Collider Time Evolution

Tom Mertens

June 7, 2017

Table of contents

Table of Contents

Motivation

- Original version (Michael Blaskiewicz) : stochastic cooling RHIC

Motivation

- Original version (Michael Blaskiewicz) : stochastic cooling RHIC
- Evolution of bunch parameters - protons and ions (R. Bruce, M. Schaumann)

Motivation

- Original version (Michael Blaskiewicz) : stochastic cooling RHIC
- Evolution of bunch parameters - protons and ions (R. Bruce, M. Schaumann)
- Luminosity prediction

Motivation

- Original version (Michael Blaskiewicz) : stochastic cooling RHIC
- Evolution of bunch parameters - protons and ions (R. Bruce, M. Schaumann)
- Luminosity prediction
- Stochastic cooling - lumi leveling (M. Schaumann)

Motivation

- Original version (Michael Blaskiewicz) : stochastic cooling RHIC
- Evolution of bunch parameters - protons and ions (R. Bruce, M. Schaumann)
- Luminosity prediction
- Stochastic cooling - lumi leveling (M. Schaumann)
- update : not possible to simulate proton-lead

Motivation

- Original version (Michael Blaskiewicz) : stochastic cooling RHIC
- Evolution of bunch parameters - protons and ions (R. Bruce, M. Schaumann)
- Luminosity prediction
- Stochastic cooling - lumi leveling (M. Schaumann)
- update : not possible to simulate proton-lead
- update : difficult to add new physics routines

Motivation

- Original version (Michael Blaskiewicz) : stochastic cooling RHIC
- Evolution of bunch parameters - protons and ions (R. Bruce, M. Schaumann)
- Luminosity prediction
- Stochastic cooling - lumi leveling (M. Schaumann)
- update : not possible to simulate proton-lead
- update : difficult to add new physics routines
- update : only two bunches

Highlights

- Allows for different particle species in each beam

Highlights

- Allows for different particle species in each beam
- Track evolution of multiple bunches (collision classes)

Highlights

- Allows for different particle species in each beam
- Track evolution of multiple bunches (collision classes)
- Colliding bunch pair luminosity evolution per IP

Highlights

- Allows for different particle species in each beam
- Track evolution of multiple bunches (collision classes)
- Colliding bunch pair luminosity evolution per IP
- Updated user interface

Highlights

- Allows for different particle species in each beam
- Track evolution of multiple bunches (collision classes)
- Colliding bunch pair luminosity evolution per IP
- Updated user interface
- More flexibility for adding new physics routines

Highlights

- Allows for different particle species in each beam
- Track evolution of multiple bunches (collision classes)
- Colliding bunch pair luminosity evolution per IP
- Updated user interface
- More flexibility for adding new physics routines
- Output flexibility, easier plotting

Internals

- Original code : FORTRAN

Internals

- Original code : FORTRAN
- Global variables in common block (separate file) $\rightarrow$ difficult

Internals

- Original code : FORTRAN
- Global variables in common block (separate file) $\rightarrow$ difficult
- Numerical routines in separate FORTRAN files, callable in Python (f2py)

```
subroutine sptransChrom(np,x,px,y,py,pt,psix,psiy,coeffchromx,coeffchromy,tpdqmin,k2L,k2Lskew)
! full turn update for x, horizontal
implicit none
integer, intent(in) :: np
!f2py intent(in) :: np
double precision, intent(inout), dimension(np) :: x,px,y,py,pt
!f2py intent(in,out) :: x,px,y,py,pt
double precision, intent(in) :: psix,psiy,coeffchromx,coeffchromy,k2L,k2Lskew,tpdqmin
!f2py intent(in) :: psix,psiy,coeffchromx,coeffchromy,k2L,k2Lskew,tpdqmin
double precision :: xk,pkx,xk1,pxk1,yk,pky,yk1,pyk1,psi1,ptk,a11,a12
integer :: k
```

Internals

- Original code : FORTRAN
- Global variables in common block (separate file) $\rightarrow$ difficult
- Numerical routines in separate FORTRAN files, callable in Python (f2py)
- FORTRAN (95) $\rightarrow$ install requires FORTRAN compiler

Internals

- Original code : FORTRAN
- Global variables in common block (separate file) $\rightarrow$ difficult
- Numerical routines in separate FORTRAN files, callable in Python (f2py)
- FORTRAN (95) $\rightarrow$ install requires FORTRAN compiler
- Python: improve user interface, sim flexibility, extendability and output

