



PLACET: New Features and Plans

Andrea Latina, Daniel Schulte (CERN)

...and the CERN CLIC Beam Dynamics Group

December 11-13, 2007 · SLAC

ILC LET Beam Dynamics Workshop

Placet Tracking Code

- it is a tracking code that simulates **beam transport** and **orbit correction** in linear colliders, developed by Daniel Schulte
- it is a real tracking code and implements several collective effects -relevant for a LC-
- it implements orbit correction algorithms and feedback loops
- it cannot simulate rings (not yet!)
- it is fully programmable (its interface is a real programming language)
- it is open to other codes, is modular and expandable

Technical Details

- It is written in C++ with a **Tcl/Tk** user interface
 - it is fast
 - it has a graphical output
- It is fully **programmable** and **modular**:
 - complex simulation scripts
 - it allows the simulation of feedback loops
 - ground motion or other dynamic effects are easy to include
- It is **open** to other codes:
 - it can read **MAD/MAD-X** deck files, as well as **XSIF** files
 - it makes use of the Universal Parser Library and **AML**
 - can be easily interfaced to Guinea-Pig
 - it can use other codes to perform beam transport
 - ⇒ e.g. Placet-BDSIM interface (see Steve Malton's talk)
- [NEW] Now it embeds **Octave**, a mathematical toolbox like MatLab (but *open-source*)
 - rich set of numerical tools
 - easy to use optimization / control system tool-boxes

Elements and Tracking

- a beamline is a *list* of Girders; each Girder contains a *list* of Elements

⇒ the following types are supported:

Quadrupole	SBend	Multipole
AccCavity	CrabCavity	Kicker
BPM	Collimator	ExternalObj

⇒ it tracks in 4d or 6d

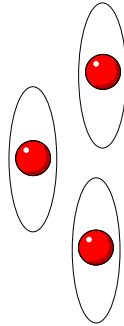
- thick lenses
- trajectory integration using thin lenses
- longitudinal motion is a boolean flag

⇒ misalignment of the elements is taken into account : offsets, rolls, and pitches

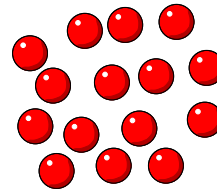
- it takes into account collective effects
 - **synchrotron radiation emission**:
 - Incoherent SR emission : all magnets
 - Coherent SR emission : sector bends
 - long/short-range **wakefields**
 - in the accelerating structures
 - in the crab cavities
 - geometric and resistive-wall wakes in the collimators
- it can simulate **ground motion**: sample sites (A,B,C,K), **ATL** Model

Beam Models: Slices and Particles

- beam is represented as a train of bunches; bunches are represented using **two** models



Macro Particles + 2nd order momenta



Single Particles

1) linear lattices → **slices**

bunches are cut longitudinally in slices; each slice contains macro-particles that have the same longitudinal position; each macro-particle is defined by a 6d-vector with second-order moments (fast tracking, LINAC)

particle : (x, x', y, y', E, z)

$$\text{moments : } \begin{pmatrix} \sigma_{xx} & \sigma_{xx'} & \sigma_{xy} & \sigma_{xy'} \\ \sigma_{x'x} & \sigma_{x'x'} & \sigma_{x'y} & \sigma_{x'y'} \\ \sigma_{yx} & \sigma_{yx'} & \sigma_{yy} & \sigma_{yy'} \\ \sigma_{y'x} & \sigma_{y'x'} & \sigma_{y'y} & \sigma_{y'y'} \end{pmatrix}$$

2) non-linear lattices → **single-particles**: set of macro-particles defined by 6d-vectors (BC, BDS)

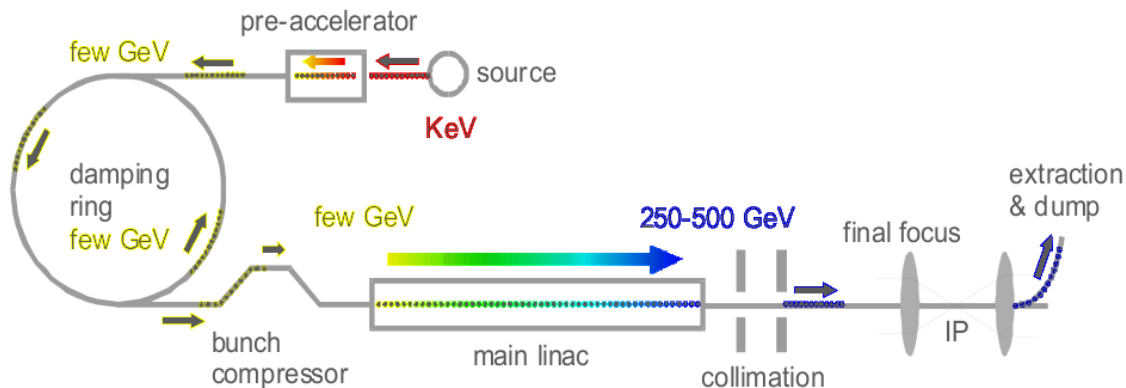
Beam Models: Slices and Particles

- it can switch between the two models during the tracking



⇒ typical numbers

- linac : 31 longitudinal slices, 11 macro-particles per slice = 341 macro-particles (particles + moments)
 - bc, bds : 50.000-100.000 single-particles
- Placet can simulate: bunch compressor, main linac, drive beam, beam delivery system (including crab cavities and instrumentation), interaction point (using **guinea-pig**)



Beam-Based Alignment Algorithms

- in a future Linear Collider, Beam-Based Alignment will be necessary to achieve the target vertical emittance at the IP
- Beam-based alignment procedures in Placet:
 - 1-to-1 Correction
 - Dispersion Free Steering
 - Ballistic Correction
 - Tuning Bumps
 - Dispersion Bumps
 - Wakefield Bumps

⇒ You need to use them all !
- These procedures take the following arguments:
 - beam : beam that has to be used during the optimization
 - machines : number of machines to be simulated (random seeds)
 - correctors : list of correctors to be used (they can be either quadrupoles or dipoles)
 - survey : misalignment schema "Clic", "Zero", "None", or an external user defined Tcl procedure (this allows the simulation of ground motion, for example)

Brief History of Placet and New Features

- 1999→2005:

- originally developed by **Daniel Schulte** (just for fun!) with contributions by **Eric D'Amico**, **Nicolas Leros** and **Peder Eliasson**

- 2005→today:

- I made some improvements :

- C++ redesign (I needed to..)

