TOUTATIS
A 3D RFQ CODE

Ver 1.3

Romuald Duperrier
CEA Saclay 91191 Gif sur Yvette FRANCE

July 27, 2015

1 TOUTATIS

1.1 Copyright and disclaimer

1.2 Acknowledgments

I acknowledge Nicolas Pichoff and Jean-Michel Lagniel for the fruitful discussions in front of a black board.

2 A few words before...

This code has been written to make it easy the understanding of the high intensity RFQ beam dynamics. An other goal was to provide to RFQ designer an accurate tool to simulate this beam dynamics. Several interrogations deal with the validity domain of analytical formulations or with simplification used in numerical algorithms. TOUTATIS has been written in the aim to minimize hypothesis in the transport simulation. The purpose of this introduction is not to give a complete description of methods used by the code, but we can recall a few major points: the fields are computed using 3D grid via a Poisson solver, this allows to compute image effect, space charge forces, and cavity fields taking into account the real shapes of electrodes (coupling gaps). No symmetry is used for external fields calculation, this means that mechanical defects can be included in the dynamics. The integrator of the motion is symplectic, this avoids phoney behaviour of emittance which may occur with a leap-frog with large step. The time is the independent

parameter, this is the only way to compute accurately self consistent forces in a time periodicity focusing channel. The interested reader could have a look in the reference [1] to get more details. The reference mentions also several cross checking results with experiments. An effort has been performed in order that PARMTEQM [2] users may easily run TOUTATIS. Good luck!...

3 Before start

3.1 Machine and OS Requirements

The physics part of the code is written in ANSI C language and the interface in the multi-platform language Tcl/Tk. This allows to compile on many platforms (unix, windows and Macintosh). At present time, TOUTATIS has been only compiled for the following OS: linux, Mac OS 10.6 and Windows. For the RAM requirement, I suggest at least 128 Mo.

64bits Linux case: In some recent linux distribution the 32bits library package is not present by default. Toutatis installator and Toutatis code need them, thus to install this requiered package type the following commands:

sudo dpkg –add-architecture i386

sudo apt-get update

sudo apt-get install libc6:i386 libncurses5:i386 libstdc++6:i386 libglib2.0-0:i386 libcups2:i386 libfontconfig1:i386 libXrandr2:i386 libXrender1:i386 libXi6:i386 libSM6:i386

Mac OSX greater then 10.6 including Mavericks: XQuartz installation is requiered and to launch GUI version of toutatis use the following consol command:

open -a /Applications/Toutatis.app –args /Applications/Toutatis.app/Contents/MacOS/example.inp

3.2 Software installation

The distribution is precompiled, so it is just necessary to uncompress the version you need whatever the platform for a graphical use. Anyhow, in case you meet graphical libraries issues, it is always possible to use the code in a console mode. Use the "*Toutatis_console_platform*" executable. Once all these stages are complete, you can run either by double click on a .inp file or in a console with the command "*Toutatis -i example.inp*". For Windows systems, an association between the extension .inp and the executable Toutatis in the bin directory is necessary to allow the double click execution. All .inp files must be executed from the Toutatis root. With the MacOS platform, the association is directly managed with the bundle. For linux distributions, it is recommended to execute the code from a console. Association for a (double) click execution is highly window manager dependant.

If the executable "*CheckKey*" and a valid activation key are not present in the executable directory, only the trial version will operate. It means that the number of

macroparticles will be limited to 1000, only 4 time steps per period and a coarse mesh will be possible. The activation key is automatically generated when the code is launched, if it is connected on the WEB and if the user has been registered in the CEA database of users. In order to use the code on a computer with no network access, you have to bring this key file with the executable to keep the user rigths.

3.3 Technical Support and Code updates

It would be appreciated to receive all remarks if you discover a (or several!) bug(s!) or if you cannot find the information you need. Send an e-mail to the following address to didier.uriot@cea.fr. For solving issues, I suggest you attach your input file and specify the used platform. To get updated version, you can use the TOUTATIS download page.

3.4 Known bugs and problems

Depending on the geometry, the code mesher may crash without any message box. Use it in a console may help to find the issue.