Internals

- Original code : FORTRAN
- Global variables in common block (separate file) $\rightarrow$ difficult
- Numerical routines in separate FORTRAN files, callable in Python (f2py)
- FORTRAN (95) $\rightarrow$ install requires FORTRAN compiler
- Python: improve user interface, sim flexibility, extendability and output
- Python decorators for checking and transforming input/output (keeps core of functions simple - re-usability of checks and transformations - updating sim logs)

Internals

- Original code : FORTRAN
- Global variables in common block (separate file) $\rightarrow$ difficult
- Numerical routines in separate FORTRAN files, callable in Python (f2py)
- FORTRAN (95) $\rightarrow$ install requires FORTRAN compiler
- Python: improve user interface, sim flexibility, extendability and output
- Python decorators for checking and transforming input/output (keeps core of functions simple - re-usability of checks and transformations - updating sim logs)
- Unit tests in Python

Internals

- Original code : FORTRAN
- Global variables in common block (separate file) $\rightarrow$ difficult
- Numerical routines in separate FORTRAN files, callable in Python (f2py)
- FORTRAN (95) $\rightarrow$ install requires FORTRAN compiler
- Python: improve user interface, sim flexibility, extendability and output
- Python decorators for checking and transforming input/output (keeps core of functions simple - re-usability of checks and transformations - updating sim logs)
- Unit tests in Python
- PEP8 standard (at least tried)

Internals

- Original code : FORTRAN
- Global variables in common block (separate file) $\rightarrow$ difficult
- Numerical routines in separate FORTRAN files, callable in Python (f2py)
- FORTRAN (95) $\rightarrow$ install requires FORTRAN compiler
- Python: improve user interface, sim flexibility, extendability and output
- Python decorators for checking and transforming input/output (keeps core of functions simple - re-usability of checks and transformations - updating sim logs)
- Unit tests in Python
- PEP8 standard (at least tried)
- Roughly 3500 lines of FORTRAN and 3500 lines of Python

Table of Contents

Physics: general

- Slow effects (hours)

Physics: general

- Slow effects (hours)
- 1 simulation turn is multiple (20000) real machine turns

Physics: general

- Slow effects (hours)
- 1 simulation turn is multiple (20000) real machine turns
- betatron motion

Physics: general

- Slow effects (hours)
- 1 simulation turn is multiple (20000) real machine turns
- betatron motion
- synchrotron motion (double RF system possible)

Physics: general

- Slow effects (hours)
- 1 simulation turn is multiple (20000) real machine turns
- betatron motion
- synchrotron motion (double RF system possible)
- radiation damping

Physics: general

- Slow effects (hours)
- 1 simulation turn is multiple (20000) real machine turns
- betatron motion
- synchrotron motion (double RF system possible)
- radiation damping
- intra-beam scattering (IBS)

Physics: general

- Slow effects (hours)
- 1 simulation turn is multiple (20000) real machine turns
- betatron motion
- synchrotron motion (double RF system possible)
- radiation damping
- intra-beam scattering (IBS)
- collisions with possibility of lumi leveling (by changing beta)

- Slow effects (hours)
- 1 simulation turn is multiple (20000) real machine turns
- betatron motion
- synchrotron motion (double RF system possible)
- radiation damping
- intra-beam scattering (IBS)
- collisions with possibility of lumi leveling (by changing beta)
- collimation

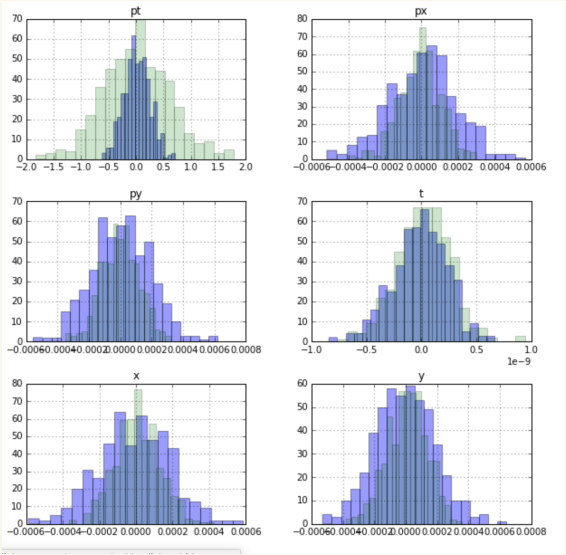
- Slow effects (hours)
- 1 simulation turn is multiple (20000) real machine turns
- betatron motion
- synchrotron motion (double RF system possible)
- radiation damping
- intra-beam scattering (IBS)
- collisions with possibility of lumi leveling (by changing beta)
- collimation
- TODO: stochastic cooling