- Octave interface

- longitudinal tracking

- new elements : collimator, crab cavities, ...

- new installation procedure ./configure, cvs repository, web page, cvs web...

- **Daniel Schulte** : ground motion, dynamic effects, ...
- **Helmut Burkhardt** and **Lionel Neukermans** : HTGEN, halo and tail generation
- **Giovanni Rumolo** : Collimator's wakefields
- **Erik Adli** : Coherent Synchrotron Radiation Emission in Sector Bends, Drive Beam Improvements

Collaborations Across Europe

- **LAL/Orsay (FR)**
 - ILC Beam-Beam Fast Feedback Simulation
 - ATF2 Simulations
 - AML/UPL Interface
- **Cockcroft Institute (UK)**
 - ILC / CLIC Crab Cavities
- **Ankara University (Turkey)**
 - Gamma-Gamma interaction at the ILC
- **John Adams Institute, Oxford (UK)**
 - ILC / ATF2 Feedback Systems
- **Royal Holloway University of London (UK)**
 - ILC Collimators, BDSIM integration
- **Manchester University (UK)**
 - ILC Collimators
- **PSI (CH)**
 - CLIC Bunch Compressors

	placet	placet-octave (improvements..)
Core	C	C++
User Interface	Tcl/Tk	Tcl/Tk + Octave
Tracking	4d Slices \Rightarrow Particles (LINAC-BDS)	6d Slices \Leftrightarrow Particles (BC-LINAC-BDS)
Element Types	Quad/Dipole/Bend/Multipole, Acc/DecCavity Acc/DecCavity BPM, TclCall	CrabCavity, Collimator, DL_Element, LinkElement
Collective Effects	Short/Long-range WF in Cavities Inc. Synrad in Bend, Quad, Multipoles	Geometric/Resistive-Wall WF in Collimators Longrange Wakefields in the CrabCavities Coherent Synrad in the SBend
I/O File Format	PLACET / MAD (tricky)	Universal Parser Library / AML (exp)
Alignment Routines	Dipole Kicker / Quadrupole Movers 1-TO-1, Ballistic, RF, DFS, ...	Generic Correctors (Attributes) Octave-1-TO-1, Octave-DFS, Simplex, MICADO, ...

Elements

- Hierarchy of C++ classes to describe element types \Rightarrow each type derives from archetypal object ELEMENT
- List of common attributes to all element types

-name	Element name [STRING]
-s	Longitudinal position [m] [READ-ONLY]
-x	Horizontal offset [um]
-y	Vertical offset [um]
-xp	Horizontal offset in angle [urad]
-yp	Vertical offset in angle [urad]
-roll	Roll angle [urad]
-length	Element length [m]
-synrad	Incoherent Synchrotron Radiation emission [BOOL]
-thin_lens	Thin Lens approximation [INT!=0]
-six_dim	Enable 6d tracking [BOOL]
-aperture_x	Horizontal aperture [m]
-aperture_y	Vertical aperture [m]
-aperture_shape	Aperture shape [STRING]

- Each sub-class inherits these attributes and defines its own attributes

An example :

```
CrabCavity -name CRABCAV -length 0.5 -frequency 3.9 -voltage 1.32 -wakelong "wakelong"
```

Elements' Attributes

- You can read/modify **each attribute** using two commands

- placet:

```
ElementSetAttribute -element 123 -attribute "length" -value 321.0  
set length [ElementGetAttribute -element 123 -attribute "length"]
```

- placet-octave:

```
QUAD=placet_get_number_list("main_linac", "quadrupole");  
placet_element_set_attribute("main_linac", QUAD, "strength", 0.0);  
QUADS=placet_element_get_attribute("main_linac", QUAD, "strength");
```

- Previously, there was one command for each property:

MultipoleSetStrengthList

DipoleSetStrengthList

QuadrupoleGetStrength

DipoleSetStrength

⇒ Advanced optimization/feedback simulation programs

⇒ A new attribute that turned out to be very useful is -name

Embedding of Octave in PLACET

- **Octave** is a high-level interactive language for numerical computations
 - it is mostly compatible with MatLab[©]
 - it can do arithmetic for real and complex scalars and matrices, solve sets of nonlinear algebraic equations, integrate functions over finite and infinite intervals, and integrate systems of ordinary differential and differential-algebraic equations
- **Octave's** interpreter and libraries have been **embedded** into PLACET
 - this means that now PLACET includes two interpreters: Tcl and Octave languages
- The backbone of PLACET is still the Tcl Language but a new Tcl keyword Octave has been added
- This keyword can be used in two ways:
 - Octave
 - ⇒ without arguments, it opens an interactive Octave shell
 - Octave { here some octave code }
 - ⇒ it runs the Octave's code and continues the Tcl script execution afterwards

Example of 1-to-1 Correction Using placet-octave

```
#!/home/andrea/bin/placet

source beamline.tcl
source beamdef.tcl
BeamlineSet -name "beamline"

SurveyErrorSet -quadrupole_y 300.0 \
               -quadrupole_roll 300.0 \
               -cavity_y 300.0 \
               -cavity_yp 300.0 \
               -bpm_y 300.0

Octave {
  B = placet_get_number_list("beamline", "bpm");
  C = placet_get_number_list("beamline", "quadrupole");
  R = placet_get_response_matrix("beamline", "beam0", B, C);

  placet_test_no_correction("beamline", "beam0", "Clic");
  b = placet_get_bpm_readings("beamline", B);
  c = -pinv(R) * b;
  placet_vary_corrector("beamline", C, c);

  placet_test_no_correction("beamline", "beam0", "None");
  [b,S] = placet_get_bpm_readings("beamline", B);
  plot(S, b);
}
```

