New Features and Performances of the Simulation Code PLACET

Andrea Latina, D.Schulte, P.Eliasson
H.Burkhardt, E.Adli, G.Rumolo, R.Tomas, ...

CLIC Meeting - November 9, 2007

- Introduction and History
- New Features
- Examples and Results

The Tracking Code PLACET

- it is a tracking code that simulates **beam transport** and **orbit correction** in linear colliders, developed by Daniel Schulte
- in what does it differ from MAD
  - **MAD**
    - is more mature, 15 years long history
    - it was originally designed for accelerator design
    - it can do tracking but with some limitations
    - it implements closed-orbit correction schemes
    - it is the standard, many tools were built on top of it (PTC, BIMAD, DMAD, ...)
  - **Placet**
    - has limited accelerator design capabilities
    - it is a real tracking code and implements several collective effects -relevant for LC-
    - it implements orbit correction algorithms and feedback loops
    - it is not symplectic, it is focused on linear colliders
    - it cannot simulate rings
    - it can virtually simulate the complete CLIC machine (Main Beam and Drive Beam)[1]
    - it is fully programmable (its interface is a real programming language)
    - it is open to other codes, modular and expandable

[1] except the Damping Rings

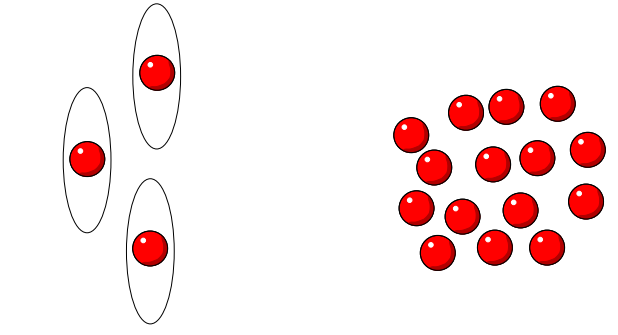
Survey on the Placet's Features

- placet simulates the beam transport and orbit correction in linear colliders
  - it implements the following element types:

    `Quadrupole, SBend, Multipole, AccCavity, DecCavity, Kicker, Beam Position Monitor, Collimator, Crab Cavity`
  - it implements `Girder`
  - it consider the misalignments of the elements : offsets, rolls, and pitches
  - it can track in 4d or 6d
- it takes into account
  - long/short-range **wakefields**
    - in the accelerating structures
    - in the crab cavities
  - geometric and resistive wall wakes in the collimators
  - it implements **synchrotron radiation emission**:
    - Incoherent SR emission : all magnets
    - Coherent SR emission : sector bends
- it can simulate **ground motion** : sample sites, ATL

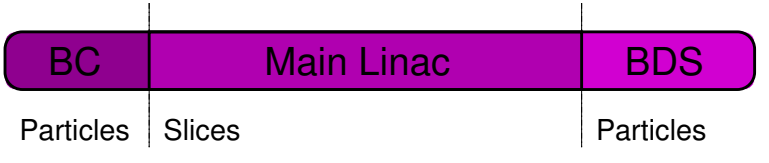
Survey on the Placet's Features

- it implements two beam models



  - **sliced beams**: longitudinal slices with macro-particles and second order moments (faster, LINAC)
  - **single particles**: macro-particles (BC, BDS)

  it can switch between these models during the tracking



⇒ PLACET can simulate: bunch compressor, main linac, drive beam, beam delivery system (including crab cavities and instrumentation), interaction point (using **guinea-pig**) and soon : post collision line

Survey on the Placet's Alignment Tools

- Available alignment techniques:
  - One-to-one Correction
  - Dispersion Free Steering
  - Ballistic Correction
  - RF Alignment
  - Tuning Bumps
    - Dispersion Bumps
    - Wakefield Bumps
- Alignment procedures are present in the code in form of "`Test...`" routines that calculate the average of several machines and take the following arguments:

  `-beam` : beam that has to be used during the optimization

  `-machines` : number of machines to be simulated

  `-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)
- Example:

```
TestRfAlignment
   -beam beam0 -machines 100 -survey load_lattice -emitt_file emitt.rf
```

Example of a Minimalistic PLACET Script

```
source /home/andrea/clic/scripts/clic_beam.tcl

set e_initial 9.0

BeamlineNew
source /home/andrea/clic/scripts/clic_linac.tcl
BeamlineSet -name linac

array set match {beta_x 6.6868 beta_y 2.7269 alpha_x -1.7211 alpha_y 0.7851}

set match(emitt_x) 6.8
set match(emitt_y) 0.1
set match(e_spread) 2.0
set match(charge) 2.56e9
set charge $match(charge)
set match(sigma_z) 30.8

set n_slice 31
set n 9
make_beam_slice beam0 $n_slice $n

SurveyErrorSet -quadrupole_y 50.0 \
  -cavity_y 10.0 \
  -cavity_yp 10.0 \
  -bpm_y 10.0

TestSimpleCorrection -beam beam0 -emitt_file simple.dat -survey Clic -bpm_res 0.1 -machines 1
```

Brief History of Placet (I)

- **1999→2005**:
  - originally developed by **Daniel Schulte** (just for fun!)
  - it was improved with contributions by **Eric D'Amico**, **Nicolas Leros** and
  - later by **Peder Eliasson**, who introduced the tuning bumps for ML alignment

⇒ until 2005, it was practically used only

- at CERN for CLIC simulations (ML+BDS, Drive Beam) and
- at RHUL, by Glenn White, for NLC Feedback Simulations (together with Merlin, that was used for the BDS)

Brief History of Placet (II)

- **2005→today**:
  - **I** took over the code and made some improvements :
    C++ redesign (I needed to..)
    enhancement of the data structures (attributes)
    longitudinal tracking
    particles to slices
    new elements : crab cavities, ...
    Octave interface
    new correction schemes ("generic" correctors, new algorithms)
    MPI Parallel module (exp)
    matching procedure, ...
    new installation procedure, web page...
  - **Daniel** : ground motion, dynamic effects, ...
  - **Helmut** and **Lionel** : HTGEN, halo and tail generation
  - **Giovanni** : Collimator's wakefields
  - **Erik** : Coherent Synchrotron Radiation Emission in Sector Bends, Drive Beam Improvements

  ⇒ at the same time, placet's popularity has grown :
  new collaborations started and...

