

User manual based on BOLIDE interface.

Introduction

The report describes the structure of software for calculation of ion beam parameters taking into account peculiarity of electron cooling, intrabeam scattering processes and beam interaction with internal target. The code was developed by JINR electron cooling group on the base of BETACOOOL program [1 - 4]. The version of the program presented here calculates:

- evolution in time of r.m.s. ion beam parameters under a common action of a few heating or cooling effects which are described in terms of characteristic times of the beam r.m.s. parameter variation. This model presumes Gaussian distribution of the ions in all degrees of freedom.
- multi-particle simulation based on amplification of the heating and cooling effects action in accordance to a large step over time during dynamics simulation. Each effect acting on the ion beam distribution function is represented by the variation of the particle momentum components after revolution in the ring. These momentum variations are amplified by the turn number, which is used as a step for dynamics simulation
- direct tracking of particles along the ring circumference with arbitrary step using Molecular dynamics technique.

The software is divided in two independent parts: physical code, which is made using only standard C++ syntax and interface part, which is an executable program working under Windows environment (Fig. 1). Connection between two parts of the program is provided using three types of the files: input, output and file used for control of the calculation process. Such a structure on the one hand allows to use the program on PC, to control and analyse results during simulations. From the other hand the physical part of the program can be compiled for UNIX operation system and used for calculations independently on interface. The interface part in this case can be used for preparation of the input file and result visualisation after completion of the calculations.

The interface part of the software consists of executable file Bolide.exe, *.dfm files containing information about BETACOOOL exterior and input files for post processing of the calculated data. Development of the BETACOOOL exterior is possible without recompilation of the Bolide.exe file. Description of the BETACOOOL exterior structure and manual for users are presented in chapter 1 of this report.

The physical part of the software consists of the executable file Betacool.exe compiled for Windows operation system and file of input parameters. For intrabeam scattering calculation one needs to use file of lattice parameters, for instance, MAD file.

The software also includes the total set of BETACOOOL initial codes: *.cpp and *.h files, project files for C++ Builder 4 and for Microsoft Visual C++ 6. The source code of the physical part of the software consists of tree relatively separated parts:

- interface part, which supports the format of input and output files common with the Bolide system,
- library of base numerical algorithms including description of dimensional variables, templates of the program self counters, procedures for matrix algebra, algorithms of numerical solution of differential equations,
- physical codes describing objects of the program and procedures with them.

Structure of the BETACOOOL program exterior corresponds to the structure of general objects in the source code, therefore main part of the object parameters is described in the chapter 1 of this report. Detailed description of the source code structure and manual for users are presented in the Chapter 2.

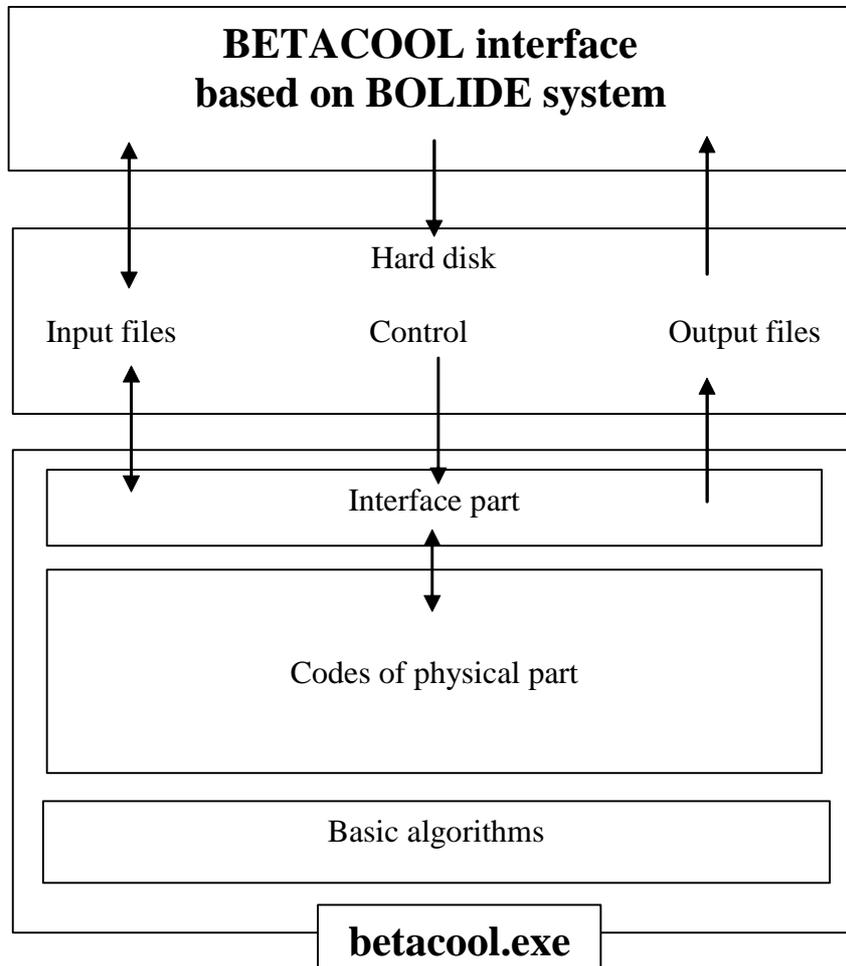


Fig.1. Structure of the software for electron cooling calculation.

The simplest way to start a work with BETACOOOL program is the following:

- to save the **Interface** files, `Betacool.exe` file, file of input parameters (*.bld format) and, if necessary MAD file of the ring lattice structure into the same folder;
- to start `Bolide.exe` file, load the input file, to edit it if necessary (with a specially developed BETACOOOL notepad);
- to start BETACOOOL program using one of the `TBrowse` components in the visual Windows of the **Interface**.
- BETACOOOL program is working in the regime of Windows-32 application and stops the calculations after their completion, or can be stopped using corresponding **Interface** tool,
- during the calculation the **Interface** program automatically reads the results from output files and represents them in numerical or graphical format in corresponding Windows.

This chapter describes the structure of the BETACOOOL program exterior for the case when working with the **Interface**. Structure of the exterior coincides with the structure of the input file and this description can be used as description of the input file format in the case when working without

Interface. The structure and dimensions of all the variables are described also in constructors, OnGet and OnSet procedures of the BETACOOOL classes in the source code of the program. For post-processing of the output BETACOOOL files one can use arbitrary graphical editor. In the file BOLIDE.doc the format of the output files is presented. Description of dimensions of output parameters is not included into the files and can be obtain from the **Interface**, or from source codes of the program. In future new format of input file with descriptions will be developed.

1. A. Lavrentev, I.Meshkov "The computation of electron cooling process in a storage ring", preprint JINR E9-96-347, 1996.
2. I .N.Meshkov, A.O.Sidorin, A.V.Smirnov, E.M.Syresin, G.V.Trubnikov, P.R.Zenkevich," SIMULATION OF ELECTRON COOLING PROCESS IN STORAGE RINGS USING BETACOOOL PROGRAM" , proceedings of Beam Cooling and Related Topics, Bad Honnef, Germany, 2001.
3. Yu.Korotaev, I.Meshkov, A.Sidorin, A.Smirnov, E.Syresin, G.Trubnikov, Software for Beam Parameter Simulations with Electron Cooling, Dubna, 2003
4. <http://lepta.jinr.ru/betacool.htm>

1. Description of the BETACOOOL interface based on BOLIDE system

1.1. General description of the interface

1.1.1. Interface files

Interface part of BETACOOOL program consists of the executable program "bolide.exe" and the following binary *.dfm files, which include information about the program exterior:

NN	File name	Window of BETACOOOL program
1	bolide1.dfm	Beam Parameters
2	Bolide2.dfm	Beam Distribution
3	Bolide3.dfm	Beam Evolution
4	Bolide7.dfm	Beam Real space
5	Bolide65.dfm	Beam Colliding beam
6	Bolide10.dfm	Effects Collision point
7	Bolide36.dfm	Effects Internal target
8	Bolide41fm	Effects Additional
9	bolide42.dfm	Effects IBS
10	Bolide6.dfm	Effects Rest Gas
11	Bolide8.dfm	Effects Stochastic cooling
12	Bolide9.dfm	Effects Particle Losses
13	bolide4.dfm	Ring Parameters
14	bolide15.dfm	Ring Lattice Structure
15	bolide21.dfm	Task Parameters
16	bolide5.dfm	Task Rates
17	bolide21.dfm	Task Parameters
18	bolide22.dfm	Task RMS dynamics
19	bolide23.dfm	Task Model beam
20	bolide25.dfm	Task Tracking
21	Bolide50.dfm	ECOOL Cooler
22	Bolide55.dfm	ECOOL Electron beam
23	Bolide60.dfm	ECOOL Friction Force
24	Bolide11.dfm	ECOOL Tabulated

The files are listed in accordance with menu items of Main window of BETACOOOL program. Name of the *.dfm file is generated by BOLIDE automatically adding to the "bolide" word the number equal to "Row" parameter in Constructor of corresponding visual window. Thus if user develops the interface part adding new windows, each new window has to have unique value of the "Row" parameter.

To install interface part one needs to copy all the *.dfm files to the same folder with Bolide.exe file. Names and number of BETACOOOL output files are described inside the program code (now there are 8 files of 3D plots *.sur and 38 files of 2D plots *.cur) and they are generated by the program and saved in the same folder with Betacool.exe file. For convenient work it is better to have interface files in the same folder with Betacool.exe.

1.2. Windows of BETACOOOL program

1.2.1. Windows of the "beam" object, list of the beam parameters

Parameters of the global variable of class `xBeam` and results of calculation of the beam parameter evolution in time are collected in the Main Menu item **Beam** which includes submenu items **Parameters, Distribution, Evolution, Real space**.

The Window **Parameters** (Fig.2) includes 5 tab sheets: **Emittance, Stability, Model Beam, Bunch, Characteristics**. The Tab Sheet **Emittance** is used to define main beam parameters. When **Model Beam** Algorithm is used user must choose Emittance Definition. Here 4 types of emittance presentation are proposed: **Root Mean Square** (usual rms emittance – when 1 sigma of Gaussian distributed particles is presumed), **Courant Snyder** (when emittance is calculated as Courant Snyder invariant), **Full Width on Half Maximum** (emittance corresponded to particles inside full width on half maximum of distribution), **Enclosed Percents** (emittance occupied by the indicated percent of beam particles).

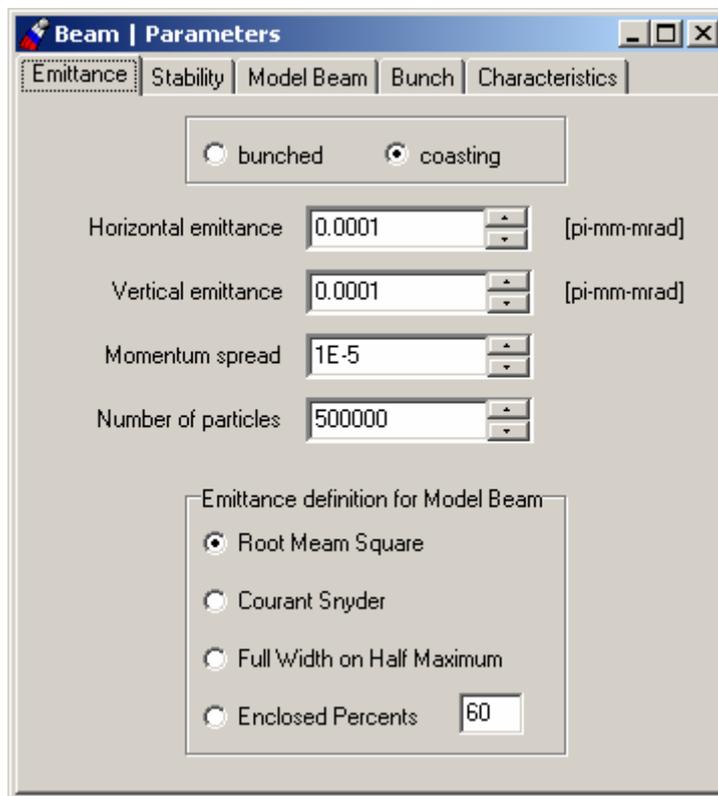


Fig.2. Tab Sheet **Emittance** of the menu item **Beam | Parameters**

The Tab Sheet Parameters (Fig. 2) includes the following input and output variables:

Variable caption	Unit	Variable in the program	Comment, Formula
Input			
Radio group "bunched – coasting"	-	Boolean variable <code>iBeam.bunched</code>	<code>bunched = true</code> for bunched beam, <code>bunched = false</code> for coasting beam
Horizontal emittance	$\pi \cdot m \cdot rad$	r.m.s. horizontal emittance, ϵ_x , <code>iBeam.Emit[0]</code>	
Vertical emittance	$\pi \cdot m \cdot rad$	r.m.s. vertical emittance,	

		$\epsilon_z,$ iBeam.Emit[1]	
Momentum spread	-	r.m.s momentum spread $\Delta p/p,$ iBeam.Emit[2]	$\epsilon_s = (\Delta p/p)^2$ Used for calculation of the longitudinal emittance
Number of particles	-	N_i iBeam.Emit[3]	
Radio group "Emittance definition for Model beam"	-	int iBeam.EmitDef	Used for Model Beam Algorithm: 0 - Root Mean Square 1 - Courant-Snyder invariant 2 - FWHM 3 - Enclosed Percent

The Tab Sheet **Stability** (Fig. 3) includes the following input and output variables:

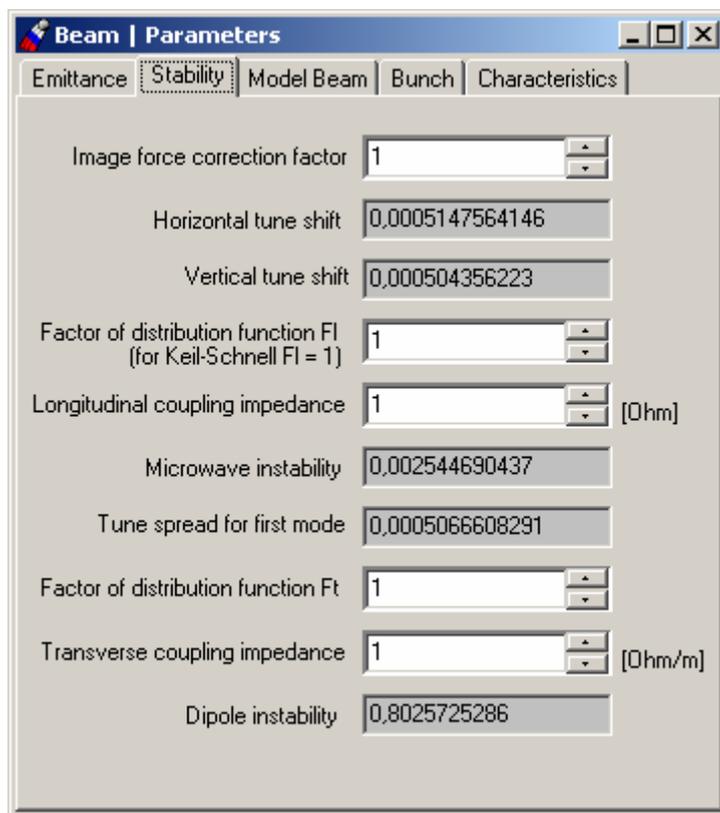


Fig. 3. Tab Sheet **Stability** of the menu item **Beam | Parameters**.

The Tab Sheet **Stability** includes the following input and output variables:

Variable lable	Unit	Variable in the program	Comment, Formula
Input			
Image force correction factor	-	F_{sc} iBeam.F_sc	Used for Lasslet tune shift calculation
Factor of distribution fansion Fl	-	F_l iBeam.F_l	Used for microwave instability threshold calculation
Longitudinal coupling impedance	Ohm	addition to longitudinal space charge impedance $Z_{Ll},$ iBeam.Z_l_l_i	$Z_L = Z_{l,sc} + Z_{L,i}$
Factor of distribution	-	F_t	Used for dipole instability

function Ft		iBeam.F_t	threshold calculation
Transverse coupling impedance	Ohm/m	addition to transverse space charge impedance $Z_{t,l}$, iBeam.Z_t_i	$Z_t = Z_{t,sc} + Z_{t,i}$
Output			
Horizontal tune shift	-	Horizontal Lasslet tune shift ΔQ_x iBeam.D_Q_x	$\Delta Q_x = \frac{F_{sc} Z^2 r_p N_i}{2\pi A \beta^2 \gamma^3 \epsilon_x \left(1 + \sqrt{\frac{\epsilon_z}{\epsilon_x}}\right) B_f}$
Vertical tune shift	-	Vertical Lasslet tune shift ΔQ_z iBeam.D_Q_z	$\Delta Q_z = \frac{F_{sc} Z^2 r_p N_i}{2\pi A \beta^2 \gamma^3 \epsilon_z \left(1 + \sqrt{\frac{\epsilon_x}{\epsilon_z}}\right) B_f}$
Microwave instability	-	Ratio between beam current and threshold current iBeam.KS	$KS = \frac{IZ_L Ze}{4F_l AU_p \beta^2 \gamma \eta \left(\frac{\Delta p}{p}\right)^2}$ $U_p = 938.2796 \text{ MeV}$
Tune spread for first mode	-	ΔQ_1 iBeam.S_Q	$\Delta Q_1 = \sqrt{\left((Q_x - 1)\eta + \xi_x\right)^2 \left(\frac{\Delta p}{p}\right)^2 + \Delta Q_z^2}$
Dipole instability	-	Ratio between beam current and threshold current, iBeam.DM	$DM = \frac{IZ_l RZe}{8F_l U_p A \beta \gamma Q_x \Delta Q_1}$

The Tab Sheet **Model Beam** includes beam parameters required for simulation by the task **Task | Algorithm | Tracking** (multi particle tracking) or **Model Beam** (dynamics of the test beam with **Particle Number** particles in accordance with chosen effects).

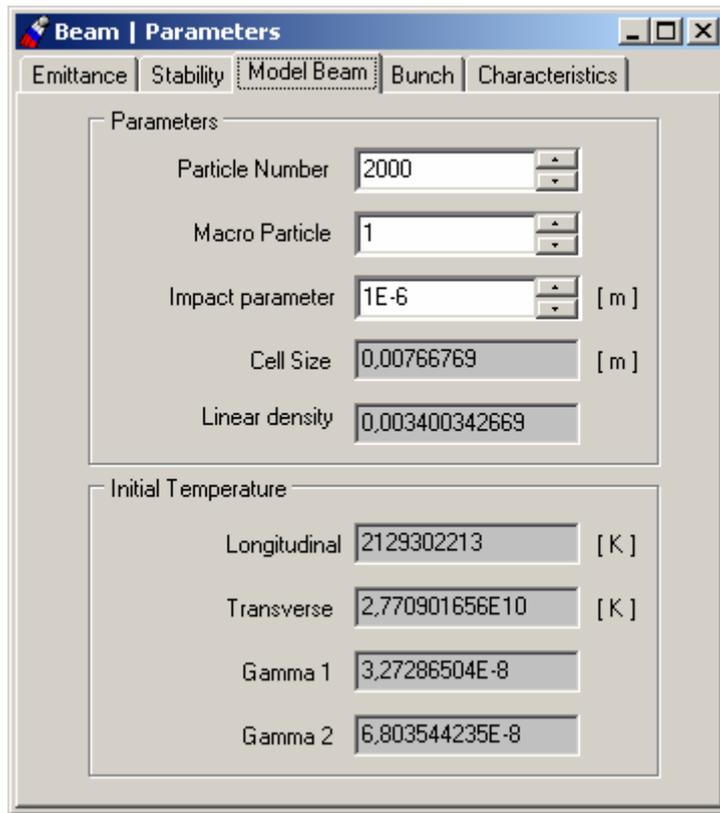


Fig. 4. Tab Sheet **Model Beam** of the menu item **Beam | Parameters**

The Tab Sheet **Model Beam** (Fig. 4) includes the following input and output variables:

Variable lable	Unit	Variable in the program	Comment, Formula
Input			
Particle Number	-	$N_{particle}$ iBeam.Number	Number of particles in the test beam
MacroParticle		iBeam.Macro	Number of macroparticles
Impact parameter		iBeam.Impact	Impact parameter – needed for crystallization simulation
Output			
Linear density	-	λ iBeam.Lambda	$\lambda_{ion} = \left(\frac{3N_i^3 r_{ion}}{8\pi^2 Q_{bet}^2 C_{ring} \gamma_0^5 \beta_0^2} \right)^{1/3}$
Longitudinal temperature	[K]	$T_{ }$ iBeam.Tlong	$T_{ } = m_i c^2 \beta^2 (\Delta p/p)^2$
Transverse temperature	[K]	T_{\perp} iBeam.Ttrans	$T_{\perp} = m_i c^2 \beta^2 \gamma^2 \left(\frac{2\pi Q_{bet} \sigma_{\perp}}{C_{ring}} \right)$
First criterion of ordering state	-	Γ_1 iBeam.Gamma1	$\Gamma_1 = \frac{Z^2 e^2}{a_{ } \sqrt{T_{ } T_{\perp}}}$
Second criterion of ordering state	-	Γ_2 iBeam.Gamma2	$\Gamma_2 = \frac{Z^2 e^2}{T_{ } \sigma_{\perp}}$

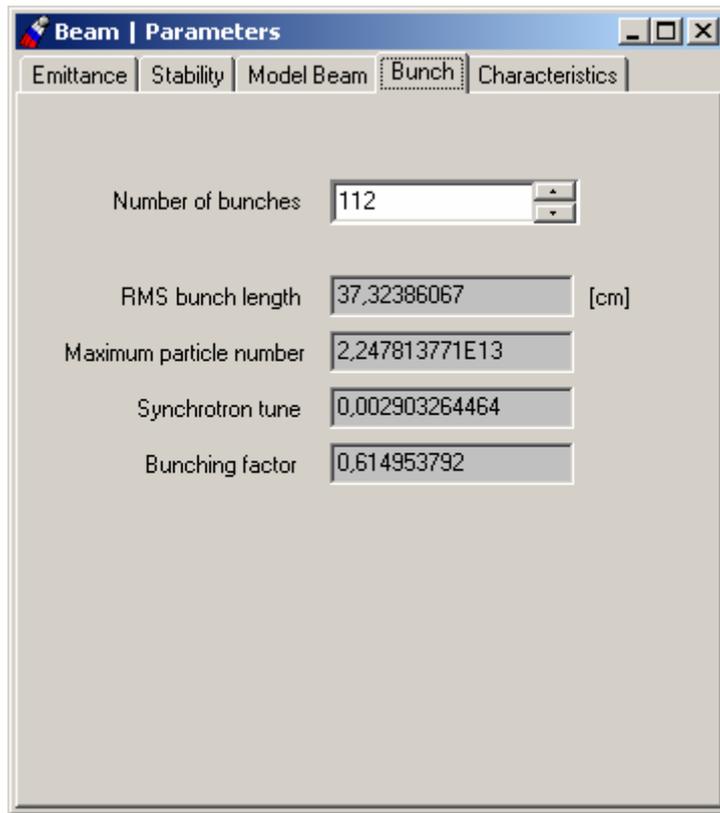


Fig. 5. Tab Sheet **Bunch** of the menu item **Beam | Parameters**

The Tab Sheet **Bunch Parameters** (Fig. 5) includes the following input and output variables:

Variable lable	Unit	Variable in the program	Comment, Formula
Input			
Number of bunches	-	N_b iBeam.N_b	Used for luminosity calculation
Output			
R.m.s. bunch length	m	σ_s iBeam.s	$\sigma_s = \beta_s \frac{\Delta p}{p}, \beta_s = \frac{ \eta R}{Q_s}$
Maximum particle number	-	Particle number corresponding to zero synchrotron tune N_{max} iBeam.N_max	$N_{max} = \frac{L_b^3 \gamma^2 heV}{3\pi ZG_L R^2 r_p U_p}$ $L_b = \sqrt{2\pi\sigma_s}$
Synchrotron tune	-	Synchrotron tune value with tacking into account beam space charge $Q_{s,N}$ iBeam.Q_s	$Q_{s,N} = Q_s \sqrt{1 - \frac{N_i}{N_{max}}}$
Bunching factor	-	B_f iBeam.B_f	$B_f = \frac{L_b}{L_{sep}}$ L_{sep} is the separatrix length

The Tab Sheet **Characteristics** includes beam parameters calculated when main parameters of the beam and ring are defined.

