
HEBL documentation

Ivan Morozov*
Fermilab, Batavia, 60510, IL

Abstract

This technical note is a set of documents for Hollow Electron Beam Lens (HEBL) numerical model. Analytical models are discussed in Sec. 1. In Sec. 2 rotations and translations are treated. Routine description is given in Sec. 3 including source code (see also `track_ext.f90`). In Sec. 4 scripts that are used for HEBL models and data processing are described. Various simulation results (including comparison with experiment) are presented in Sec. 5. All LifeTrack files, scripts, and latex sources are available at [\[link to AFS to be set here\]](#).

*imorozov@fnal.gov

Contents

1	HEBL models	4
1.1	Integration. Drift–kick split	5
1.2	Electrons trajectories in HEBL	5
1.3	Ideal HEBL model	6
1.4	HEBL sources of imperfections	7
1.5	HEBL model with angular imperfections	8
1.6	HEBL model with radial imperfections	11
2	Euclidean Group in LifeTrack	15
2.1	Free space propagation	16
2.1.1	Hamiltonian z -parametrization	16
2.1.2	Exact Drift	17
2.1.3	Paraxial Drift	17
2.1.4	Drift element map	18
2.2	External Elements in LifeTrack	18
2.2.1	Tracking in Lifetrack	18
2.2.2	Finite length external element	18
2.3	Locally correct Euclidean Group	19
2.3.1	Translations	20
2.3.2	Rotations	22
2.4	EG in solenoid	24
2.4.1	Ideal solenoid	24
2.4.2	Kick element inside ideal solenoid	24
2.4.3	Alignment inside ideal solenoid	24
3	HEBL Routines for LifeTrack	26
3.1	Ideal HEBL lens : EXT_EL00	27
3.2	HEBL with radial imperfections : EXT_EL01	29
3.3	HEBL with angular imperfections (1) : EXT_EL02	31
3.4	HEBL with angular imperfections (2): EXT_EL03	33
3.5	Field–free propagation routine : EXT_EG00	35
3.6	Euclidean group direct routine : EXT_EG1A	36
3.7	Euclidean group inverse routine : EXT_EG1B	39
3.8	Ideal solenoid (1) routine : EXT_EGS0	42
3.9	Ideal solenoid (2) routine : EXT_EGS1	43
3.10	Element nesting tips	45
4	HEBL models scripts	46
4.1	2D particles distribution	47
4.2	Angular model script	48
4.2.1	Angular distribution	49
4.2.2	Harmonic parameters	49
4.3	Radial model script	52
4.3.1	Normalized radial distribution from 2D	52

CONTENTS

4.3.2	Normalized electric field	53
4.3.3	Transport of density and electric field	53
4.3.4	Electric field interpolation	55
4.4	Lost particle distribution script	56
4.4.1	Loss rate	56
4.4.2	Histograms	56
5	HEBL simulations	60
5.1	Harmonics	60
5.2	Aligment	60
5.3	Experiment	60

1 HEBL models

In this section HEBL models with electron beam imperfections are developed. Electron beam is assumed to be infinite in longitudinal direction. This allows to treat HEBL as a conventional transverse element. To include deviations of hollow beam transverse profile from ideal case two simple models based on drift-kick split are developed in reduced phase space (x, p_x, y, p_y) . Parameters for these models can be obtained from hollow beam measurements or simulations. Electron beam alignment is described in Sec. 2. Edge effects and other possible models are under development.