4 Getting started

In the root of Toutatis install directory or in the directory /Toutatis.app/Contents/MacOs with MacOs, the file *example.inp* describes a RFQ structure which is formated for Toutatis simulations. It compiles several commands with comments. Most of them are detailed following.
Simulation is done from the inside of the entry plate to the inside of the output plate.

The *Coupling gap* command allows to include several radio frequency coupling gaps with elliptical shapes (see figure 1). The position parameter is the longitudinal location of the gap center (in meter). The gap parameter is the distance between the two segments. The width and height parameters are dimensions of the ellipse (see figure 2).

NumberOfCouplingGap defines the number of gap used.

TheGeometryFileFlag *type N dz*

- if type 0 → default, 50 steps per cell
- if type 1 → read geometry file, N is the complete path to the file without blank
- if type 2 → if (N==0) then the step size is dz, otherwise N steps by cell

The *theGeometryFileFlag* command permits to use an ASCII file (.vane) for the geometry of each pole. The format is composed by 17 columns. The first column is the longitudinal location of the other columns parameters. The 16 columns are composed by four blocks of 4 columns (1 block per pole). Each block is divided as follow:

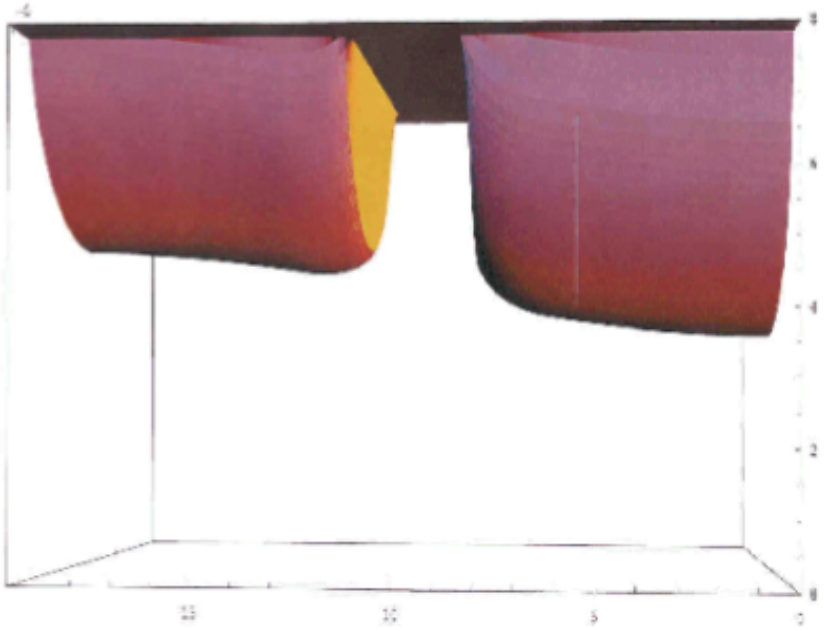


Figure 1: 3D view of RF coupling gap with elliptical shape.

