperture to remove unstable particles
    elements_list.append(xt.LimitRect(min_x=-100e-3, max_x=100e-3, min_y=-100e-3, max_y=100e-3))
    elements_names_list.append(f'transverse_rectangular_aperture_{ii_rf_station+1}')


line = xt.Line(elements=elements_list, element_names=elements_names_list)
```

Create the XSuite line used for tracking

Simulation setup

International MUON Collider Collaboration

- This process is repeated for all RCS we want to study
- Each RCS parameter can be set with the configuration file + inputs inside the scripts (number and location of dampers, wakefield model to use…)



- Now we need a **distribution of particles** that will be tracked through the different lines

Simulation setup

If we are currently studying the first RCS in the chain, we must generate the particle distribution beforehand.
Otherwise we use the distribution that comes out of the previous line.

Longitudinal bunch matching.
Xsuite routines are the same as PyHEADTAIL’s.

Given:
- the RF bucket parameters
- and the target longitudinal emittance

The matcher will try generate the longitudinal distribution

Transverse coordinates generation

A particle distribution is then created, and will be tracked through the different lines

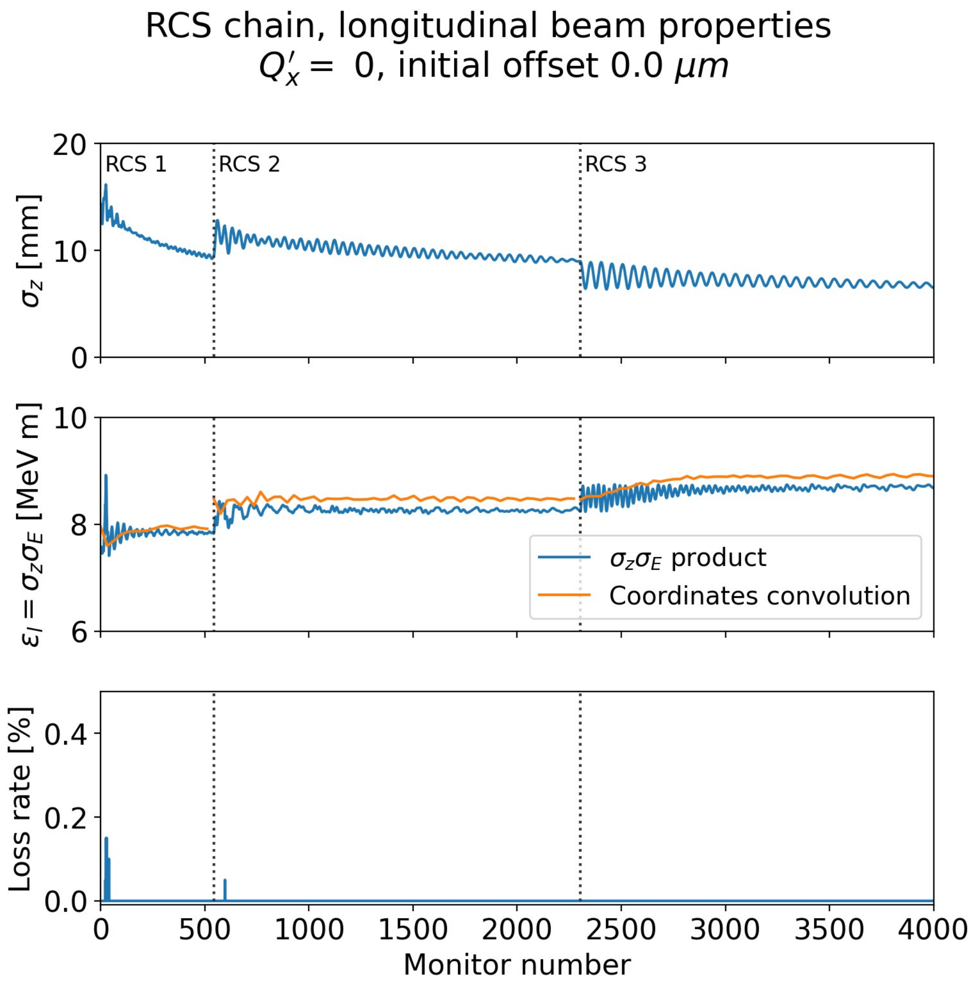
```python
    # We generate the particle distribution only if we are looking at the first RCS simulated
    if rcs_to_study == 'RCS1':

        # Define the reference particle for the simulations using the parameters specified beforehand
        particle_ref = xp.Particles(p0c=particle_p0c, mass0=particle_mass_eV,
                                    q0=particle_charge_number, x=0 , y=0, zeta=0)

        p_increment = energy_increment_per_turn * e / c

        rfbucket = RFBucket(circumference=accelerator_parameters.circumference,
                            gamma=gamma,
                            mass_kg=particle_mass_kg,
                            charge_coulomb=particle_charge,
                            alpha_array=np.atleast_1d(momentum_compaction_factor),
                            # alpha_array=np.atleast_1d(1.6e-4),