1.1 Integration. Drift–kick split

Drift-kick split is a first order symplectic integrator. Currently it is the only integration method that is available for HEBL. New positions (x, y) and new moments (p_x, p_y) on the next step of integration are obtained by the following rule,

$$\begin{aligned} p_{n+1} &= p_n + f(q_n) \\ q_{n+1} &= q_n + p_{n+1} \Delta s \end{aligned}$$

where function $f(q) = f(x, y; s)$ is related to the derivative of the transverse field potential, Δs is the length of a drift subsection. HEBL element is treated as a conventional transverse element, i.e. it is assumed to have only transverse field. Then transverse Lorentz force acting on a test particle is,

$$\begin{aligned} F_x &= qE_x - q\beta_b c B_y \\ F_y &= qE_y + q\beta_b c B_x \end{aligned}$$

Next we express F_x and F_y in terms of electron beam rest frame electric field,

$$\begin{aligned} F_x &= q\gamma_e(1 \pm \beta_e\beta_b)E_x \\ F_y &= q\gamma_e(1 \pm \beta_e\beta_b)E_y \end{aligned}$$

where γ_e , β_e are electron beam relativistic parameters and β_b is circulating beam relative speed. If $\vec{v}_e\vec{v}_b < 0$ then one should take "–" sign and the force is maximum. Transverse kicks in Cartesian frame are the following,

$$\Delta p_x = \Delta s \frac{q}{\beta_b c p_b} \gamma_e (1 + \beta_e \beta_b) E_x \quad (1)$$

$$\Delta p_y = \Delta s \frac{q}{\beta_b c p_b} \gamma_e (1 + \beta_e \beta_b) E_y \quad (2)$$

1.2 Electrons trajectories in HEBL

If one wants to use parameters from beam measurements then information about electron beam (such as transverse profile) is not generally available in the interaction region, i.e. inside the main solenoid. But one may know beam parameters in several places outside the main solenoid. Then to recover particles positions inside interaction region we assume that electrons obey the following equation of motion,

$$\frac{r_m}{r_g} = \sqrt{\frac{B_g}{B_m}} \quad (3)$$

where r_m is electron position in the main solenoid, r_g is electron position at the gun (or any other position with known magnetic field), B_m and B_g are magnetic fields inside the main solenoid and the gun respectively. By doing so we neglect torsion in longitudinal magnetic field and collective effects.

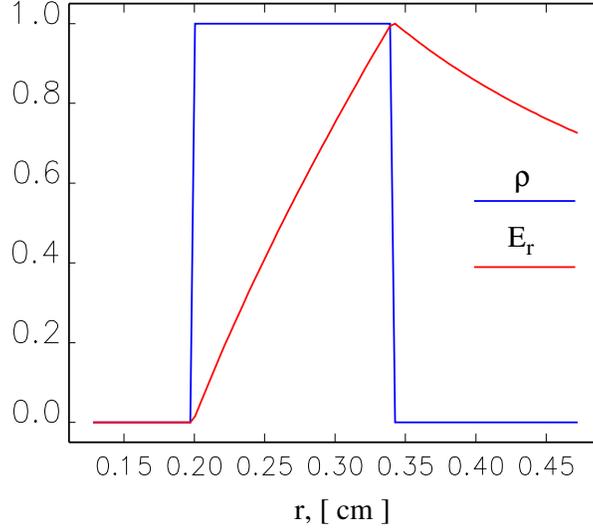


Figure 1: Ideal HEBL profile and electric field (normalized, matched to 4σ with $\sigma = 0.05[cm]$).

1.3 Ideal HEBL model

On Fig. 1 transverse density distribution and radial electric field for ideal HEBL are shown. Inner r_1 and outer r_2 electron beam radius are obtained with the help of eq. (3) since their values at the gun position are known. Then using the results of the first subsection transverse kicks that are found to be ($\vec{v}_e \vec{v}_b < 0$),

$$\Delta p_r = \begin{cases} 0 & \text{if } r < r_1 \\ 2\Omega_e \frac{r^2 - r_1^2}{r(r_2^2 - r_1^2)} \Delta s & \text{if } r_1 < r < r_2 \\ 2\Omega_e \frac{1}{r} \Delta s & \text{if } r > r_2 \end{cases} \quad (4)$$

$$\Omega_e = 0.3 \times 10^{-7} \frac{I_e(A)}{p_b(GeV/c)} \gamma_e \frac{1 + \beta_e \beta_b}{\beta_e \beta_b}$$

where,

- I_e – electron beam current
- p_b – circulating particle designed momentum
- γ_e and β_e – electron beam relativistic parameters
- β_b – circulating particle relative velocity