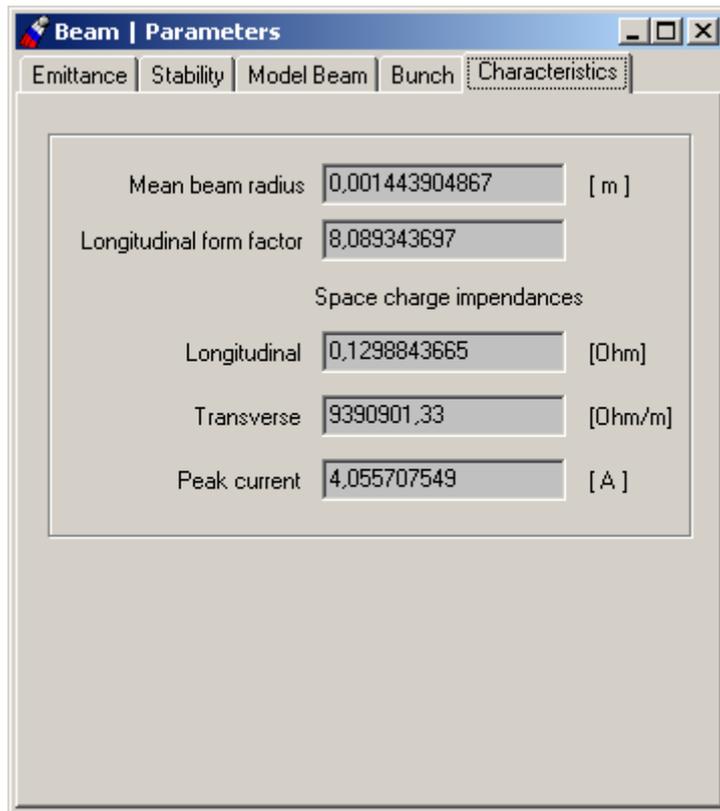


Fig. 6. Tab Sheet **Characteristics** of the menu item **Beam | Parameters**

This Tab Sheet (Fig. 6) includes the following input and output variables:

Mean beam radius	m	a iBeam.a	Used for longitudinal form-factor calculation $a = 2\sqrt{\sigma_x \sigma_z}$
Longitudinal form factor		G_L iBeam.G	$G_L = 1 + 2 \ln \frac{b}{a}$ b is the mean radius of the vacuum chamber
Longitudinal	Ohm	Space charge longitudinal coupling impedance $Z_{L,sc}$, iBeam.Z_l_sc	$Z_{L,sc} = \frac{Z_0 G_L}{2\beta\gamma^2}$ $Z_0 = 377 \text{ Ohm}$
Transverse	Ohm/m	Space charge transverse coupling impedance $Z_{t,sc}$ iBeam.Z_t_sc	$Z_{t,sc} = \frac{Z_0 R}{\beta^2 \gamma^2} \left(\frac{1}{a^2} - \frac{1}{b^2} \right)$ R is the ring mean radius
Peak current	A	Current of the coasting beam or peak current for the bunched beam, I iBeam.I	$I = \frac{ZeN_i}{B_f T_{rev}}$ B_f is the bunching factor

The Window **Beam | Evolution** (Fig. 7) is used for visualization of the beam parameter time dependencies.

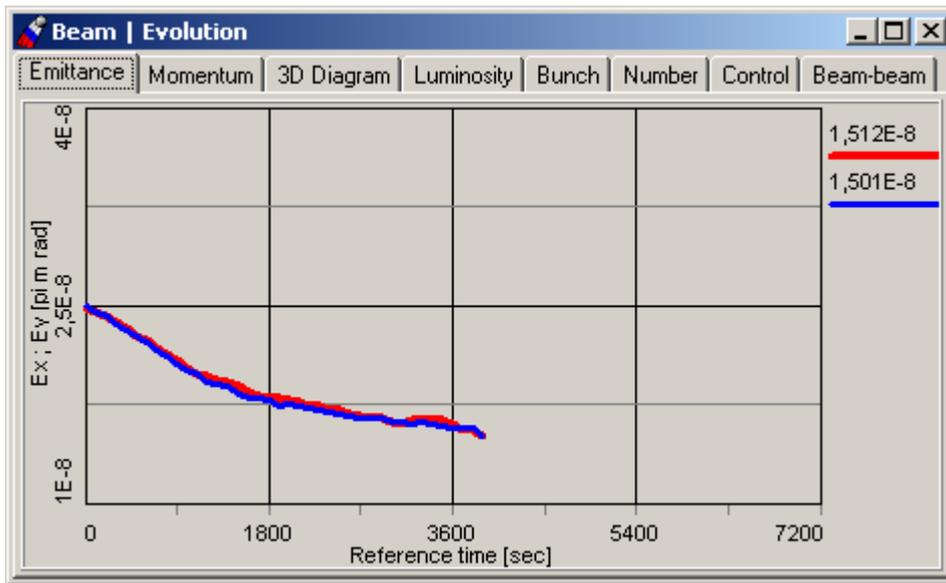


Fig. 7. Window of the **Beam | Evolution** menu item.

Tab Sheet **Emittance** contains 2D plot for output two curves: `Ex2t.cur`, `Ey2t.cur` - time dependencies of horizontal and vertical emittances. Tab Sheet **Momentum** contains 2D plot for output curve `Dp2t.cur` - time dependence of momentum spread. Tab Sheet **3D Diagram** contains plot for output `gamma2.cur` and `gamma3.cur` - criterion $\Gamma_2=\pi$ and the dependence (evolution) of the horizontal emittance on the momentum spread `txy2t.cur`. Tab Sheet **Luminosity** contains plot for output `Lum2t.cur` - luminosity time dependence. Tab Sheet **Bunch** contains plot for output `Bunch2t.cur` for time dependence of bunch length. Tab Sheet **Number** contains plot for output `Num2t.cur` - particle number time dependence. Tab Sheet **Beam-beam** contains plots for output `kappax.cur` and `kappay.cur` - both transverse beam-beam parameters time dependence.

The new useful feature of Betacool windows is TabSheet **Control** (Fig.8), which is added to every window that contains any plot.

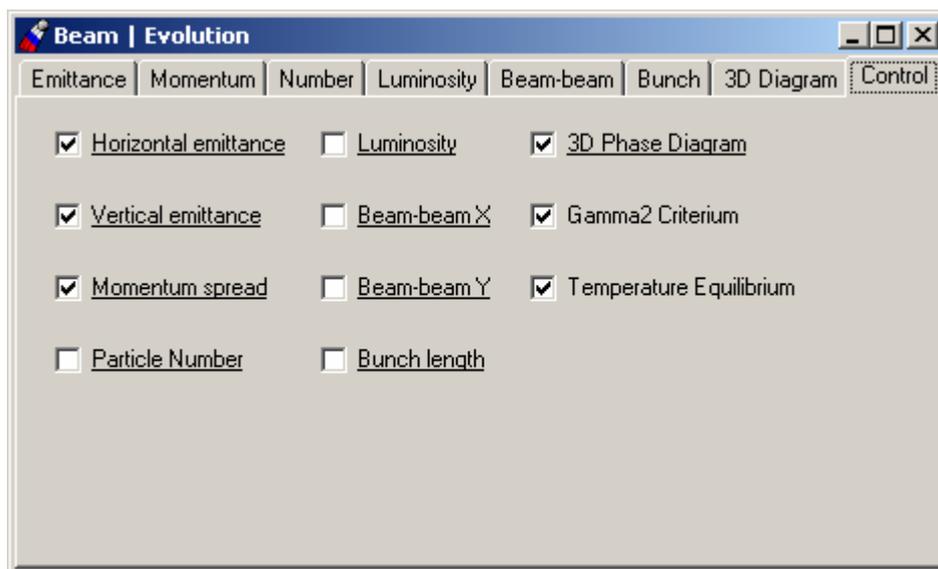


Fig. 8. Window of the **Beam | Evolution|Control** TabSheet.

This Tab Sheet allows user managing of the plots visualization. It contains list of all the plots of the current window which presented by Checkboxes. User must check plots which are needed to be

redrawn on-line and (or) check off unnecessary plots. Such a scheme allows to manage CPU resource in order to fasten calculation time.

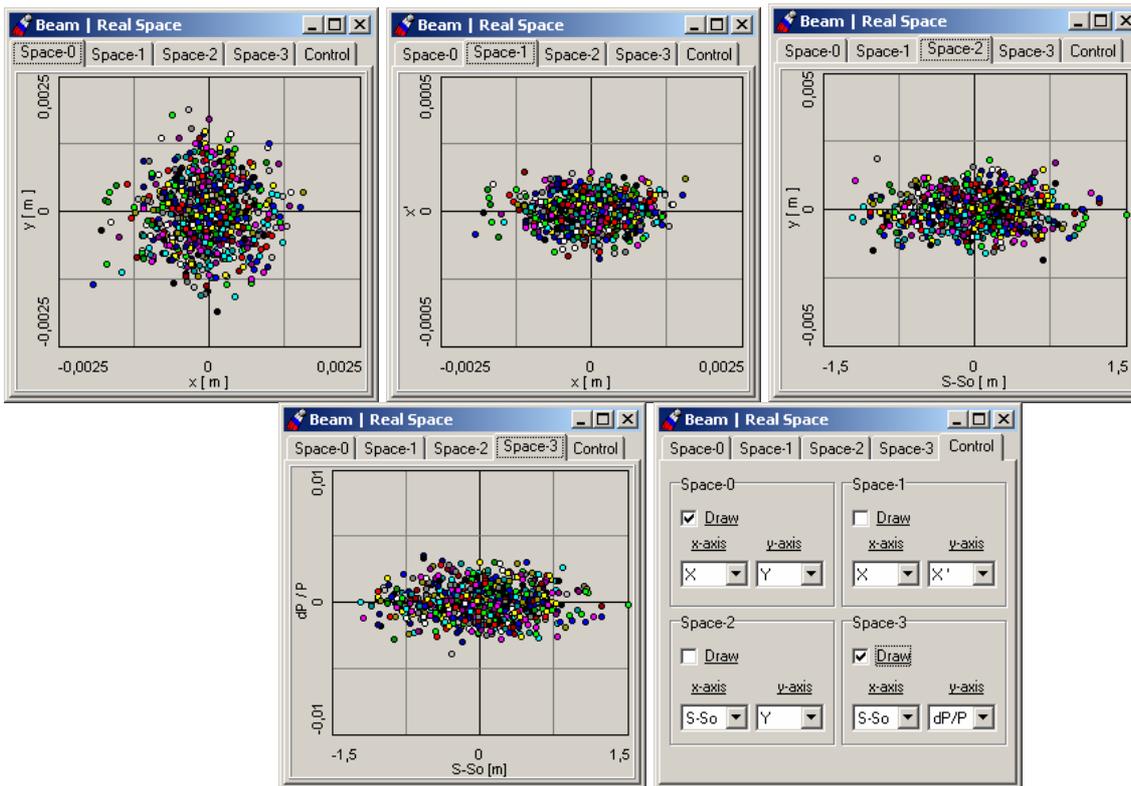


Fig. 9. Windows of the **Beam | Real Space** menu item.

The Window **Beam | Real Space** (Fig. 9) is used for visualization of the particle distribution in different planes during tracking procedure. Tab Sheet **Control** here is a manager of plots – user can choose which plot is active for on-line redrawing and what a coordinate plane will be represented on it (for example **Y-X** – transverse real space of particles, **X'-X** – horizontal phase space, **Y-S** – longitudinal real space etc).

The Window **Beam | Distribution** (Fig. 10) is used for visualization of the real particle distribution for test beam. Tab Sheet **Coordinate** – is the dependence of particle number (in percent) on momentum deviation (dist_sp.cur) and coordinate (dist_sx.cur and dist_sy.cur) normalized on corresponding rms parameter and particle number ($100\% \cdot (\Delta P/P) / (\sigma_p \cdot N)$). Tab Sheet **Profile** - is real particle distribution for every coordinate (analogue of previous plot) averaged on betatron or synchrotron oscillations (dist_ix.cur , dist_iy.cur , dist_ip.cur). Tab Sheet **Emittance** - is plot of all three particle invariants (dist_ex.cur , dist_ey.cur , dist_dp.cur)– actually it shows particle number (in percent) which occupies corresponding emittance (abscissa axis).

Tab Sheet **Evolution** – 3D plot for evolution of real particle distribution in time (evolution.sur).

Tab sheet **Control** – settings for plot on all Tab Sheets listed above. As on every plot window user can choose which plots he wants to be redrawn on-line. Group Box **Coordinate**, **Profile** correspond to first two Tab Sheets and defines: **Sigma** – is range in number of sigmas, **Division** – split number, **normalized on emittance** – choice of rms parameter for sigma to be normalized on . Group Box **Evolution** defines settings for 3D plot of particle distribution deviation: **Slices** – number of steps in time scale, **Step, sec** – value of step in time.

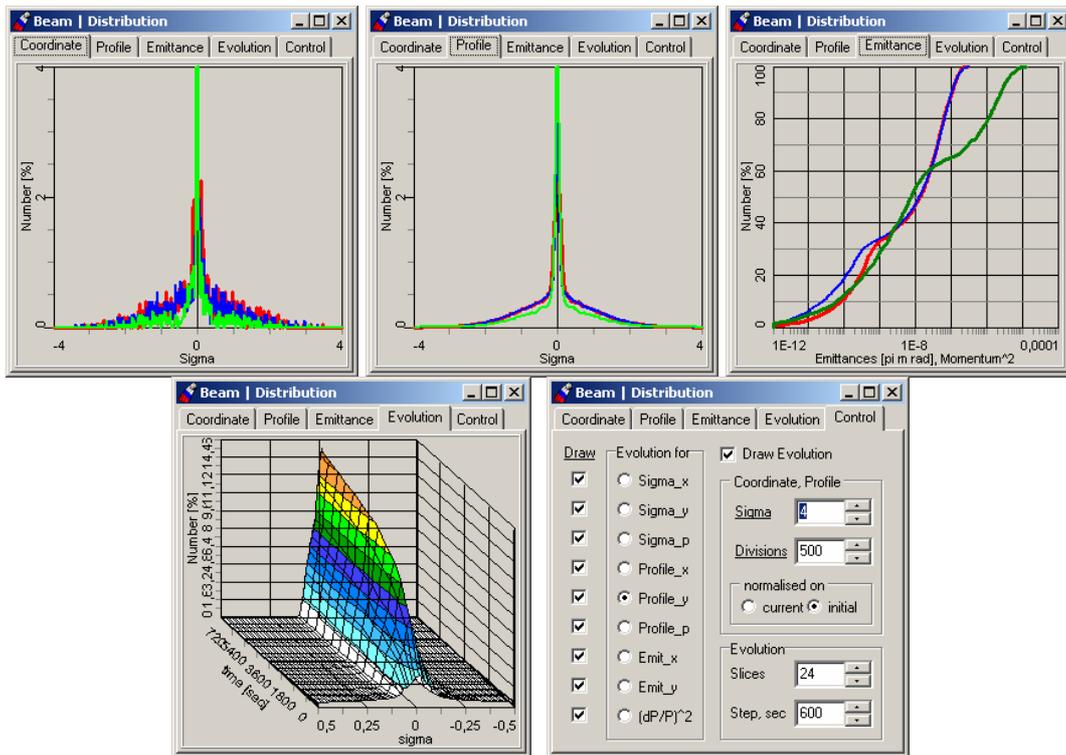
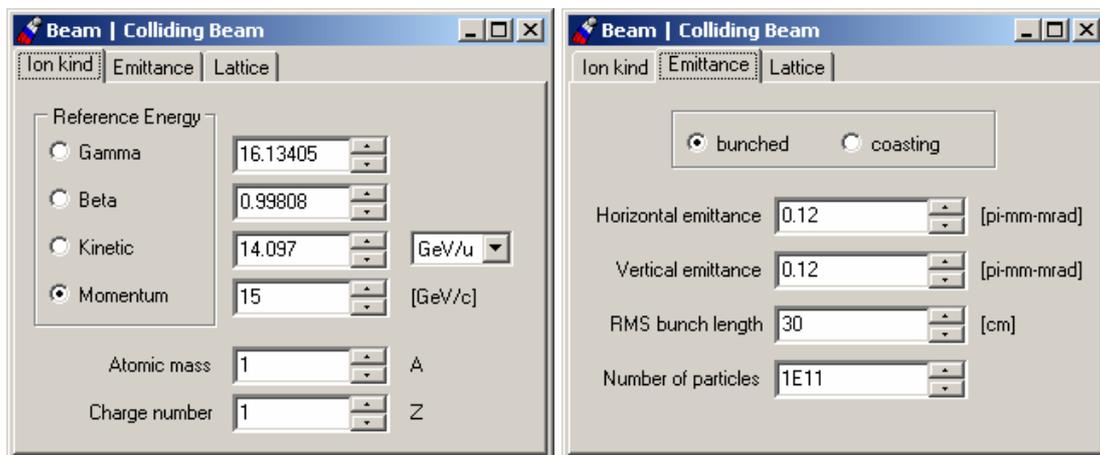


Fig. 10. Windows of the **Beam | Distribution** menu item.

The Window **Beam | Colliding Beam** (Fig.11) includes 3 Tab Sheets: **Ion kind**, **Emittance**, **Lattice**. This Form is intended to define parameters of the second (colliding) beam, if it differs from the first (main) beam. List of parameters presented on Tab Sheets of this form are analogue to parameters of the main beam and have the same meaning:

- the Tab Sheet **Ion kind** is used to define main beam parameters: *Reference energy*, *Atomic mass* and *charge number*.
- The Tab Sheet **Emittance** has main list of main beam characteristics to define: *horizontal emittance* (ϵ_x), *vertical emittance* (ϵ_y) measured in [pi-mm-mrad], *RMS bunch length* (σ_s), and *number of particles* (N).
- The Tab Sheet **Lattice** allows to define values of lattice parameters in collision point – *horizontal* (β_h) and *vertical* (β_v) *beta-functions*.