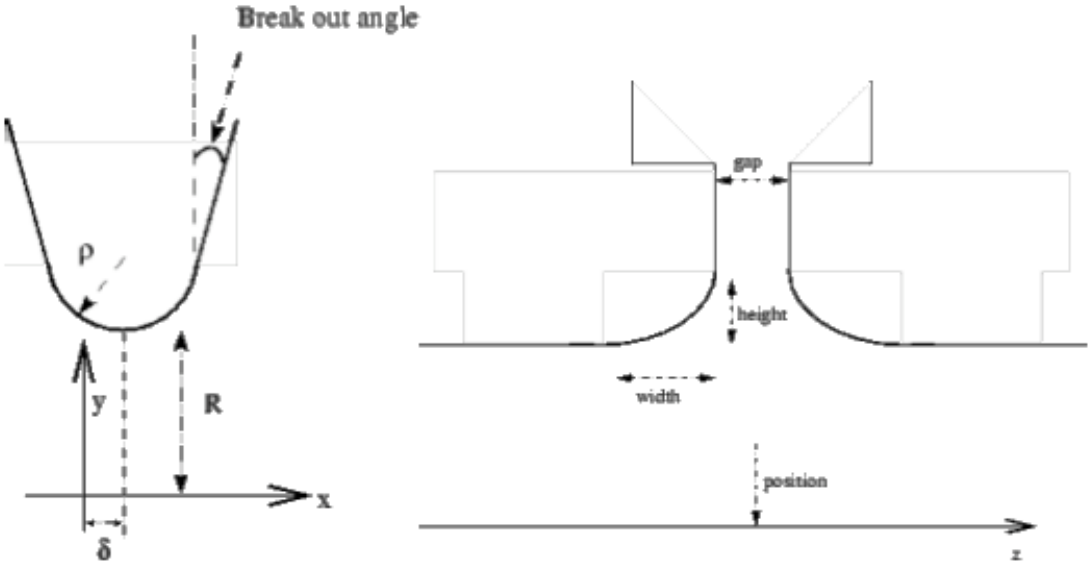


Figure 2: Scheme for the gap description.

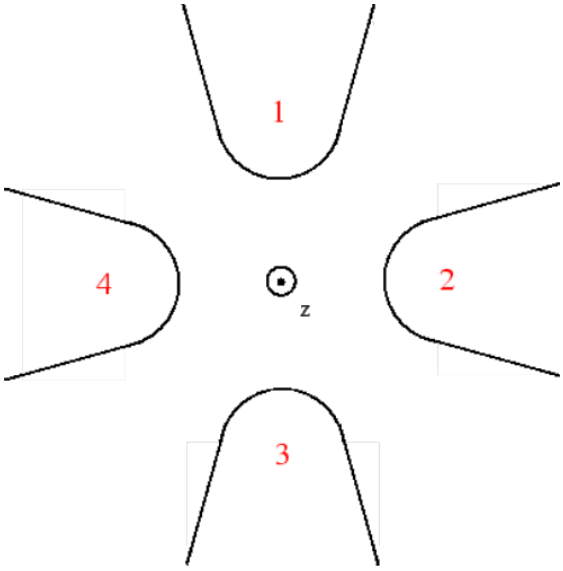


Figure 3: Convention for pole order description in .vane file.

R (meter) ρ(meter) V (volt) δ(meter)

R is the distance between the axis and the pole (longitudinal profile included), ρ is the curvature of the electrode, V the pole voltage equal to u/2 if u is the inter vane voltage (take care about the sign in order to induce quadrupolar and dipolar symmetries), and δ the shift of the pole in the direction perpendicular to the electrode (see figure 2). This last value can be negative. Each pole is individually managed. The sequence for blocks is described by the figure 3.

If geometry file is not given, the code creates a toutatis.vane which describes the structure built from the input file. The longitudinal step may be constant (use *dz* entry in meter) or a fraction of the length of the cell (use *Number of step / cell*). See the example file for the details.

If a geometry vane file is specified, its length must be the same than the vane file generated by the inp file alone. The geometry file length is the RFQ length - the front-end gaps length. In other terms, the inp file is still read and only the copper part of the geometry will be replace by the geometry defined in the vane file.

The *theLossesCriteriaFlag* command supposes two possible criterion: Electrodes or Square. The first choice implies that a particle is declared lost when it hits the pole surface. This is the default. The second choice is the PARMTEQM convention: a particle is declared lost when it gets out of a square with an edge equal to two times the minimum aperture. This last choice allows to quantify on a particular design the impact of such approximation.

Mesh refinement level with the *theAccuracyFlag* command allows the user to define how the finest grid is refined. According to the multigrid method, the different possi-

Mesh refinement level	0	1	2	3
Number of nodes in finest grid for Laplace	33^3	65^3	65^3	129^3
Number of nodes in finest grid for Poisson	17^3	17^3	33^3	65^3

Table 1: Correspondence between mesh refinement level and the number of nodes in finest grid.

bilities for the number of nodes per direction are as 2n+1 law. The table 1gives the correspondence.