Collaborations Across Europe

- **LAL/Orsay (FR)**
  - ILC Beam-Beam Fast Feedback Simulation (PAC07)
  - ATF2 Simulations
  - AML/UPL Interface
- **Cockcroft Institute (UK)**
  - ILC / CLIC Crab Cavities (PAC07, EPAC2008??)
- **Ankara University (Turkey)**
  - Gamma-Gamma interaction at the ILC
- **John Adams Institute, Oxford (UK)**
  - ILC / ATF2 Feedback Systems (EPAC2008??)
- **Royal Holloway University of London (UK)**
  - ILC Collimators, BDSIM integration (PAC07, EPAC2008??)
- **Daresbury Lab (UK)**
  - ILC Collimators (PAC07)
- **PSI (CH)**
  - CLIC Bunch Compressors (PAC07)

	placet	placet-octave (improvements..)
Core	C	C++
User Interface	Tcl/Tk	Tcl/Tk + Octave
Tracking	4d Slices ⇒ Particles (LINAC-BDS)	6d Slices ⇔ 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, ...

Element's Attributes

- Defined a hierarchy of C++ classes to describe the element types. Each type derives from 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             Enalbe 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"
```

Element's Attributes

- One 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
```

⇒ This allows for 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 overs 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 continue the Tcl script execution afterwords

placet-octave's instructions set

- placet_get_name_number_list(beamline, name)
- placet_get_number_list(beamline, type)
- placet_get_bpm_readings(beamline)

- placet_get_response_matrix(beamline, beam, correctors, bpms)

- placet_test_no_correction(beamline, beam, survey)
- placet_test_simple_correction(beamline, beam, survey)

- placet_vary_corrector(beamline, beam, corrector, variation)

- placet_element_get_attributes(beamline, element)
- placet_element_get_attribute(beamline, element, attribute)
- placet_element_set_attribute(beamline, element, attribute, value)

- Tcl_Eval(expression)
- Tcl_GetVar(name)
- Tcl_SetVar(name, value)

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
    $\Rightarrow$ 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

Example: CLIC BC2 Simulation

⇒ Beam parameters:

	unit	entrance of Bunch Compressor 2	entrance of Main Linac
energy	[GeV]	9	9
unc. energy spread	%	2.0	2.0
no.of particles	[#]	$2.56 \cdot 10^9$	$2.56 \cdot 10^9$
charge	[nC]	0.41	0.41
sigmaz	[μm]	250	30
h. norm. emittance	[nm·rad]	570	680
v. norm. emittance	[nm·rad]	4	5

⇒ BC2 Parameters:

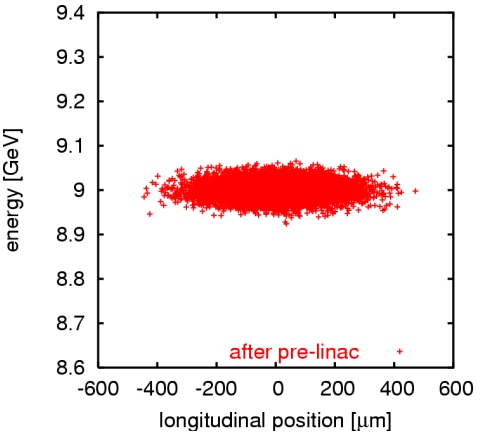
- Magnetic chicane:

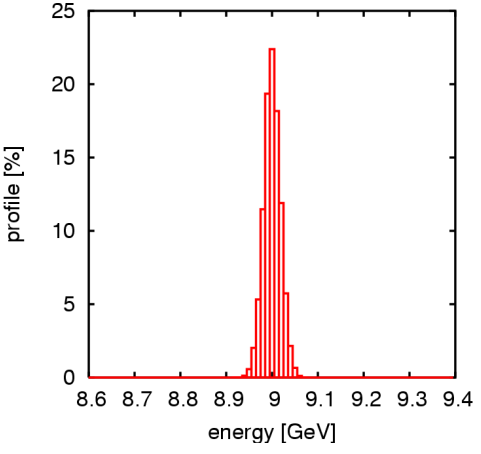
$\mathbf{R_{56}} =$ -14 mm
norm. $\mathbf{s} - \mathbf{E}$ corr. = -70.5 1/m

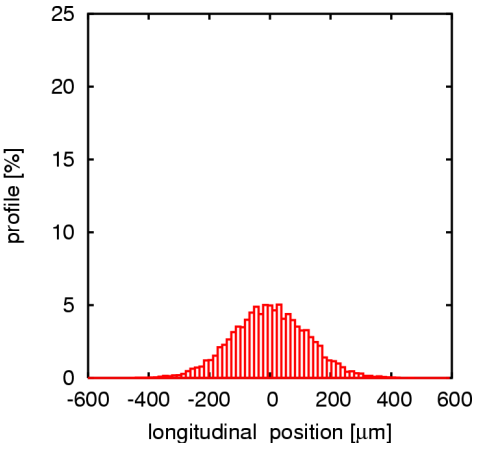
- RF system:

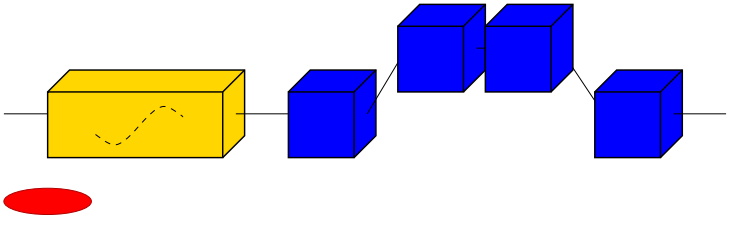
$V =$	1009.14 MV	integrated voltage
$\Phi =$	$k\pi$	phase
$f =$	30 GHz	frequency