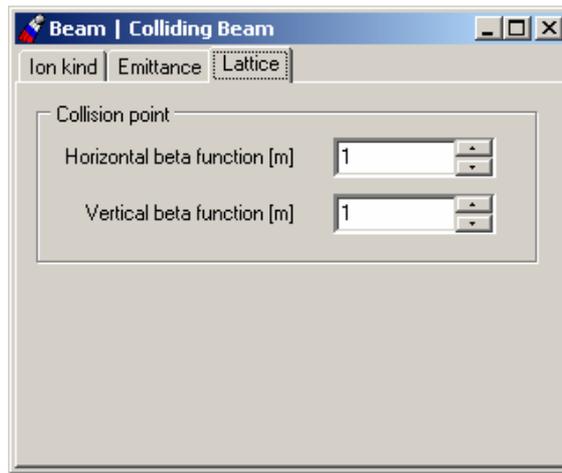


Fig.11. Window **Beam | Colliding Beam**

The Tab Sheets of Window **Beam | Colliding Beam** (Fig. 11) includes the following input and output variables:

Variable caption	Unit	Variable in the program	Comment, Formula
<i>Tab Sheet Ion kind</i>			
Atomic number	-	A , cBeam.Energy.A	
Charge number	-	Z , cBeam.Energy.Z	
<i>Tab Sheet Emittance</i>			
Radio group "bunched – coasting"	-	Boolean variable cBeam.bunched	bunched = true for bunched beam, bunched = false for coasting beam
Horizontal emittance	$\pi \cdot \text{mm} \cdot \text{mrad}$	r.m.s. vertical emittance, ϵ_x , cBeam.Emit[0]	
Vertical emittance	$\pi \cdot \text{mm} \cdot \text{mrad}$	r.m.s. vertical emittance, ϵ_z , cBeam.Emit[1]	
RMS bunch length	cm	σ_s , cBeam.Sigma_s	$\epsilon_s = (\Delta p/p)^2$ Used for calculation of the longitudinal emittance
Number of particles	-	N_i , cBeam.Emit[3]	
<i>Tab Sheet Lattice</i>			
Horizontal beta function	m	β_x , cBeam.Lattice.betax	$\beta_x = R/Q_x$
Vertical beta function	m	β_z , cBeam.Lattice.betax	$\beta_z = R/Q_z$

1.2.2. Windows of the "ring" object, list of the ring parameters

Parameters of the global variable of class `xRing` and results of calculation of the ring mean parameters are collected in the Main Menu item **Ring** that includes submenu items **Lattice Structure** and **Parameters**.

1.2.2.1. Window Lattice Structure

Window **Optics Structure** (Fig 12) contains tab sheets **Lattice Filename**, **Output MAD format**, and three tab sheets connected to Lattices of optic structure and its transformation matrix: two of 2D plots **beta functions** and **alfa functions**, and **Matrix**.

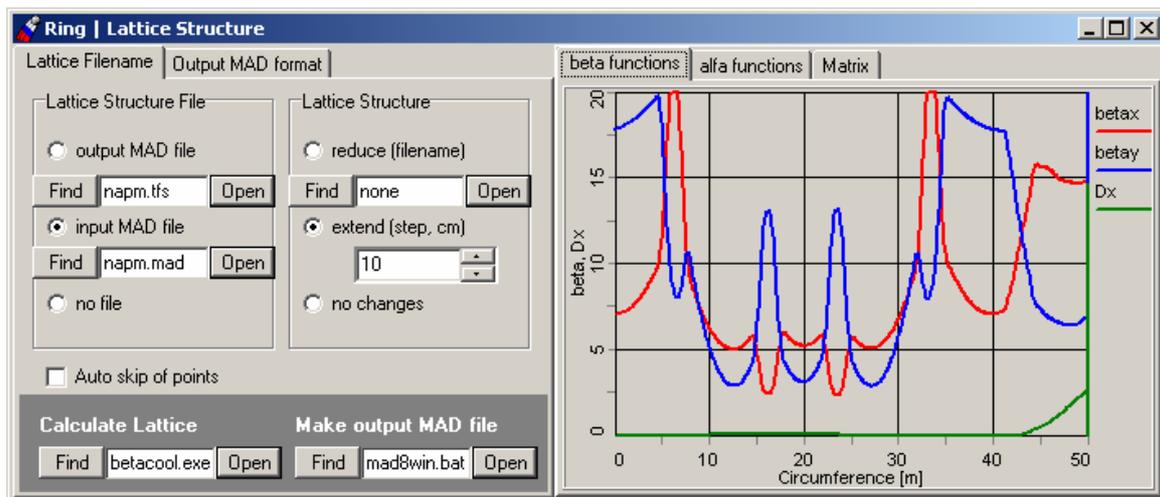


Fig. 12. Window of the **Ring | Optic structure** menu item. Tab sheet **File names**.

Tab sheet **Lattice Filename** is used to select and save in the input file the names of files containing information about ring optics structure. This Tab Sheet contains:

- Radio Group **Lattice structure File** with three possible positions: **output MAD format**, **input MAD format**, **no file** and three of **TBrowse** components. This Radio Group is used when user has a file with some lattice structure obtained from MAD program or collected in accordance with input MAD file standard. **No file** option is chosen when user does not need Lattice file.
- Radio Group **Lattice Structure** with three possible positions: **reduce (filename)**, **extend (step, cm)**, **no changes**. Here choice of active option depends on the definition made in described above **Lattice structure File** Radio Group.

For IBS calculations using Martini or Jei Wei models one needs to find appropriate lattice structure file, for example MAD output file. Button **Find** of the **TBrowse** component opens the file manager window. Button **Open** opens the file using internal text editor. The chosen file name is indicated in the edit window of the **TBrowse** component and saved in the input BATACOOL file. This name is used for initialization of the ring structure after start of the program. Validity of the file can be checked using **TBrowse** component **Calculate Lattice**.

Button **Open** of this **TBrowse** starts BETACOOOL with the parameter `/lattice`. At this parameter BETACOOOL read MAD output file, transform lattice parameters into internal format and save them into the files `BetaX.cur`, `BetaY.cur`, `DispX.cur`, `AlfaX.cur`, `AlfaY.cur`, `DispX_.cur`. During this procedure the program checks validity of the data in all positions of lattice structure file in accordance with the description tacking from the tab sheet **Output MAD format** (Fig. 13).

Another **TBrowse** component at this Form: **Make output MAD file** allows to launch MAD application in the framework of BETACOOOL and to create file with lattices in format of MAD output file. Here user must indicate MAD input file with **find Input MAD file** component and then to launch **TBrowse** component **Make output MAD file** to force MAD program generate output file with tracked lattices.

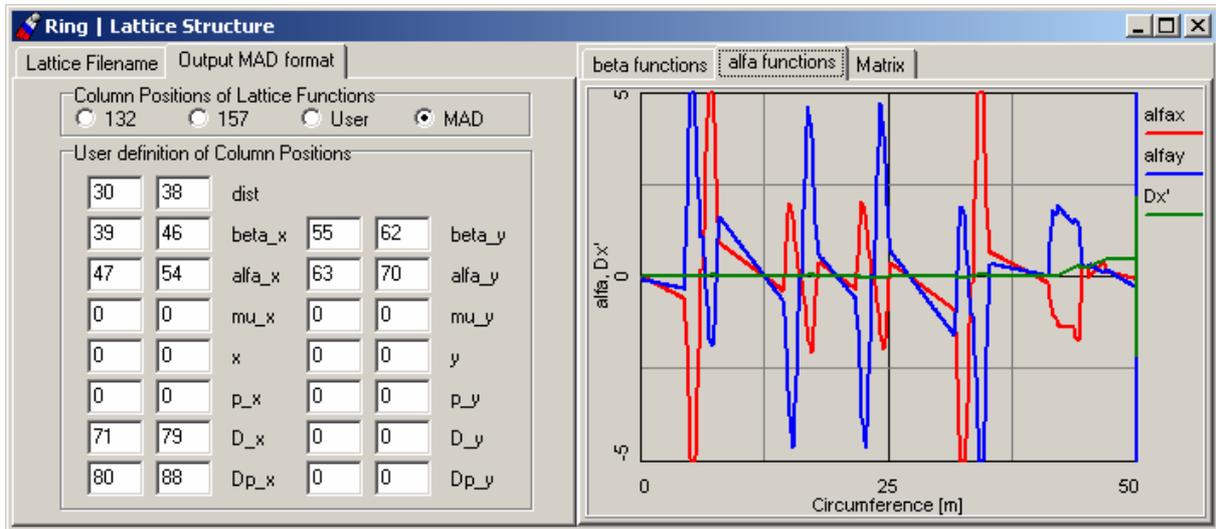


Fig. 13. Window of the **Ring | Optic structure** menu item. Tab sheet **Output MAD format**.

The Tab Sheet **Output MAD format** contains radio button **columns** which positions **132** and **157** correspond to the standard output MAD files with 132 or 157 columns correspondingly. In case of nonstandard lattice file user can prepare its specification by himself – here radio button **User** must be chosen. For this goal one needs to introduce initial and final columns in the string for all lattice functions. If user set the initial and final columns in zero value then these lattice functions will be equal zero. In the case of mistakes in the Lattice structure file specification BETACOOOL generates corresponding message in the `Betacool.war` file.

If radio button **MAD** is chosen here – then definition of the column position in MAD output file will be taken correspondingly to the version of MAD program which lies in the current folder (this application is used when **TBrowse** component **Make output MAD file** is called).

Input MAD format – is used for choosing the input file name for multi particle tracking. BETACOOOL program can read optics and lines from the standard input MAD file and translate to the optics structure of the storage ring. The result of translation is saved to file with same name as MAD input file and extension `*.use`. Now BETACOOOL can translate the following elements from MAD file:

- DRIFT;
- SBEND (ANGLE – bend angle, E1 and E2 – edge angles);
- QUADRUPOLE (K1 – quadrupole gradient, TILT – rotation);
- SEXTUPOLE (K2 – sextupole gradient);
- RFCAVITY (FREQ – RF frequency, V – RF Voltage)
- SOLENOID (Ks – solenoid gradient);
- LINE;
- USE.

All the elements have obligatory parameter **LENGTH**

Electron cooling object can be added into the MAD file if any element has name **XECOOL**:

XECOOL: drift, l=2.1

Target object is included into the MAD file as zero-length element and should have name XTARGET:

XTARGET: marker, type=TARGET

Depending on the task, the BETACOOOL program is a powerful instrument for processing with ring lattices and (or) transformation matrices of optics element. If user has a MAD input file with optics element consequence, there is an algorithm that provides a calculation of transformation matrix for whole ring, then the second step – is calculation of lattice parameters in the point S_0 , and after tracking of lattice parameters takes place. If one has a lattice at the chosen point and matrix after, it is possible to calculate lattices after matrix, and so on and so force. For this procedure user has to choose **Input MAD format** radio button and to push **Calculate Optics Structure**. Passing time, calculated lattices will be visualized onto corresponded plots.

Reverse procedure is also provided – co called “reverse-tracking” – when one chooses the **Output MAD format** radio button – the consequence of transformation matrices for optical elements of the structure will be restored from the consequence of lattices.

If checkbox **Auto skip of points** is switched on then the following procedure takes place: when the number of points in any curve (here the curves with lattice parameters) is to be exceeded then because of the limited size of the curve with every next point inserted to the curve array one loses first point from the array. So if checkbox is ON, every second point in this array will be deleted and all left points will be suppressed, then at least a half of the curve array will be released for the expected calculated data. This procedure can be eternal - always when array is nearly to be overfilled, every second cell in it will be released.

Reduce(filename) – is useful option foreseen for large lattice (optics) structure. Here one must built a special file *.red with reduced structure of the ring. User has to leave only optical elements and corresponded which he wants to be taken into account for calculations and visualization. Here the following algorithm takes place: for the tracking using matrices the whole structure is taken into account. But necessary matrices will be build only in points selected by user (by multiplying of all the intermediate matrices between selected points). Then only these matrices will track the beam. And IBS effect will be calculated only in these points. Finally, only lattices in selected points will be plotted. So the calculation time may be sufficiently reduced.

Extend (step, cm) – this option is used only when Tracking algorithm is presumed and Molecular dynamics technique is used. Here the step over longitudinal coordinate is defined between points where transformation matrix is calculated. This choice is active when option **Matrixes** is switched on in **Task | Algorithm | Tracking | Equations of motion** window. The second necessary condition – is **Input MAD format** must be chosen in **Lattice structure File** Radio Group

Next Tab sheet **Matrix** (see Fig 14) is rather comfortable tool for the monitoring the transformation matrix of the whole ring or selected optic element. One can choose to monitor either a transformation matrix of the whole ring (**Ring**) or the selected element from optic structure (**Optics**) by choosing its number with **Index** counter. Here complete transformation matrix is visualized. There is an option to look at any part of numerical appearance of matrix elements, if it is introduced as complex one. One can choose the representation by switching **Real | Imag | Abs | Arg** radio buttons. Here edit window **|1-Det|** introduces the precision of calculated matrix determinant.

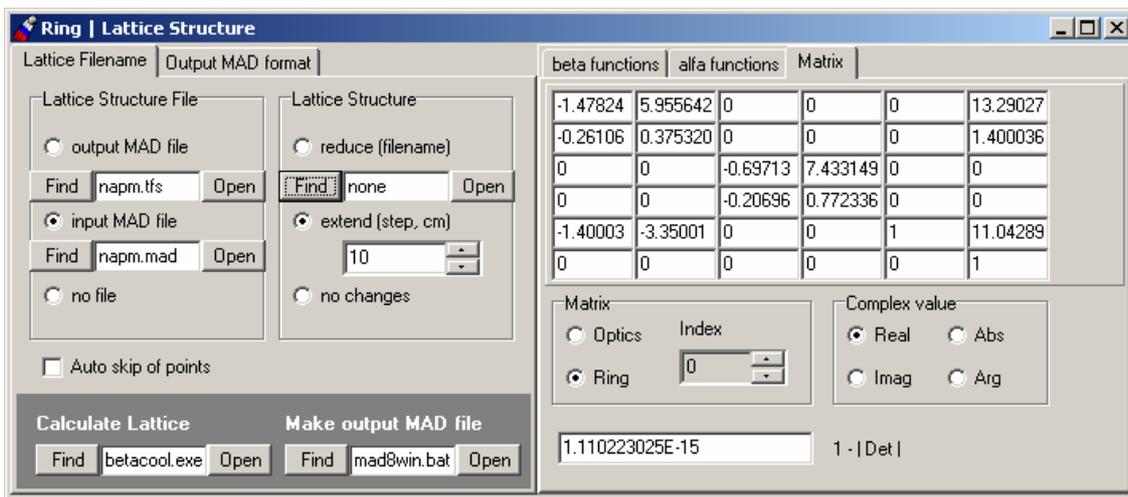


Fig. 14. Window of the **Ring | Optic structure** menu item. Tab sheet **Matrix**.

Here a useful table is presented for different modes of usage the **Ring | Lattice Structure | Lattice Filename** radio buttons for selected tasks (for details see description of **Task | Algorithm**):

<i>Radio button</i> <i>Task</i>	Output MAD format	Input MAD format	No optics	Comments
RMS Dynamics	√	√	√*	* - in case when Piwinsky model of IBS calculation was chosen
Model Beam	√	√	√**	** - if chosen No optics – user has to fill (type in) the ring transformation matrix
Tracking	—	√	—	

1.2.2.2. Window **Ring | Parameters**

Window **Parameters** (Fig. 15 – 18) contains the tab sheets **Ion kind**, **Lattice**, **Mean params**, **RF system**.

The tab sheet **Ion kind** (Fig. 15) contains radio button **Reference energy**. Depending on chosen button one can input one of the following parameters for the reference particle:

- Lorenz factor (**Gamma**),
- Velocity in the speed of light units (**Beta**),
- Kinetic energy (Kinetic) or
- Momentum in GeV/c (**Momentum**).

Units of the particle kinetic energy can be determined using ComboBox near corresponding edit window. Only one parameter from this group can be chosen as input one, all others parameters are output.

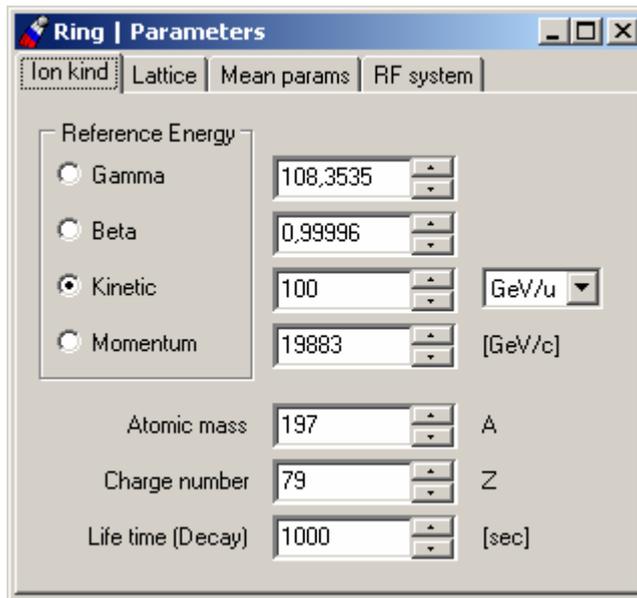


Fig. 15. Window of the **Ring | Parameters** menu item. Tab Sheet **Ion kind**.

Other parameters from this tab sheet are listed in the following table:

Variable lable	Unit	Variable in the program	Comment, Formula
Input			
Atomic number	-	A, iRing.A	
Charge number	-	Z, iRing.Z	For calculation of the ion life time due to the interaction with residual gas and internal target one supposes that the ions are completely stripped
Life time	sec	iRing.Tlife	used in Decay effect for the loss rate calculation

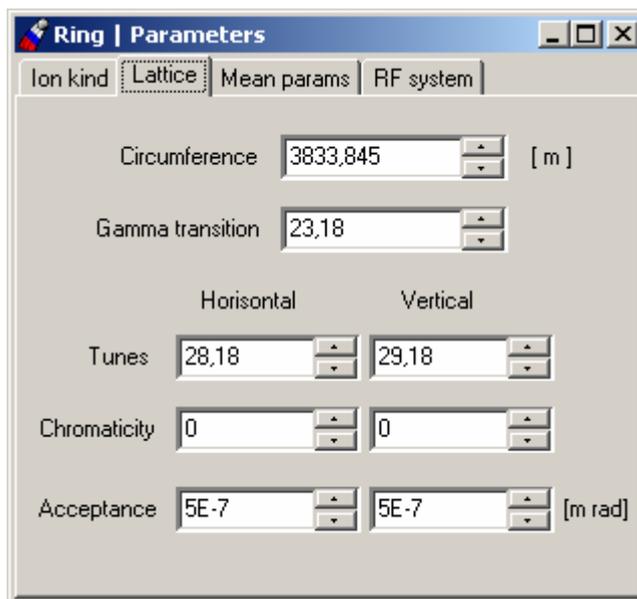


Fig.16. Window of the **Ring | Parameters** menu item. Tab Sheet **Lattice**.

The tab sheet **Lattice** (Fig. 16) includes the following input variables:

Variable lable	Unit	Variable in the program	Comment, Formula
Circumference	m	C , <code>iRing.Circ</code>	
Gamma transition	-	γ_{tr} <code>iRing.GammaTr</code>	used for off momentum factor calculation
Tunes	-	Q_x, Q_z <code>iRing.TunesH</code> , <code>iRing.TunesV</code>	
Chromatisity	-	ξ_x, ξ_z <code>iRing.HromatH</code> , <code>iRing.HromatV</code>	used for tune spread calculation
Acceptance	$\pi \cdot m \cdot rad$	A_x, A_z <code>iRing.AcceptH</code> , <code>iRing.AcceptV</code>	used for calculations of life time due to single scattering on big angles

The tab sheet **Mean params** (Fig. 17) includes the following output variables:

Variable caption	Unit	Variable in the program	Comment, Formula
Mean radius	M	R , <code>iRing.R_m</code>	$R = C/2\pi$
Horizontal beta function	M	β_x , <code>iRing.BetaH</code>	$\beta_x = R/Q_x$
Vertical beta function	M	β_z , <code>iRing.BetaV</code>	$\beta_z = R/Q_z$
Dispersion	M	D , <code>iRing.Dispersion</code>	$D = \beta_x/Q_x$
Revolution period	Sec	T_{rev} , <code>iRing.Trev</code>	$T_{rev} = C/\beta c$
Off momentum factor	-	η , <code>iRing.Eta</code>	$\eta = \frac{1}{\gamma^2} - \frac{1}{\gamma_{tr}^2}$

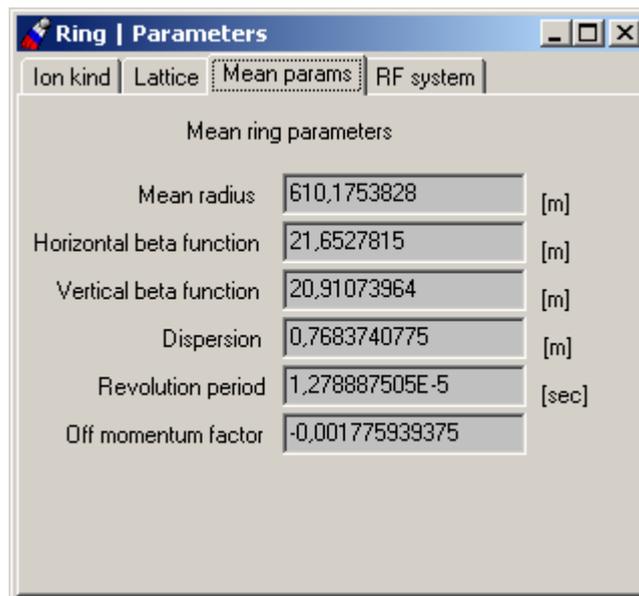


Fig. 17. Window of the **Ring | Parameters** menu item. Tab Sheet **Mean params**.