NStep sets the time discretization. It must be an even value. The periodicities for Poisson and Laplace calculations options permit faster but less accurate calculations.

theAcceleratedFlag permits to select particles to insert in .dst output file and for emittance calculations. The limit for the accelerated is the energy acceptance defined by the following formula:

$$\Delta W \simeq 1.6\sqrt{\beta_s^2 qAVE_0(\Phi_s cos(\Phi_s) - sin(\Phi_s))}$$

with β_s, the reduced speed, and Φ_s, the relative phase at the cell middle, for the synchronous particle; A, the accelerating efficiency, V, the inter-vane voltage, E_0 the rest mass and q the charge.

Compression_ Factor (-1/0/1/2) command permits to generate a "plt" file. If it set to -1, no "plt" file is generated, otherwise (0/1/2) are the compression level of "plt" file (see PlotWin or TraceWin help for "plt" file details). This file size could rapidly become huge.

Dipole PerCentVoltage SPStem command permits to add a dipolar horizontal component to the field allowing to simulate a bad tuning of this mode.

$$E_x = PerCentVoltage.\frac{1}{4}.\frac{V}{\rho}.sin(wt).cos(\frac{\pi}{SPStem.z})$$

Monopole PerCentVoltMono SPStemMono command permits to add a monopolaire component at the field at the end of the RFQ allowing to simulate a bad tuning of this mode.

$$E_z = PerCentVoltMono.e^{-\frac{(z-RFQFinish)^2}{2.SPStemMono^2.}}.sin(wt)$$

TheTrancellFlag if it is set to 1, the last cell is a transition cell.

TheRandomFlag command permits to set fix the seed for particle generation.

TheLossesCriteriaFlag, if 1, a square (like old parteqm) is used to estimate beam losses, if 0 (default) it is electrode shape.

The2TermSFlag command permits to set modulation not sinusoidal but defined as a scalar equipotential 2-term potential well known in the world of the RFQ (see CERN School Acc).

WallAperture defines the plate aperture radius (m).

TheStartingGmodualtion (defaut 1) sets to -1 allows to change the starting modulation convention, it is equivalent to turn the RFQ by 90 degrees.

if *DatFlag* is set to 0, an input *rfq.dat* file (input TraceWin file) is generated (Use TraceWin code to simulate an RFQ suppose user get a Toutatis license).

if *TheFourRodsFlag* is set, the modulation is performed by periodic reduction of the radius of the poles and not by a distance from the axis of the pole. This allows to simulate four rods RFQ bar.
if *The2termsFormulationFlag* is set, fields are not 3D calculate numerically and the 2 terms potential is used (analytical formulation).

SemiWidthWall defined the half thickness of the input and output plates, (0.0025 m is the defaut). It is used for field computation at the begenning of the structure and don't change the RFQ geometry

RFQFinish defines the z location for distrib.dst, RFQ length is the default

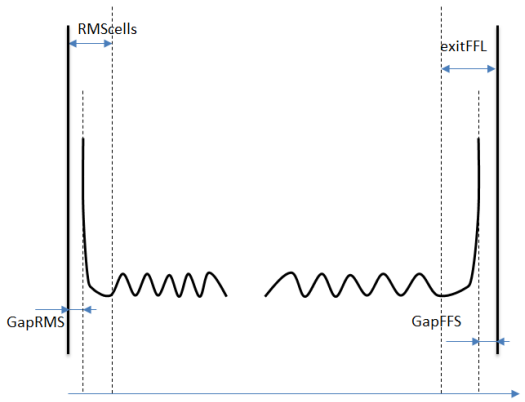


Figure 4: *Front-end cells*

GapRMS see figure 4

GapFFS see figure 4

NbRMSCell: number of cells for the RMS. Don't include cell 0. The profile of the RMS cells is defined by an analytic formula frozen in the code. If you want another profile, use an extern vane file to define it.

Smoothing command permits to apply your low pass filter to smooth *a, R0, m, Φs, V* and ρ.