Example: CLIC BC2 Simulation 1/3

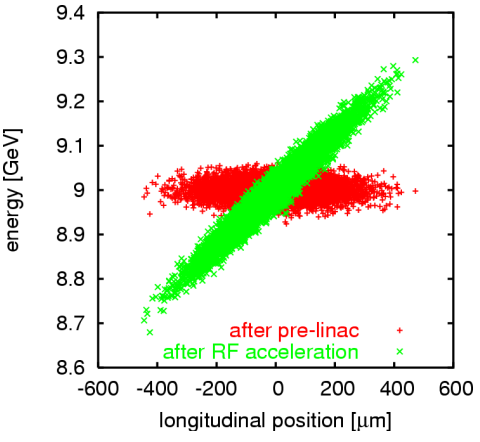


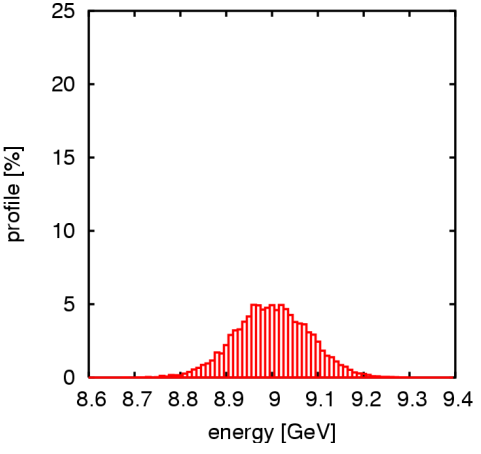


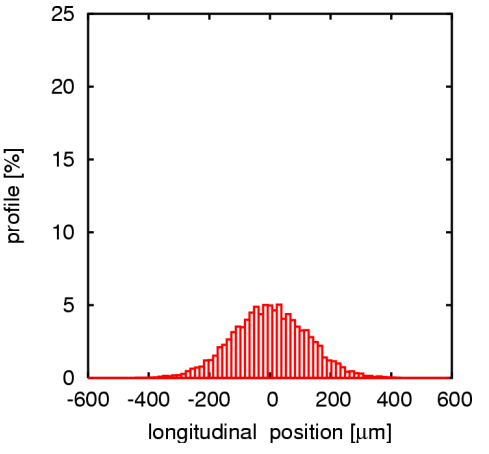


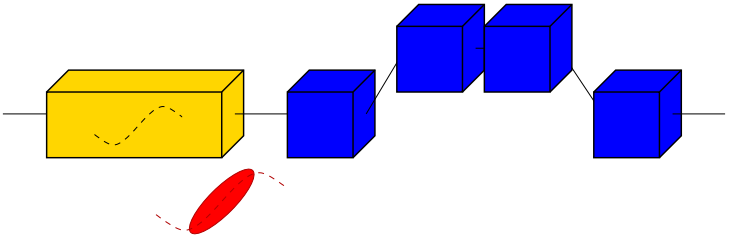


Example: CLIC BC2 Simulation 2/3

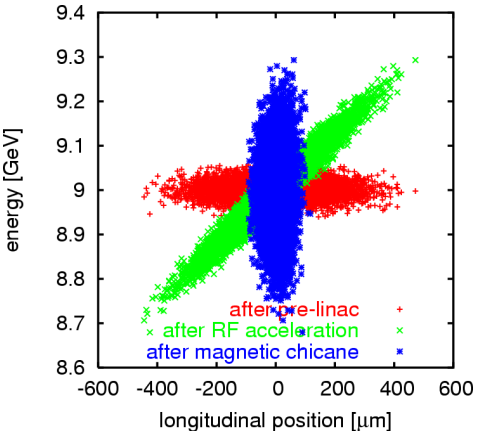


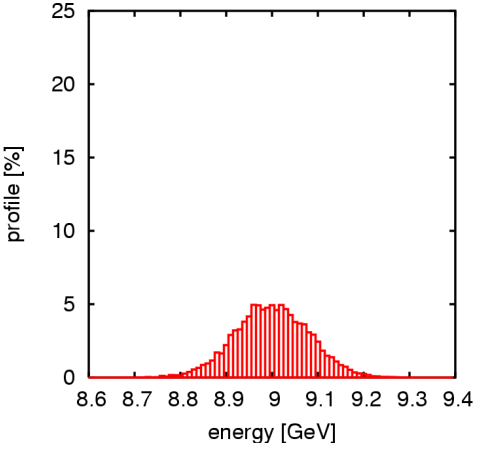


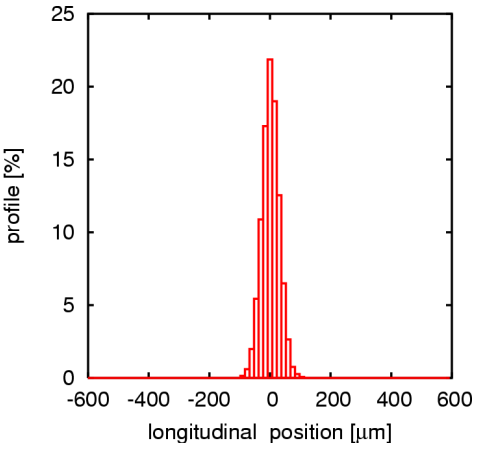


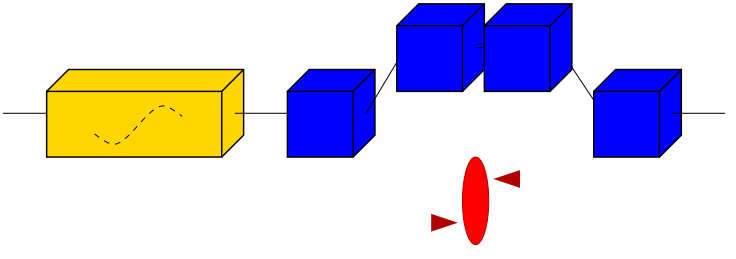


Example: CLIC BC2 Simulation 3/3



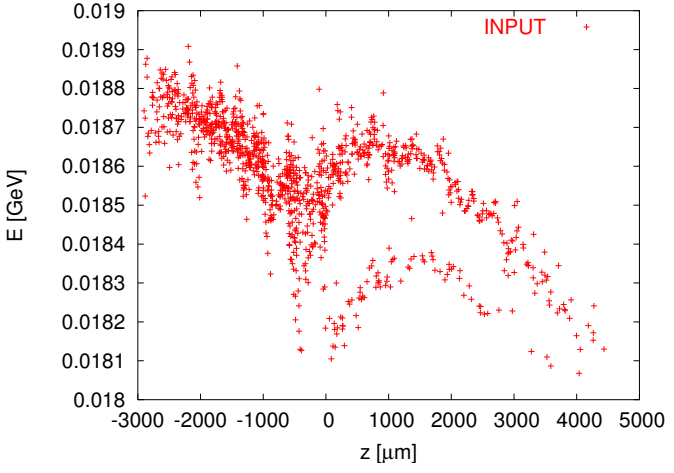


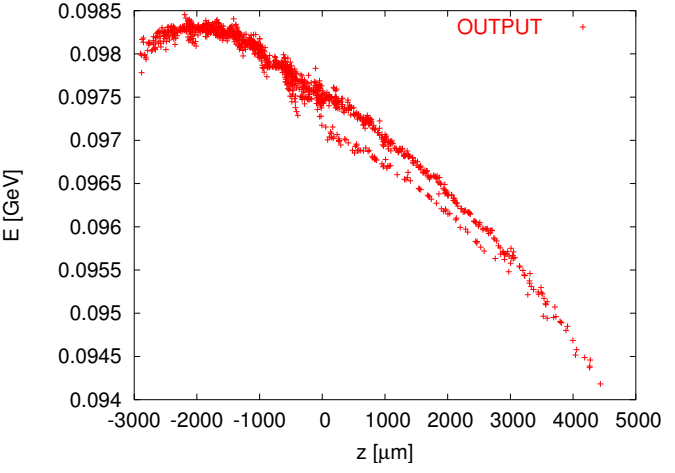