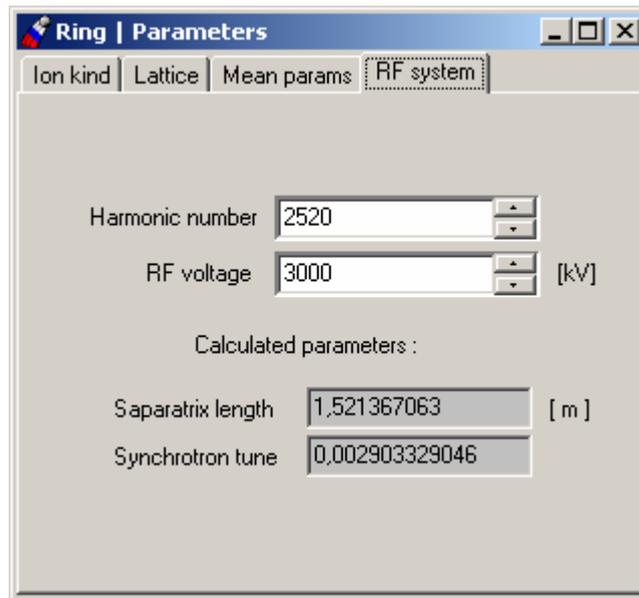


Fig. 18. Window of the **Ring | Parameters** menu item. Tab Sheet **RF system**.

The tab sheet **RF system** (Fig. 18) includes the following input and output variables:

Variable caption	Unit	Variable in the program	Comment, Formula
Input			
Harmonic number	-	H , iRing.h	
RF voltage	kV	V iRing.V	
Output			
Separatrix length	m	L_{sep} iRing.L_s	$L_{sep} = C/h$
Synchrotron tune	-	Q_s iRing.Q_s	$Q_s = \frac{1}{\beta} \sqrt{\frac{eh \eta ZV}{2\pi U_p A\gamma}}$

1.2.3. Windows of the "ECOOL" object, list of the cooler parameters

In this BETACOOOL version electron cooling is treated as an **Effect** acting on the r.m.s. parameters of the ion distribution function and changing the ion number. Like other **Effects** (**IBS**, **Additional Heating** and so on) the electron cooling model returns heating and loss rates and, in principle, it is a part of the Effect library. Separate menu item **ECOOL** is used for electron cooling due to complicated structure of this **Effect**. To calculate the cooling rates user needs to determine:

- models of the cooler, general parameters of the cooling section, ion beam;
- model of the electron beam;
- formula for the friction force calculation from corresponding library.
- parameters of the tables of pre-calculated friction forces

To make the procedure clear these three steps are divided between three submenu items and can be done independently using corresponding windows.

1.2.3.1. Models of the cooler and ion beams

The tab sheet **Cooler** of the **ECOOL | Model** Window (Fig. 19) contains radio button **Calculation model** which provide the choice between calculations of the ion coordinates after crossing the

cooling section in the frame of the **Thin lens** model and numerical integration of the ion motion equation. The numerical integration can be performed by two methods: Euler (**Euler method**) or 4-th order Runge Kutta (**RK method**). For the numerical integration one needs to determine the number of integration steps along the cooling section.

In the program the electron beam models are switched by variable `iEcool.e_model` which is equal:

- `iEcool.e_model = 1`, for Thin Lens model;
- `iEcool.e_model = 2`, for integration with Euler method;
- `iEcool.e_model = 3`, for integration with Runge Kutta method.

The variable `iEcool.integr_step` corresponds to the integration step number for both numerical methods.

Also here on this Tab Sheet presented main parameters of the Electron Cooler:

Variable caption	Unit	Variable in the program	Comment, Formula
Input			
Cooler length	m	L_{cool} , <code>iEbeam.CoLength</code>	
Magnetic field	kG	B , <code>iEbeam.F.mfield</code>	
Section number	-	<code>iEcool.SectionNumber</code>	This parameter is used only when Model Beam algorithm is used

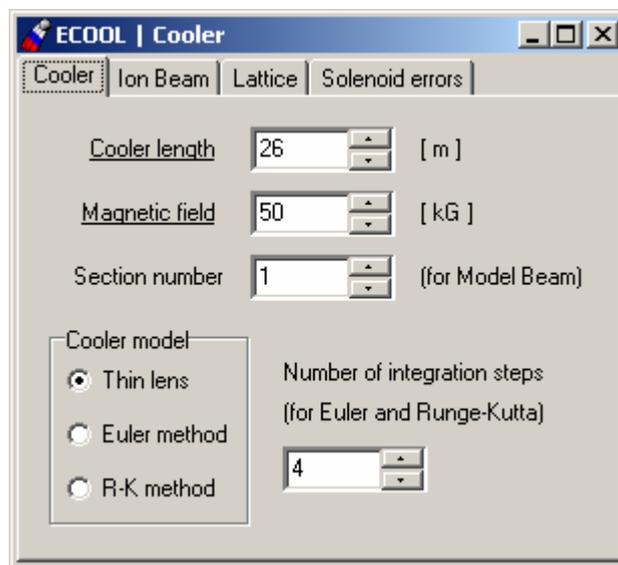


Fig. 19. Window of the **ECOOL | Models** menu item. Tab Sheet **Cooler**.

The cooling rates can be calculated for r.m.s. particle or by averaging over the particle distribution function using Monte Carlo method. Choice between these possibilities is provided by radio button **Calculation model** of the tab sheet **Ion beam** of the **ECOOL | Model** Window (Fig. 20).

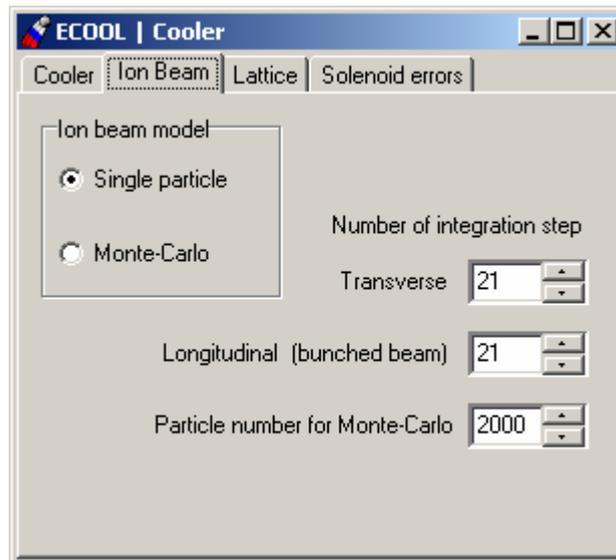


Fig. 20. Window of the **ECOOL | Cooler** menu item. Tab sheet **Ion Beam**.

In the program these models are switched by variable `iEcool.i_model`, which is equal to 1 for r.m.s. particle calculation and 2 for Monte Carlo calculation.

For r.m.s. particle the cooling rates are calculated by averaging over phases of betatron and synchrotron oscillations. Numbers of integration steps over the phases are input in corresponding edit windows. Corresponding variables in the program are `iEcool.step_tr` and `iEcool.step_long`.

For Monte Carlo method one needs to determine number of particles used for cooling rate calculation. In the program corresponding variable has a name `iEcool.num_MC`.

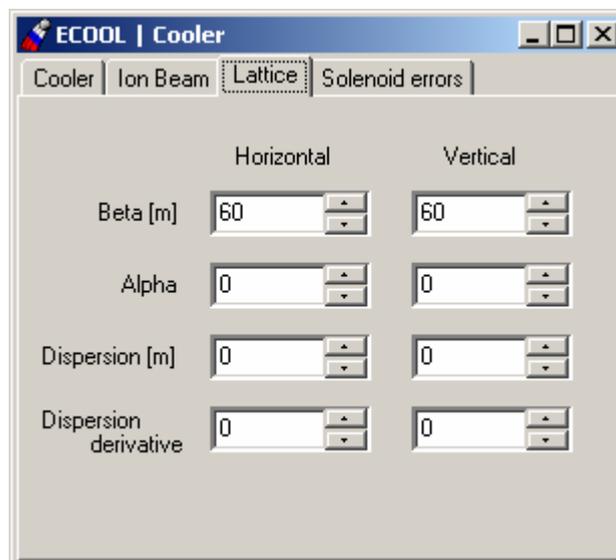


Fig. 21. Window of the **ECOOL | Parameters** menu item. Tab sheet **Lattice**.

The lattice parameters of the ring in the cooler position are input using tab sheet **Lattice** (Fig. 21). They are collected in the variable `iEcool.Lattice` of the `xLattice` class.

Variable caption	Unit	Variable in the program	
Input			
Beta (Horizontal / Vertical)	m	iEcool.Lattice.betax / betay	
Alpha (Horizontal / Vertical)		iEcool.Lattice.alfax / alfay	
Dispersion (Horizontal / Vertical)	m	iEcool.Lattice.Dx / Dy	
Dispersion derivative (Horizontal / Vertical)		iEcool.Lattice.Dpx / Dpy	

Tab Sheet **Enable Solenoid Errors** is intended to take into account magnetic field errors in solenoids of cooling section (Fig. 22). There are two possibilities of errors definition:

- with the edit windows **Initial coordinate [m]**, **Final coordinate [m] (Horizontal / Vertical)** which are intended to input the position of the electron beam orbit at the entrance and at the exit of the cooling section respectively to the ion equilibrium orbit in [cm]. Corresponding variables in the program are: `iEbeam.x_0`, `iEbeam.y_0`, `iEbeam.x_f`, `iEbeam.y_f`.
- to read magnetic field errors from the text file. This file contains three-tuples of coordinates [m] from which angle of the field line is calculated. User can choose file with *TBrowse* component

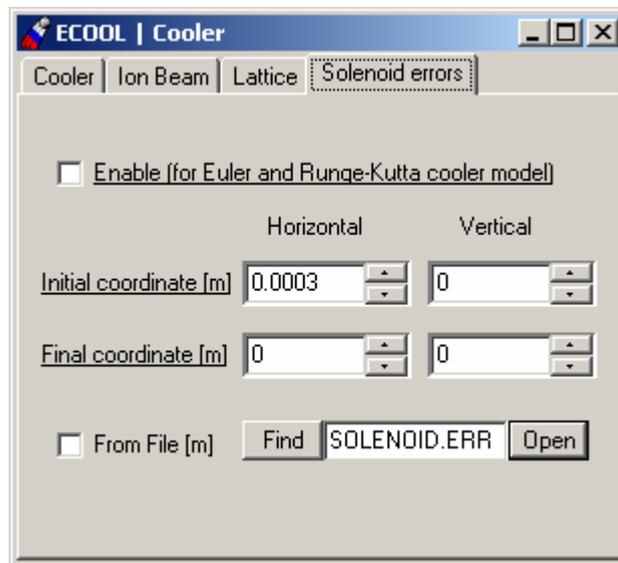


Fig. 22. Window of the **ECOOL | Cooler** menu item. Tab sheet **Solenoid errors**.

1.2.3.2. Models of the electron beam

General parameters of the cooling section are collected in the Window of the **ECOOL | Electron beam** menu item. This Window contains the following tab sheets:

- Model (here Uniform cylinder)**
- Uniform bunch**

Gaussian cylinder.
Gaussian bunch,

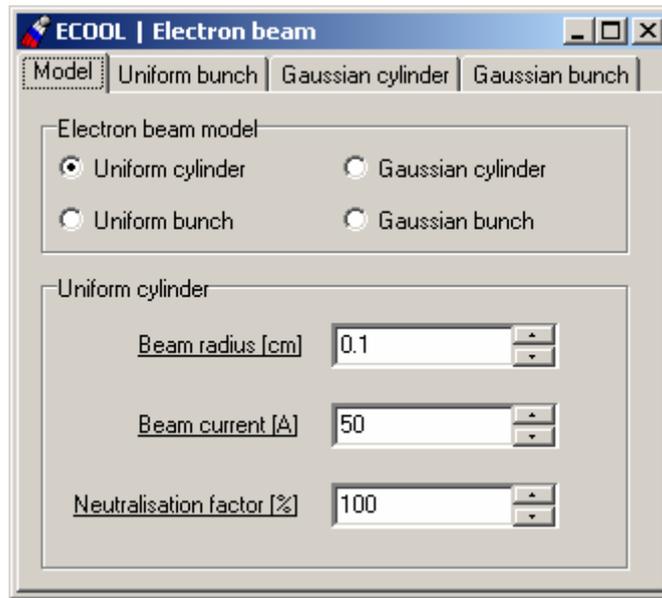


Fig. 23. Window of the **ECOOOL | Electron beam** menu item. Tab sheet **Model**.

The Tab Sheet **Model** (Fig. 23) includes the following input and output variables:
Radio Button **Electron beam model** which provides a choice between three models of the electron beam. In the program the models are switched by variable `i.Ebeam.GeoModel`, which is equal to the following numbers:

Electron beam model	<code>i.Ebeam.GeoModel</code> value
DC cylindrical electron beam with uniform electron density	0
Electron bunch with with uniform electron density	1
DC electron beam with elliptic cross-section and Gaussian distribution in the transverse plane	2
Electron bunch with elliptic cross-section and Gaussian distribution in all degrees of freedom	3

At the same time this Tab Sheet is used for definition of the parameters of the **Uniform cylinder** electron beam model:

Variable caption	Unit	Variable in the program	Comment, Formula
Input			
Beam radius	Cm	a_e , <code>iEbeam.bradius</code>	
Beam current	A	I_e , <code>iEbeam.bcurrent</code>	
Neutralization factor	-	η_{neutr} , <code>iEbeam.Neutralization</code>	has to be positive and less than 1

The Tab Sheet **Uniform bunch** (Fig. 24) includes the following input and output variables:

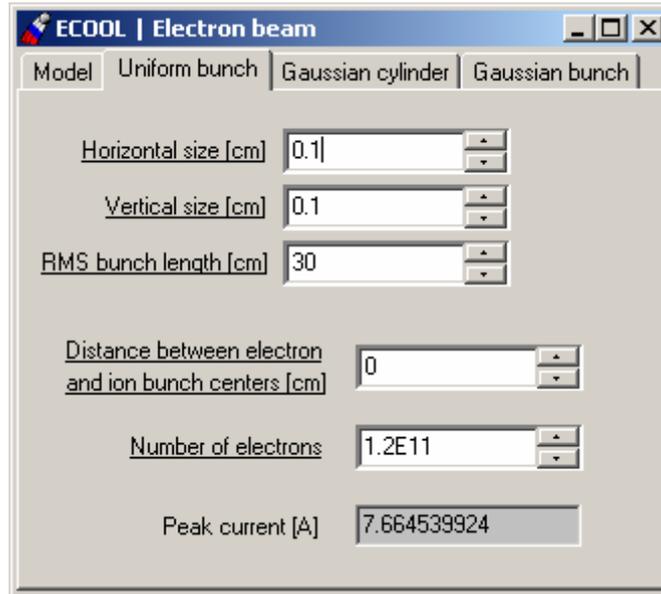


Fig. 24. Window of the **ECOOOL | Electron beam** menu item. Tab sheet **Uniform bunch**.

Variable caption	Unit	Variable in the program	Comment, Formula
Input			
Horizontal size	cm	$\sigma_{e,x}$, iEbeam.size_x	
Vertical size	cm	$\sigma_{e,z}$, iEbeam.size_y	
RMS bunch length	cm	λ_e , iEbeam.size_s	
Distance between electron and ion bunch centers	cm	iEbeam.dist	
Number of electrons	-	N_e , iEbeam.Ne_uni	
Output			
Peak current	A	I_e , iEbeam.Ie_uni	$I_e = \frac{\beta c e N_e}{\sigma_{e,s} \sqrt{2\pi}}$

The Tab Sheet **Gaussian cylinder** (Fig. 25) includes the following input and output variables:

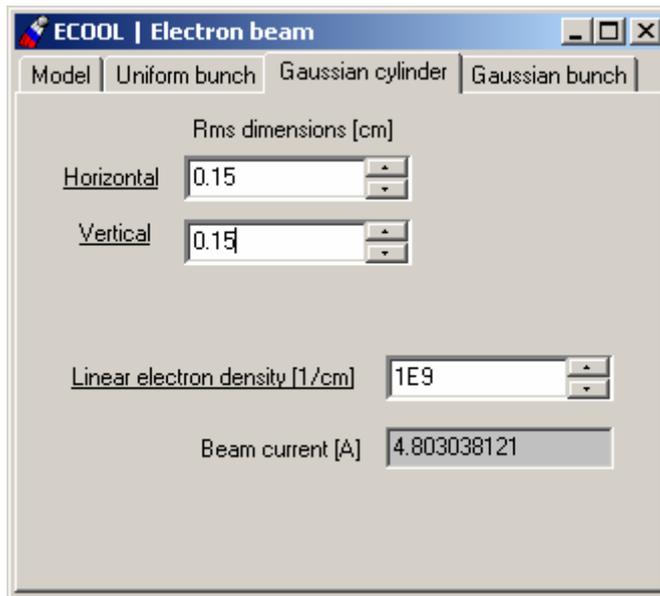


Fig. 25. Window of the **ECOOL | Electron beam** menu item. Tab sheet **Gaussian cylinder**.

Variable caption	Unit	Variable in the program	Comment, Formula
Input			
Horizontal	cm	$\sigma_{e,x}$, iEbeam.sigma_x_cil	
Vertical	cm	$\sigma_{e,z}$, iEbeam.sigma_y_cil	
Linear electron density	1/cm	λ_e , iEbeam.Ne_cil	
Output			
Beam current	A	I_e , iEbeam.Ie_cil	$I_e = e\lambda_e\beta c$

The Tab Sheet **Gaussian bunch** (Fig. 26) includes the following input and output variables:

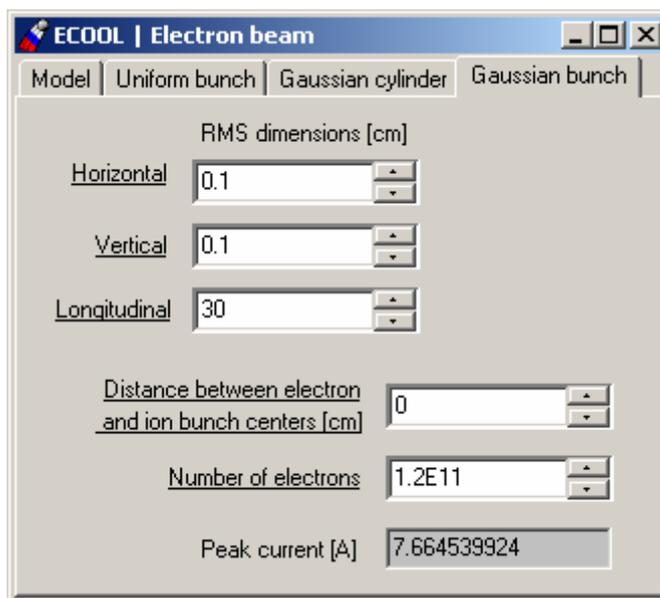


Fig. 26. Window of the **ECOOL | Electron beam** menu item. Tab sheet **Gaussian bunch**.

Variable caption	Unit	Variable in the program	Comment, Formula
Input			
Horizontal	cm	$\sigma_{e,x}$, iEbeam.sigma_x	
Vertical	cm	$\sigma_{e,z}$, iEbeam.sigma_y	
Longitudinal	cm	$\sigma_{e,s}$, iEbeam.sigma_s	
Distance between electron and ion bunch centres	cm	iEbeam.dist	
Number of electrons	-	N_e , iEbeam.Ne	
Output			
Peak current	A	I_e , iEbeam.Ieb	$I_e = \frac{\beta c e N_e}{\sigma_{e,s} \sqrt{2\pi}}$

1.2.3.3. Library of the friction forces

The friction force components acting on the ion inside the electron beam can be calculated using different analytic formulae and, in principle, using results of numerical calculations. Choice between different presentation of the friction force is provided by radio button **Model** of the tab sheet **Model** of the **ECOOOL | Friction force** menu item window (Fig. 27). Now in the program realized the following formulae:

Budker formula

Non-magnetized;

Derbenev-Skrinsky-Meshkov;

Parkhomchuk;

Tabulated.

In the program the friction force formulae are switched using variable `iForce.Type` which is equal to the following numbers:

Type of force presentation	iForce.Type value
Budker	0
Non-magnetized	1
Derbenev-Skrinsky-Meshkov	2
Parkhomchuk	3
Tabulated	4

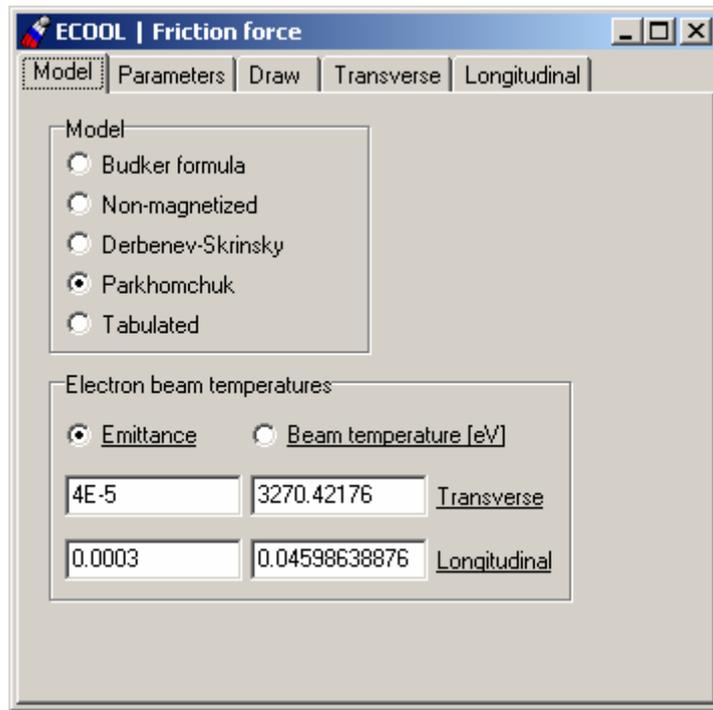


Fig. 27. Window of the **ECOOL | Friction force** menu item. Tab sheet **Model**.