Improvements in the Tracking Module

- **Longitudinal motion** is now considered
 - longitudinal track is a boolean flag of each element
 - ⇒ entire segments of the lattice can be considered as 6d (i.e. the BC but not the ML)
 - DRIFT, QUADRUPOLE, DIPOLE, SBEND, CAVITY, SEXTUPOLE and MULTIPOLE's are already implemented
- **Thin-lens approximation** is being implemented as a boolean flag (both 4d and 6d)
- simple **MPI parallel tracking module** exists (never distributed!)
 - beam is *scattered* among the CPUs (# of particles / # of CPUs)
 - beam *reduction* after tracking correctly updates the beamline status (i.e. bpm readings)
 - collective effects are not considered yet

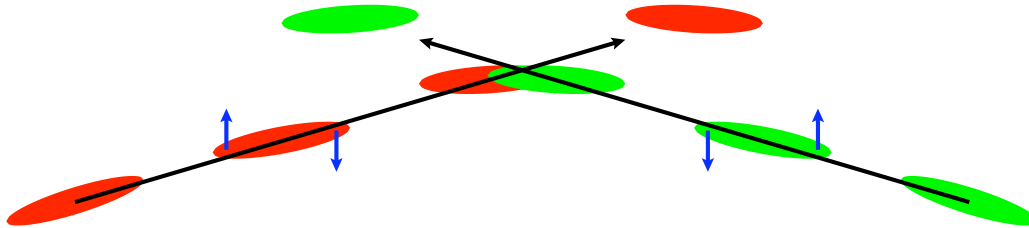
Some Examples

- The results showed in the other presentations
- **Beam Delivery System**
 - Collimator wakefields
 - Crab cavity imperfections
 - Halo tracking and background
- Preliminary: ATF2 static alignment

Wakefields in the BDS

Crab Cavity

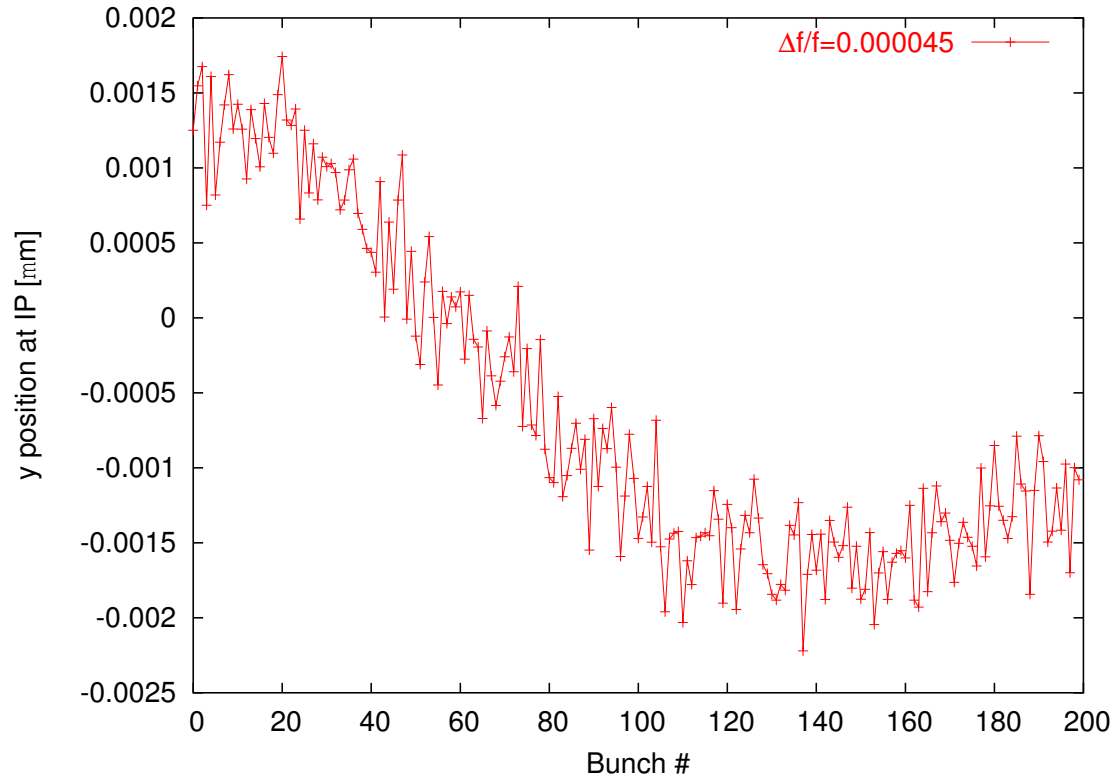
- Crab cavities are required for ILC and CLIC



- The crab cavity is a deflection cavity operated with a 90° phase shift.
- A particle at the center of the bunch gets no transverse momentum kick and hence no deflection at the IP.
- A particle at the front gets a transverse momentum that is equal and opposite to a particle at the back.
- The quadrupoles change the rate of rotation of the bunch.

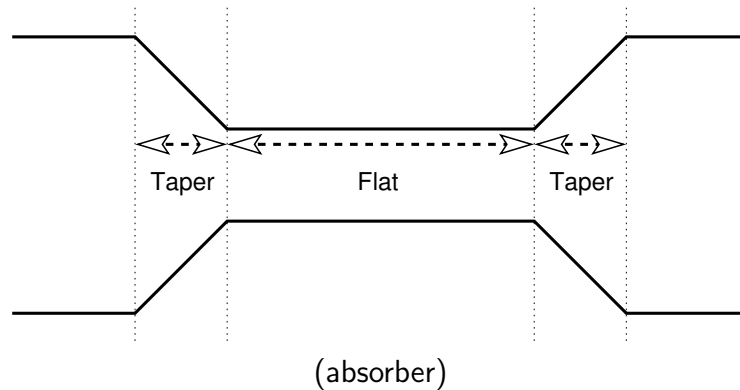
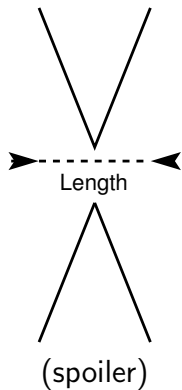
ILC Crab Cavity Wakefields

Vertical offset at the IP due to longrange wakefields in the Crab Cavities in case of frequency jitter.