Example: Longitudinal Phase Space in the CTF3 Linac (H.Shaker)

- Wakefields of the accelerating cavities are taken into account





Bunch to bunch longrange wakefields are taken into account.

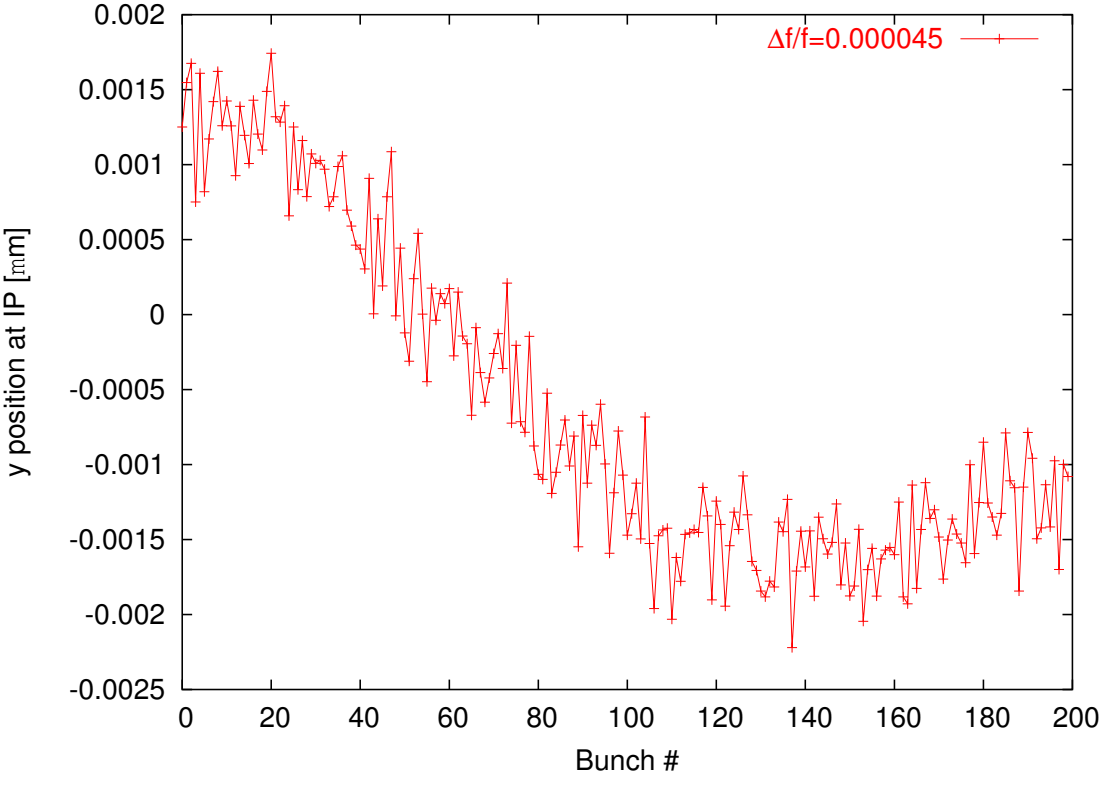
⇒ PLACET has already been tested against real measured data from CTF3:

1) PLACET Response Matrix used to predict BPM readings

2) Measured Response Matrix using PLACET to calculate the correction

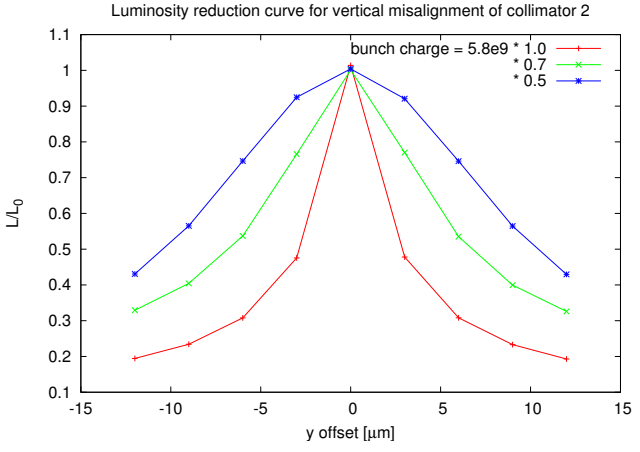
Example: ILC Crab Cavity Wakefields (G.Burt, A.Dexter - Cockcroft)

