BEAM PHYSICS DEPARTMENTS

How to use ISYN, a longitudinal tracking code in particle accelerator
Manual

Sandra Aumon, Paul Görgen
October 13, 2014

1 HOW TO USE ISYN

Due the frequent and very significant changes in the code, the syntax of the input files are varying. Therefore, the user manual and the how to will come later.

2 CODE COMPARISON WITH BEAM DYNAMICS: BENCHMARKS

This section is dedicated to compared Isyn simulation results to the longitudinal beam dynamics theory in very specific and well known scenarii used in accelerators. Most of the cases will be apply to the FAIR-SIS100 and also to the CERN PS machine. The following longitudinal beam simulations will be presented here, the aim being to compare the results of Isyn to the theory and observe the behavior of the tracking code:

1. The simple case of a stationary single harmonic bucket, without collective effect. (DONE)
2. A bunch rotation with a proton beam. (DONE)
3. A bunch rotation with a proton beam with longitudinal space charge. (DONE)
4. Acceleration through transition energy.
5. A stationary single harmonic bucket with longitudinal space charge.
6. Acceleration with longitudinal space charge through transition energy.

For each cases, an short description of the interface python main file will be given and explained as well as the theory used to compare with. Then the results of the simulations are shown. Remarks and discussions can be triggered if the special effects or code artefacts are found.

2.1 SINGLE HARMONIC STATIONARY BUCKET

The case of a simple single harmonic bucket has been simulated in Isyn. The particle coordinates have been generated within an elliptical distribution, and the simulation is done with the SIS-100 beam parameter around injection energy. The particles are tracked with one kick per turn for 50 000 turns, as a first step.

In this paragraph, the namely file Isyn_StationaryBucket_noSc.py is described. For this simulation, we need the following imports for Python,

```

from accelerator.Accelerator import Accelerator
from accelerator.Accelerator_Info import Accelerator_Info
from beamofparticle.Beam import Beam
from radiofrequency.RFcavity import RFcavity
import cProfile # Useful to monitor the time taking by each module of the code. Not
    mandatory.
```

First an Accelerator Python Object is created, as referred in Section 1.

```

#---------------- Declare the Object Accelerator ------------#

sis100 = Accelerator() # here my accelerator is sis100
sis100.setRadius(172.4603) # the machine radius
sis100.gammat = 8.9 # the transition gamma
sis100.bdot = 0.0 # the dB/dt , here zero since no acceleration
sis100.gammadot = 0.0

sis100.setParameterFilename('sis100') # any comment that you like for your file name in
    output
sis100.dumpDistributionEveryNturn(1000) # Particle coordinates written into a file every
     1000 turns
```

Then a Beam Python Object is created, as refered in Section 1.

```

# ---------------- Declare the Object Beam ------------#

beam = Beam() # I create a Beam Object beam
beam.setGamma(5.0) # with an gamma of 5
beam.setRestEnergy(938272046.0)
beam.setCharge(1.0) # Protons are used
beam.setNParticle(100000) # 100k macroparticles will be tracked
beam.parabolicDistribution(0.509, 0.001418, 0.0) # The distribution is chosen Parabolic.
```

This part prints some info about the simulation you created. The momentum compaction factor is then given, computed from the Objects beam and sis100.

```
# ---------------- Get the Object Accelerator_Info ------------#

info = Accelerator_Info(sis100, beam)
```

Then, a Python Object RFcavity is created to simulate the RF kick given to the particles each turn.

```
# ---------------- Declare the Object Rfcavity ------------#

rf = RFcavity(sis100)
rf.setHarmonic(10) # RF Harmonic
rf.setRFvoltage(60000.0) # the voltage in Volt
rf.setStablePhase(0.0) # the stable phase, here zero.
sis100.addElement(rf) # Add the RF cavity to the beam line.
```

Finally, the tracking can start, according to two possibilities. If there an interest to monitor the speed of the code, one can use cProfile in order to see how long each module is taking to perform the computations.

```
# ---------------- Tracking ------------#

cProfile.run('sis100.track(beam, 50000)','restats')
import pstats
p = pstats.Stats('restats')
p.sort_stats('cumulative').print_stats(10)
```

Or, without time monitoring,

```
# ---------------- Tracking ------------#

sis100.track(beam, 50000)
# Here start the tracking of 100k macroparticles over 50k turns
```

Later, the bunch profiles were damped and given to a program of tomography developed at CERN [2], here 33 frames of 512 points every 40 turns. One has to be careful to cut the empty bins, otherwise the tomogram does not converge. The longitudinal phase space is reconstructed in Fig. 2.3 and the beam parameters are retrieved such the match bunch area. The bunch intensity is not correct in Fig. 2.3.