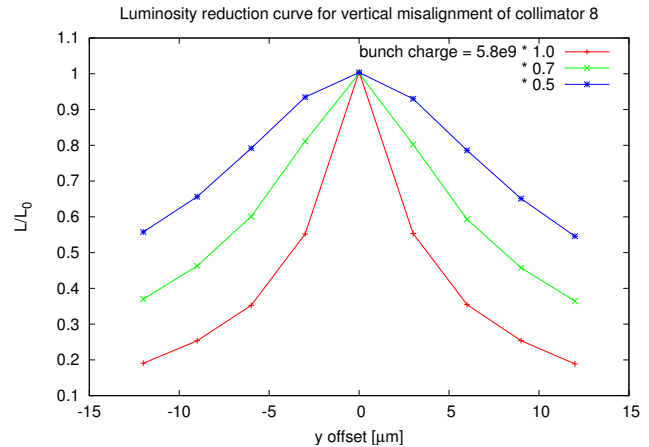
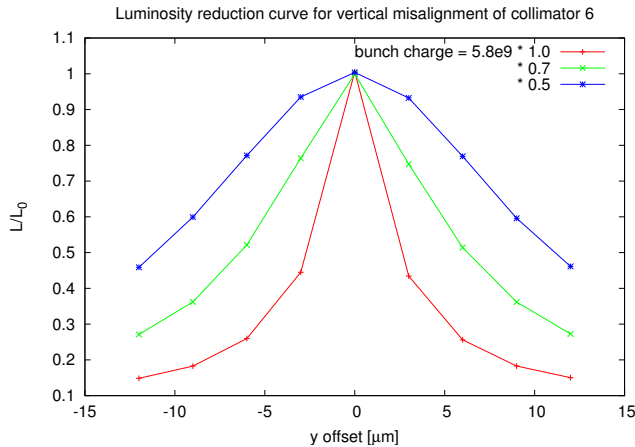
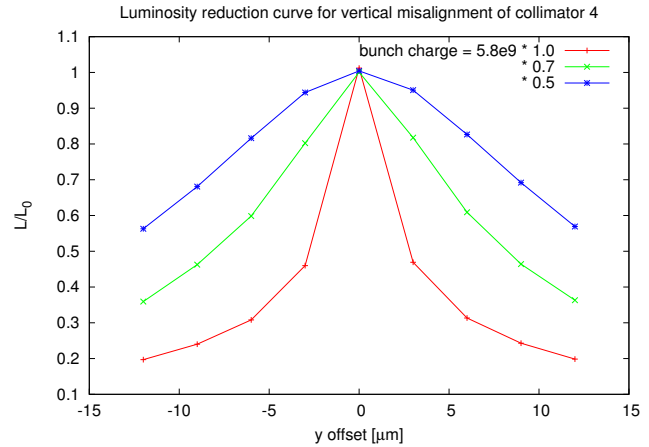
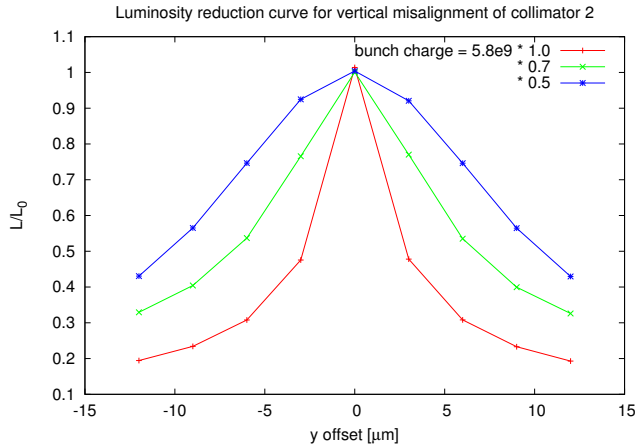
Collimators Wakefields

- Wakefield kick (Stupakov, Yokoya):
 - geometric and resistive components are evaluated
 - inductive or diffractive for the geometric wake fields, short- or long-range, intermediate regimes
 - the bunch is subdivided into slices
 - the KICK depends both on the longitudinal and on the transverse coordinates of the particles
- Input parameters:
 - geometry of the collimator (width, initial and final height, taper length, ...)
 - properties of the material (conductivity σ , relaxation time τ)
 - type (spoiler/absorber, vertical/horizontal)

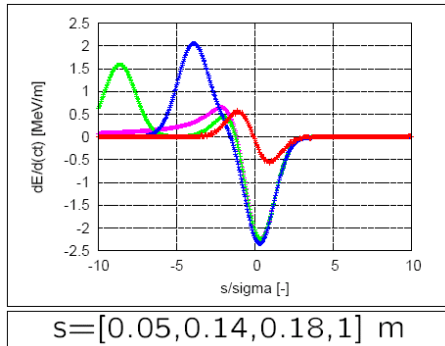


CLIC BDS Collimator Wakefields

- Luminosity reduction curves due to vertical misalignment of the collimators



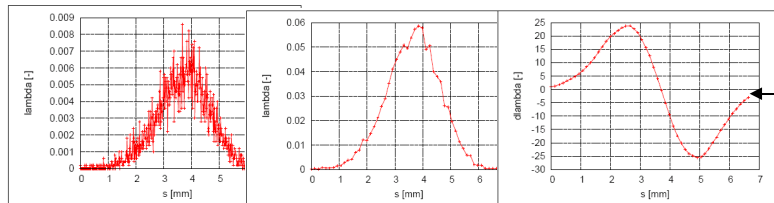
CSR Module (E.Adli)



A CSR module has been implemented (E. Adli).