If **Tabulated** model is chosen then user must point file with tabulated values of friction force component pre-calculated by another program (see **ECOOL | Tabulated**).

Here temperature (or emittance) for the electron beam can be defined with corresponding parameters:

Variable caption	Unit	Variable in the program	Comment, Formula
Input			
Emittance/temperature		iEbeam.emit_temp	Choice of the electron beam temperature presentation
Beam temperature Transverse	eV	T_{\perp} , iEbeam.F.Ttemp_centre	
Longitudinal	eV	T_{\parallel} , iEbeam.F.Ltemp	
Beam emittance Transverse	m	iEbeam.e_emit_tr	
Longitudinal		iEbeam.e_dpp	

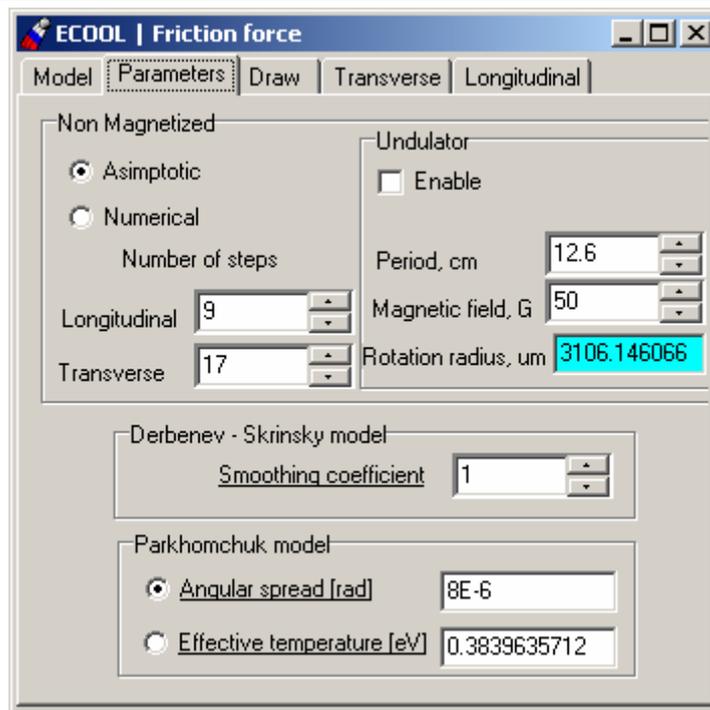


Fig. 28. Window of the **ECOOL | Friction force** menu item. Tab sheet **Parameters**.

The Tab Sheet **ECOOL | Friction force | Parameters** (Fig.28) is used to tune and define parameters for the chosen friction force model calculation.

If **Non-Magnetized** friction force model is chosen, user can change the following parameters in the corresponding **Panel**:

Radio button **Asimptotic** or **Numerical** – choice of the calculation method for this model – either using asymptotic assumption (asymptotic formulae obtained with Coulomb analogy are used), or integrating over longitudinal and transverse ion velocities (variable `iForce.asimptotic` for choice and variables `iForce.dl`, `iForce.dt` in the program code correspondingly).

Another case of non-magnetized cooling – is using of “4- π undulator” – here Larmor rotation of electrons is substituted with rotation of them due to influence of undulator.

This case can be taken into assumption when Check Box **Enable** is switched in **Undulator** Panel. User must put main undulator parameters:

Variable caption	Unit	Variable in the program	Comment, Formula
Input			
Period	cm	<code>iEbeam.F.lambda</code>	$\lambda = \frac{\theta 2\pi pc}{eB}$
Magnetic field	G	<code>iEbeam.F.B_field</code>	
Output			
Rotation radius	μm	<code>iEbeam.F.r_0</code>	$r_0 = \frac{\theta \lambda}{2\pi} = \frac{eB \lambda^2}{4\pi^2 pc}$

If **Derbenev-Skrinsky-Meshkov** model is chosen, user can define so-called **Smoothing coefficient** (variable `iEbeam.F.Smoos` in the code)

If **Parkhomchuk** model for the friction force calculation is chosen, user can define parameter **Effective temperature [eV]** (or it can be interpreted as **Angular spread [rad]**) which is used in

Parhomchuk formula (variable `iEbeam.F.TempEff` or `iEbeam.F.Theta_Eff` correspondingly in the code). There are Edit

Tab Sheet **ECOOL | Friction Force** (Fig. 29) contains a helpful toolkit for checking 3D shape of friction force. There are Edit windows for boundaries of transverse and longitudinal ion velocities (minimum and maximum values) and division number in every range. In the program code they are following variables:

Variable caption	Unit	Variable in the program	Comment, Formula
Input			
Transverse velocity - Minimum value - Maximum value	[m/sec]	<code>iForce.Vtr_min</code> <code>iForce.Vtr_max</code>	
Long. velocity - Minimum value - Maximum value	[m/sec]	<code>iForce.Vlong_min</code> <code>iForce.Vlong_max</code>	
Divisions		<code>iForce.div</code>	

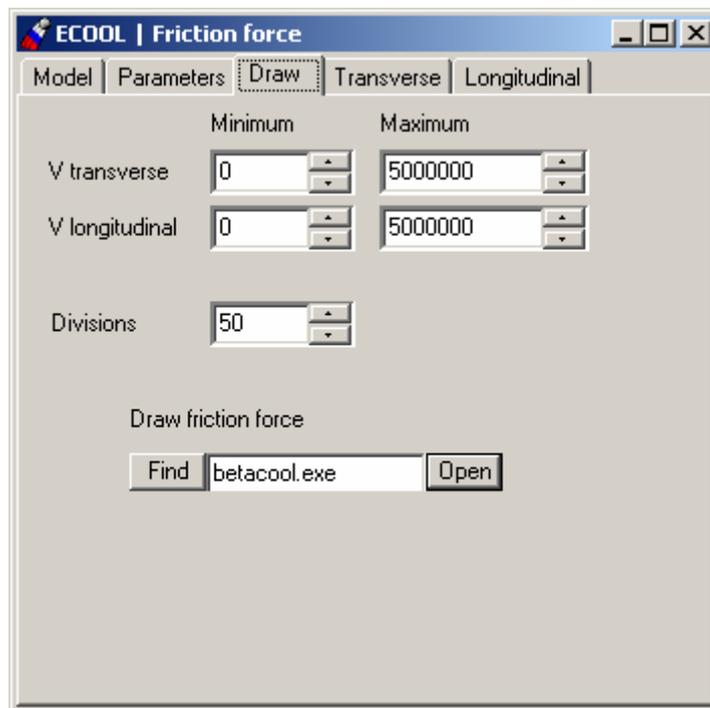


Fig. 29. Window of the **ECOOL | Friction force** menu item. Tab sheet **Draw**.

Component **TBrowse Draw force shape** is used for visualization of the friction force dependence on ion velocity. Button **Open** of the **TBrowse** starts **BETACOOOL** program with parameter `/fr`. At this parameter **BETACOOOL** calculates and saves into the disk two 3D plots: `FFtr.sur` and `FFlong.sur`. The 3D plots are loaded and visualized into corresponding tab sheets **Transverse** and **Longitudinal** of the **ECOOL | Friction force** menu item window (see example of the friction force calculation in the Fig. 30).

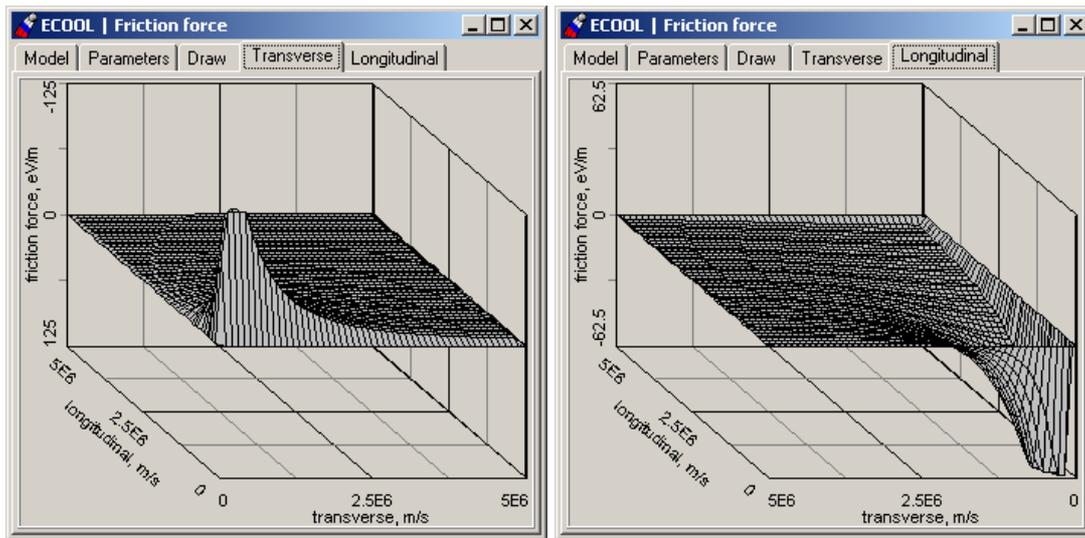


Fig. 30. Example of the friction force shape calculation.

The friction force is performed in [eV/m] units, the velocity components are measured in [m/sec].

1.2.3.4. Tabulated friction force

Window **ECOOL | Tabulated** is a special toolkit for testing and interpolation of tables with pre-calculated transverse and longitudinal friction force values which are created by another code. This window has following Tab Sheets:

Tab Sheet **Control** (Fig. 31) – here 3 **TBrowse** components – two for choosing files containing tables with transverse and longitudinal friction force values:

- **File with transverse velocity table**
- **File with longitudinal velocity table**

These files have to have a special extensions: for transverse component - *.tvt, for longitudinal component *.lvt.

And one **TBrowse** component

- **Generate table with velocities**

Here user must launch betacool.exe to generate tables with friction force values in accordance with mesh parameters and interpolation method which are defined in next Tab Sheets on this window;

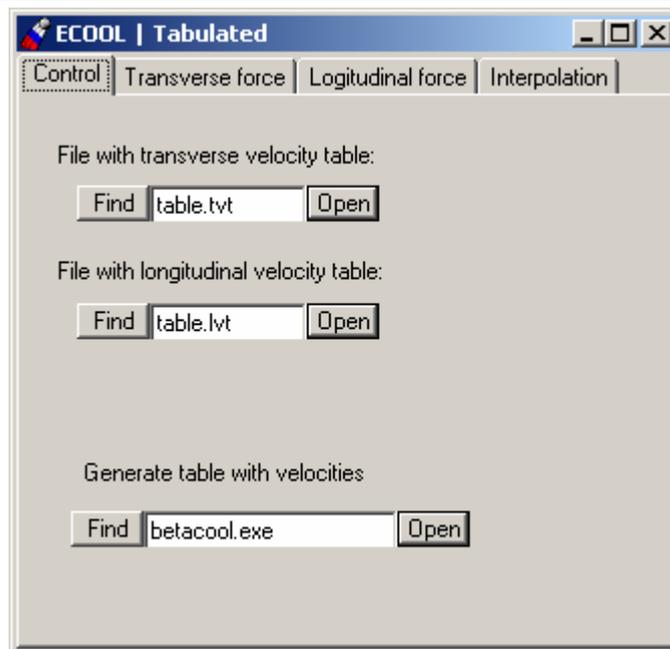


Fig. 31. Window of the **ECOOL | Tabulated** menu item. Tab sheet **Control**.

Tab Sheet **Transverse Force** (Fig 32) is used for definition of the parameters of the table for transverse component of the friction force. This table is a mesh with friction force values in nodes which is generated versus transverse and longitudinal ion velocity values. Mesh for this table has 3 ranges for every velocity with independent splitting in every range. Parameters of this mesh are presented on this Tab Sheet. First range is from 0 to **1st point** value with splitting **steps in range**, second range is from **1st point** to **2nd point** with splitting **steps in range** corresponding to **2nd point**, third range is from **2nd point** to **3rd point** with splitting **steps in range** corresponding to **3rd point**. Tab Sheet **Longitudinal Force** (Fig 32) is used for definition of the parameters of the table for longitudinal component of the friction force. All the parameters are analogue to Tab Sheet **Transverse Force**.

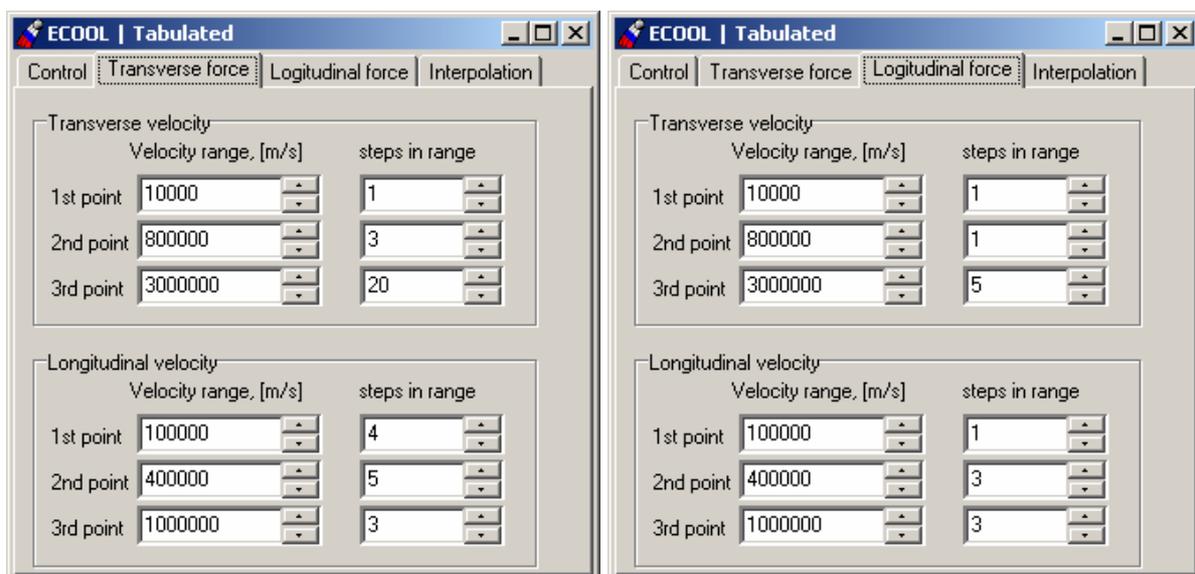


Fig. 32. Tab Sheets **Transverse force**, **Longitudinal force**.

Tab Sheet **Interpolation** (Fig 33) is used for definition of the interpolation method for processing of the tables with pre-calculated friction force values. Here Radio group Type is presented and user can choose one of three possible interpolations:

- **Linear** (linear interpolation – closest nodes. For uniform mesh)
- **Bilinear** (non-linear interpolation. For uniform mesh)
- **Triangle** (non-linear interpolation. For arbitrary non-uniform mesh).

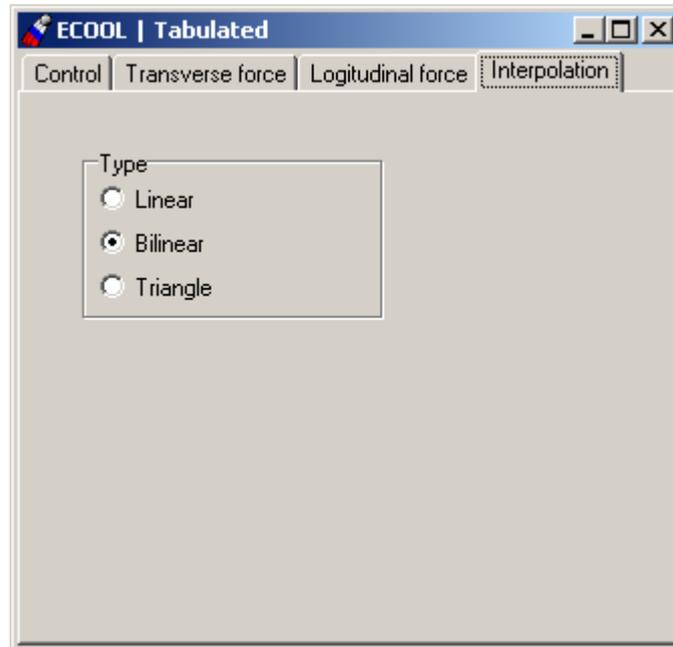


Fig. 33. Tab Sheet **Interpolation**.

1.2.4. Menu item **Effects**. **Library of the Effects**

All the effects using in BETACOOOL but **Electron Cooling** are collected in the menu item **Effects** including the following submenu items:

Collision Point
Internal Target
Intrabeam Scattering
Rest Gas
Additional Heating
Particle Losses.

1.2.4.1. Menu item **Effects**. **CollisionPoint Effect**.

Window of the **Effects | Collision Point** (Fig. 34) includes Toolkit for Luminosity and connected parameters calculation in the collision points. Here the choice of the **Luminosity calculation model** is provided and four Tab Sheets for the parameters definition: **Parameters, Divisions, Luminosity, Beam-beam**.

A possibility to use parameters of the colliding beam if it has another species to the main one is foreseen – Check Box **Use Colliding Beam Parameters**.

Model of luminosity calculation includes three types of model and is represented in the program code as `int xColl::LuminosityModel` variable. Algorithm in different models are following:

Local density – here luminosity is calculated for every particle of the model beam through the density of the head-on beam in the point of the particle position.

Coordinate ellipsoid – here luminosity is calculated for the elliptical layers inside the model beam through the density of the corresponding layer of the head-on beam. Density in the head-on beam is calculated through the particle coordinates.

Profile density – here luminosity is calculated for the spherical layers inside the model beam through the density of the corresponding layer of the head-on beam. Density in the head-on beam is calculated with the averaging over betatron oscillations.

CheckBox **Real-time calculation** is intended for the fast calculation of the luminosity dependence on the particles per division (see plot on the **Luminosity** Tab Sheet of the current window).

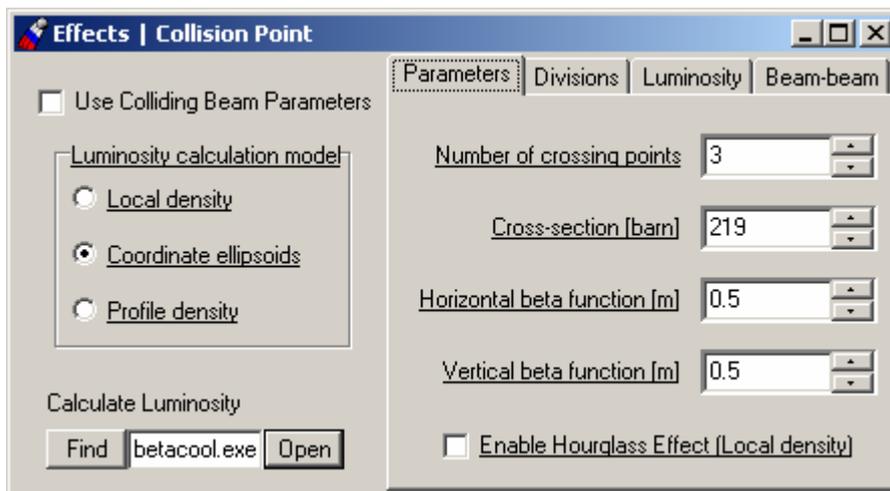


Fig. 34. Window of the **Effects | Collision** menu item. TabSheet **Parameters**

The Tab Sheet **Parameters** (Fig.34) is intended for the definition of the main characteristics of the collision point and includes next parameters needed for calculation:

Variable caption	Unit	Variable in the program	Comment, Formula
Input			
Number of crossing points	-	int iColl.Points	Number of interaction (collision) points in the ring
Cross-section	barn	int iColl.Cross	Cross-section of the collision
Horizontal beta_function	m	β_x , Lattice.betax	Horizontal beta-function in the collision point
Vertical beta_function	m	β_y , Lattice.betay	Vertical beta-function in the collision point

CheckBox **Enable Hourglass effect (Local density)** – when checked and the model of the luminosity Local Density is chosen – takes into account the so-called hourglass effect in the collision point.