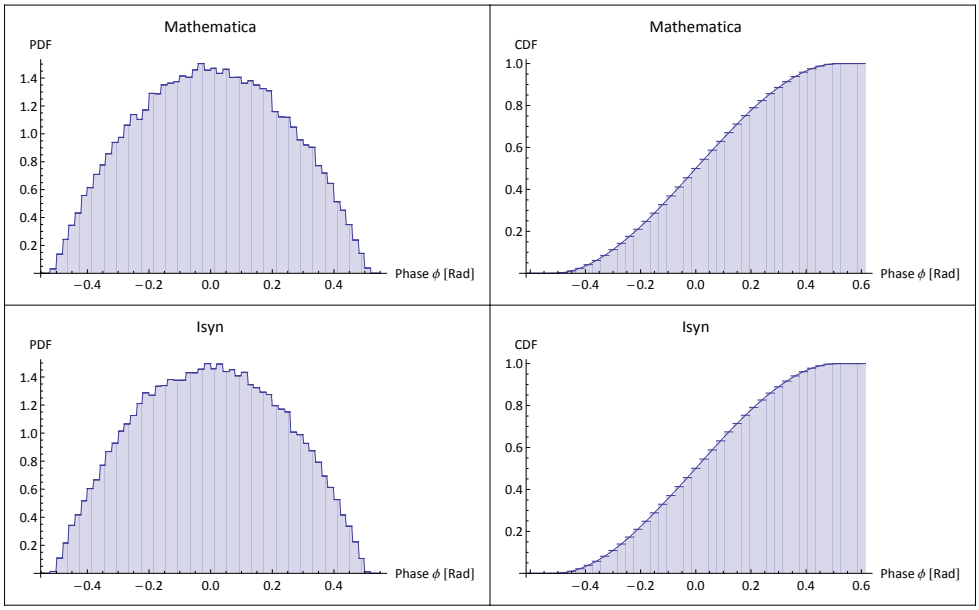

Figure 2.1: CDF .