CSR can be activated in the S-bend element, with a simple switch. Implementation is based on the proven Elegant-implementation, based on paper by Saldin et al.

(Limitations of this approach: 1D model, currently no shielding effects taken into account).



User can (and should) adjust number of bins and filter length for best performance.

Currently undergoing final benchmarking against Elegant and CSRTrack (E. Adli and F. Stulle)

Easy to use: described by on-line help:

```
sbend -help
.
-csr                Coherent Synchrotron Radiation (CSR) [BOOL]
-csr_charge         CSR: [REQUIRED] total charge of input distribution [C]
-csr_enable_driftwake Enable csr wake propagation into trailing drift [BOOL]
                    drift wake will be applied until 1% remains or until next SBend
-csr_nbins          CSR: # of bins ( >= 10 ) [#]
-csr_nhalffilter     CSR: savitzky-Golay filter half-width ( >= 1 ) [# of bins]
-csr_nsectors        CSR: # of dipole sectors ( >= 1 ) [#]
-csr_savesectors     CSR: save data for each sector [nbin s lambda dlambda dE_ds [GeV/m] ] [BOOL]
-csr_attenuation_length CSR: attenuation length for csr drift (default value is 1.5*overtaking length) [m]
-csr_enforce_steady_state CSR: enforce steady state mode (infinite slippage length) [BOOL]
```

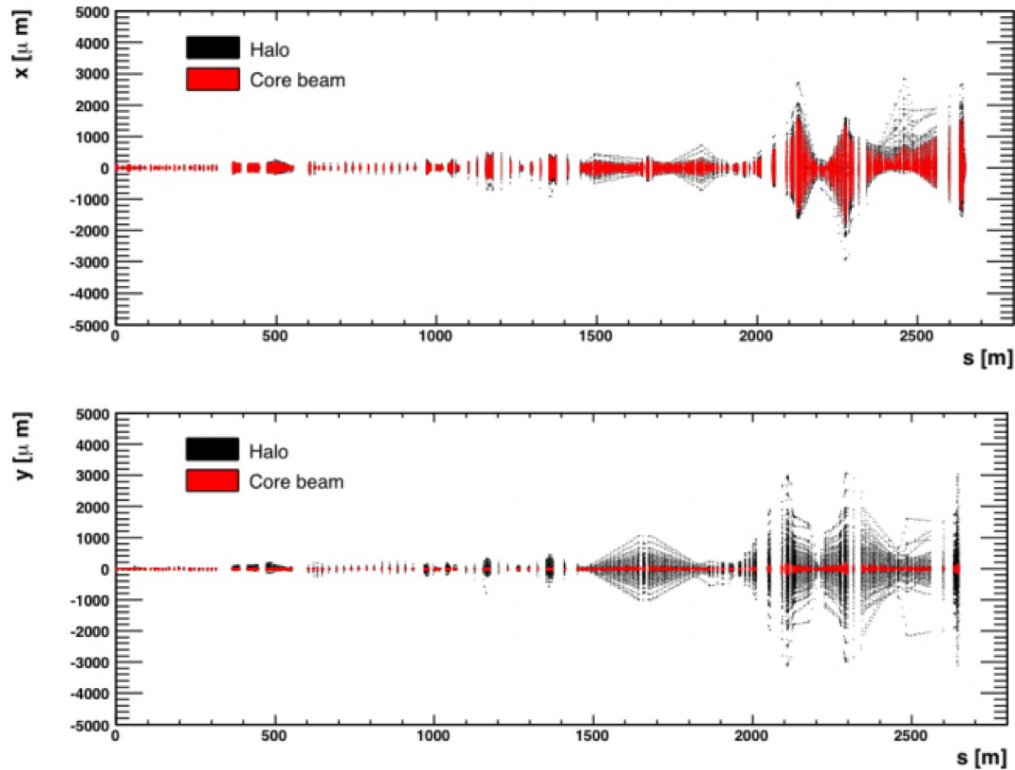
Halo generation and tracking

- HTGEN is part of ILPS task
- Identify and study critical issues:
 - Halo sources
 - Transfer lines (collimation, final focus ...)
- Provide a generic tool for beam halo studies
 - beam-gas generator, halo tracking, photon tracking, multiple scattering in spoilers are already available Placet package
 - More details and code available in

<http://n.home.cern.ch/n/neukerma/www>

Halo generation and tracking

- The beam gas pressure and apertures can be separately specified for each element
- The particles hitting the beam-pipe are considered lost

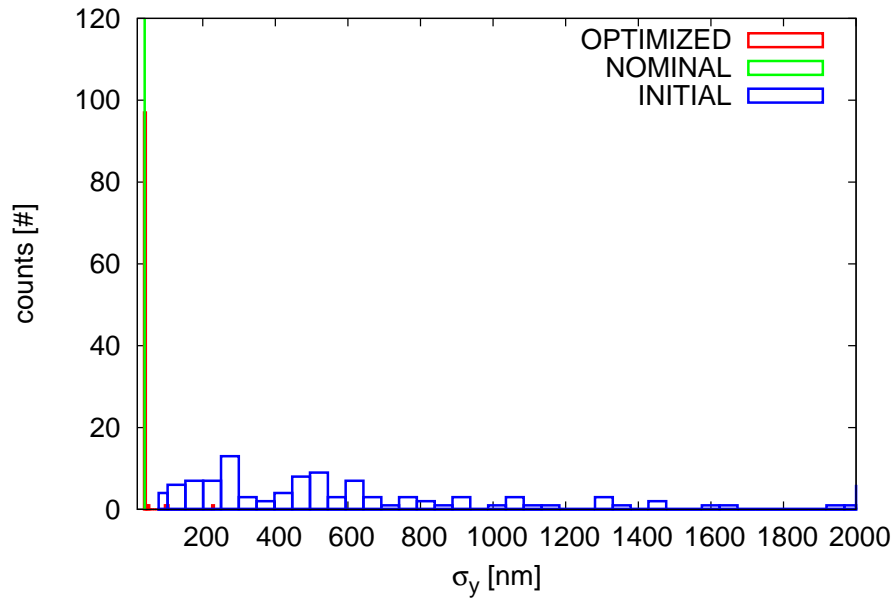


⇒ beam-gas scattering from LINAC and BDS: a fraction of 10^{-4} of the particles impacts on the spoilers