                            harmonic_list=np.atleast_1d(rf_harmonic_number),
                            voltage_list=np.atleast_1d(rf_voltage),
                            phi_offset_list=np.atleast_1d((rf_lag_degrees)*np.pi/180),
                            p_increment=p_increment)

        matcher = RFBucketMatcher(rfbucket=rfbucket,
                                  distribution_type=ThermalDistribution,
                                  #   sigma_z=None,
                                  epsn_z=4*np.pi*emit_z)

        z_particles, delta_particles, _ = matcher.generate(macroparticlenumber=n_macroparticles)

        line.particle_ref = particle_ref.copy()
        line.particle_ref.zeta = 0

        x_in_sigmas, px_in_sigmas = xp.generate_2D_gaussian(n_macroparticles)
        y_in_sigmas, py_in_sigmas = xp.generate_2D_gaussian(n_macroparticles)

        particles = line.build_particles(
                    zeta=z_particles-rfbucket.z_sfp,
                    # zeta=z_particles,
                    delta=delta_particles,
                    x_norm=x_in_sigmas, px_norm=px_in_sigmas,
                    y_norm=y_in_sigmas, py_norm=py_in_sigmas,
                    nemitt_x=norm_emit_x, nemitt_y=norm_emit_y,
                    weight=initial_bunch_intensity/n_macroparticles)
        particles.circumference = accelerator_parameters.circumference
```

Contents

- XSuite presentation
- Simulation setup
- Example of start-to-end simulations and future work

Example of start-to-end simulations

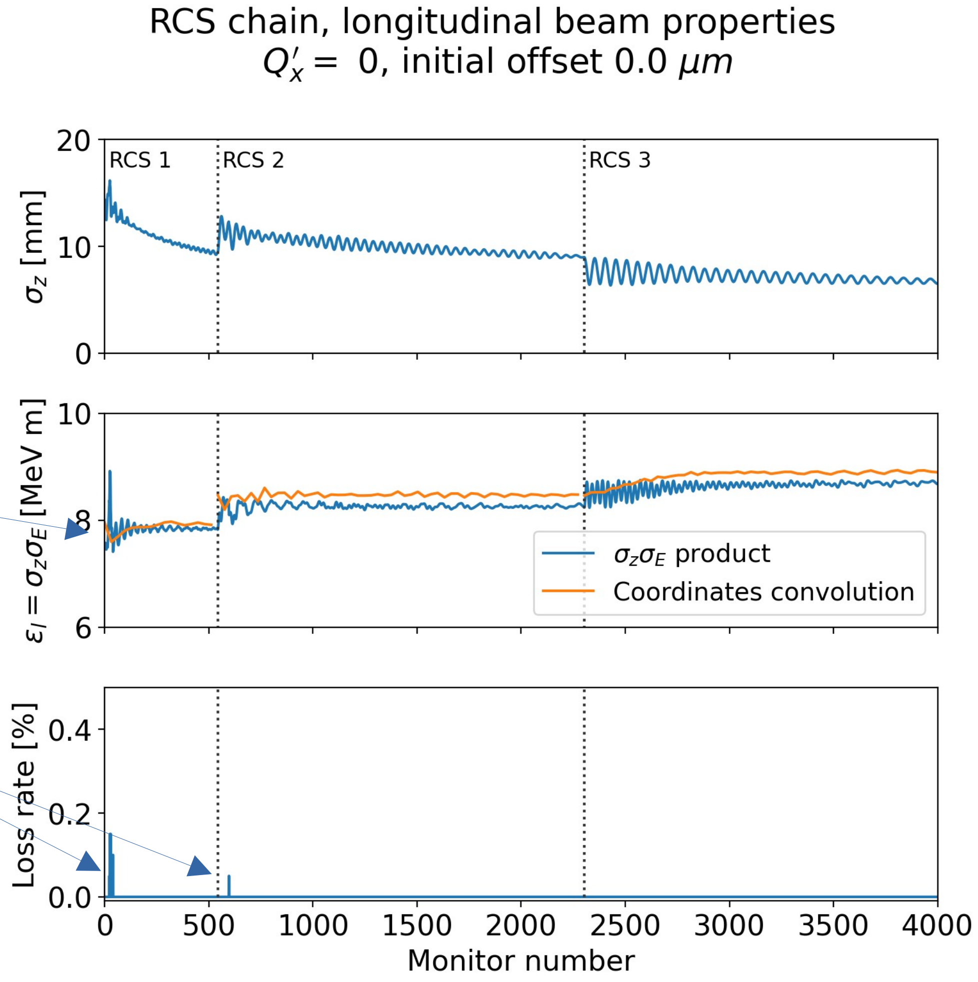
- Example of a simulation in RCS 1, RCS 2 and RCS 3 chain
  - 17/55/66 turns of acceleration in RCS1/2/3
  - 32 RF stations in each RCS
  - Chromaticity Q’ = 0, no impedance, no initial transverse offset
- There is a beam monitor at each RF station
  - Total of (17+55+66) * 32 = 4416 measurement points



Example of start-to-end simulations

Bunch is matched longitudinally at injection into RCS 1

Some particles are lost in the first turns after injection

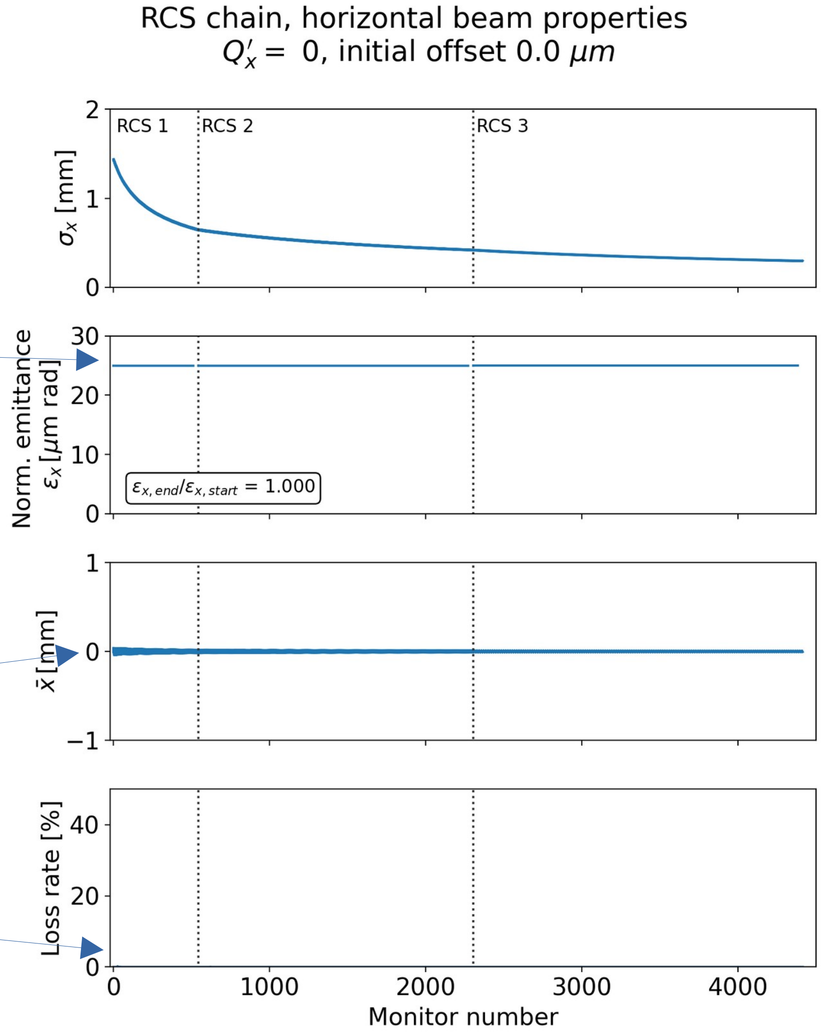


Example of start-to-end simulations

Bunch is matched transversely at injection into RCS 1