TheDistribution flag none commands permits to replace the default 4D WB input ditribution (*flag=k*) by an input dst file (*flag=f*) and *none* the complete path of the file.

theSpeciesFlag n permit to inject several beams with different q/A if n >0, a sequence of n lines with a block which describe each q/A with its current fraction, reduced Mass, charge, a boolean to indicate to Toutatis that it is the reference q/A. Example: to transport a reference beam of protons with 80 mA and 20 mA of H2+ and 5 mA of H3+, you have to write:
theSpeciesFlag 3
0.761 1.007 1 1
0.19 2.014 1 0
0.049 3.022 1 0
The total number of macroparticles is set with the input command, the kinetic energy is the same for all (see *linac* command in example.inp) , the sum of the current fraction must be equal to 1. In case the command *theDistribution* is set to *f* for each line n in the block, the code will search a file distribn.dst (here, distrib1.dst, distrib2.dst and distrib3.dst). If the command *theDistribution* is set to *k*, a 4D water bag is generated with the same twiss parameters. In case, you use .dst files, take care about the compatibility of the respective declarations for the mass, number of particles and current.

TheThoR0RatioInRMS command set to 1 permits to keep constant the ratio between Rho and R0 in the RMS.

TheBreakOutAngle see figure 2.

Diaphragme defines the aperture radius allowing to cut the beam entered in the RFQ.

Harmonic command permtis to take into account that the input beam has the same frequency as the RFQ, example if the beam bunch frequency is 176 MHz and the RFQ RF is 352 MHz set *Harmonic* = 2.

TheSpaceChargePeriod defines the refresh rate of space charge computation, for instance, 2 means every 2 steps.

OutputFileField field_file X max (m) Y max (m) NStep bin_flag, permtis to generate a 3D file of the RFQ fields. *NStep* defines the number of step between -X/Ymax and X/Ymax, the Z step is variable. The output file format is ASCII (*X(m), Y(m), Z(m),*

Ez (V/m), Ey(V/m), Ez(V/m)), if *bin_flag*=0, otherwise the output format is binary (6 x double). The output is organized in row-major storage (Z, Y, X). if *Nstep* is set to 0, the output file is 1D field, *Ez(z, Xmax, Ymax)*.

Exitfffl see figure 4.

Vfac allows to reduce the RFQ tension $V_{new} = Vfac.V_{vane}$.

Rho defined the ratio ρ/Ro along the structure.

Scheff defined the beam current in mA.

END to finish the command section.

Most of the commands must be located before the *END* command, few have to be located between *END* command and the channel block, some others must be located after the channel block. Respect the order defined in the *Inp* example file.

4.1 The channel block

In this section, we call "channel block" the part of the input file which describes the cells parameters. The channel block format is the same as PARMETQM input file. But several columns are ignored by TOUTATIS. Only the following parameters are taken into account:

cell number, Voltage[kV], null, null, null, A, Φ_s, a[cm], m, R0[cm], ρ[cm], null, cell length[cm], null, null, null

It is important to remark that the 'null' values are not used but are read. The code supposes that the number of columns (16) is the same compared to an output file (.out) of PARI, the Los Alamos code. The cell number 0 are not use in the geometric vane generation, it's just a starting point similare to PARMATEQM but not used in Toutatis code. The Phase parameter is not use by Toutatis. It is needed to generate TraceWin input file.

5 Running the code

5.1 Running the code in console mode

The code can be run in a console without graphical outputs. It supposes that the input file is configured. The syntax is "$INSTALLPATH/toutatis -i inputfile -nox". Or, you can

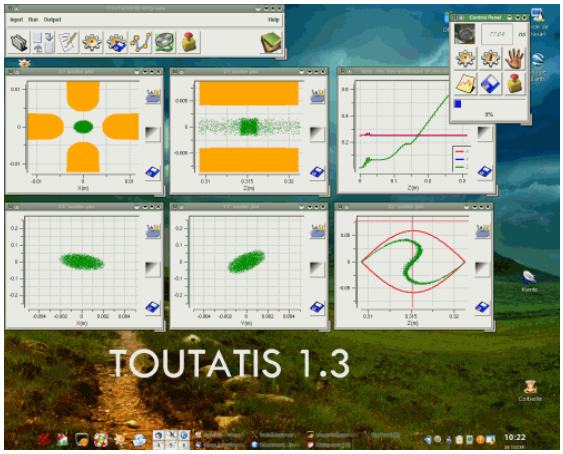


Figure 5: TOUTATIS launching logo.

use the exe Toutatis_console_xxx. Then only "$INSTALLPATH/toutatis_console_xxx -i inputfile " is needed.