The Tab Sheet **Divisions** (Fig.35) is intended for the real-time luminosity calculation for the fast estimation of the luminosity vs number of particles. It includes next parameters needed for calculation:

Variable caption	Unit	Variable in the program	Comment, Formula
Input			
Particles/Divisions	-	int iColl.Divisions	Number of particles in the unit division of density

from		int iColl.From	Start and final parameters, and step for the calculation
upto		Lattice.Upto	
step		Lattice.Steps	

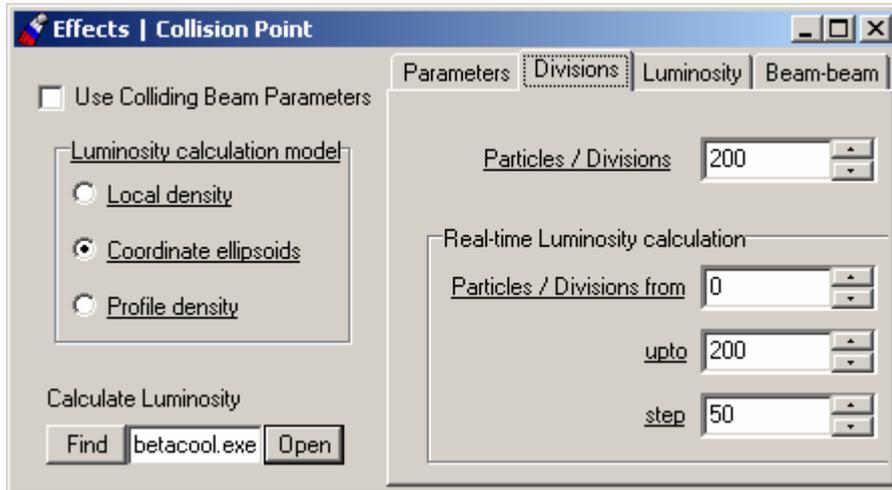


Fig. 35. Window of the **Effects | Collision** menu item. TabSheet **Divisions**

The Tab Sheet **Luminosity** (Fig. 36) contains plot with dependence of luminosity on particles-per-cell. It is necessary for making real-time fast estimation of the luminosity.

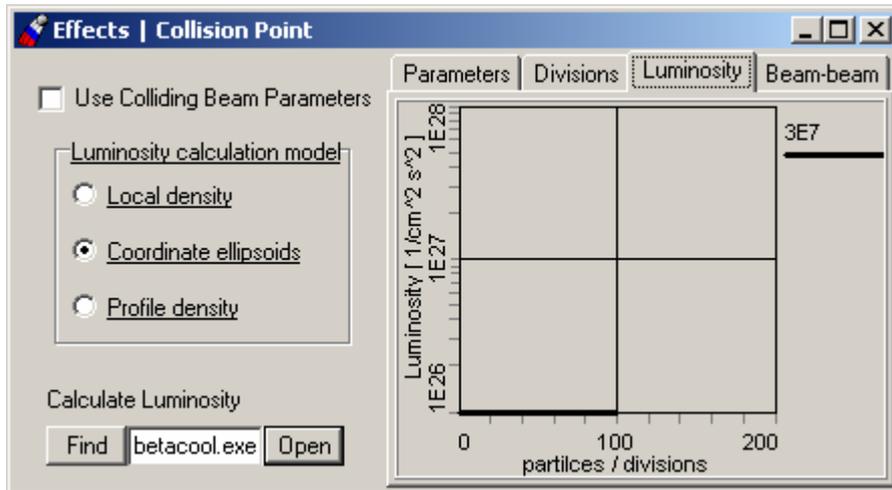


Fig. 36. Window of the **Effects | Collision** menu item. TabSheet **Luminosity**

The Tab Sheet **Beam-beam** (Fig. 37) is intended for the definition how beam-beam parameters (horizontal and vertical) will be calculated. These parameters are represented in the code and are calculated as a following:

Variable caption	Variable in the program	Comment, Formula
Beam-beam parameter	$\xi_{x,y}$ iColl.Kappa_x iColl.Kappa_y	$\xi_{x,y} = \frac{N \cdot \beta_{x,y}^*}{4\pi} \frac{Z^2 e^2}{A \cdot m_p c^2} \frac{(1 + \beta^2)}{2\gamma\beta^2} \frac{2}{\sigma_{x,y}(\sigma_{x,y} + \sigma_{y,x})}$

Choice for the **Emittance for the beam-beam** (which the $\sigma_{x,y}$ are calculated via by) defines which emittance value will be taken for the calculation: usual RMS, FWHM value, or the emittance value

occupied by pointed percent of particle number. Correspondingly the number of particles for the beam-beam formula will be recalculated and taken in accordance with the chosen emittance. Evolution of the Luminosity and Beam-beam parameters in time is calculated and is visualized into the window **Beam | Evolution Tab Sheets Luminosity and Beam-beam**.

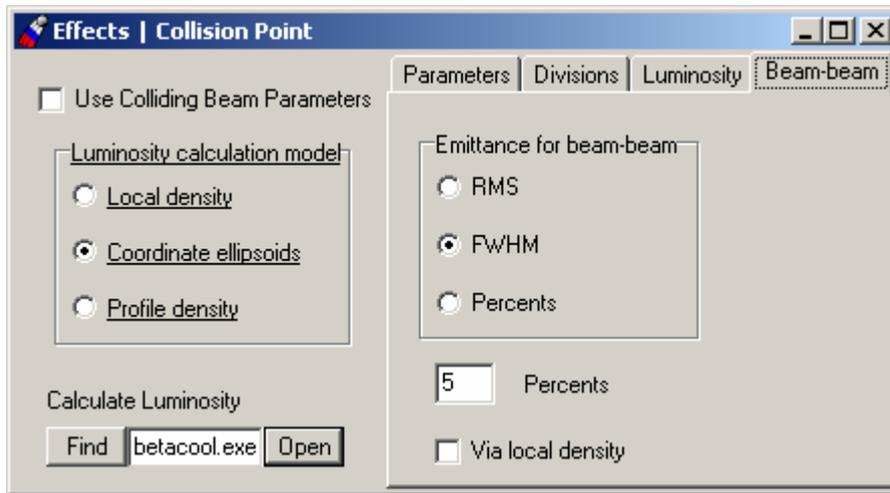


Fig. 37. Window of the **Effects | Collision** menu item. TabSheet **Luminosity**

1.2.4.2. Menu item Effects. Internal Target Effect.

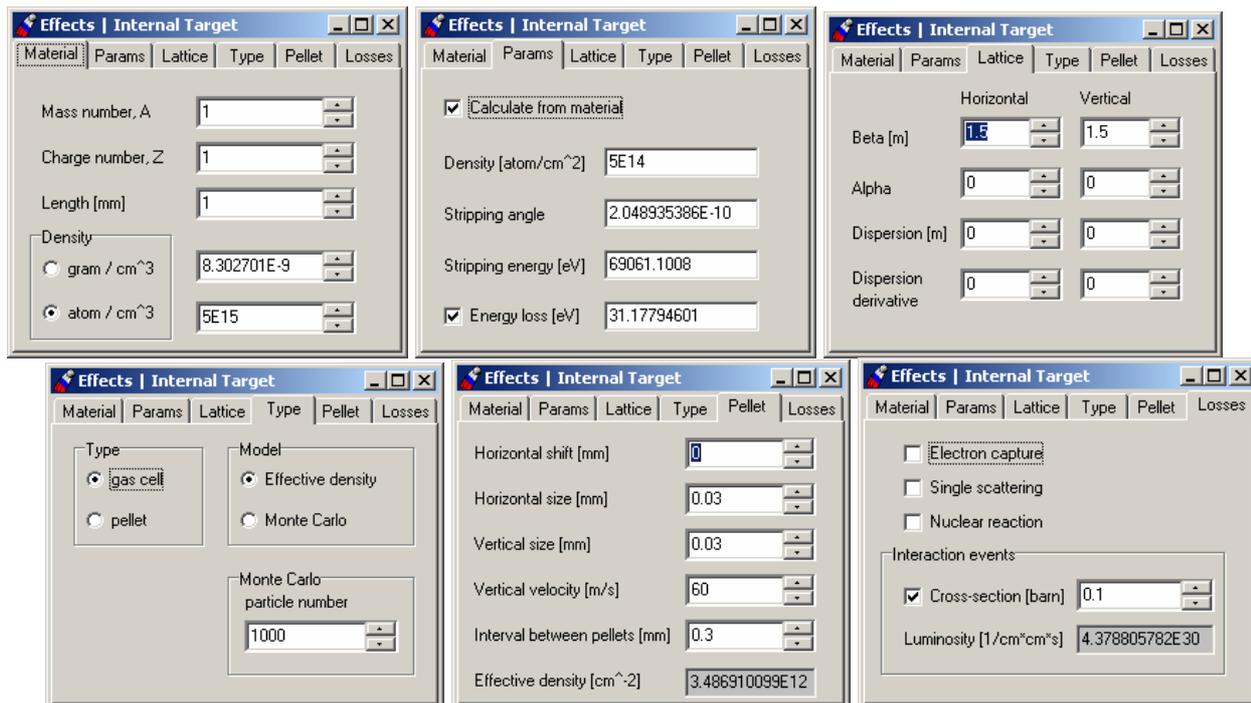


Fig. 38. Window of the **Effects | Target** menu item.

Window of the **Effects | Target** (Fig.38) includes five tab sheets: **Material**, **Params**, **Lattice**, **Type**, **Pelleta** and **Losses**. Tab sheet **Material** defines Mass number and Charge number of the target atoms, Length and Density of the target. Tab sheet **Material** shows parameter of the target which can be calculated from material if check box **Calculate from material** is switch on. Check box **Energy loss** indicates the using of the energy loss value in the simulation. Tab sheet **Lattice** defines the lattice functions in the target position. Tab sheet **Type** is used for choosing of the target **Type** (gas jet or pellet) and simulation **Model** (Effective density or Monte Carlo). Tab Sheet **Pellet** describes parameters of the pellet target.

Tab Sheet Losses gives a possibility to take into account particle losses on target interaction. There are 3 kind of losses are foreseen

- Electron capture (recombination)
- Single scattering
- Nuclear reactions

Cross-section for probability of every event can be defined as a value in [barn].

In the program code parameters from this Tab Sheet are presented as a following variables:

Variable caption	Unit	Variable in the program	Comment, Formula
Input			
Electron capture	-	iTarget.ElectronCapture	
Single scattering		iTarget.SingleScattering	
Nuclear reaction		iTarget.NuclearReaction	
Cross-section	[barn]	iTarget.CrossSection	

More detailed explanation of the target object is given in physical description of BETACOOOL program.

1.2.4.3. Menu item Effects. Intrabeam Scattering Effect.

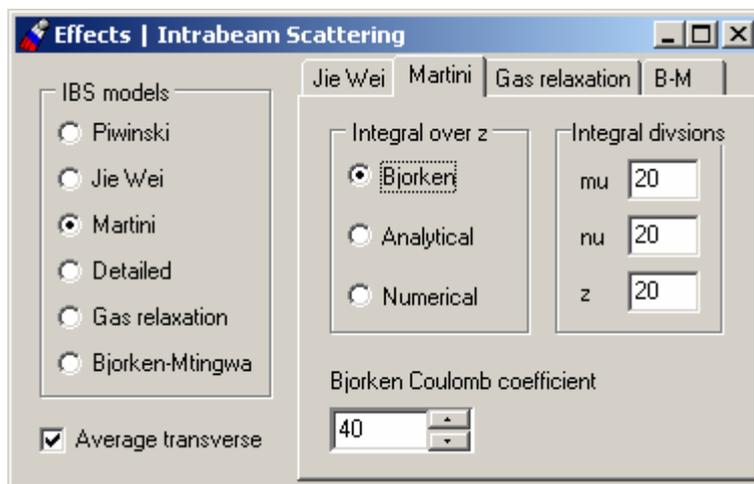


Fig. 39. Window of the **Effects | IBS** menu item.

Window of the **Effects | IBS** menu item (Fig. 39) includes radio group **IBS models** and four Tab Sheets for more detailed description and definition of the chosen calculation model.

Also it contains Check Box **Average transverse rates**. When it is checked (in the program variable `iIBS.coupled` is equal `true`) the both transverse growth rates are set to be equal half-sum of calculated horizontal and vertical rates.

Radio group **IBS models** provides a choice between analytical models of diffusion coefficient calculation. **Piwinski** model calculates the growth rates for smoothed ring structure (for mean beta functions and dispersion) and it does not require loading optics structure of the ring. **Jie Wei, Martini and Bjorken-Mtingwa** models calculate the rates by averaging over the ring tacking into account dependence of the lattice parameters on longitudinal co-ordinate. These models require loading optics ring structure from output MAD file (or from another type file). The fourth model is **Detailed** IBS calculation, here for rates calculation this model needs mean beta functions and dispersion. But if dynamics algorithm chosen – one needs to load optics structure of the ring. The fifth model of IBS is **Gas Relaxation** – calculation of growth rates is provided using well known formula of characteristic times of gas relaxation from plasma physics. This model needs mean beta-functions and dispersion, which can be taken as pre-defined parameters or calculated from real lattices by averaging.

Jie Wei model calculates the growth rates analytically (without numerical evaluation of integrals). In the program the models are switched with variable `iIBS.Model` which is equal to 0 - for **Piwinski**, to 1 - for **Jie Wei**, to 2 - for **Martini model**, 3 for **Detailed IBS** calculation, 4 – for the **Gas Relaxation** formula, and 5 – for **Bjorken-Mtingwa** theory.

Martini model calculates integral over three variables. The radio button **Martini integral over z** (Fig. 39) provides a choice of method for calculation of the integral over z variable (Coulomb logarithm calculation). **Bjorken** model sets the value of this integral to some constant ($40/D$ in the text of the program). **Analytical** model calculates the integral as a sum of a few first members of the expansion into series. The **numerical** model provides integration numerically. Edit windows **step over mu**, **step over nu**, **step over z** input number of integration steps over corresponding variable. Corresponding variables in the program are: `iIBS.stepmu`, `iIBS.stepnu`, `iIBS.stepz`. **step over z** is used for numerical model only. These models are switched with variable `iIBS.Model2` in the program, which is equal to 0 for **Bjorken**, to 1 for **Analytical** and to 2 for **numerical** model.

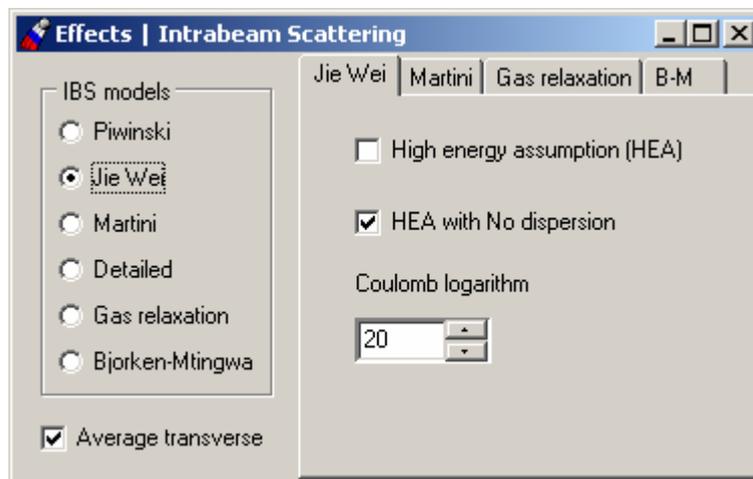


Fig. 40. Window of the **Effects | IBS** menu item. Jie Wei Model.

For the Jie Wei model (Fig. 40) there are two check boxes to choose some modifications in this model. For details see the BETACOOOL description, section, dedicated to IBS modelling.

If check box **High energy assumption** is checked (in the program variable `iIBS.coupledJie` is equal **true**) then works case for beams which are stored at energies much higher then the transition energy. Due to coupling and injection conditions, the horizontal and vertical betatron amplitudes are about the same. Here in formulae for growth rates the value of transition energy will be taken instead of usual gamma factor.

If check box **HEA with no Dispersion** is checked (in the program variable `iIBS.NoDispersion` is equal **true**) together with **High energy assumption** Check Box this is a case when r.m.s. beam parameters (σ_x and σ_p) are related only by the average dispersion D_p . And the asymptotic configuration for form factor d can be reached ($d \approx 2n/(1+2n)$).

For the **Detailed** model – there are no switches. The formula for longitudinal rate calculation is analytic and was obtained by A. Burov. It has two-dimensional integral inside. To define number of steps for those integrals following variables in the program are used: `iIBS.stepmu`, `iIBS.stepnu`. To change these values one has to use edit windows **step over mu**, **step over nu** on the Tab Sheet intended for Martini model.

For the **Gas Relaxation** model (Fig. 41) there is a Tab Sheet with two edit windows for output of calculated average dispersion and dispersion derivative of the investigated structure.

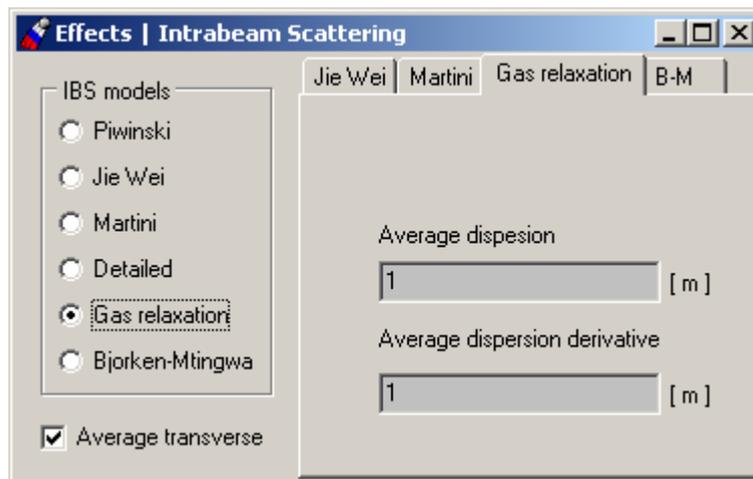


Fig. 41. Window of the **Effects | IBS** menu item. **Gas Relaxation** Model.

For the **Bjorken - Mtingwa** model (Fig. 42) there is a Tab Sheet **B-M** with following parameters of the calculation to be defined:

Check Box **Coulomb Log** – numerical value for the Coulomb logarithm in formula for timerate in Bjorken-Mtingwa assumption (variable `iIBS.B_M_log` in the code).

Two next parameters are defining limits and number of steps for main integral in theory that is defined in terms of matrix Λ :

Edit window **Upper limit** – upper limit of integral (variable `iIBS.lam_limit` in the code) in units of the maximal of following three parameters: $\frac{\beta_x}{\varepsilon_x}$, $\frac{\beta_y}{\varepsilon_y}$ or $\frac{\gamma_0^2}{\sigma_p^2}$ which are analyzed during calculation.

Edit window **number of steps** – number of integration steps (variable `iIBS.step_lam` in the code).

Check Box **High Energy assumption** – if checked, then this IBS effect is calculated using Gas Relaxation formalism (variable `iIBS.BMHEA` in the code).

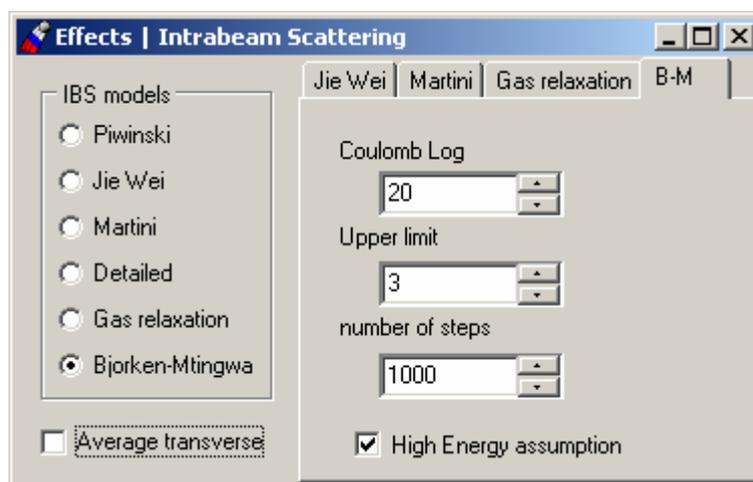


Fig. 42. Window of the **Effects | IBS** menu item. **Gas Relaxation** Model.

1.2.4.4. Menu item Effects. RestGas Effect.

Window of the **Effects | Rest Gas** (Fig. 43) includes a kit of parameters characterized vacuum composition in the ring. The list of following parameters on the Tab Sheet **Vacuum composition** is presented:

Pressure [Torr] (variable `iRestGas.Pressure` in the program code);

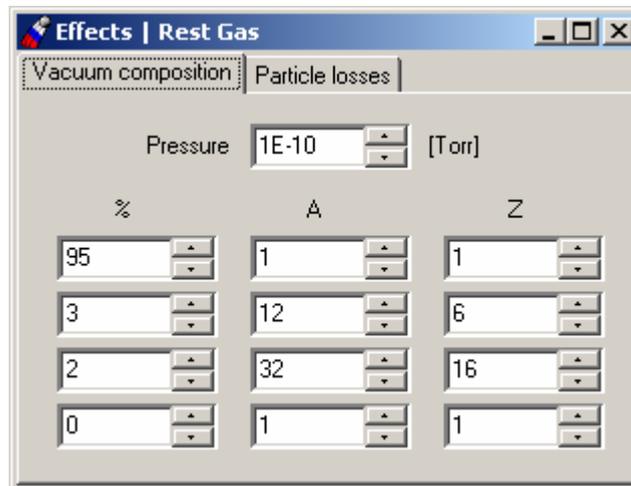


Fig. 43. Window of the **Effects | RestGas** menu item. Tab Sheet **Vacuum composition**.

Four-component vacuum is presumed for now in the program. Below the above mentioned parameters the table of the vacuum components is situated. For every component provided the following kit of parameters: percentage of the component in a whole composition (%), Atomic number of the component (**A**) and Charge number correspondingly (**Z**). In the program these are variables: `xMaterial iRestGas.Comp[]`.