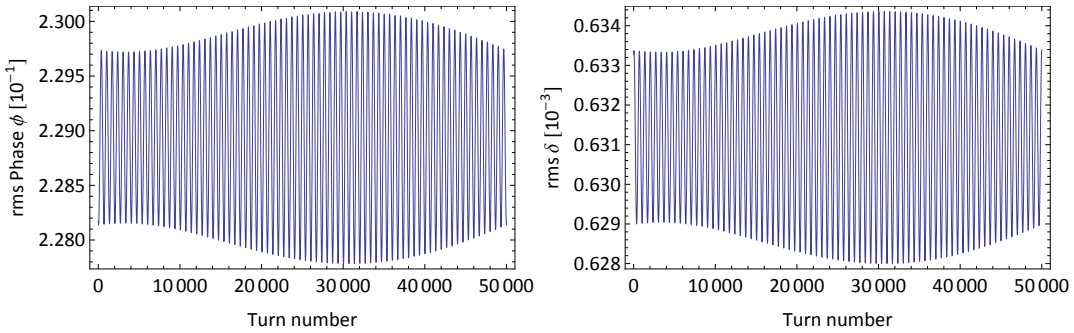

Figure 2.2: Bunch length and momentum spread

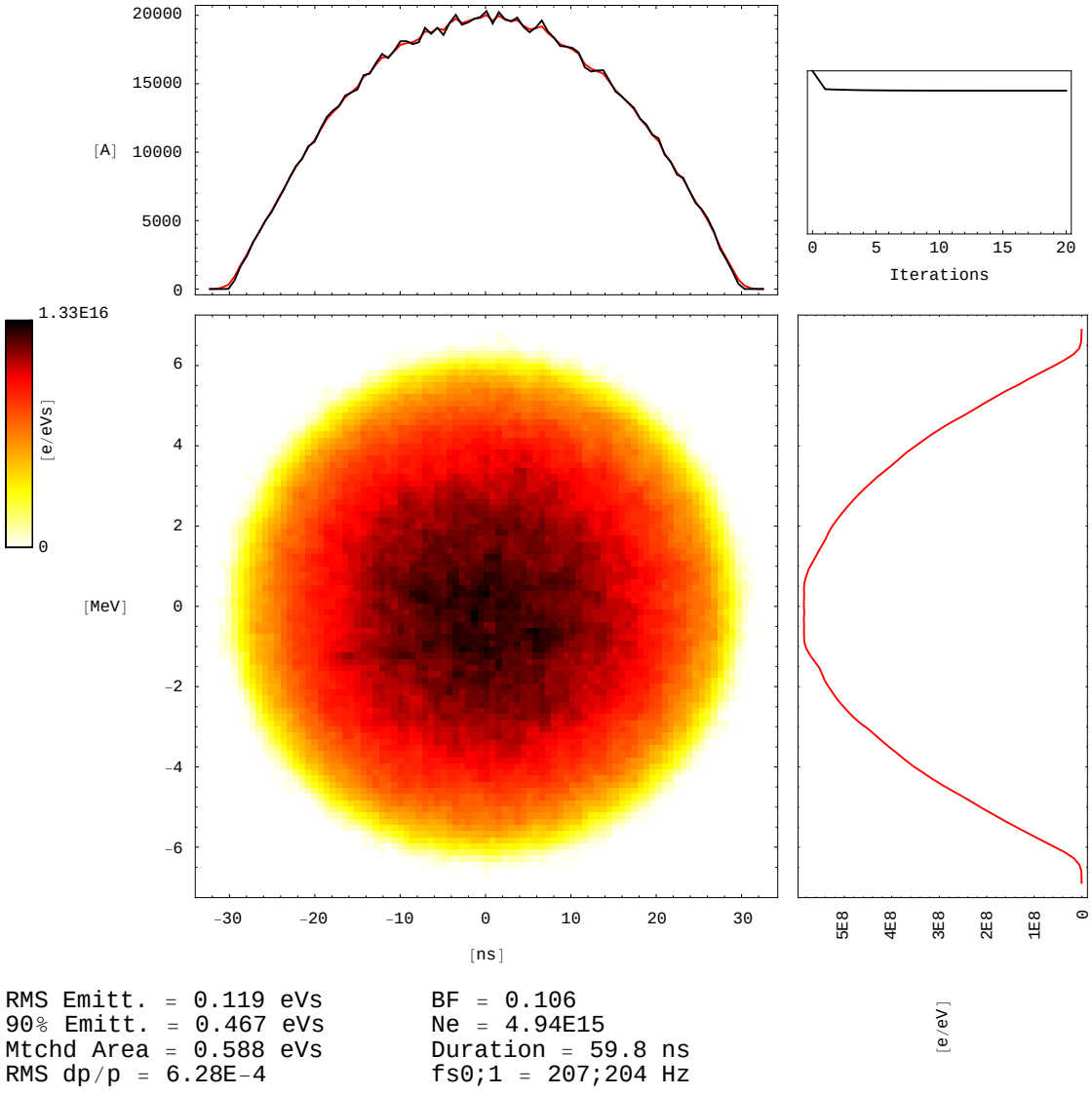

Figure 2.3: Results of the tomogram on the simulated data for the stationary bucket without longitudinal space charge.

In this example, a matched bi-parabolic distribution is generated and the voltage of the RF cavity object is ramped up, in a quasi-adiabatic way. The longitudinal profiles in phase are then observed and the resulting bunch length and momentum spread are compared to theoretical formulas. In this paragraph, the namely file Isyn_StatBucket_LinearRamp.py is described.

First an Accelerator Python Object is created, as referred in Section 1.

```
#---------------- Declare the Object Accelerator -------------#

sis100 = Accelerator()
sis100.setRadius(172.4603)
sis100.gammat = 8.9
sis100.bdot = 0.0

sis100.setParameterFilename('sis100')
sis100.dumpDistributionEveryNturn(500) # dump the distributions and profiles every 500
    turns.
```

Then, the Beam Object,

```
# ---------------- Declare the Object Beam -------------#

beam = Beam() # I create a Beam Object beam
beam.setGamma(5.0) # with an gamma of 5
beam.setRestEnergy(938272046.0)
beam.setCharge(1.0) # Protons are used
beam.setNParticle(200000) # 100k macroparticles will be tracked
beam.parabolicDistribution(0.509472, 6.31e6, 0.0) # The distribution is chosen Parabolic
    .
```

The ramp of the RF voltage should be set at the RF node. In this example, the ramp is linear. The user should define a Python function with *def* definition, according the following example. The ramp function should use a time variable. Then the name of the function, here *func* should be given to the command

```
rf.setRFvoltageFunction(function_name)
```