This possibility is interesting for multiparameter calculations. The executable "Check-Key" and a valid key must be present in the executable directory, otherwise a trial version will be executed.

During calculations, several informations are printed in the console such: the RFQ length, the peak field in Kilpatrick unit and the evolution of transmission in respect to the longitudinal position.

5.2 Running the code in graphical console mode

By omitting the -nox option in a console or by double clicking a .inp file (previously, on win32 and Gnome/Kde platform, an association has to be created with the executable Toutatis in bin), the code is launched graphically. The splash logo should be displayed (figure 5).

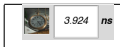
A few moment after, it depends on the required calculations, the control panel appears. The following section describes this widget.

5.3 The control Panel

During the run, the time loop can be controlled using the control panel (figure 6). This widget is able for example to stop, to perform a pause and to restart calculations.

The following table summarizes the different control panel widgets:

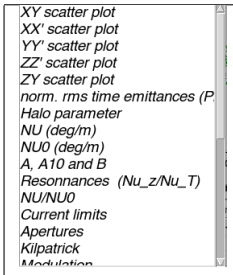
Figure 6: The control panel.

Icon	widget type	function
	label	Indicates the current time in the simulation
	button	Launches the run until the end or an other action is required
	button	Performs one calculation step and refreshes all plots
	button	Performs a pause in the run and refreshes all plots
	button	Displays or undisplays the diagnostics list box
	button	This option permits to dump the current distribution in a .dmp
	button	Quit the calculation
	progress bar	Indicates position in respect to the cells number as %

5.4 The diagnostic box

This box compiles all plots. Each plot may be called by double clicking on its corresponding title or by selecting the title and push the open button (see table below). This box may be closed using the window manager (click on the white cross in the window border for the figure 7) or by pushing the diagnostics button of the control panel. All

windows are closed if the user pushes the kill button. The following table summarizes the diagnostics box commands:

Icon	widget type	function
	list box	Compiles and permits selection of plots
	button	Open the selected plot
	button	Close all windows

5.5 2D plot window

During the run, scatter plots and curves may be displayed. For a few windows, RFQ structures are drawn statically (X envelop plot) or dynamically (XY scatter plot). The figure 8 and 9 show a XY scatter plot and a rms time emittance plot respectively.

In each window, several successive zoom may be performed. The zoom doesn't work with the MacOS and linux version, but a rescaling will provide the same effect. This can be done by defining a rectangle with the left button of the mouse (see figure 10). By clicking on the right button, the previous views are recovered. Several plots embed a legend located by default at right bottom corner. If, for any particular reason, user wishes to move this legend at other location or outside the plot, it can be drag and drop with the right click.

The following table summarizes the 2D plot window commands:

Icon	Widget type	function
	button	Displays a scale box
	button	Background color selector
	button	Allows to save the plot in different formats

It is important to notice that the output format is determined by the file extension and not by the file filter of the browser. The different formats for saving are:

- Encapsulated PostScript (.eps), the font family becomes times with 12 points for the size.