Growth rates due to the scattering on the residual gas are calculated with the same mechanism to the internal target effect (for example gas-jet target). Accordingly to the residual gas pressure the effective density of the gas is calculated, then energy loss on scattering with Bethe-Bloch formulae, r.m.s. angles of ions after scattering on atoms of the residual gas, and emittance and momentum deviation are calculated consequently. After in accordance with the percentage of components the characteristic growth rates are obtained. More detailed information is given in physical description of BETACOOOL program.

Tab Sheet **Particle losses** (Fig 44) has one Edit window and three Check Boxes to define type of particle losses to take into account during calculations:

Edit window **Mean vacuum chamber radius** [cm] (this variable is presented in program code as `iBeam.a` variable),

Variable caption	Unit	Variable in the program	Comment, Formula
Input			
Electron capture	-	<code>iRestGas.ElectronCapture</code>	
Single scattering		<code>iRestGas.SingleScattering</code>	
Nuclear reaction		<code>iRestGas.NuclearReaction</code>	

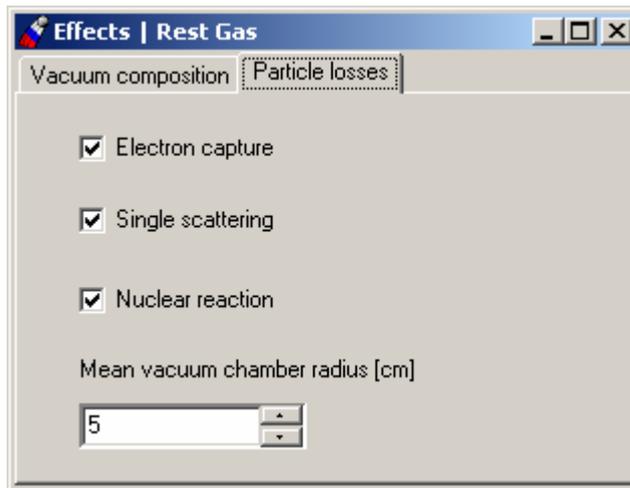


Fig. 44. Window of the **Effects | RestGas** menu item. Tab Sheet **Particle losses**.

1.2.4.5. Menu item Effects. Additional Heating Effect.

Window of the **Effects | Heating** menu item (Fig. 45) includes for Check Boxes to switch different types of additional heating:

- **Constant** rate growth
- **Linear** deviation of emittances
- **Diffusion** power of rate variation
- **Diffusion** heating of emittances

and four Tab Sheet: **Constant rate, Linear deviation, Diffusion power, and Diffusion heating**. Each Tab Sheet has factor of characterizing parameters deviation:

Tab Sheet Constant Rate

Variable caption	Unit	Variable in the program	Comment, Formula
Input			
Horizontal rate	1/sec	iHeat.Rate[0]	
Vertical rate	1/sec	iHeat.Rate[1]	
Longitudinal rate	1/sec	iHeat.Rate[2]	

Tab Sheet Linear Deviation

Variable caption	Unit	Variable in the program	
Input			
Horizontal emittance	$\pi \cdot \text{mm} \cdot \text{mrad}/\text{sec}$	iHeat.Linear[0]	
Vertical emittance	$\pi \cdot \text{mm} \cdot \text{mrad}/\text{sec}$	iHeat.Linear[1]	
Momentum spread	mrad^2/sec	iHeat.Linear[2]	

Tab Sheet Diffusion power

Variable caption	Unit	Variable in the program	
Input			
Horizontal power	1/sec	iHeat.Power[0]	

Vertical power	1/sec	iHeat.Power[1]	
Longitudinal power	1/sec	iHeat.Power[2]	

Tab Sheet **Diffusion heating**

Variable caption	Unit	Variable in the program	
Input			
Horizontal emittance	$\pi \cdot \text{mm} \cdot \text{mrad}^2/\text{sec}$	iHeat.Diffusion[0]	
Vertical emittance	$\pi \cdot \text{mm} \cdot \text{mrad}^2/\text{sec}$	iHeat.Diffusion[1]	
Momentum spread	mrad^4/sec	iHeat.Diffusion[2]	

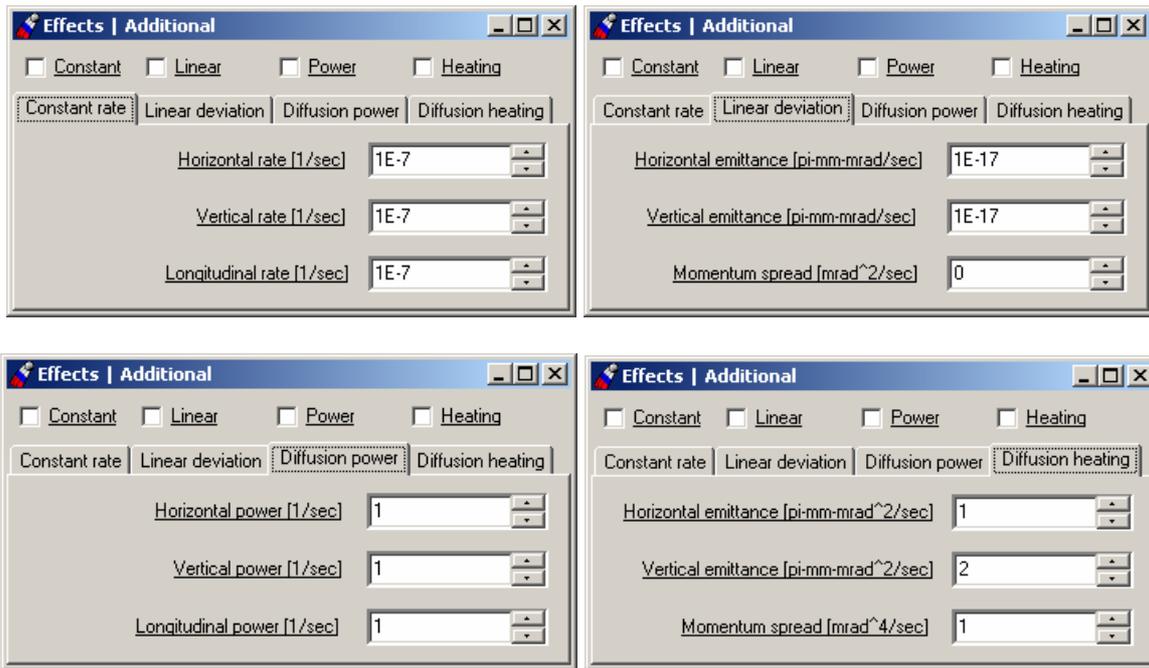


Fig. 45. Window of the **Effects | Heating** menu item. Tab Sheet **Constant rate**.

1.2.4.6. Menu item Effects. Particle Losses Effect.

Window of the **Effects | Heating** menu item (Fig. 46) includes list of the effects and factors which lead to the particle losses in the storage ring. Here are two lists presented. The first one is for the **Active Effects** and it contains four factors which cause particle losses due to the active switched effects:

- **Electron Capture in ECOOL**
- **Scattering on Rest Gas**
- **Internal Target**
- **Collision Point**

All this effects lead to the decreasing of the particle number in the beam.

When the check Box **Electron capture n ECOOL** is checked the particle losses due to the recombination in the cooling section are tacking into account in the loss rate calculation. In this case the Boolean variable `iEcool.Loss` is equal `true` in the opposite case – `false`.

When Check Box **Scattering on Rest Gas** is checked the particle losses due to the scattering on residual gas are taken into account. In this case the Boolean variable `iRestGas.Loss` is equal `true` in the opposite case – `false`.

When Check Box **Internal Target** is checked the particle losses due to the scattering on the switched target are taken into account. In this case the Boolean variable `iTarget.Loss` is equal `true` in the opposite case – `false`.

When Check Box **Collision Point** is checked the particle losses due to the effects in the interaction point are taken into account. In this case the Boolean variable `iColl.Loss` is equal `true` in the opposite case – `false`.

The second list on the form contains factors which cause **Additional Losses**:

- **Life time (Decay)**
- **Acceptance**
- **Separatrix Length**

User can choose which effect is to be taken in calculations.

When Check Box **Life time (Decay)** is checked the particle losses due to the natural decay of the particles is taken into account. In this case the Boolean variable `iLosses.Decay` is equal `true` in the opposite case – `false`.

When Check Box **Acceptance** is checked the particle losses due to the aperture boundaries is taken into account. In this case the Boolean variable `iLosses.Acceptance` is equal `true` in the opposite case – `false`.

When Check Box **Life time (Decay)** is checked the particle losses due to the particles coming out of the separatrix is taken into account. In this case the Boolean variable `iLosses.Separatrix` is equal `true` in the opposite case – `false`.

GroupBox **Generate new particle on distribution** – allows to recover lost particle in the test beam. BETACOOOL works with two arrays of particles (beams) – the first one (major) – it contains real number of particles. The second one – is test beam – it usually contains several thousands particles and is used for dynamics simulation in accordance with active effects. When particle is lost due to some factors we have to decrease number of particles in the major beam, but the number of particles in the test beam has to stay unchanged.

So this Group Box allows to make a choice how lost particles will be recovered in the test beam in accordance with defined distribution from the list: **Gaussian**, **Real**, or **None**. If Group Button **Off** is chosen – then the option of particle re-generation is switched off.

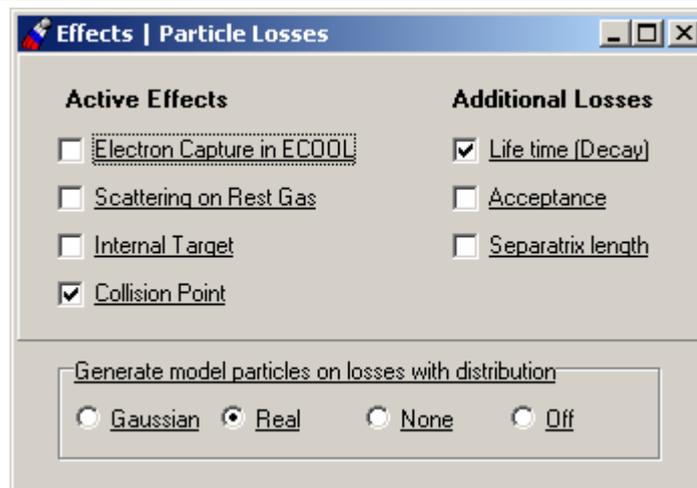


Fig. 46. Window of the **Effects | Particle Losses** menu item.

1.2.5. Menu item **Task. Procedures of BETACOOOL program**

Input parameters for general procedures of the program are collected in the menu item **Task**, which includes submenu items: **Parameters, Rates, RMS Dynamics, Model Beam, Tracking**.

The windows of submenu item **Task | RMS Dynamics, | Model Beam and | Tracking** are used to provide a beam tracking and beam parameters recalculation correspondingly to the chosen numerical model. TBrowse components of this window start selected calculation process.

The window of submenu item **Task | Rates** collects input parameters for calculation of the sum of inverse characteristic times of emittance variation for all active effects. The TBrowse component of this Window is used for calculation the sum of the rate as a set of four numbers, or as three 3D plots.

The window of submenu item **Task | Parameters** is a tool to control parameters of the launched algorithms and output.

1.2.5.1. Calculation of emittance variation rates

The Window of the **Task | Rates** menu item (Fig. 47) includes:

- TBrowse component **Calculate Diagrams of Rates** which starts the calculations;
- the panel **Active Effects** including the check boxes for all effects determined in the program;
- four edit windows for representation of sum of the rates of corresponding beam parameter;
- five tab sheets for representation of results in graphic format.

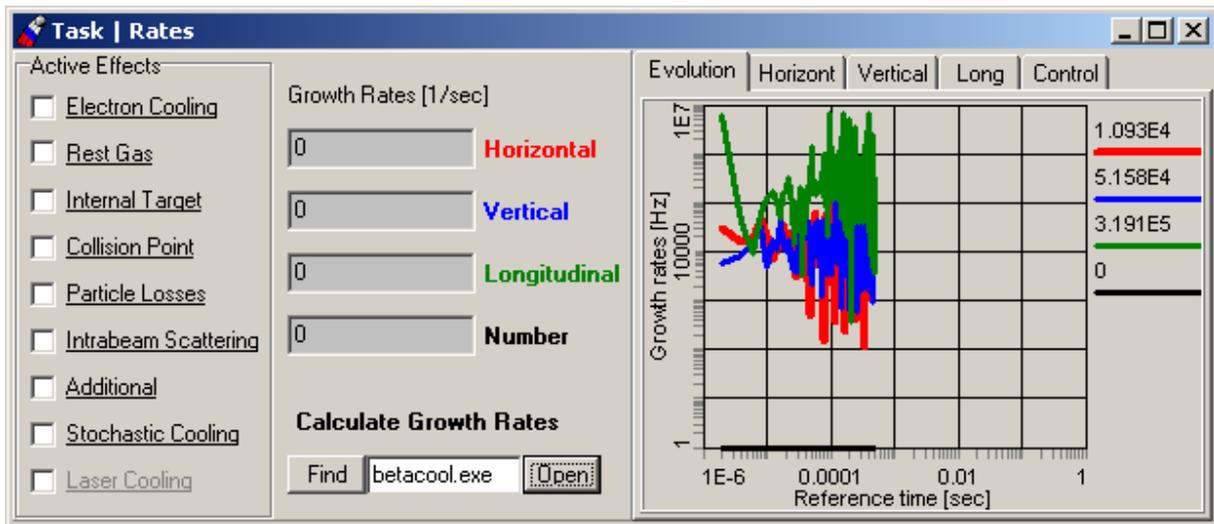


Fig. 47. Window of the **Task | Rates** menu item. Tab sheet **Evolution**.

All the Effect classes have the same parent class `xEffect`, which has a Boolean variable `use`. When the corresponding check box is checked this variable of the effect is **true**. All the variables Effect are put in the array using self counter system. The program calculates sum of the rates in cycle calculating the rates of the effects at `use = true`. In the current version of the program not a whole list of the effects from last version of BETACOOOL is realized. The font color of effect name, which can be used in calculations now, is black. For effects under development the color is gray.

Next useful Tab Sheet on this window is **Control**. It contains:

- the Check Boxes **Evolution**, and **3D Diagrams**;

The check box **3D Diagram** determines the calculation procedure, which starts when push the button **Open** of the TBrowse **Calculate Rates**. If the check box is not checked the program calculates the sum of the rates at the beam emittance values determined in the window of the menu item **BeamParameters**, and represents them into corresponding edit windows. If the check box is ON the program calculates the sum of the rates as three 3D diagrams in the range of the beam emittances determined in the tab sheet **Control** (Fig. 48).

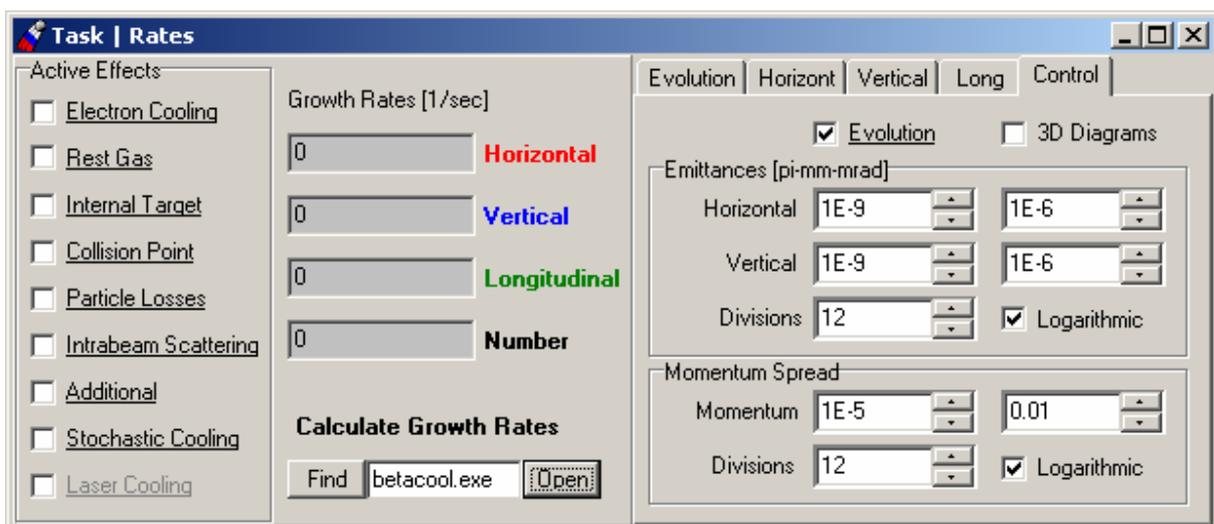


Fig. 48. Window of the **Task | Rates** menu item. Tab Sheet **Control**.

If the Check Box **Evolution** is checked then evolution of the calculated rates (absolute value of the growth rates vs reference time) will be visualized onto plot on the **Evolution** Tab Sheet when pushed the button **Open** of the TBrowse **Calculate Rates**.

An example of 3D diagram of the sum of the rates is presented in the Fig. 49. The tab sheet **Horizont** contains 3D diagram of the sum of the rates of the horizontal emittance variation, **Vertical** – the vertical emittance variation, **Long** – the longitudinal one.

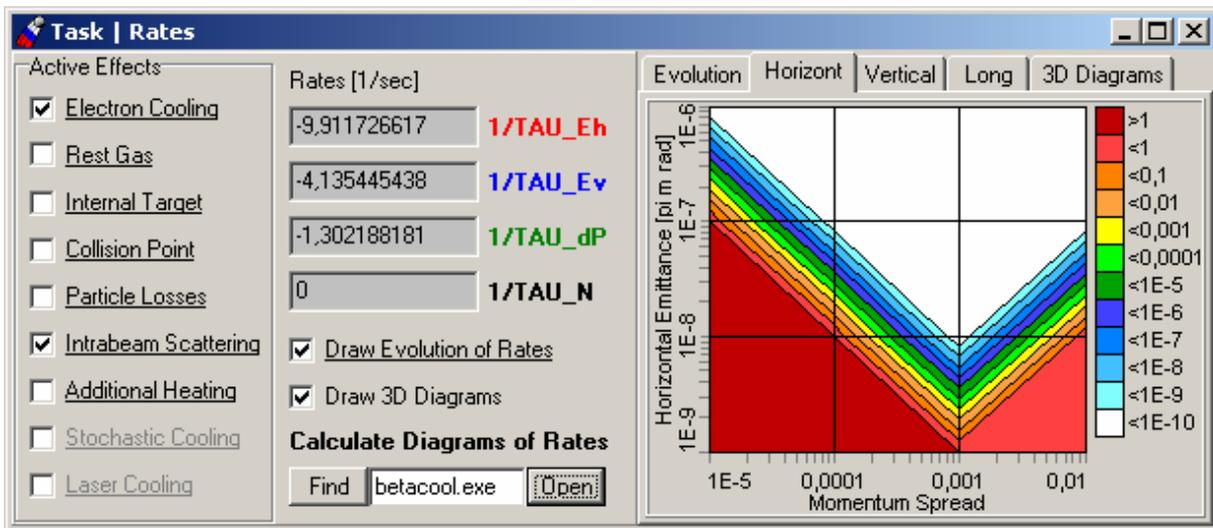


Fig. 49. Window of the **Task | Rates** menu item. Tab sheet **Horizont**.

1.2.5.2 Window Task | Parameters

This window includes two Tab Sheets: **Output** and **Points** (Fig 50).

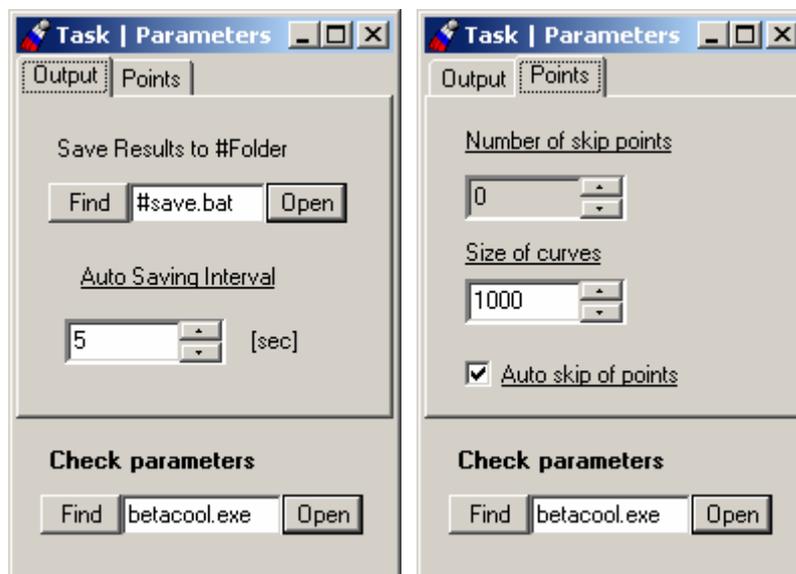


Fig. 50. Window Task | Parameters

The Tab Sheet **Output** controls the process of data output. Edit window **Save Results to #Folder** determines the path to the folder where all the files with calculated results (curves, tables, etc) will be saved. The name of the folder will be identical to the name of the file of initial parameters (for example #rhic.bld). The batch file (save.bat) pointed in this edit window contains a list of files with curves and data which are intended to be saved.

Edit window **Auto Saving Interval** determines how frequently the program saves all calculated results to the indicated folder.

The Tab Sheet **Points** controls the process of data output. Edit window **Number of skip Point** determines how many points of calculated curves will be skipped when saving to disc. Edit window **Size of curves** determines the size of curve (size of array of points) to be saved on disk.