Transverse starting distribution : bi-Gaussian ($\sigma = \sqrt{\epsilon\beta}$)

```
! rotate in y-py plane
         ptk   = pt(k)
         psi1  = psiy + ptk * coeffchromy
         a11   = cos(psi1)
         a12   = sin(psi1)
         yk    = y(k)
         pky   = py(k)
         yk1   = yk * a11 + pky * a12
         pyk1  = pky * a11 - yk * a12
         y(k) = yk1

! rotate in x-px plane
         psi1 = psix + ptk*coeffchromx
         a11   = cos(psi1)
         a12   = sin(psi1)
         xk    = x(k)
         pkx   = px(k)
         xk1   = xk * a11 + pkx * a12
         pxk1  = pkx * a11 - xk * a12
         x(k) = xk1

! now have dqmin part - coupling between x and y
         px(k) = pxk1 + tpdqmin * yk1
         py(k) = pyk1 + tpdqmin * xk1

! thin sextupole kick
         px(k)= px(k) + 0.5 * k2L * (xk1**2 - yk1**2) - k2Lskew * (xk1 * yk1)
         py(k)= py(k) + k2L * (xk1 * yk1) + 0.5 * k2Lskew * (xk1**2 - yk1**2)
```

Longitudinal starting distribution:

- smoke ring distribution
- from file
- bi-Gaussian
- pseudo-Gaussian



Radiation damping

Radiation integral calculations:

- smooth approximation using ring averages
- lattic : calculate for each element and sum over ring
- manual : lifetimes and equilibrium sizes are given manually

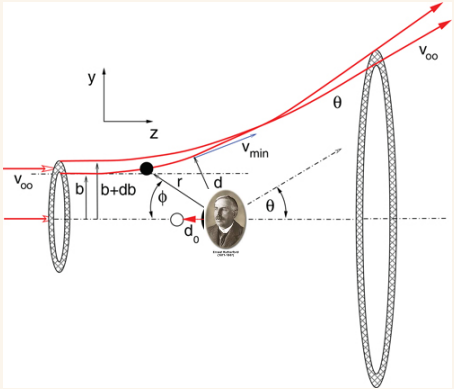
Radiation damping code

```
@check_set_particle_atomicn(fn_log_file)
@check_set_particle_charge(fn_log_file)
@check_set_particle_gamma(fn_log_file)
@check_set_raddamp_method(fn_log_file)
@check_set_raddamp_switch(fn_log_file)
def init_radiation_damping(dict_sim, df_twiss_data, trev, part_charge_key, part_atom_key, part_gamma_key, part_mass_key,
                           betax_avg, betay_avg):
    """
    Returns radiation lifetimes and equilibrium sizes
    :param dict_sim: simulation settings dictionary
    :param df_twiss_data: twiss data
    :param trev: particle revolution time
    :param part_charge_key: particle charge dict key
    :param part_atom_key: particle atom number dict key
    :param part_gamma_key: particle gamma dict key
    :param part_mass_key: particle mass key
    :param betax_avg: ring average betax
    :param betay_avg: ring average bety
    :return: tradperp, tradlong, siglong, sigperp
    """
    rIon1 = (float(dict_sim[part_charge_key]) ** 2 / float(dict_sim[part_atom_key])) * 1.54e-18
    # C_alpha*E^3/C in handbook formulas p 221 eq 11
    CalphaE3C = rIon1 * float(dict_sim[part_gamma_key]) ** 3 / (3 * trev)

    if dict_sim['phys_raddamp_method'] == 'approx':
        I2, I3, I4x, I4y, I5x, I5y = get_raddamping_integrals_approx(dict_sim, betax_avg, betay_avg)
        tradperp, tradlong, siglong, sigperp = get_raddamping_lifetimes_equi_sizes_from_integrals(dict_sim, I2, I3, I4x,
                                                                                                  I4y, I5x, CalphaE3C,
                                                                                                  part_mass_key,
                                                                                                  part_gamma_key)

    elif dict_sim['phys_raddamp_method'] == 'lattic':
        I2, I3, I4x, I4y, I5x, I5y = get_raddamping_integrals_lattic(dict_sim, df_twiss_data)
        tradperp, tradlong, siglong, sigperp = get_raddamping_lifetimes_equi_sizes_from_integrals(dict_sim, I2, I3, I4x,
                                                                                                  I4y, I5x, CalphaE3C,
```

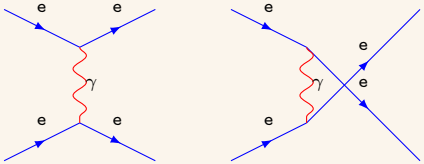
IBS

- CTE software has several IBS models (FORTRAN)