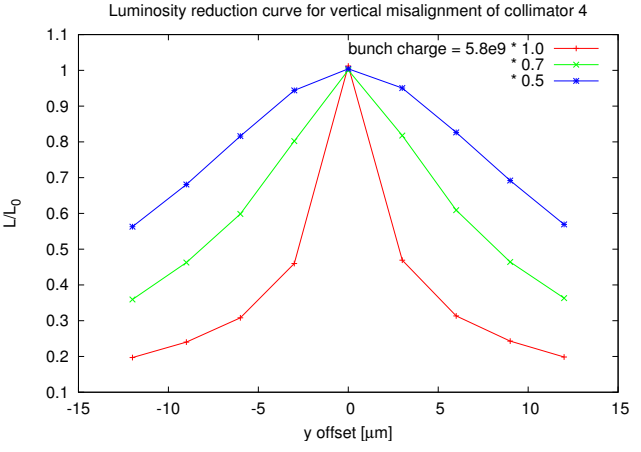
Vertical offset at the IP due to longrange wakefields in the Crab Cavities in case of frequency jitter.

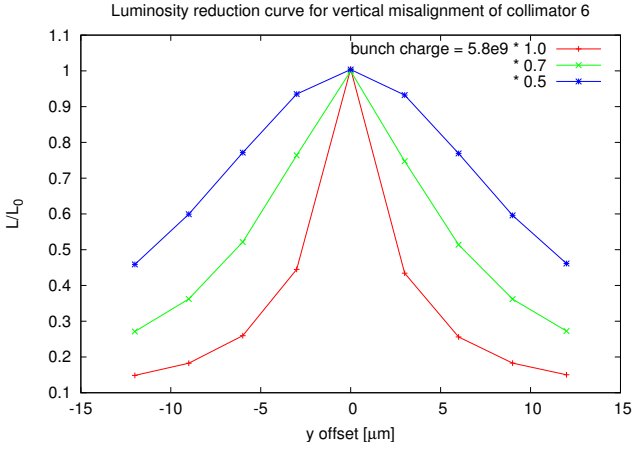


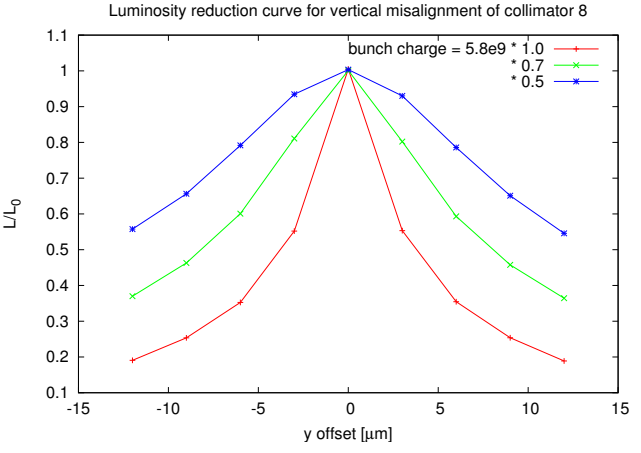
Example: CLIC BDS Collimator Wakefields (G.Rumolo)