Bunch centroid motion is stable

The losses are only longitudinal (not visible on this scale)

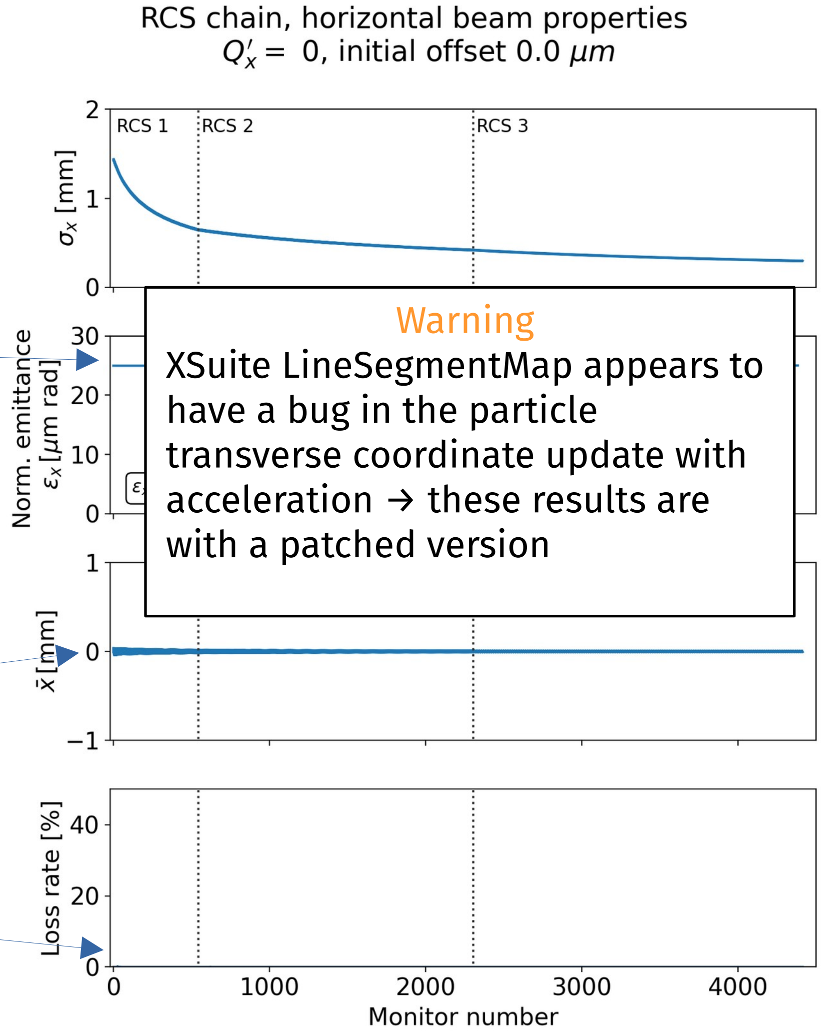


Example of start-to-end simulations

Bunch is matched transversely at injection into RCS 1

Bunch centroid motion is stable

The losses are only longitudinal (not visible on this scale)

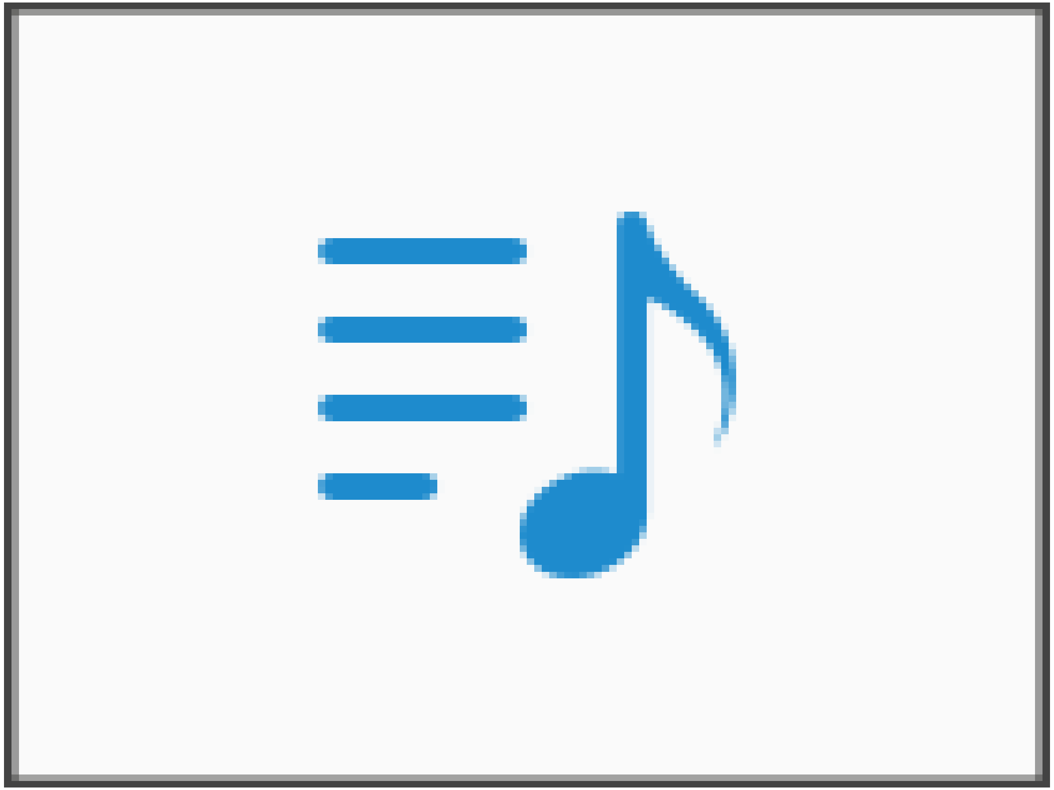


Warning

XSuite LineSegmentMap appears to have a bug in the particle transverse coordinate update with acceleration → these results are with a patched version

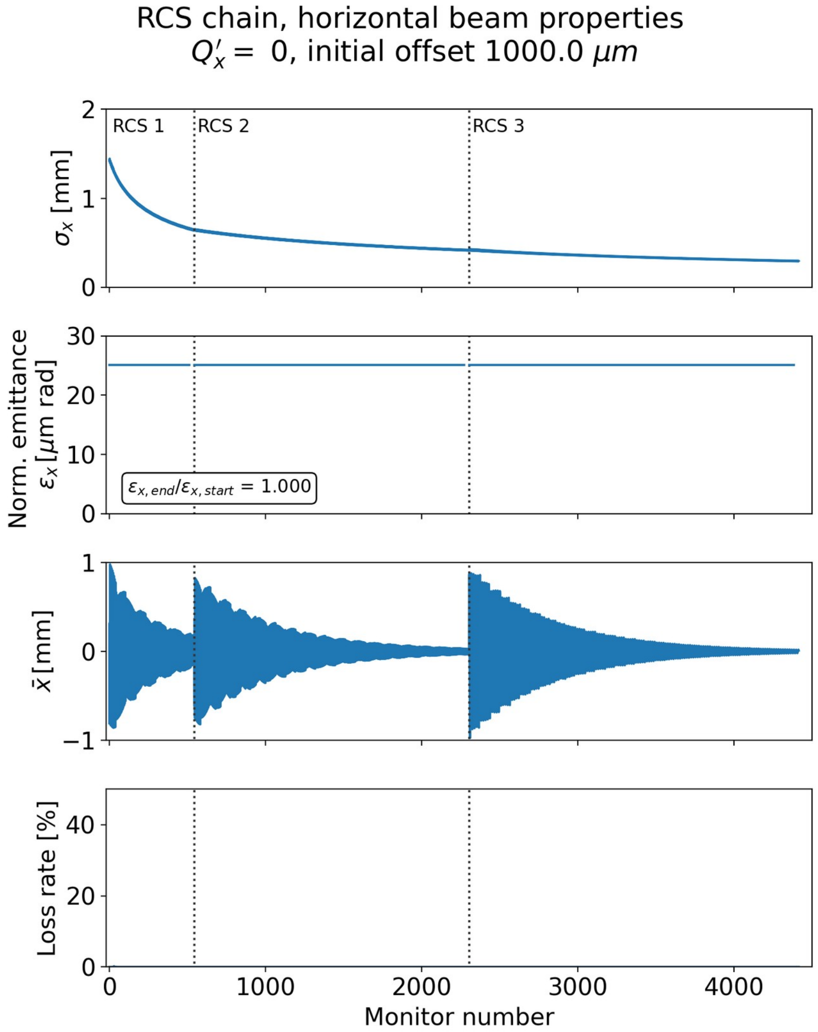
Example of start-to-end simulations

- Animation of the longitudinal phase space evolution over the three RCS



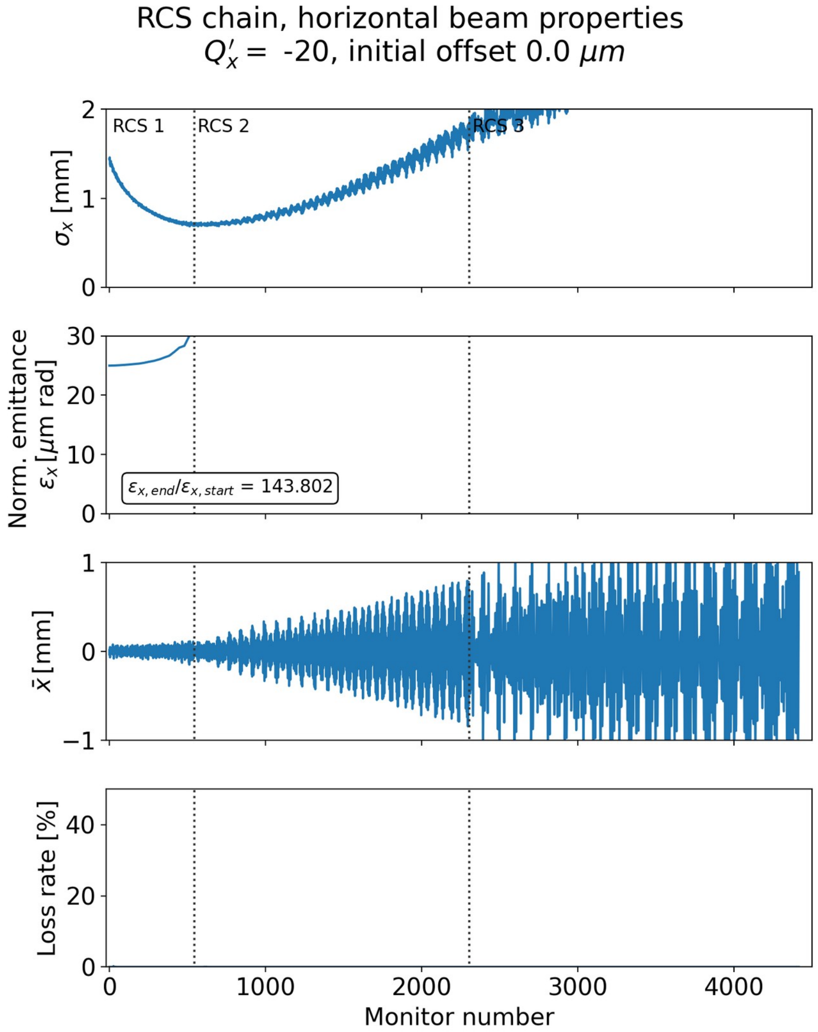
Example of start-to-end simulations

- Chromaticity Q’ = 0
- No impedance
- **Initial transverse offset = 1 mm at each machine injection**
- A **20-turn transverse damper** is included in each ring (at station #9)



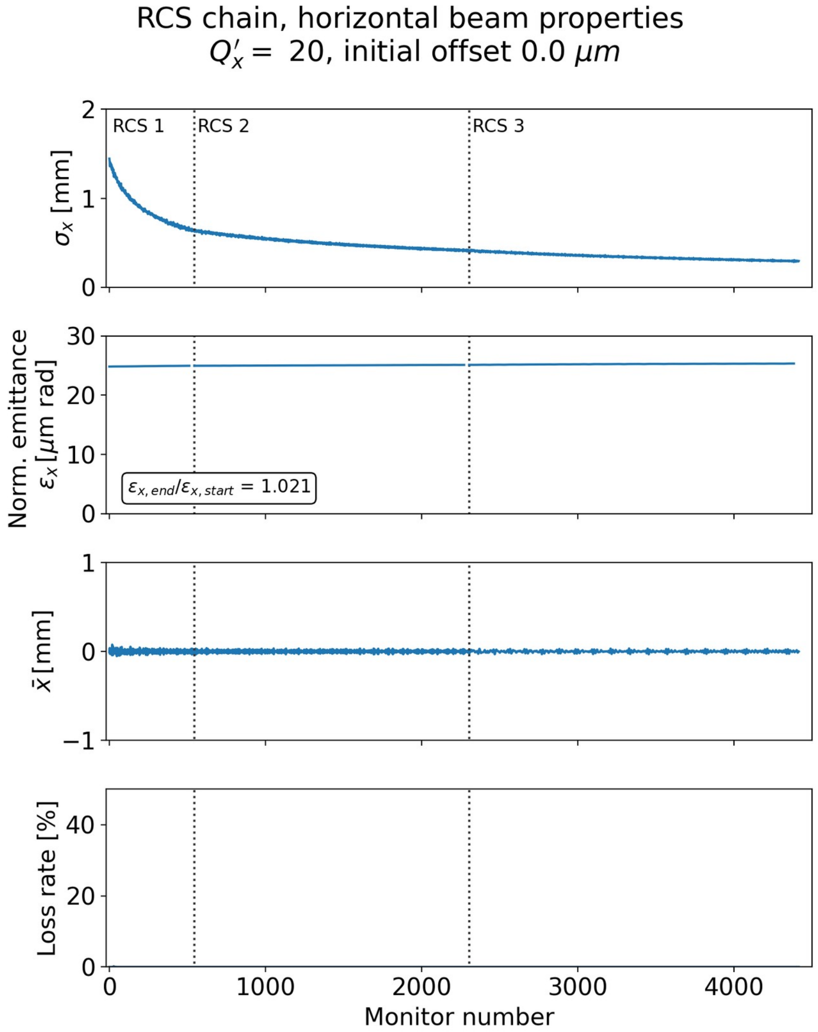
Example of start-to-end simulations

- Chromaticity **Q' = -20** (natural chromaticity)