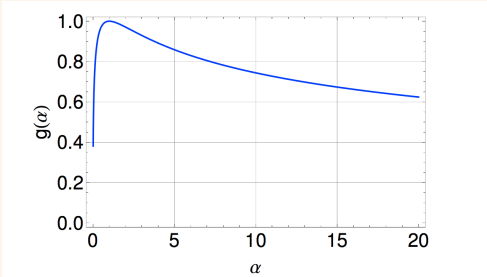
- CTE software has several IBS models (FORTRAN)
  - Piwinski (smooth, lattice, modified - using D')



- CTE software has several IBS models (FORTRAN)
  - Piwinski (smooth, lattice, modified - using D')
  - Bjorken-Mtingwa - Tri-linear interpolation (MAD-X)



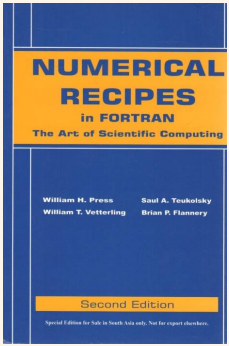
- CTE software has several IBS models (FORTRAN)
  - Piwinski (smooth, lattice, modified - using D')
  - Bjorken-Mtingwa - Tri-linear interpolation (MAD-X)
  - Bane - high energy



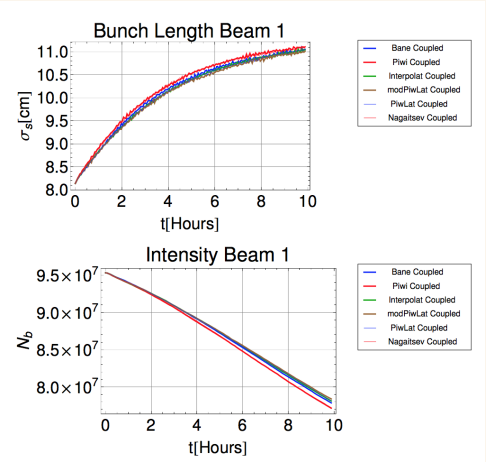
- CTE software has several IBS models (FORTRAN)
  - Piwinski (smooth, lattice, modified - using D')
  - Bjorken-Mtingwa - Tri-linear interpolation (MAD-X)
  - Bane - high energy
  - Nagaitsev - Carlson's Elliptical Integrals

$$R_D(x,y,z) = \frac{3}{2}\int_0^\infty \frac{dt}{\sqrt{(t+x)(t+y)(t+z)^3}} \tag{1}$$

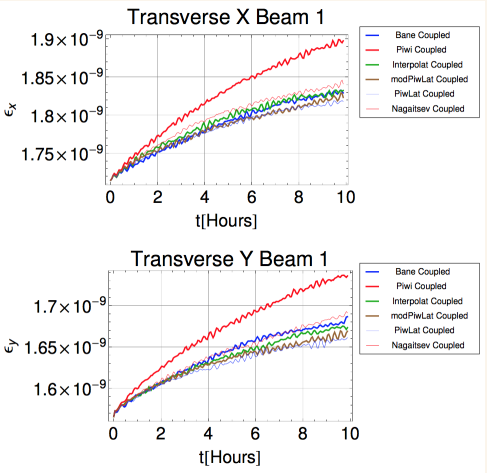
$$R_D(x,x,x) = x^{-\frac{3}{2}} \tag{2}$$



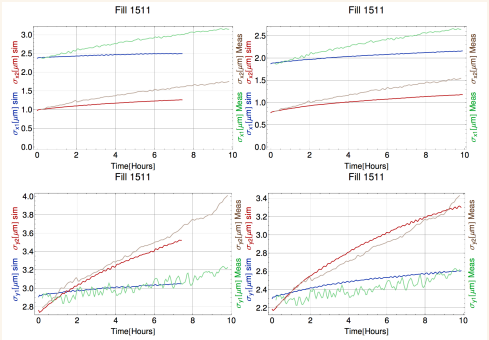
- CTE software has several IBS models (FORTRAN)
  - Piwinski (smooth, lattice, modified - using D')
  - Bjorken-Mtingwa - Tri-linear interpolation (MAD-X)
  - Bane - high energy
  - Nagaitsev - Carlson's Elliptical Integrals
- How do the IBS models compare?