- Luminosity reduction curves due to vertical misalignment of the collimators



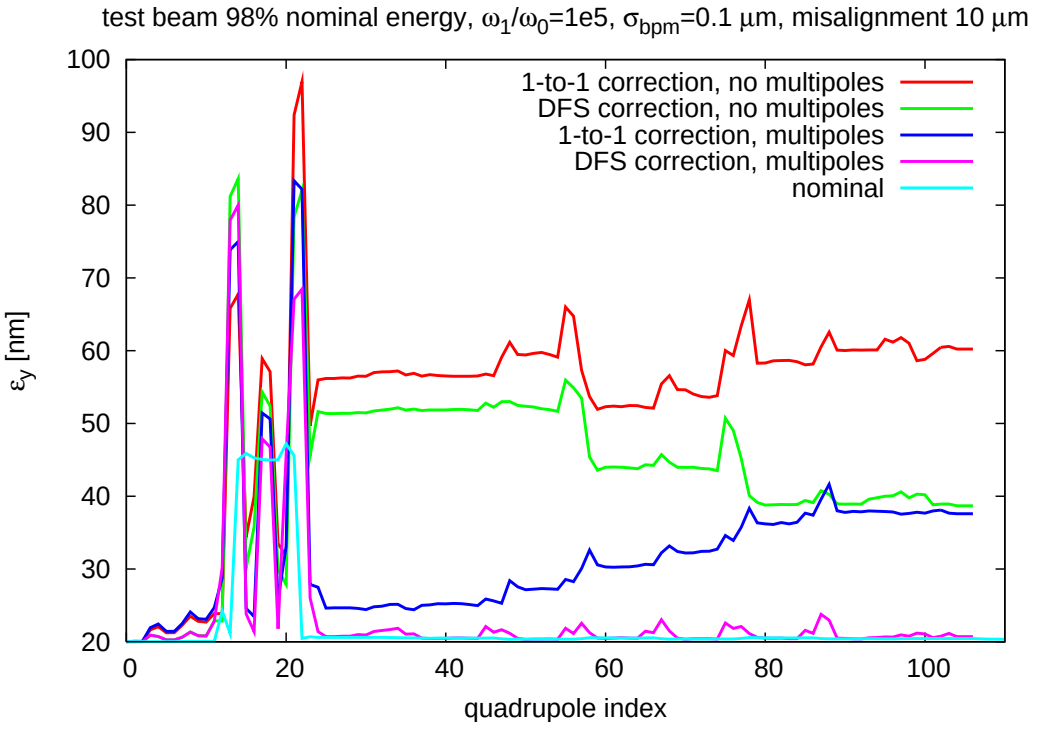






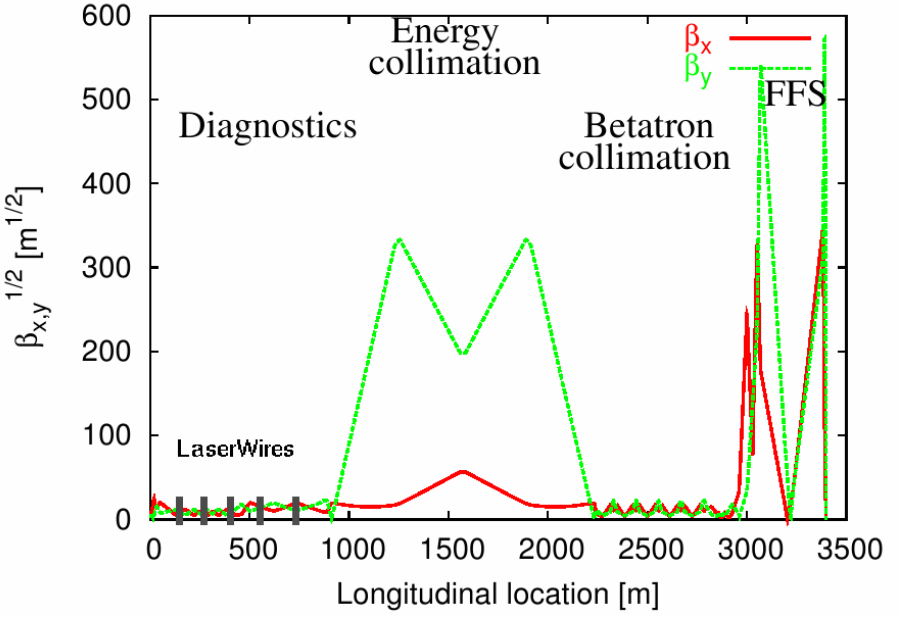
DFS in the BDS Collimation System (R.Tomas)

- using one test beam with $E = 98\% E_0$
- alignment in 4 steps...



$\Rightarrow$ final emittance growth is $\boxed{\Delta\epsilon = 0.7 \text{ nm}}$

Diagnostics in the CLIC BDS (R.Tomas)



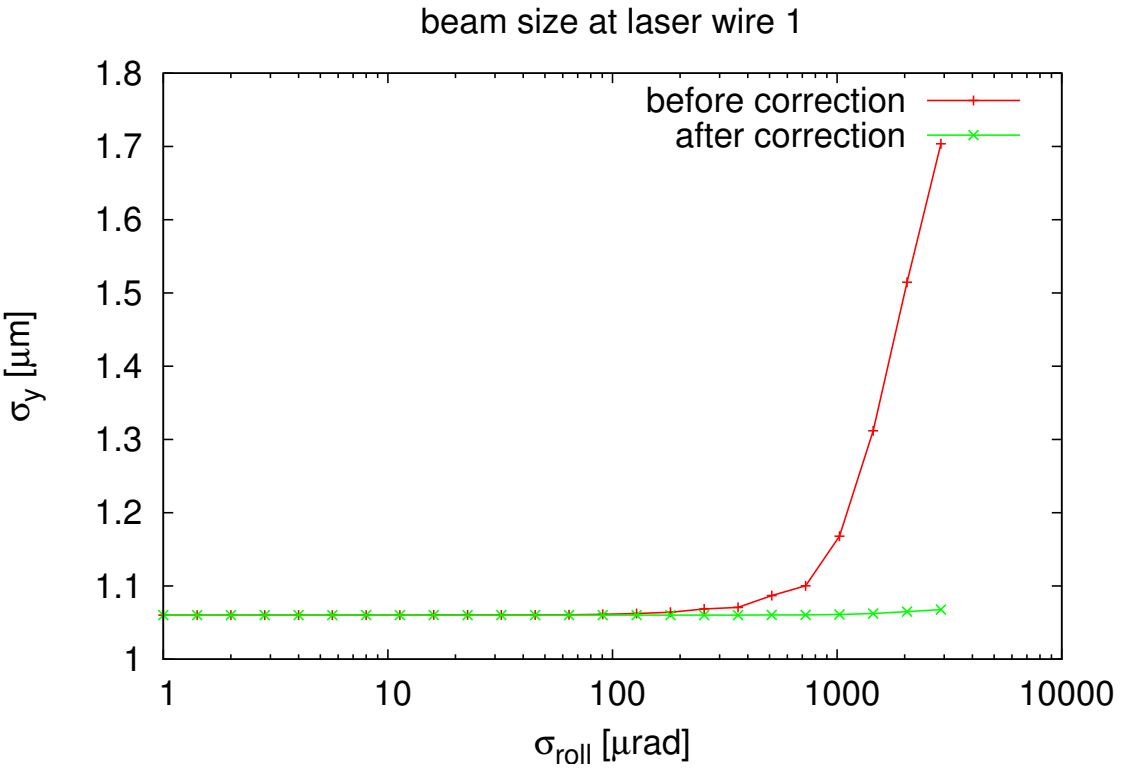
We want to remove the couplings due to quadrupole rolls, using skew quadrupoles.