The *setRFvoltageFunction(functionname)* has to be use when a RF voltage ramp is simulated in Isyn. Finally, a RF node can be add to the lattice.

```
# ---------------- Declare the Object Rfcavity -------------#
# for linear ramp
Vi = 60000.0 # Initial voltage
Vf = Vi*2.0 # The final or second point voltage
duration = 0.005 # in second, duration of the ramp
linear_slope = (Vf-Vi)/duration
# Now we have V(t) = Vi + linear_slope * time
def func(time):  # The user can build a Python function, allowing to give every kind of
    RF ramp. The variable should the time.
    return  Vi + linear_slope * time

```

```
rf = RFcavity(sis100)
rf.setHarmonic(10)
rf.setRFvoltage(60000.0) # Not mandatory in the case of RF ramp.
rf.setStablePhase(0.0)

# Set a RF function
rf.setRFvoltageFunction(func)
sis100.addElement(rf)
```

The command *setRFvoltageFunction(functionname)* is declared in RFcavity.py Object

```
def setRFvoltageFunction(self, func): # func is a voltage as a function of time defined
    by the user
    self.RF_Ramp = func  # RF_Ramp is a property of the Obect RF cavity.
```

```
    @property
    def rampVoltage(self):
        return self.RF_Ramp(self.accObject.time)
```

The half bunch length and energy spread are retrieved from the simulation and compared to the following theoretical formulas,

$$\Delta E = \beta^2 E_{tot} \sqrt{\epsilon_l} \sqrt{\frac{\omega_0}{\pi \beta E_{tot}}} \left(\frac{heV \cos\phi_s}{2\pi\beta^2 E_{tot} |\eta|} \right)^{1/4} \tag{2.1}$$

and the bunch length in phase

$$\Delta\phi = h\sqrt{\epsilon_l} \sqrt{\frac{\omega_0}{\pi \beta E_{tot}}} \left(\frac{2\pi\beta^2 E_{tot} |\eta|}{heV \cos\phi_s} \right)^{1/4} \tag{2.2}$$

Eq. 2.2 and Eq. 2.1 are compared to the simulations and the results are presented in Figs. 2.4, which show a very good agreement.

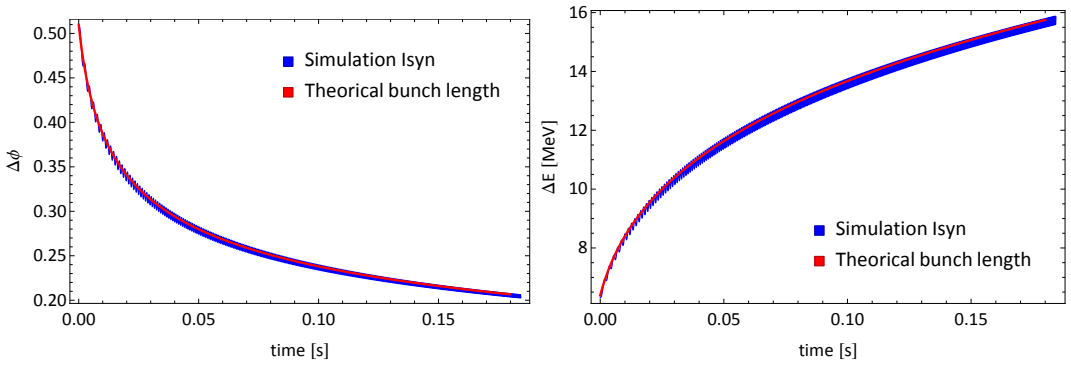

Figure 2.4: Bunch length and energy spread from Isyn compared to the analytical formulas Eq. 2.2 and Eq. 2.1.

The longitudinal profiles are obtained by using the interpolation of the particles to a grid and are dumped every 500 turns, as presented in Fig.2.5