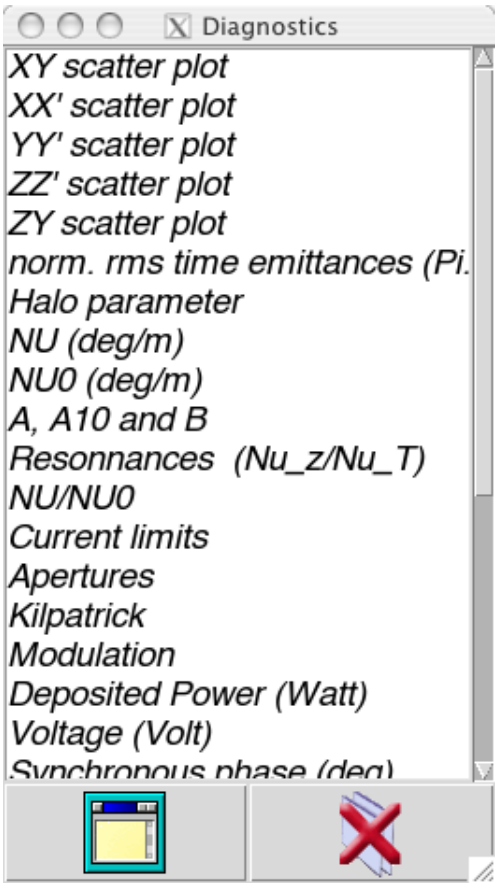


Figure 7: The diagnostics box.

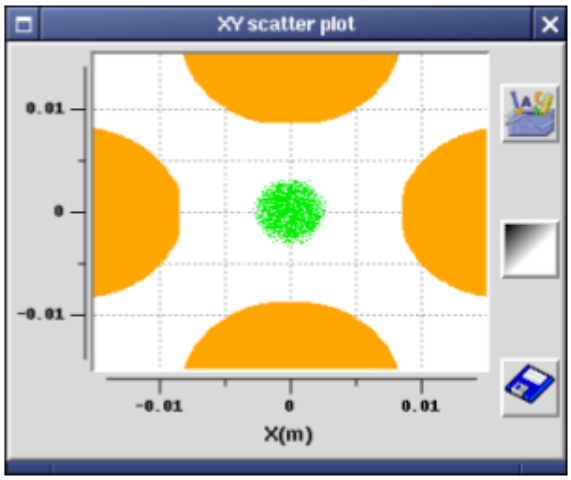


Figure 8: XY scatter plot.

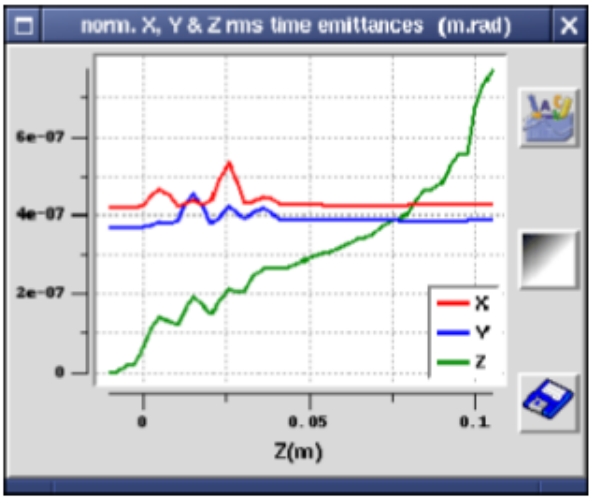


Figure 9: rms emittance plot.