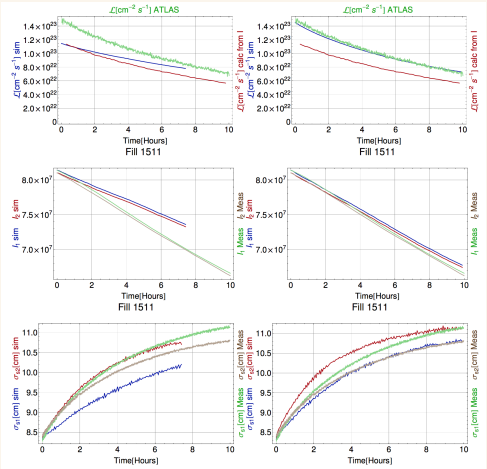


- CTE software has several IBS models (FORTRAN)
  - Piwinski (smooth, lattice, modified - using D')
  - Bjorken-Mtingwa - Tri-linear interpolation (MAD-X)
  - Bane - high energy
  - Nagaitsev - Carlson's Elliptical Integrals
- How do the IBS models compare?



IBS


Fill 1511
Fill 1511
Fill 1511
Fill 1511
Time[Hours]
Time[Hours]
Time[Hours]
Time[Hours]


ℒ[cm⁻² s⁻¹] ATLAS
ℒ[cm⁻² s⁻¹] ATLAS
Fill 1511
Fill 1511
Fill 1511
Fill 1511
Time[Hours]
Time[Hours]
Time[Hours]
Time[Hours]

Collisions

Two different routines available:

- assuming Gaussian distribution
- using binning and no Gaussian assumption (slow)

CTEPY contains a function to determine order of collisions and collision locations (here LHC version).

```
def genlumikeys(b1,b2):
    tdc = OrderedDict()
    lumip1keys = []
    lumip2keys = []
    lumip5keys = []
    lumip8keys = []
    lumip1 = OrderedDict()
    lumip2 = OrderedDict()
    lumip5 = OrderedDict()
    lumip8 = OrderedDict()

    for i in range(3564*2):
        res = [None]*4
        bb1 = [(b-i/2.)%3564 for b in b1]
        bb2 = [(b-i/2.)%3564 for b in b2]
        for j in range(len(b1)):
            for k in range(len(b2)):
                if (bb1[j]==0) and (bb2[k]==0):
                    res[0]=[j,k]
                if (bb1[j]==3564-445.5) and (bb2[k]==445.5):
                    res[1]=[j,k]
                if (bb1[j]==1782) and (bb2[k]==1782):
                    res[2]=[j,k]
                if (bb1[j]==447) and (bb2[k]==3117):
                    res[3]=[j,k]
        if res != [None]*4:
            tdc[i/2.] = res
            if res[0] is not None:
                lumip1keys.append('b1-'+str(b1[res[0][0]])+'-b2-'+
                        str(b2[res[0][1]]))
            if res[1] is not None:
                lumip2keys.append('b1-'+str(b1[res[1][0]])+'-b2-'+
                        str(b2[res[1][1]]))
            if res[2] is not None:
                lumip5keys.append('b1-'+str(b1[res[2][0]])+'-b2-'+
                        str(b2[res[2][1]]))
            if res[3] is not None:
                lumip8keys.append('b1-'+str(b1[res[3][0]])+'-b2-'+
                        str(b2[res[3][1]]))
    for k in lumip1keys:
        lumip1[k]= pd.DataFrame(columns= ['sim.turn','t(hours)', '
                L(cm^-2␣s^-1)','red␣factor','betas'])
    for k in lumip2keys:
        lumip2[k]= pd.DataFrame(columns= ['sim.turn','t(hours)', '
                L(cm^-2␣s^-1)','red␣factor','betas'])
    for k in lumip5keys:
        lumip5[k]= pd.DataFrame(columns= ['sim.turn','t(hours)', '
                L(cm^-2␣s^-1)','red␣factor','betas'])
    for k in lumip8keys:
        lumip8[k]= pd.DataFrame(columns= ['sim.turn','t(hours)', '
                L(cm^-2␣s^-1)','red␣factor','betas'])
    return tdc,lumip1,lumip2,lumip5,lumip8
```

Table of Contents

Command Line Interface

- Example:

```
python ctepy -f "settings.in" -b1 "bunchesb1.csv" -b2 "bunchesb2.csv" -s "12376" -t "./results/test_sim_"
```

Command Line Interface

- Example:

```
python ctepy -f "settings.in" -b1 "bunchesb1.csv" -b2 "bunchesb2.csv" -s "12376" -t "./results/test_sim_"
```

- Python parser to read CLI arguments

Command Line Interface

- Example:

```
python ctepy -f "settings.in" -b1 "bunchesb1.csv" -b2 "bunchesb2.csv" -s "12376" -t "./results/test_sim_"
```

- Python parser to read CLI arguments
  - -f : file containing simulation settings