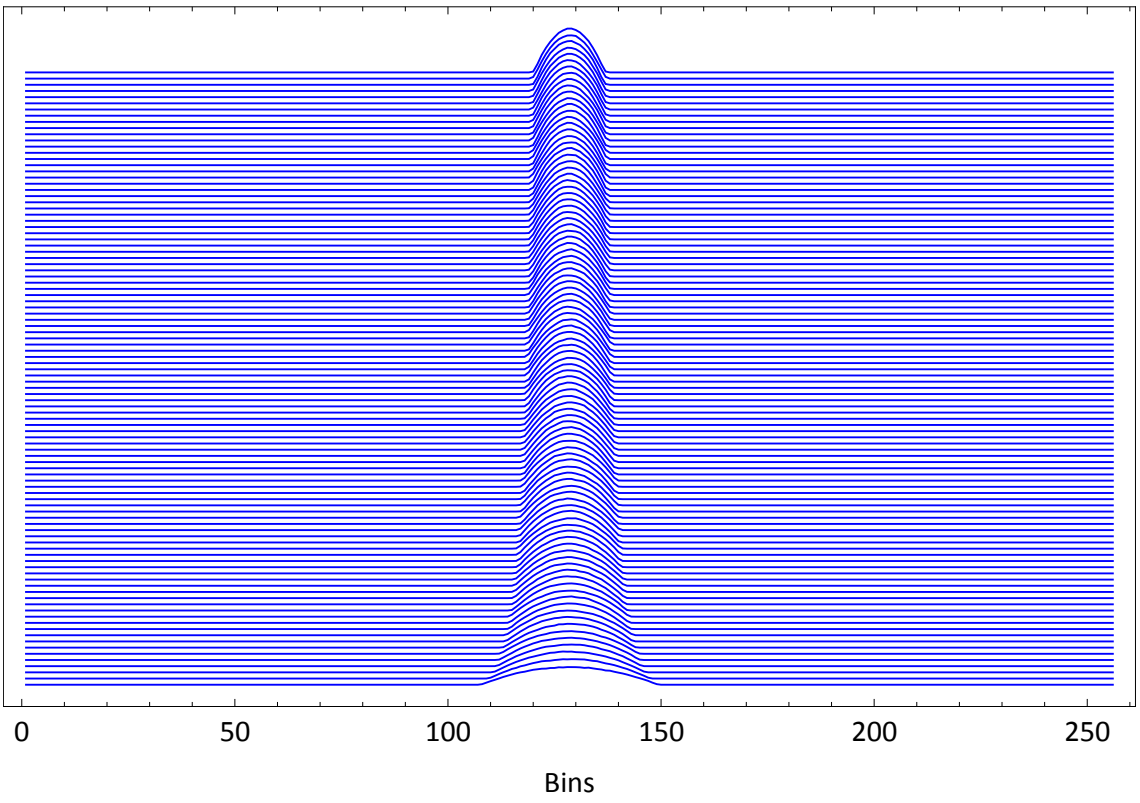

Figure 2.5: Longitudinal profiles dumped every 500 turns.

2.3 BUNCH ROTATION WITHOUT LONGITUDINAL SPACE CHARGE

This example focuses on an example of bunch compression simulation with Isyn. The resulting bunch length and momentum spread are compared to those predicted by the longitudinal envelop equation.

The bunch rotation (or compression) consists of a rotation of the beam in the longitudinal phase space while the RF cavities voltages are suddenly raised. The beam is mismatched in purpose with respect to the bucket and rotates in the phase space. After a quarter of a synchrotron period, the bunch length is very short (and consequently the momentum spread large), which make the beam properties interesting for experiments on target, due to the short pulse length.

This example is performed without longitudinal space charge, which is the subject of the next example. The beam is supposed to bi-parabolic, as hypothesizes in the derivation of the envelop equation. The beam parameters are taken from the SIS-100, however this example is not foreseen for the FAIR project. Therefore, the choice of the beam energy, the RF conditions, were used as purely pragmatic example to cross check the code with envelop equation and illustrate later the effect of longitudinal space charge in the next example.

2.3.1 INPUT FILE FOR BUNCH ROTATION

```
import os
import sys
from accelerator.Accelerator import Accelerator
from accelerator.Accelerator_Info import Accelerator_Info
from beamofparticle.Beam import Beam
from radiofrequency.RFcavity import RFcavity
import cProfile

#---------------- Declare the Object Accelerator -------------#

sis100 = Accelerator()
sis100.setRadius(172.4603)
sis.setRadiusOfCurvature(52.3)
sis100.gammat = 8.9
sis100.setBdot(0.0)

sis100.setParameterFilename('sis100')
sis100.dumpDistributionEveryNturn(13)

# ---------------- Declare the Object Beam -------------#

beam = Beam()
beam.setGamma(5.0)
beam.setRestEnergy(938272046.0)
beam.setCharge(1.0)
beam.setNParticle(100000)
beam.parabolicDistribution(0.509472, 6.31e6, 0.0)

# ---------------- Get the Object Accelerator_Info -------------#

```

```
info = Accelerator_Info(sis100, beam)

# ---------------- Declare the Object Rfcavity -------------#

rf = RFcavity(sis100)
rf.setHarmonic(10)
rf.setRFvoltage(600000.0)
sis100.addElement(rf)

# ---------------- Tracking -------------#

cProfile.run('sis100.track(beam, 200)','restats')
# Again those commands are not necessary. Only useful if you want to check the
    performances of the code.
import pstats
p = pstats.Stats('restats')
p.sort_stats('cumulative').print_stats(10)
```