Preliminary: ATF2 Static Alignment

- Simulation Parameters

- atf2 lattice
- $\sigma_{RMS} = 100\mu\text{m}$ quadrupole and sextupole positions
- $\sigma_{q\text{-strength}} = 0.01\%$ quadrupoles' strength jitter
- observable : final vertical beam size $\sigma_{\text{res}} = 1\text{ nm}$



Example of ATF2 Alignment Using placet-octave

```
Octave {
    global nIteration=0;
    function beam_size = FinalSize(beamline)
        global nIteration;
        target_sx = 2.17071070688;
        target_sy = 0.0387071349;
        [E,B]=placet_test_no_correction(beamline, "beam0", "None");
        beam_sizey=std(B(:,3))+(randn()*0.001);
        beam_sizex=std(B(:,2))+(randn()*0.001);
        beam_size=sqrt((beam_sizey - target_sy)**2 + (beam_sizex - target_sx)**2/100.0);
        nIteration++;
    endfunction

    # we define the correction

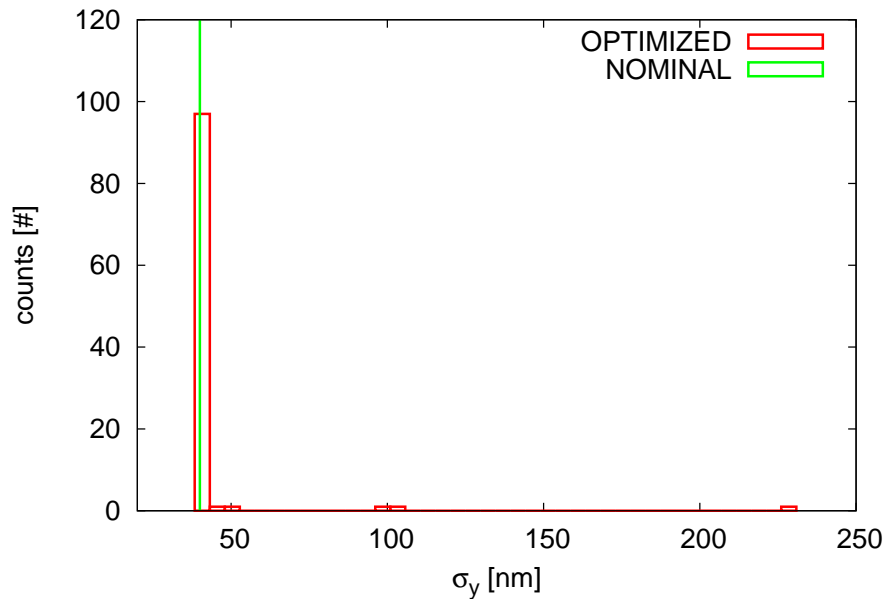
    QI=placet_get_number_list("ATF2", "quadrupole");
    SI=placet_get_number_list("ATF2", "sextupole");

    CORR = [ QI, QI, QI, SI, SI ];
    LEVR = [ repmat("x", size(QI'));
             repmat("y", size(QI'));
             repmat("strength", size(QI'));
             repmat("x", size(SI'));
             repmat("y", size(SI')) ];
    STEP = [ 10*ones(size([QI, QI])), 0.0001*ones(size(QI)), 5*ones(size([SI, SI])) ];

    [optimum, merit] = placet_optimize("ATF2", "FinalSize", CORR, LEVR, STEP);
}
```

Preliminary: ATF2 Static Alignment

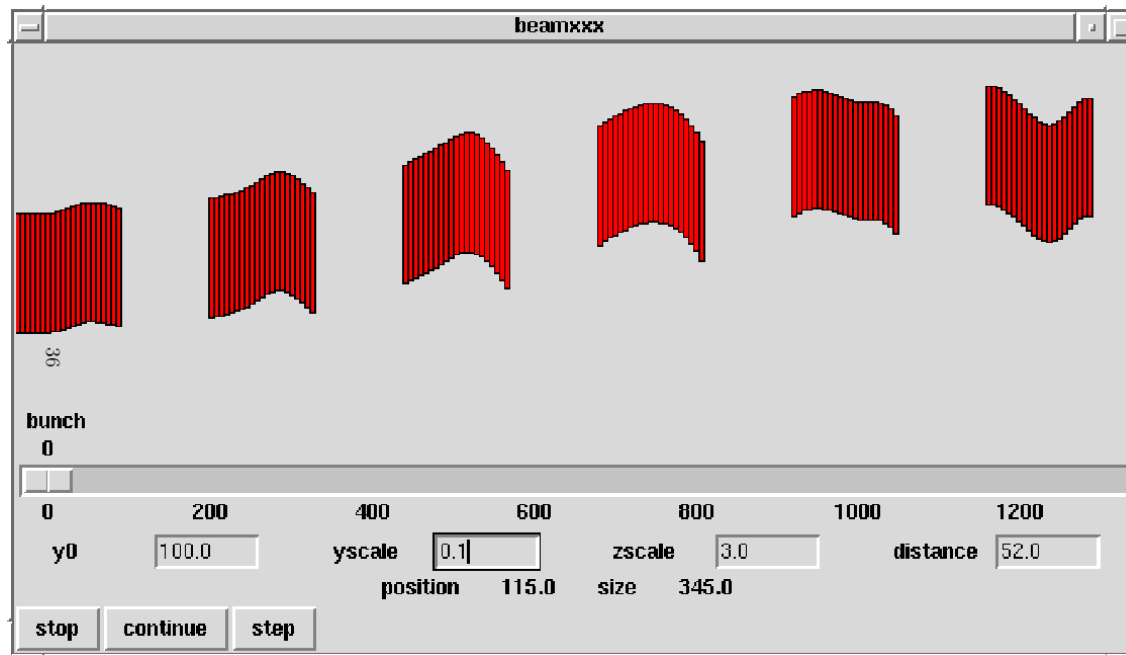
- Optimization Procedure
 - 1-to-1 correction using the quadrupoles
 - simplex optimization using quadrupole and sextupoles as correctors
 - quadrupole : movers, strength
 - sextupole : movers



⇒ about 1200 iterations required!