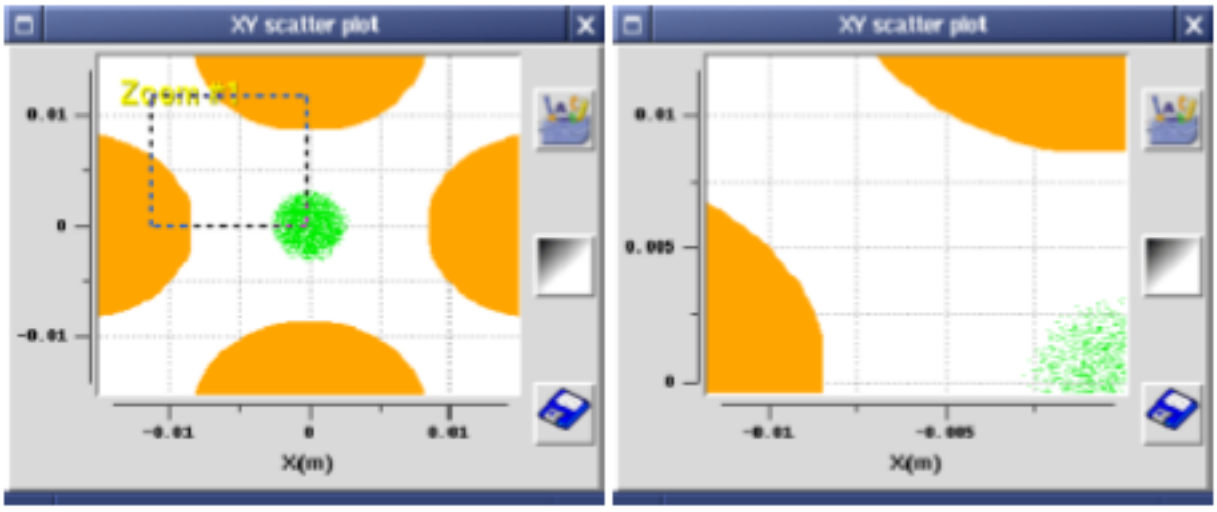


Figure 10: Zooming action.

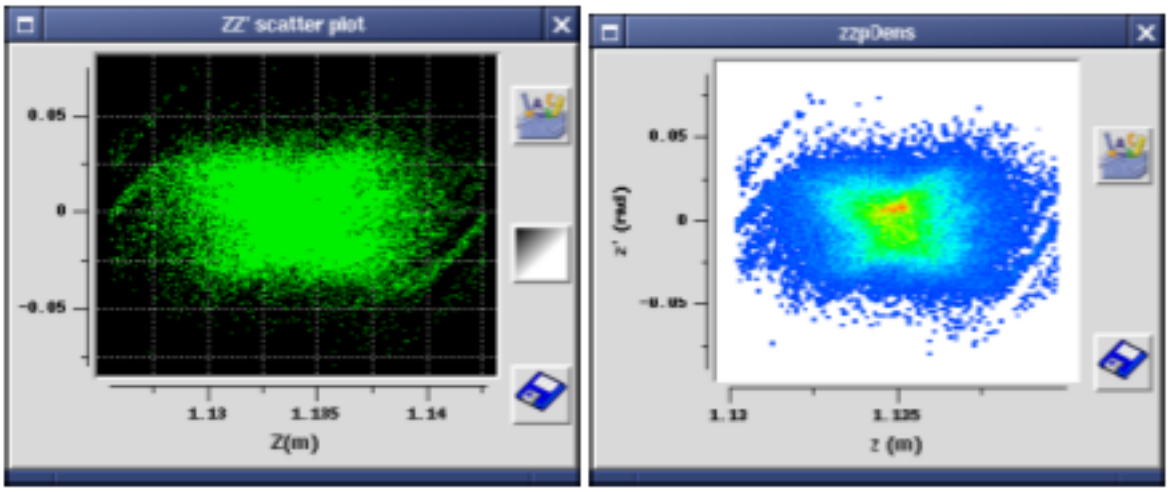


Figure 11: A zz' scatter plot and its associated density plot.

- ASCII (.txt), all physical data are printed in the selected file in an ASCII format, it is then possible to use them for other treatment.
- GIF (.gif), the font family becomes times with 12 points for the size.
- TIFF (.tif), the font family becomes times with 12 points for the size.
- JPEG (.jpg), the font family becomes times with 12 points for the size.
- PNG (.png), the font family becomes times with 12 points for the size.

All image hardcopies don't work with MacOs. This a BLT issue which I didn't solve yet. A capture is always possible.

5.6 Density contour plot

This plot has been implemented for simulations with a big number of macro particles. The density is normalized (values between 0 and 1). The figure 11 shows an example of such plot.

References

[1] R. Duperrier, Ph. D. thesis number 6194, University of Orsay, 2000.

[2] K. R. Crandall, J. H. Billen, R. S. Mills, D. L. Schrage, R. H. Stokes, G. H. Neuschaefer, T. P. Wangler and L. M. Young, "RFQ Design Codes", LANL tech. rep. n LA-UR-96-1836, 1997.

[3] http://irfu.cea.fr/Sacm/logiciels/