2.3.2 COMPARISON TO THE LONGITUDINAL ENVELOP EQUATION

The longitudinal envelop equation used here can be also found in the Ref. [3, 4]. The max bunch length in meter follows the following differential equation,

$$z_m'' + k_0^2 z_m - \frac{K_l}{z_m^2} - \frac{\epsilon_L^2}{z_m^3} = 0 \tag{2.3}$$

with $k_0^2 = eZh\eta/(2\pi R^2\gamma\beta^2 Amc^2)$ the linearized RF force, the longitudinal perveance $K_l = 3g_0N(Z^2/A)r_0\eta/(2\beta^2\gamma^3)$, with g factor, $\epsilon_L = |\eta|z_m(\delta p/p_0)_0$, see Fig. 2.6. In this simulation, no intensity effect is taken into account, $K_l = 0$.

The results in maximum bunch length and momentum spread from the simulation are compared to the longitudinal envelop equation 2.3 in Fig 2.7: the outputs of Isyn agree very well with the dynamics predicted by the envelop equation.

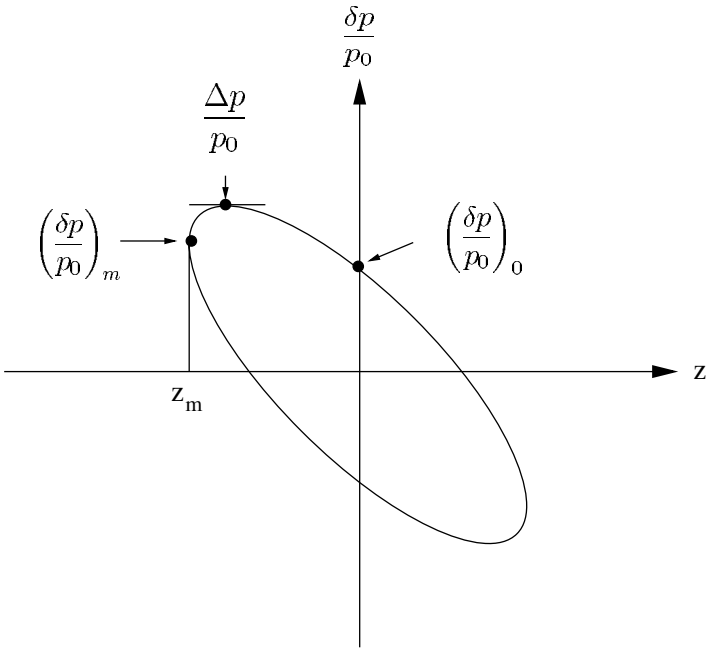

Figure 2.6: Definition of parameters for a rotated ellipse in the fast compression scheme.