- Example:

```
python ctepy -f "settings.in" -b1 "bunchesb1.csv" -b2 "bunchesb2.csv" -s "12376" -t "./results/test_sim_"
```

- Python parser to read CLI arguments
  - -f : file containing simulation settings
  - -b1 /-b2 : files containing the initial conditions for the bunches to be simulated

- Example:

```
python ctepy -f "settings.in" -b1 "bunchesb1.csv" -b2 "bunchesb2.csv" -s "12376" -t "./results/test_sim_"
```

- Python parser to read CLI arguments
  - -f : file containing simulation settings
  - -b1 /-b2 : files containing the initial conditions for the bunches to be simulated
  - -s : random seed

- Example:

```
python ctepy -f "settings.in" -b1 "bunchesb1.csv" -b2 "bunchesb2.csv" -s "12376" -t "./results/test_sim_"
```

- Python parser to read CLI arguments
  - -f : file containing simulation settings
  - -b1 /-b2 : files containing the initial conditions for the bunches to be simulated
  - -s : random seed
  - -t : location and prefix for output files (possible update to write to database)

- Example:

```
python ctepy -f "settings.in" -b1 "bunchesb1.csv" -b2 "bunchesb2.csv" -s "12376" -t "./results/test_sim_"
```

- Python parser to read CLI arguments
  - -f : file containing simulation settings
  - -b1 /-b2 : files containing the initial conditions for the bunches to be simulated
  - -s : random seed
  - -t : location and prefix for output files (possible update to write to database)
- Decorators perform checks on arguments

- Example:

```
python ctepy -f "settings.in" -b1 "bunchesb1.csv" -b2 "bunchesb2.csv" -s "12376" -t "./results/test_sim_"
```

- Python parser to read CLI arguments
  - -f : file containing simulation settings
  - -b1 /-b2 : files containing the initial conditions for the bunches to be simulated
  - -s : random seed
  - -t : location and prefix for output files (possible update to write to database)
- Decorators perform checks on arguments
- Several input files required

Input files

- Settings file

```
op_voltage_rf1_b2       0.0
op_nharm2_b1    35640
op_nharm1_b2    360
op_voltage_rf1_b1       0.0
sim_nturns      2000
op_nharm2_b2    35640
op_ip_settings_fn       ./../TestInputFiles/test_ip_settings.csv
op_voltage_rf2_b2       -14000000.0
part_a_atom_b2  1
op_tauhat1      1.25e-09
op_tauhat2      1.25e-09
sim_timeratio   20000
switch_raddamp  True
part_a_atom_b1  208
phys_raddamp_method     approx
```

Input files

- Settings file
- Bunches initial conditions

```
bucket  exn      eyn      npart    blen
1       1.5e-06 2.1e-06 11000000000.0   0.076
5       1.8e-06 2.3e-06 13000000000.0   0.085
```

Input files

- Settings file
- Bunches initial conditions
- IP settings

ip_id	ip_rf_loc		ip_beta	ip_xing	ip_level_switch	ip_level_value
ip1	0	0.6	0.000145		False	1.8e-26
ip2	891	0.8	0.0001375		False	0.0
ip5	1782	0.6	0.000145		False	0.0
ip8	3117	1.5	0.000285		False	0.0

Table of Contents

Output files

- Intensity Evolution (per beam per bunch)

```
,sim.turn,t(hours),N1_macro,N1_real,NlostLum1,SumL1,NlostDebunch1,SumD1,NLostBet1,Sumb2,NlostMom1,SumM1
0,100.0,0.04940248245,990.0,148500000.0,10.0,10.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0
1,200.0,0.0988049648999,985.0,147750000.0,5.0,15.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0
2,300.0,0.14820744735,980.0,147000000.0,4.0,19.0,1.0,1.0,0.0,0.0,0.0,0.0,0.0
3,400.0,0.1976099298,972.0,145800000.0,8.0,27.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0
4,500.0,0.24701241225,966.0,144900000.0,6.0,33.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0
```

- Emittance Evolution (per beam per bunch)
- IBS life times (per beam per bunch)
- Luminosity Evolution (per IP per colliding pair)