We minimize the following merit function:

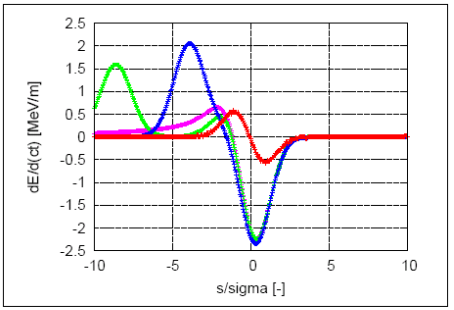
$$M = \sum_{i=1}^{5} \frac{\left(\sigma_{y,i} - \sigma_{0\,y,i}\right)^2}{\sigma_{0\,y,i}^2}$$

Skew Quadrupoles Optimization

- Beam size at the laserwire as a function of the Quadrupoles roll



CSR Module (E.Adli)

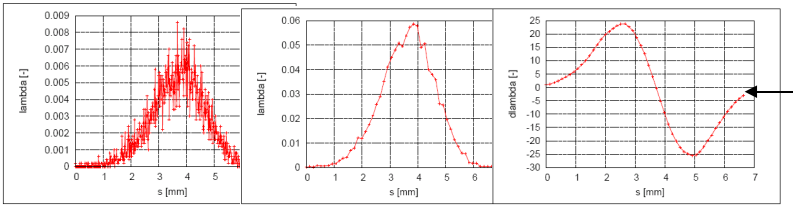


s=[0.05,0.14,0.18,1] m

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

CSR can be activated in the Sbend 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)

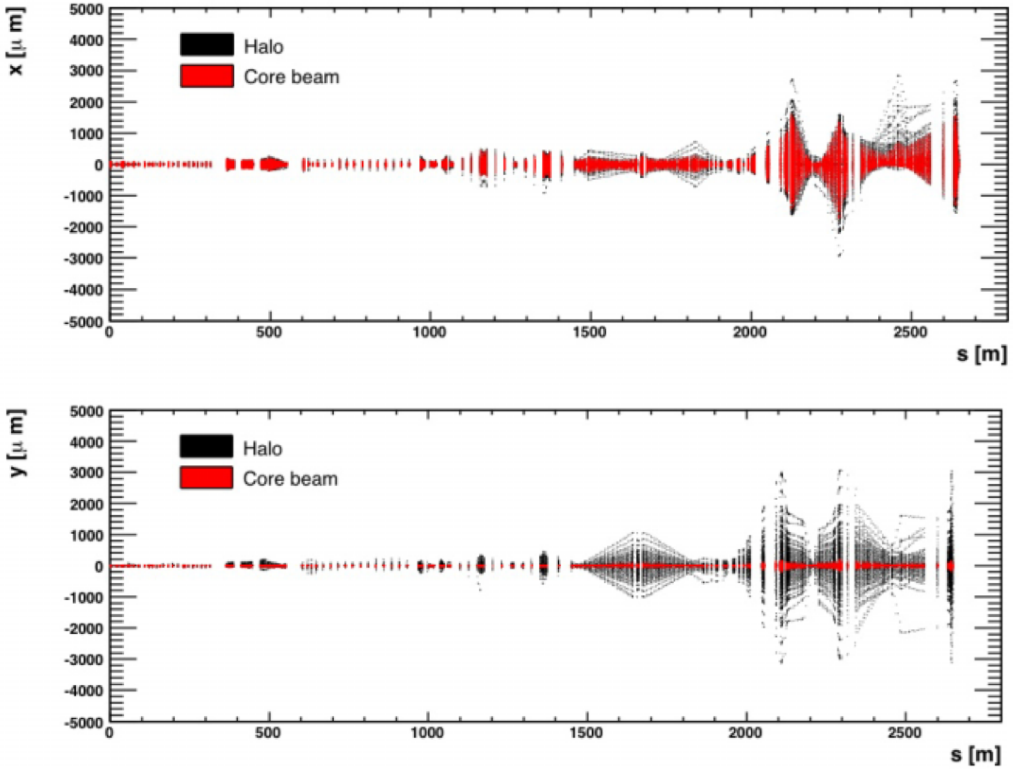
Easy to use: described by on-line help:

```
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]
```

HTGEN

Halo generation and tracking (H. Burkhardt)

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



$\Rightarrow$ beam-gas scattering form LINAC and BDS: a fraction of 10^{-4} of the particles impacts on the spoilers

Overview of the new features

- 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

- 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 wake-field kick
- Future Developments

  $\Rightarrow$ Complete the Static Alignment of the Final Focus (I and Rogelio are obtaining very good results!)

  $\Rightarrow$ Test of Dispersion Free Steering in CTF3

  $\Rightarrow$ Can placet-octave be used in the 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`