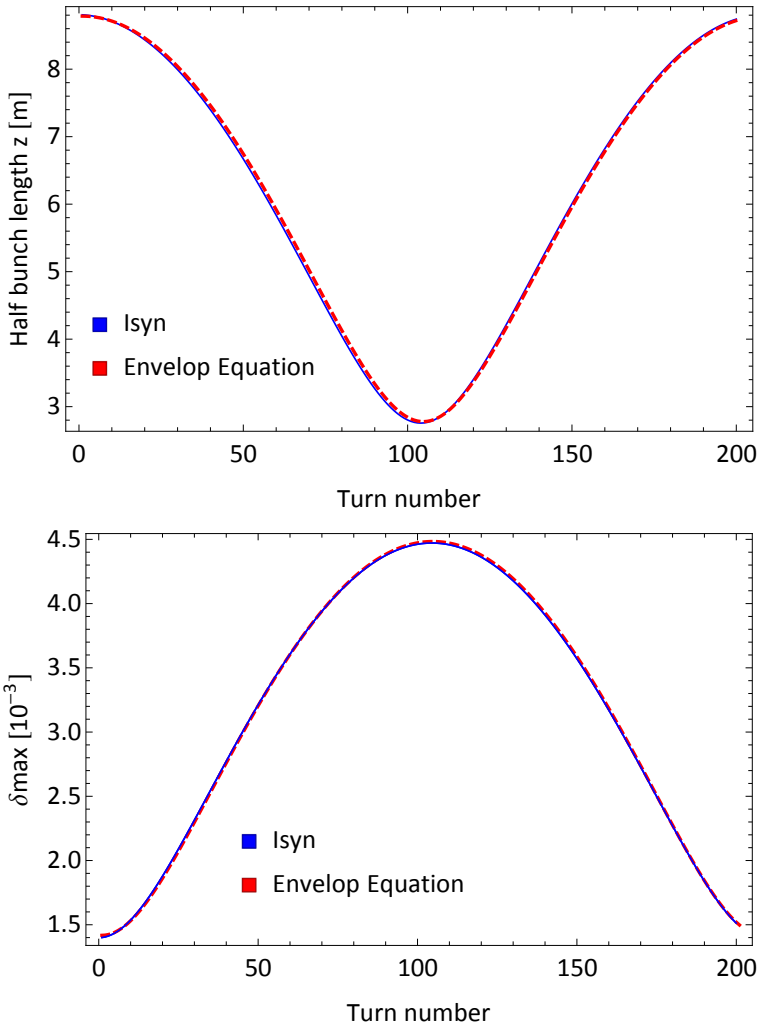

Figure 2.7: Bunch length and momentum spread

2.4 ACCELERATION OF PROTON BEAM THROUGH TRANSITION ENERGY WITH LONGITUDINAL SPACE CHARGE

This example presents a simulation of an intense proton beam in SIS-100 during an acceleration through transition energy with longitudinal space charge. The result in bunch length from Isyn is compared to the longitudinal envelop equation derived by Ng in [1], written as,

$$\frac{d}{dx}\left(\frac{1}{x}\frac{d\hat{\Delta\phi}}{dx}\right) + sgn(x)\hat{\Delta\phi} + \frac{\kappa_{sc}^{l}}{\hat{\Delta\phi}^2} - x\frac{(Sn/\pi)^2}{\hat{\Delta\phi}^3} = 0 \tag{2.4}$$

with $x = t/T_c$, t is time with respect to transition energy, T_c is the nonadiabatic time written as

$$T_c = \left(\frac{\beta^2 E_0 \gamma_t^4}{4\pi f_0^2 \dot{\gamma} h e V_{rf} |\cos\phi_s|}\right)^{1/3} \tag{2.5}$$

with e the charge of the particle, V_{rf} the total voltage, h the RF harmonic number, $\dot{\gamma}$ the change of γ with time, i.e. $d\gamma/dt$, f_0 the revolution frequency. Let consider the time t with $t = 0$ being the transition energy. Then,

$$S_n = \frac{2h^2\omega_0^2\dot{\gamma}T_c^2}{\beta^2\gamma^4 E_{rest}} S \tag{2.6}$$

S is the bunch area, and

$$\eta_{sc} = \frac{\kappa_{sc}^{l}}{\hat{\Delta\phi}^3} \quad \text{and} \quad \kappa_{sc}^{l} = \frac{3\pi N_b r_p E_{rest} g_0 h^2 Sgn(\eta)}{R\gamma^2 eV|\cos\phi_s|} \tag{2.7}$$

2.4.1 INPUT FILE

The distribution of the beam comes from a previous simulation where the RF voltage and Bdot are ramped up to the desired values, in order to start from a stationary matched beam to the bucket with longitudinal space charge to end up in a accelerated bucket. The object Beam is then damped and reloaded in the following input file. The beam should be matched to the bucket and is accelerated through transition energy for SIS-100 beam parameters.

```
import os
import sys
from accelerator.Accelerator import Accelerator
from accelerator.Accelerator_Info import Accelerator_Info
from beamofparticle.Beam import Beam
from radiofrequency.RFcavity import RFcavity
from lspace_charge.space_charge import spaceChargeNode
import cProfile
import pickle
import copy

#---------------- Declare the Object Accelerator ------------#

sis100 = Accelerator()
sis100.setRadius(172.4603)
sis100.setRadiusOfCurvature(52.3)
```

```
sis100.gammat = 8.9
sis100.setBdot(4.0)

sis100.setParameterFilename('sis100_AccPicklelSc')
sis100.dumpDistributionEveryNturn(5000)

# ---------------- Declare the Object Beam ------------#
mybeam = pickle.load(open("SaveBeamLsc.p","rb"))

beam = Beam()
beam.setNParticle(1000000)
beam = copy.deepcopy(mybeam)
beam.setGamma(mybeam.getGamma())
#beam.setRestEnergy(938272046.0)
beam.setCharge(1.0)

print beam.getGamma()
print beam.getRestEnergy()
print beam.momentum()

# ---------------- Get the Object Accelerator_Info ------------#

info = Accelerator_Info(sis100, beam)

# ---------------- Declare the Object Rfcavity ------------#

rf = RFcavity(sis100)
rf.setHarmonic(10)
rf.setRFvoltage(280000.0)
sis100.addElement(rf)

# ---------------- Longitudinal space charge node ------------#

sc_node = spaceChargeNode(sis100)
sis100.addElement(sc_node)
sc_node.setNumberParticleinBunch(500.0e10)
#sc_node.setMacroparticle(100000)
sc_node.setNumberBins(128)
sc_node.setRatio(5)
sc_node.phi2z(rf.getHarmonic()) # This is temporary
sc_node.setStablePhase(0.0) # temporary

# ---------------- Tracking ------------#

cProfile.run('sis100.track(beam, 15000)','restats')
import pstats
p = pstats.Stats('restats')
p.sort_stats('cumulative').print_stats(10)
```