- **TESLA cavities impedance model is included**
- No initial transverse offset
- A 20-turn transverse damper is included in each ring (at station #9)



Example of start-to-end simulations

- Chromaticity **Q' = +20**
- **TESLA cavities impedance model is included**
- No initial transverse offset
- A 20-turn transverse damper is included in each ring (at station #9)



Overview and next steps

International MUON Collider Collaboration

- The **start-to-end simulation in the RCS** chain is now laid-out with **XSuite**
  - Easy to change a machine parameter (config files), easy to add a RCS (RCS 4 for acceleration to 5 TeV)
  - Already some effects can be studied: impedance, transverse damper, chromaticity...
  - Effects implemented in XSuite can be added: beam-beam, detailed lattice, apertures...
- Codes are available on gitlab https://gitlab.cern.ch/muon-collider-bd
  - muc-impedance repository for the impedance and beam dynamics code
  - Muon Docker for the docker image developed by Erik to launch simulations on the batch syst

Overview and next steps

- Restart the **parametric studies**
  - Find the limits for the cavity and beam pipe impedance
  - Chromaticity, transverse damper, possibly Landau damping as mitigation measures for coherent instabilities
- The **acceleration model could be refined** to account for the real ramp function of the RCS
  - Possible interfacing with the rcsparameters class being developed

International MUON Collider Collaboration

Thank you!

Beam and machine parameters for the RCS 1

Beam parameters	Unit	Value
Synchrotron tune Q_s		1.8
Synchrotron period	turns	0.55
Bunch length 1σ	**mm**	**5.7**
Bunch intensity	**Particles per bunch**	**2.6e12**
ε_x / ε_y	**µm rad**	**25**
# of macropaticles		**50000**

Parameters from F. Batsch RCS tables

Machine parameters	Unit	Value
Circumference	m	5990
Beam momentum	GeV/c	63
Energy increase per turn	GeV	14.7
Rev. frequency	kHz	50
RF frequency	MHz	1300
Harmonic number		25957
RF voltage	GV	20.9
α_p		0.0024
Avg. beta x/y	m	50 / 50
Chromaticity Q'_x/Q'_y		0/ 0
Detuning from octupoles x/y	m^{-1}	0 / 0