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

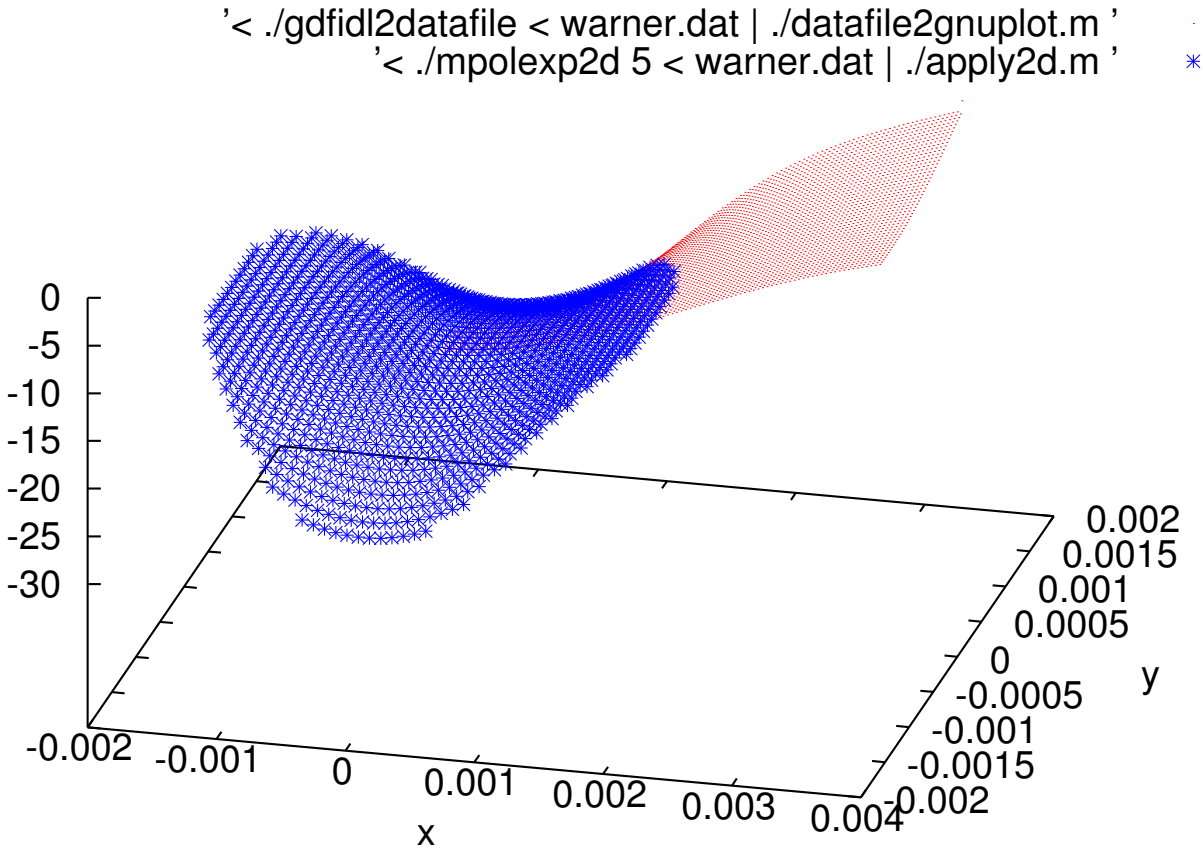
- First problem : data compression, the amount of data produced by GdfidL can be as big as several hundreds Mb's per each collimator
  - we need to make a multipole expansion of the Wake-Potentials
  - we need to calculate the wakefield kick from the expansion

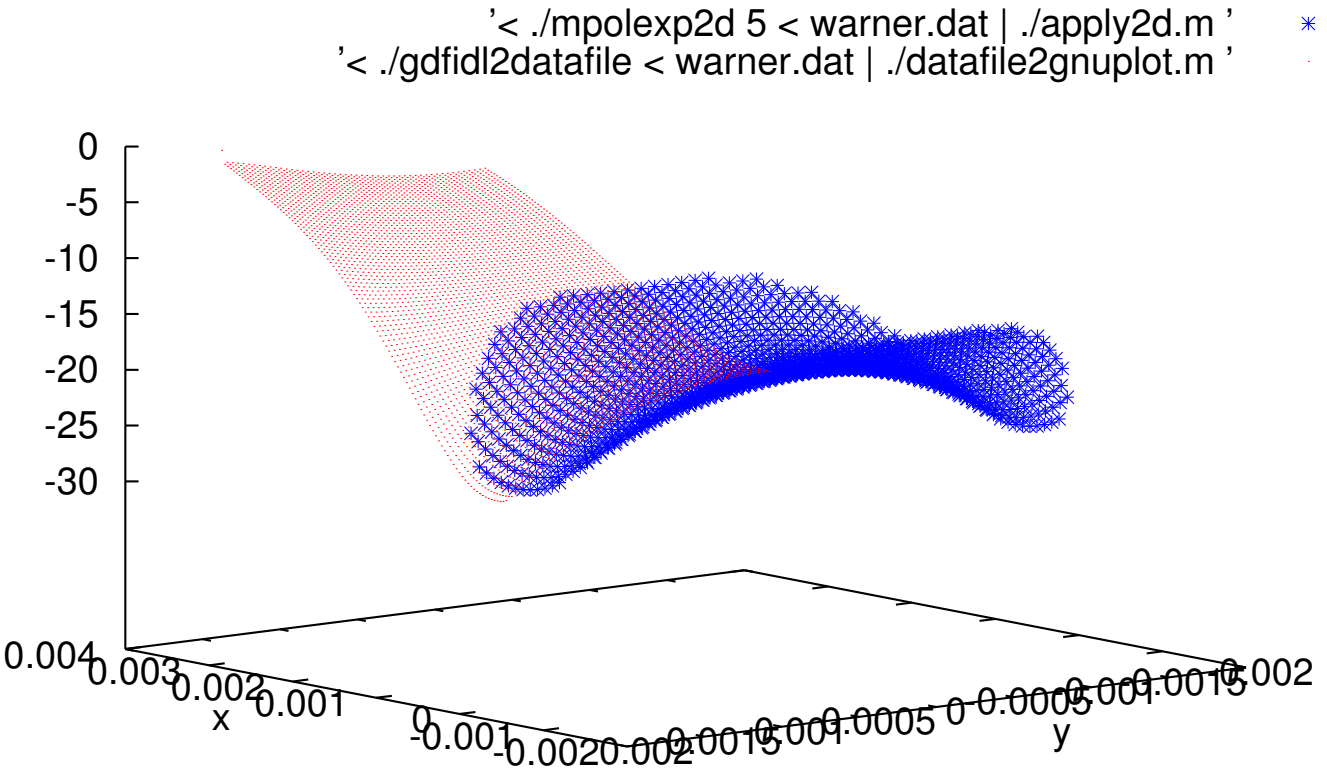
After several discussions with Warner, we agreed about a file format for the (uncompressed) wake-potentials (GdfidL $\longrightarrow$ multipole expansion)

- 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
mpolex2d - 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 '
0
-5
-10
-15
-20
-25
-30
-0.002
-0.001
0
0.001
0.002
0.003
0.004
x
0.002
0.0015
0.001
0.0005
0
-0.0005
-0.001
-0.0015
-0.002
y


'< ./mpolexp2d 5 < warner.dat | ./apply2d.m '
'< ./gdfidl2datafile < warner.dat | ./datafile2gnuplot.m '
0
-5
-10
-15
-20
-25
-30
0.004
0.003
0.002
0.001
0
-0.001
-0.002
x
0.002
0.0015
0.001
0.0005
0
-0.0005
-0.001
-0.0015
-0.002
y

- 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