2.4.2 RESULTS

Fig. 2.8 and Fig. 2.9 present the results of the simulations compared to the theory 2.4, again the distribution is supposed bi-parabolic.

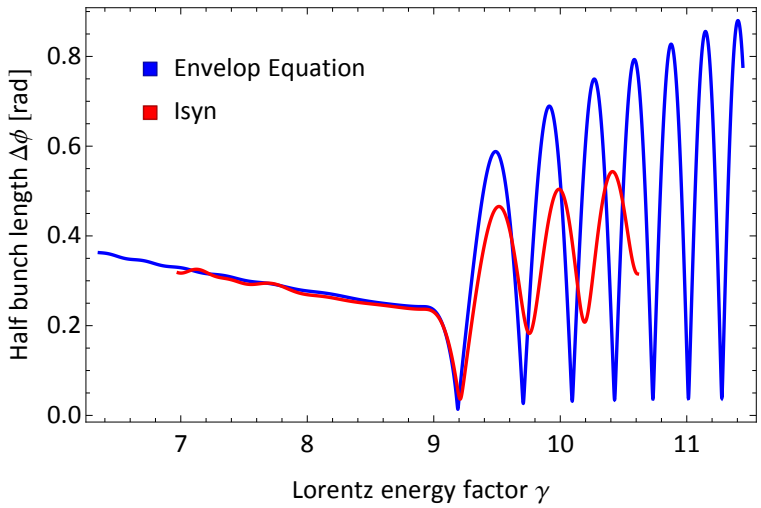

Figure 2.8: Half bunch length through transition energy.

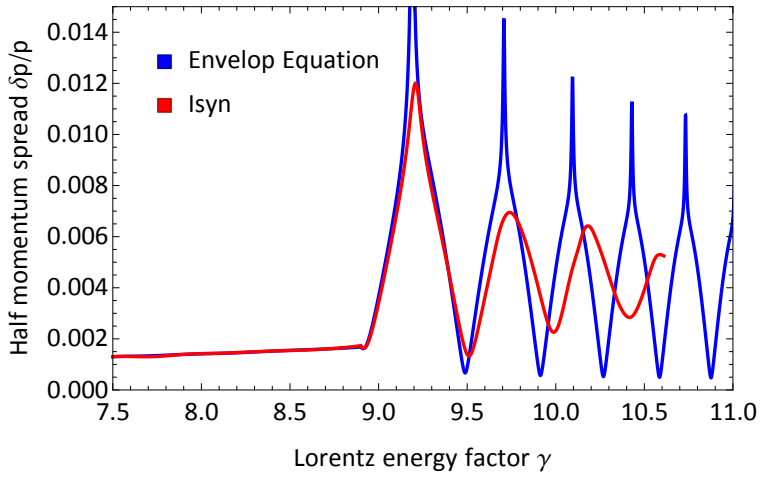

Figure 2.9: Half momentum spread through transition energy.

REFERENCES

[1] K.Y. Ng, *Physics of Intensity Dependant Beam Instabilities*, World Scientific Publishing 1st ed., 2006.

[2] S. Hancock, P. Knaus, M. Lindroos, Tomographic measurements of longitudinal phase space density, European Particle Accelerator Conference, Stockholm, Sweden, 22 - 26 Jun 1998. Publ. in: Proceedings, (CERN-PS-98-030-RF).

[3] Franchetti et al., Effect of space charge on bunch compression near the transition, Physical Review Special Topics Accelerators and Beams, Volume 3, 084201 (2000).

[4] M. Reiser, Theory and Design of Charged Particle Beams, Wiley.