PLACET Graphical Output

- Longitudinal Beam Profile under the effects of transverse wakefield



Overview of the new features (I)

- PLACET has undergone a redesign toward pure C++ code
 - it is faster
 - easy to maintain and extend
- Collimator wake fields (both geometric and resistive wall) have been implemented to allow full tracking
 - luminosity reduction curves due to the wake fields have been obtained for initial jitters and different configurations of collimator misalignments
 - the performances of the nonlinear collimation system including wake fields have to be studied
- The model for wake fields includes nonlinear and near-wall effects
- Octave interface opened the way to even more complex simulation scripts and much more, thanks to its high-level language for scientific computations
 - Correction algorithms and optimizations are easy to write
 - Feedback loops can take advantage of the extensive Octave's library of optimization routines
 - Excellent Results in BDS Alignment

Overview of some other new features (II)

- Longitudinal motion is a boolean flag : only segments of lattice can be 6d (like option -synrad)
- ParticlesToSlices, besides the existing SlicesToParticles
- CrabCavities in the code, with independent longrange wakefields
- Possibility of creating external, *dynamically loadable*, elements
- Interfacing with BDSIM, for accurate HALO tracking in the Collimators (RHUL):
 - parallel tracking: placet/BDSIM, with exchange of halo data at each Collimator
 - placet tracks the core bunch along the BDS, and the HALO inside the collimators
 - BDSIM tracks the halo along the BDS, and receives from placet the Collimators wakefield kick
- **Future Developments**
 - ⇒ PLACET/GdfidL interface
 - ⇒ Test of Dispersion Free Steering in CTF3, Flight Simulator for ATF2
 - ⇒ Can placet-octave be used in a control room?

<http://savannah.cern.ch/projects/placet>

Appendix

New PLACET Installation Procedure

- Now we have a `./configure` script:
 - it checks the characteristics of the computer in use and writes the appropriate makefiles
 - it accepts the following options:

`./configure`

<code>-prefix=DIR</code>	installs placet in \$DIR
<code>-enable-htgen</code>	enables the use of HTGEN
<code>-enable-octave</code>	enables the use of Octave
<code>-with-gsldir=DIR</code>	GNU Scientific Library is installed in \$DIR
<code>-with-octdir=DIR</code>	Octave is installed in \$DIR
<code>-with-htgendir=DIR</code>	HTGEN is installed in \$DIR

- Installation procedure

```
$.> ./configure --enable-octave --with-gsldir=$HOME/gsl-1.8 --prefix=$HOME
```

```
$.> make
```

```
$.> make install
```

Installation procedure on AFS

- There is a special version of `./configure` for computers with `/afs`, that automatically sets all these paths to the proper directories on AFS. On a CERN computer, you can install `placet` with

```
$.> ./configure.AFS --prefix=$HOME
```

```
$.> make
```

```
$.> make install
```

⇒ “stable” makefile creates :

```
${prefix}/bin/placet
```

```
${prefix}/bin/placet-htgen
```

⇒ “development” makefile creates :

```
${prefix}/bin/placet-development
```

```
${prefix}/bin/placet-htgen
```

```
${prefix}/bin/placet-octave
```

- and also : `ground`, `grid`, `mad2gp`, `gp2mad`, ... and other utilities

Interfacing of PLACET with GdfidL

- Original idea was to create a “wakefields file format” to exchange wakefields data between different tracking codes
- The wakefields would be generated by GdfidL or similar programs
- First problem : data compression, the amount of data produced by GdfidL can be as big as several hundreds Mb's per each collimator
 - we need to make a multipole expansion of the Wake-Potentials
 - we need to calculate the wakefield kick from the expansion

After several discussions with Warner, we agreed about a file format for the (uncompressed) wake-potentials (GdfidL → multipole expansion)

- GdfidL writes on disk a set of “slices” containing the Longitudinal Wake-Potential on a grid

- I have created a set of utilities to perform the multipole expansion and to calculate the transverse components of the Wake-Potential
- Multipole Expansion:

```
$.> mpolexp2d --help
mpolexp2d - 3D Multipole Expansion of a (Wake)Potential, version 0.1
Written by Andrea Latina <andrea.latina@cern.ch>, Apr 26 2007
```

```
USAGE:  mpolexp2d [OPTIONS] K [FILE]...
```

DESCRIPTION:

This program calculates the multipole moments of an input function, up to the 'K'-th term.

OPTIONS:

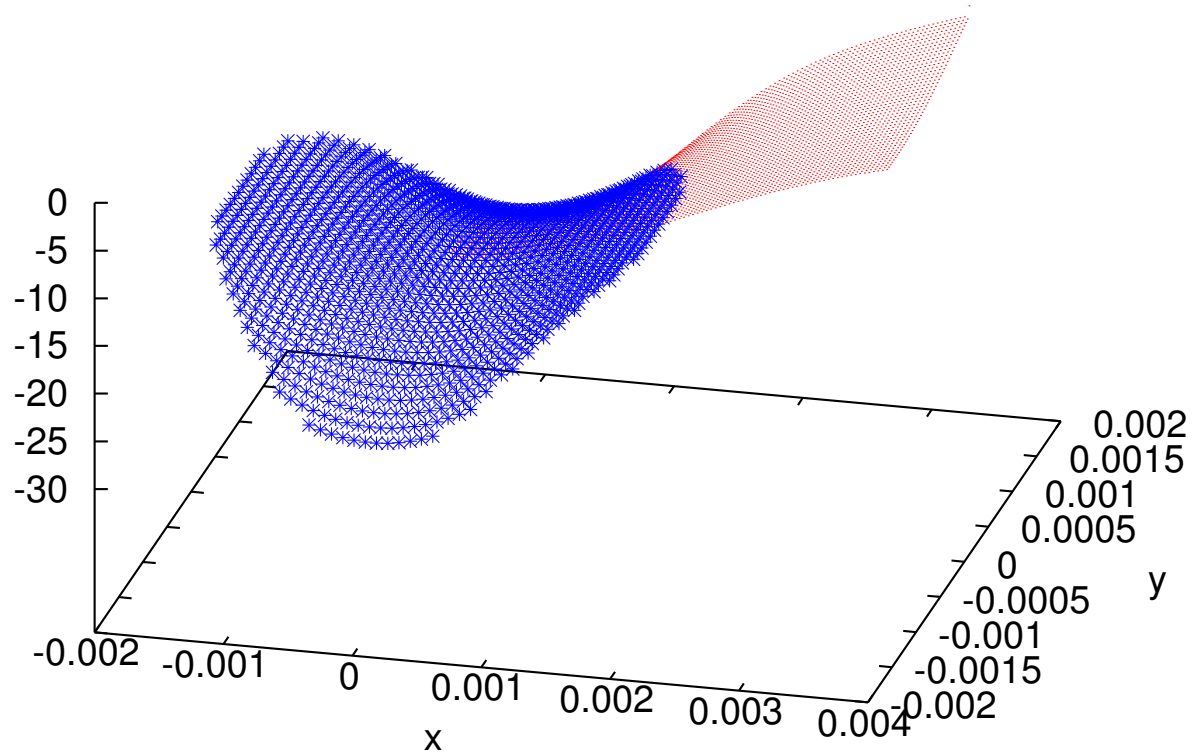
--help	Display this help
--version	Display version information
--rx	Horizontal radius of the integration disk ('XX' or 'XX%')
--ry	Vertical radius of the integration disk ('YY' or 'YY%')
--nr=N	Number of radial points for the integration (default 50)
--bilinear bicubic	Use [bilinear bicubic] interpolation (default bilinear)
--gdfidl octave	Assume the input is in [Octave GdfidL]'s text format (default GdfidL)

With no FILE, or when FILE is -, read standard input.

```
$.>
```

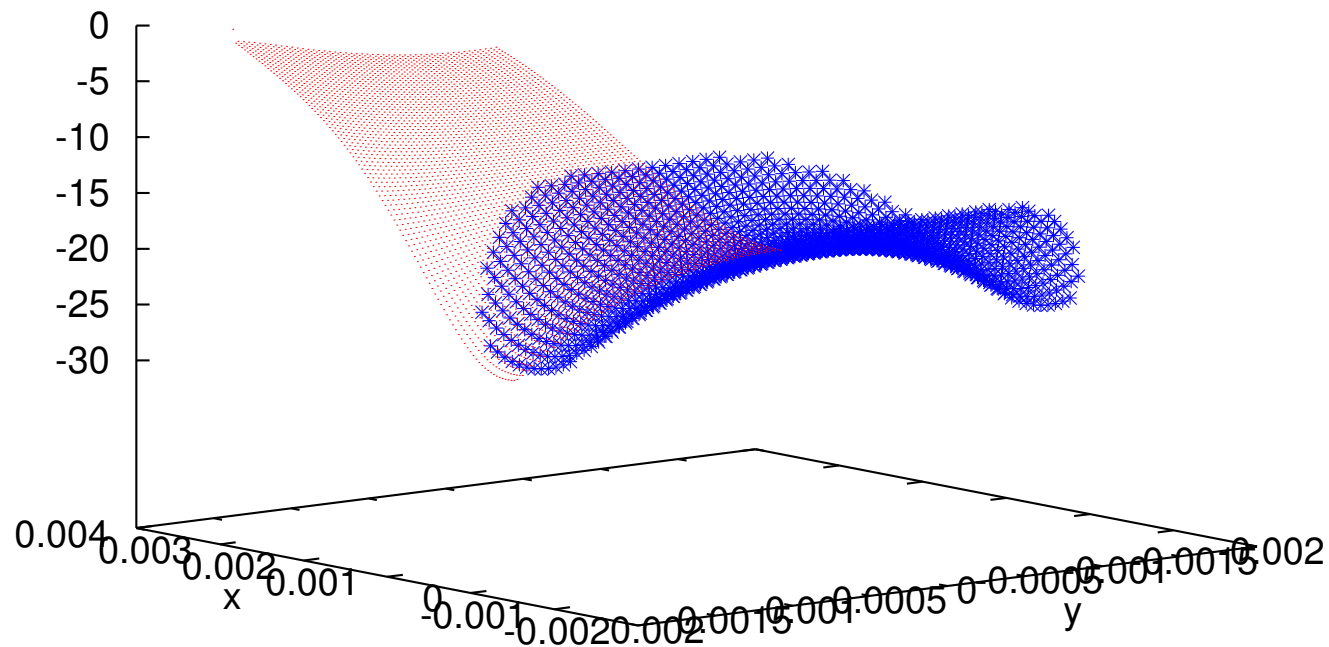
```
'< ./gdfidl2datafile < warner.dat | ./datafile2gnuplot.m '  
'< ./mpolexp2d 5 < warner.dat | ./apply2d.m '
```

*



```
'< ./mpolexp2d 5 < warner.dat | ./apply2d.m '  
'< ./gdfidl2datafile < warner.dat | ./datafile2gnuplot.m '
```

*



- Calculation of the transverse components of the Wake-Potential

```
$> ./mpolexp2Wpotential
```

```
mpolexp2Wpotential - Reads the Multipole Expansion of a Wake-Potential  
and returns its components at any point, version 0.1
```

```
Written by Andrea Latina <andrea.latina@cern.ch>, Apr 26 2007
```

```
USAGE:  mpolexp2Wpotential wakepotential.dat < input.dat
```

```
Where 'input.dat' is a set of X Y Z triplets
```

```
$.>
```

- In which I calculate the transverse components, applying the Panofsky-Wenzel theorem

- TO DO LIST

- deal with the charge distribution
- include this code into placet