Output files

- Intensity Evolution (per beam per bunch)
- Emittance Evolution (per beam per bunch)

```
,sim.turn,t(hours),ex1(m),ey1(m),el1(eV/s/charge),sig1_T1,sig1_dP/P_2
0,0.0,0.0,5.5825769063e-10,5.39433790991e-10,0.469071511379,2.51828545989e-10,9.12058214449e-05
1,100.0,0.04940248245,5.6129215906e-10,5.40293321836e-10,0.487367635197,2.57958067577e-10,9.25115655377e-05
2,200.0,0.0988049648999,5.64613375881e-10,5.39773833543e-10,0.498639674164,2.68272894516e-10,9.10119657798e-05
3,300.0,0.14820744735,5.70276964844e-10,5.38991703571e-10,0.507743064796,2.65008880096e-10,9.38149460536e-05
4,400.0,0.1976099298,5.72806052518e-10,5.39316255512e-10,0.513817308456,2.58815455793e-10,9.72091133559e-05
5,500.0,0.24701241225,5.81267935014e-10,5.40179947552e-10,0.532681410349,2.71776785229e-10,9.59718007271e-05
```

- IBS life times (per beam per bunch)
- Luminosity Evolution (per IP per colliding pair)

Output files

- Intensity Evolution (per beam per bunch)
- Emittance Evolution (per beam per bunch)
- IBS life times (per beam per bunch)

```
,sim.turn,t(hours),Tp(hours),Tx(hours),Ty(hours)
0,100.0,0.0494024831861,1.99135178794,5.53714454666,-22935.2000234
0,200.0,0.0988049663722,1.99528245505,5.77796324126,-23871.4348998
0,300.0,0.148207453239,2.12068578922,5.84764345996,-23996.5108119
0,400.0,0.197609932744,2.27011494092,5.82620991508,-23824.4177956
0,500.0,0.24701241225,2.32592380837,6.2401060832,-25378.3411978
```

- Luminosity Evolution (per IP per colliding pair)

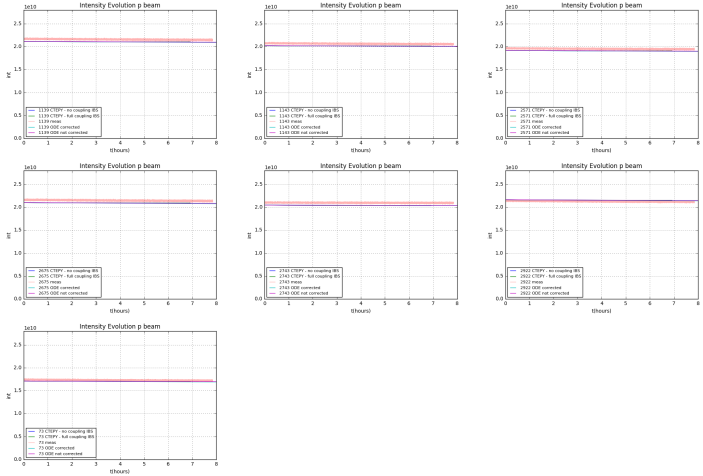
Output files

- Intensity Evolution (per beam per bunch)
- Emittance Evolution (per beam per bunch)
- IBS life times (per beam per bunch)
- Luminosity Evolution (per IP per colliding pair)

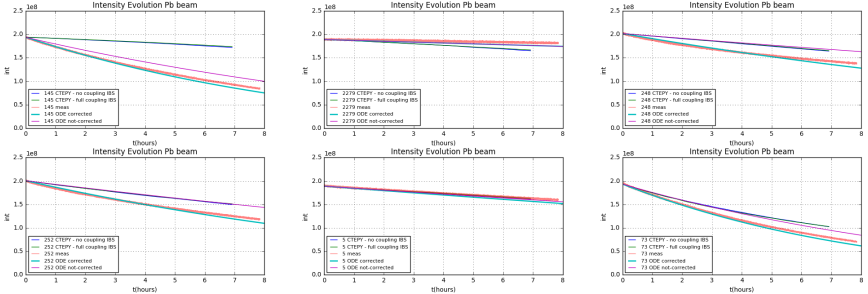
```
,sim.turn,t(hours),L(cm^-2 s^-1),red factor,betas
0,1.0,0.0004940248245,5.43172198611e+24,0.834262730505,0.8000003615
1,100.0,0.04940248245,5.32563765713e+24,0.833414126217,0.8000003615
2,200.0,0.0988049648999,5.20354748571e+24,0.82727548401,0.8000003615
3,300.0,0.14820744735,5.07332067055e+24,0.820042334529,0.8000003615
4,400.0,0.1976099298,5.03359018282e+24,0.829682807118,0.8000003615
5,500.0,0.24701241225,4.91015007074e+24,0.825243108757,0.8000003615
```