1.2.5.3 Numerical algorithms of beam evolution

Main menu Task has 3 items to launch different algorithm models: **RMS Dynamics**, **Model Beam** and **Tracking**. Each item has its own window to input the calculation process parameters. Below description of those windows is done.

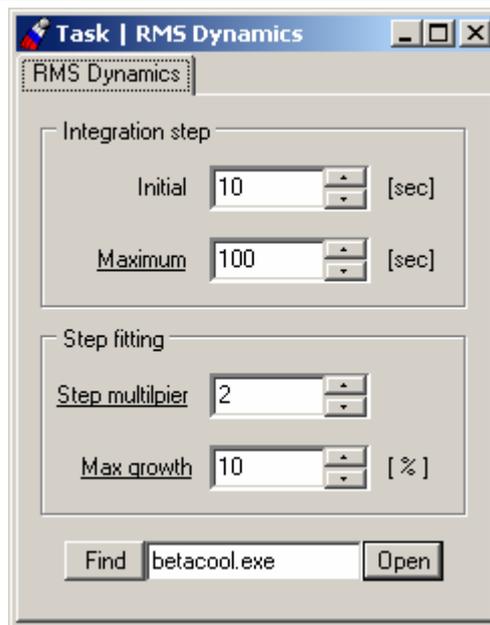
<i>Radio button</i> <i>Task</i>	MAD output file	MAD input file	No optics file	Comments
Dynamics	√	√	√*	* - in case when Piwinsky model of IBS calculation was chosen
Model Beam	√	√	√**	** - if chosen No optics – user has to fill (type in) the ring transformation matrix
Tracking	—	√	—	

Crosses of algorithm models and mode of Optics structure required.

1.2.5.4 Calculation of the beam RMS dynamics

RMS Dynamics – is original BETACOOOL method for beam r.m.s. parameters dynamics (evolution of r.m.s. parameters). To launch this calculation method one beforehand must choose Ring Optics (lattice structure) – with help of Ring Optics window – user must switch on either **Lattice structure file (MAD output)** or **MAD input file** (see below a special table of modes to be switched). The result of knitting of the MAD lines to the continuous chain of optic elements, which will be used for r.m.s. parameters tracking one can see in *.use file. So the goal of the **Dynamics** algorithm is calculation of growth rates of beam r.m.s. parameters evolution.

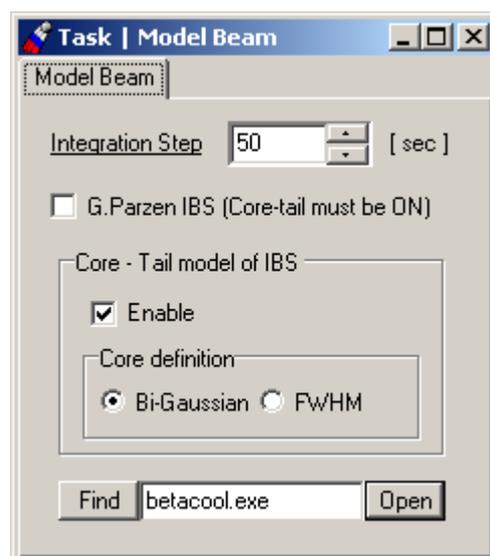
Here one must use the window **RMS Dynamics** (Fig. 51) to change some necessary parameters. In case of this model selection program calculates the beam parameter evolution by Euler method at variable step (variable `iDynamics.Variation` is *true*). At each step over time program checks all the beam emittance and if expected emittance variation is higher then determined in the edit window **Max growth** of the tab sheet **RMS Dynamics**, or expected emittance values after the step have a negative sign, the program divided the step value by the factor determined in the edit window **Step multiplier**. In the program this procedure is controlled by the variables `iDynamics.ratio`, `iDynamics.ifmore` and local Boolean variable `negative` of the procedure `xDynamics::Dynamics()`. **Initial step** of integration can be input from the panel and is controlled by variable `iDynamics.dt`, **Maximum step** can also be input from the panel and is a variable `iDynamics.Maxtdt` in the binary code.

Fig. 51. Window **Task | Algorithm**.

1.2.5.5 Calculation of the Model Beam algorithm

This method of beam tracking through the optics structure is based on the algorithm, realized in program code Simcool made by V.Parkhomchuk. The basic scheme of the algorithm is the following:

- on the first stage a beam is generated with defined parameters;
- in the selected point of the ring in accordance to the current lattices so called “kicks” from active effects are calculated (coordinates and angles of every particle are changed correspondingly). In the earlier versions of the BETACOOOL kicks for two effects only were calculated (electron cooling and IBS). The last version presumes calculation of the influence from any effect in the list as a kick
- obtained vector of coordinates is multiplied *Turn step* times by the transformation matrix of the whole ring;
- go to the first step;

Fig. 52. Window **Task | Model Beam**.

This algorithm has the parameter to be initialized from the window **Task | Model Beam** (Fig. 52) - **integration step**. This parameter is step of the integration in seconds – this is itself the number of the beam turns, by other words – a number of acts of the ring transformation matrix to the vector of coordinate. In the program code this parameter is initialised with double `iDynamics.IntegrationStep` variable.

Check Box **Core/Tail model of IBS Enable** if checked determines that effects of Core-Tail distribution for IBS is taken into account during the calculation. Here user can choose which assumption for core definition – either Bi-Gaussian presentation of particle distribution in core or FWHM parameter of the particle distribution to estimate transverse size (sigmas) of the beam. Another assumption to define distribution parameters in core can be calculated if to choose **G.Parzen IBS** Check Box. This is option which lets to take theory derived by George Parzen (BNL) of Two-Gausses to describe distribution in core

1.2.5.6 Calculation of the particle tracking



Fig. 53. Tab Sheets of the **Task | Algorithm | Tracking** Window.

This method of particle tracking over the optics structure is an eternal cycle of particles over the optics elements. All the elements are collected in a special array `iRingOptics[n]`. Every element has its own index. Every member of this array contains a kit of optic element parameters. The step of tracking is initialized from the **Task | Tracking: Integration step for Forces [cm]**. The number of element particle to “enter” is taken from the `iRing.Index`. Also there is a variable `xTime.s0` that indicates the coordinate of particle inside the optics element. All the necessary parameters of the element (length, etc) are taken accordingly to the `xRing.Index`. New vector of coordinates is recalculated at every step in the presence of acting forces inside the element (right parts of the motion equations) with selected method of integration (Runge-Kutta of the 4th order, Symplectic integrator or Euler). For the IBS effect method of Molecular Dynamics is used.

The Window of the **Task | Tracking** (Fig. 53) includes also two Check Boxes:

- **Check longitudinal crossing** – if switched the number of crossings in the longitudinal direction will be checked. In the program code this parameter is initialised with `bool iDynamics.CheckCrossing` variable.

- **Initial crystalline distribution** – if switched then particles in the beam will generated as an ordered distribution. This option is used when crystalline beam dynamics is simulated. In the program code this parameter is initialised with `bool iDynamics.Crystall` variable. And the Group Box **Equations of motion** which gives a choice what formalism will be used for the particle coordinate vector tracking – either via transformation **Matrixes** or integrating the vectors of Right Parts (**Forces**).

1.3. Work with BETACOOOL independently on interface.

1.3.1. Format of the input files and launching the program.

The input file for the last version of BETACOOOL is developed to use in a convenient way the program in cross-platform regime. For now the interface for Linux version is under development, that's why the user friendly way to use BETACOOOL is independent editing of the input file and a possibility to launch a program with pre-defined parameters and further investigation of the results with any Windows or Linux application.

The format of the input file is developed in such a way that it is easy to collect and edit it in any simple editor. All the parameters are provided with comments and structure of parameters list is connected to the program algorithm.

All the parameters (input and output) which are processing during calculation are kept in global array `Data`. The size of this array is set in accordance with the input file: number of rows corresponds to the maximal number value of the effect (object) or the tag of the Form or Panel with the maximal number (if interface is used). Number of columns in the row corresponds to the number of parameters of the processed effect (object).

This array is generated in accordance to the mentioned numbers and every cell is filled with input parameters.

All the objects and effects in program which have parameters to be input from the form include functions `int OnGet()`, `int OnSet()` and `int OnRun()` as obligatory part of the structure.

Function `OnGet()` accordingly to the effect(object) tag reads necessary parameters from the corresponded rows. Here is the example of the input file is presented:

```
[row=1] Beam | Parameters | Emittance
2.4e-08= Horizontal emittance, pi*m*rad
2.5e-08= Vertical emittance, pi*m*rad
0.001= Momentum spread
1000000000= Number of particles
0= bunched(0) / coasting(1)
1= Collider regime (0/1)
0.001443904867= Mean beam radius, m
8.089343697= Longitudinal form factor
0.1298843665= Longitudinal space charge impedances, Ohm
9390901.33= Transverse space charge impedances, Ohm/m
4.055707549= Peak current, A
0= Emittance definition: RMS(0),CS(1),FWHM(2),%(3)
100= Divisions for FWHM
50; Percents
```

```
[row=2] Beam | Parameters | M.D.
2000= Particle per Cell
```

1= Macro Particle
 1e-06= Impact parametr, m
 0.00766769= Cell size, m
 0.003400342669= Linear density
 2129302213= Initial longitudinal temperature, K
 2.770901656e+10= Initial transverse temperature, K
 3.27286504e-08; Gammal

Parameter [row=] is corresponded to the Form or Panel tag and simultaneously number of Row in the Data array. On the next strings of the input file the list of parameters is presented. The value is separated from the comments with “=”. The last parameter of the effect (object) (means the last column in the row) has to have a marker “;” after equaling.

In this way is easy to understand that values (necessary parameters themselves) can be separated from the comments with “]”, “=” or “;” markers.

After all the parameters are filled in accordance with the input file and are redrawn in edit windows if the interface is used, for some pre-calculation or on-line update the function OnSet () is used. It makes some intermediate calculations and update Data array cells and edit windows on interface.

Another one very useful feature which BETACOOOL gives is OnRun () function. All the parameters which have underlined labels in interface can be changed while program executes and all the procedures will re-calculate with on-line changed parameters. Effects or objects which need such a possibility must have a function OnRun () inside its own class.

To activate this function user must edit necessary parameter and push the button with yellow clock on the main window of the interface (see Fig. 54). Changed parameters will be saved to the file, file will be reloaded and calculation process will be continued with new parameters.

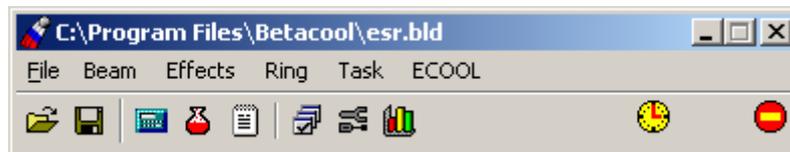


Fig.54. Main window of the BETACOOOL program

To stop the program with correct cleaning the OS memory it is advised to use red button Stop on the main window. In this case a special file-marker will be created in the current directory and as soon as program will find it then it immediately will shutdown with correct saving all the calculated results.

The BETACOOOL program can be started independently on interface program. For this purpose one needs to input in the command line the following:

<path>\betacool.exe <inputfilename> /<parameter>

The program analyses only the first letter in the parameter name. The parameter can be equal to the following letters:

Parameter	Executable procedure, output files of simulation
a	iDynamics. Algorithms () - calls one of the indicated simulation model. Ex2t.cur Ey2t.cur Dp2t.cur Num2t.cur Lum2t.cur Th2t.cur Tv2t.cur

	Tp2t.cur Tn2t.cur
l	iDynamics.Lattice() – used for check a validity of MAD output file output files: BetaX.cur, BetaY.cur, DispX.cur, AlfaX.cur, AlfaY.cur, DispX_.cur.
r	iDynamics.Rates() – calculates growthrates of the rms beam parameters in accordavce with active effects. RateEh.sur RateEv.sur RatedP.sur
f	iDynamics.FFTest() – used for visualisation of the friction force shape output files: FFtr.sur FFlong.sur

1.3.2. Format of the output files.

The files *.cur as a rule contain time dependencies of the beam parameters. Example of the *.cur file format:

```

0      0.0003849600109
18     0.0003703453546
36     0.0003563822554
54     0.0003430464386
72     0.0003303146755
90     0.0003094731154
99     0.0002900512001
108    0.0002719605886
117    0.0002551191811

```

The first column of the files with beam parameter time dependencies contains the current time in [sec], second column – values of corresponding variable. The columns are divided by **tabulator** symbol. Strings are finished by the **end of line** symbol.

The files *.sur contain functions of two variables. Example of *.sur file:

```

      0      0.000333333 0.000666666 0.001
0      -0      -0      -0      -0
0.000333333 2.98404e-12 2.98364e-12 2.98243e-12 2.9804e-12
0.000666666 2.16460e-12 2.16452e-12 2.16426e-12 2.1638e-12
0.001 1.01881e-12 1.01879e-12 1.01873e-12 1.01864e-12

```

First element in the first string of the file is empty. Other elements in the first string contain the value of first variable. The elements of the first column contain the values of second variable. Corresponding value of the function are placed at the cross of the string and column. Numbers in the string are divided by **tabulator** symbol. Strings are finished by the **end of line** symbol.

2. Description of the source code

This chapter describes the structure of BETACOOOL initial codes, that includes *.cpp and *.h files, project files for C++ Builder 4 and for Microsoft Visual C++ 6. The code structure was developed in the way, which simplifies understanding the structure of the objects and provides the tools for fast modification and development of the base physical models of the program. Text of the code includes comments to all general variables of the program inside the header files and comments to calculation algorithm steps inside the *.cpp files. Dimensions of all dimensional variables used in the program are declared in the text of the code. General input and output variables of the program are described in the chapter 1 and their dimensions and physical sense can be verified using BOLIDE interface of the program. By this way we tried to make the source code and the software as a whole in the form of a self-documented system. Any case overview of the code structure is necessary for understanding a way of connection between its functional parts. From the other hand, the code fragments presented here are aimed to be initial manual for user how to develop the code in parallel with development of the program exterior described in the chapter 1.

2.1. Source files

The source code consists of

- **Betacool.cpp** file, which contains **main** procedure of the program,

- and following units:

doubleu.cpp
datau.cpp
bolideu.cpp
xbeam.cpp
xdistributor.cpp
xdraw.cpp
xdynamic.cpp
xebeam.cpp
xecool.cpp
xeffect.cpp
xforce.cpp
xibs.cpp
xlibrary.cpp
xoptics.cpp
xpowell.cpp
xrestgas.cpp
xring.cpp
xrunge.cpp
xstoch.cpp
xtarget.cpp

with corresponding header files;

file **xbeam.inc**

- two header files:

BTempinc.h

BTemplat.h

with the BOLIDE Self Pointer Templates;

- header file

StdAfh.h

generated by Microsoft Visual C++.

To compile the program by Borland C++ Builder 4 the project file **Betacool.bpr** is used.

Betacool.dsk, generated by Borland C++ Builder 4, with saved desktop of the interface

To compile the program by Microsoft Visual C++ 6 the working space file **Betacool.dsw** and project file **Betacool.dsp** are used.

2.2. Structure of the source code

The structure of the source code can be shared between three functional parts and its structure is illustrated on the Fig. 55.

The source code consists of:

- interface part, which supports the format of input and output files common with the BOLIDE system,
- library of base numerical algorithms including description of dimensional variables, templates of the program self counters, procedures for matrix algebra, algorithms of numerical solution of differential equations,
- physical codes described objects of the program and procedures with them.

The interface part of the code and library of algorithms were developed and tested independently on physical part of the program. They are aimed to give user the tools for development of the physical part of the code. Correspondingly, modification of the physical code of the program, if it is necessary to improve models of investigated processes, can be done without any changes of the interface part. Therefore here we give only brief description of the interface and algorithm parts of the code and describe the physical code in more details.

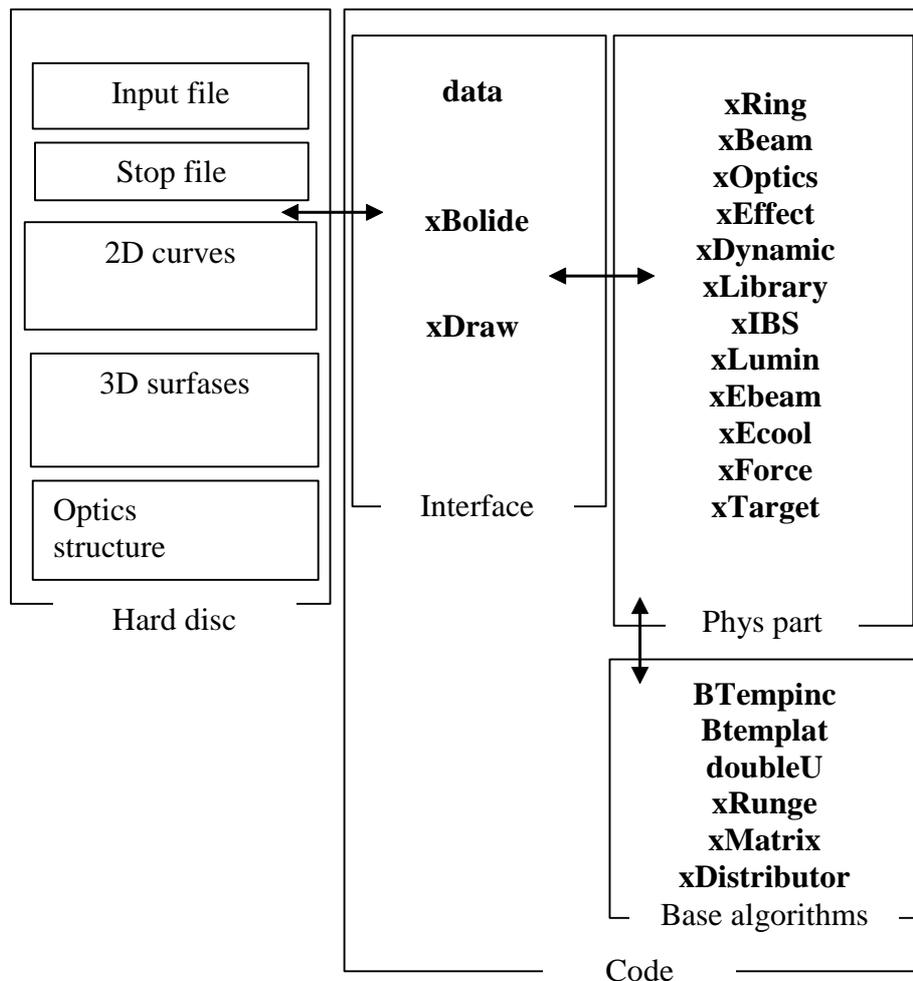


Fig. 55. Schematics of the code structure.

2.3. Interface part of the source code

Interface part of the code provides a work with file structure on a hard disk and it is described in the files **datau.cpp**, **datau.h**, **bolideu.cpp**, **bolideu.h**, **xdraw.cpp**, **xdraw.h**.

Class **xData** described in the files **bolideu.cpp**, **bolideu.h** is the parent class for all general objects of the code. This class includes two-dimensional array of the **BData** class, which provides loading and saving to disk the input file of the program. This array has variable column number in the rows and variable row number and is described in the files **data.cpp**, **data.h**. Structure of this array corresponds to the structure of input file. In the program global variable **xData Data** is declared and all objects of the program get the values of their variables from the elements of the global array. Output variables are put into the corresponding elements of the global array. The array **Data** is generated dynamically during reading of an input file.

All the names of output curves and surfaces of the program are collected in the object **xDraw**, where they are associated with the corresponding names of output files. Interface part described in the files **bolideu.cpp**, **bolideu.h** saves the calculated curves and surfaces into corresponding files. The data writing to the disk is repeating periodically in accordance with corresponding parameters in the input file.

2.4. Basic algorithms

Basic algorithms used by physical part of the code includes:

- algorithm of work with Self pointers in the program (files **BTempinc.h**, **Btemplat.h**),
- description of the class of dimensional variables (files **doubleu.cpp**, **doubleu.h**),
- procedures for numerical solution of systems of differential equations (files **xrunge.cpp**, **xrunge.h**),
- templates and classes of dimensional vector and matrix releasing the matrix algebra (**matrixu.cpp**, **matrixu.h**),
- generator of the array of 6D dimensional vectors in accordance with given distribution function (**xdistributor.cpp**, **xdistributor.h**).

The Self Pointers algorithm is the base of calculation of the sum of the rates for active effects of the task. Brief description of the algorithm is given in the chapter 2.5.3.

The hierarchy of classes for the description of dimensional variables described in the files **doubleu.h**, **doubleu.cpp**. The dimensional calculations in the program are based on (**second**, **Quloumn**, **meter**) system of units. Dimensional variable in the code has five parameters: **value**, **powers of base units and coefficient** for recalculation the value to the units of the variable. All the standard operations with the dimensional variable and elementary functions are overloaded. Appropriation and comparison operators include the checking of dimension equality (**CheckUnits** and **CheckZero** functions), which prompt the user in case of errors. To increase the calculation speed of the program after completion of the debug process user can switch off the checking of dimensions. For this case one has to comment first line in **doubleU.h** file:

```
#define POWERS // Do comment this line to speed up calculation and recompile the program.
```

The files **xrunge.cpp**, **xrunge.h**, **matrixu.cpp**, **matrixu.h** describe the standard mathematical algorithms for dimensional variables. Description of mathematical methods used in the object **xDistributor** is given in the part of report dedicated to the basic physical models of the software.