Table of Contents

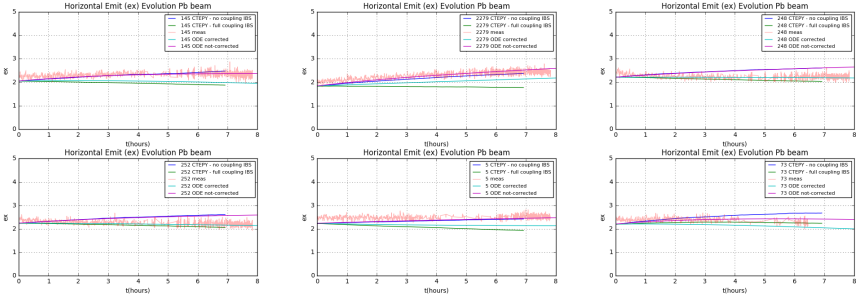
Intensity - protons



Intensity - ions



Emittance - ions



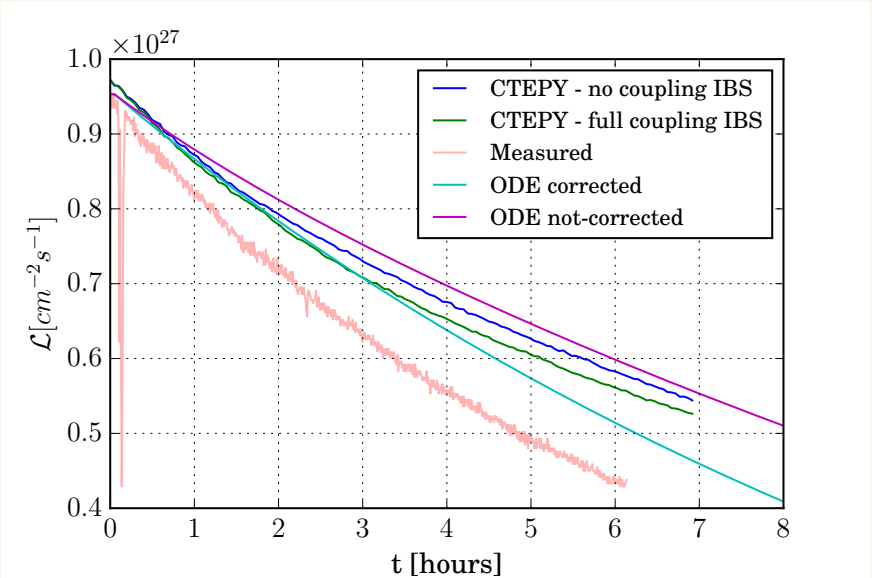
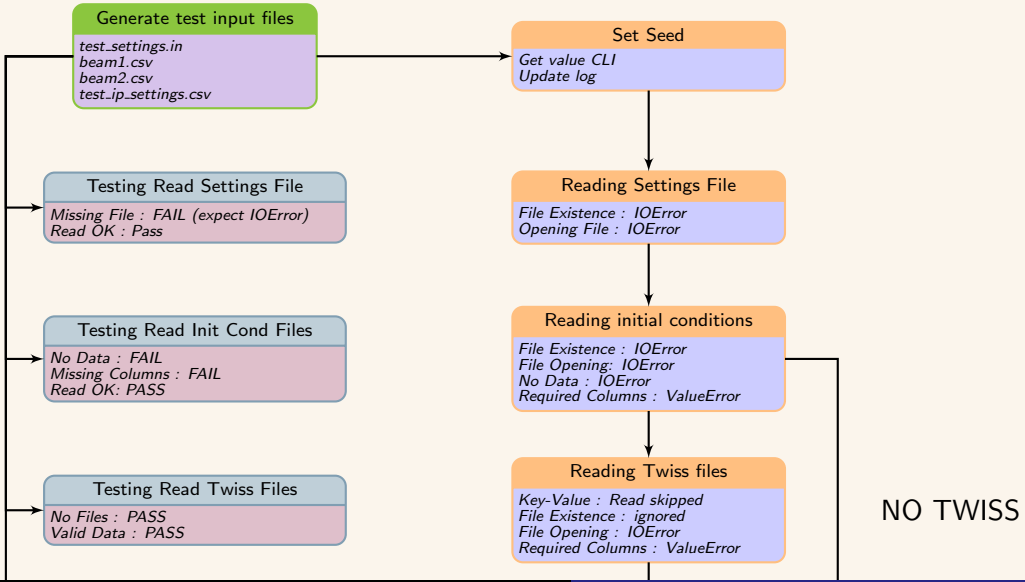

$\times 10^{27}$
1.0
0.9
0.8
0.7
0.6
0.5
0.4
$\mathcal{L}[cm^{-2}s^{-1}]$
0
1
2
3
4
5
6
7
8
t [hours]
CTEPY - no coupling IBS
CTEPY - full coupling IBS
Measured
ODE corrected
ODE not-corrected

Table of Contents

Flowchart example



Questions?