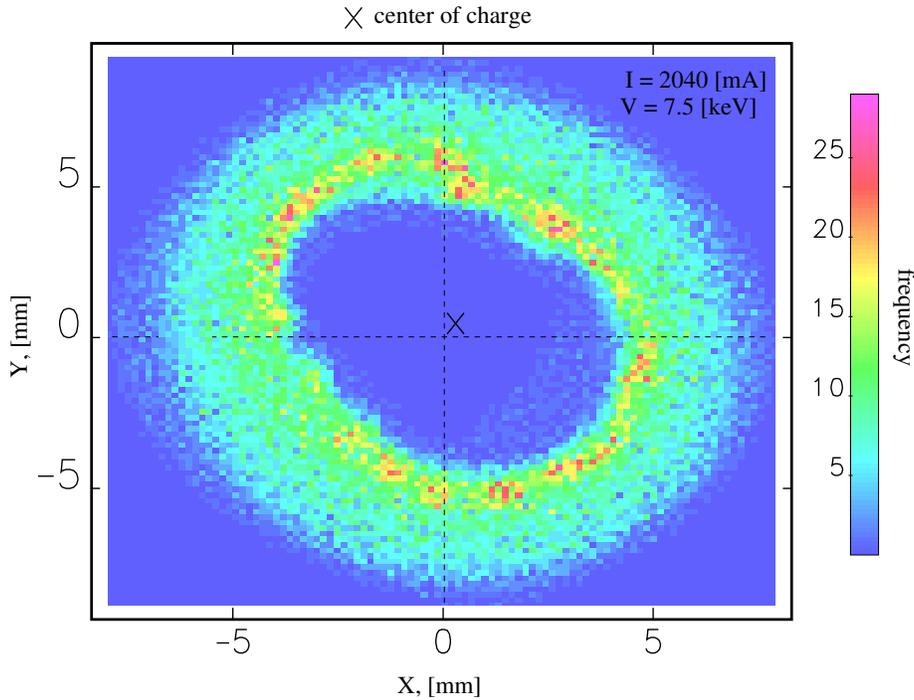


Figure 2: An example of measured transverse HEBL density profile after the main solenoid ($B = 0.3$ [T]).

1.4 HEBL sources of imperfections

Several sources of imperfections for HEBL are listed below:

- lack of the axial symmetry (see Fig. 2)
- non ideal radial profile (see Fig. 2)
- electron beam alignment (see Sec. 2)
- effects introduced by bends in electron beam

Models for first two cases are developed in this note based on conventional drift-kick split. Electron beam model is described in Sec. (1.1) and (1.2). Angular model deals with axial imperfections and allows to estimate each harmonic contribution. This model is of a special interest when studying effects on the circulation beam core. Radial model helps to estimate how disturbed profile effects particles removal rate. These two models can be combined if necessary, i.e. one can put two kick elements successively (if two models are valid simultaneously) and that gives a kick element with zero length again.

1.5 HEBL model with angular imperfections

In this case we want to investigate how angular imperfections influence the circulating beam (core). Electron lens is assumed to be infinitely long and thin with charge distribution in the beam rest frame,

$$\rho(r, \theta) = \frac{f(\theta)}{2\pi r_e} \delta(r - r_e)$$

where r_e is cylinder (beam) radius, $f(\theta)$ is some function that defines angular distribution shape. Then we express charge distribution in the following form,

$$\rho(r, \theta) = \frac{I_e}{\beta_e c} \frac{\delta(r - r_e)}{2\pi r_e} \sum_{m=0}^{\infty} \xi_m \cos(m\theta + \delta_m)$$

where ξ_m is a relative amplitude of m -th harmonic (relative to zero harmonic's amplitude, i.e. to the beam line charge density), δ_m is harmonic's phase, I_e is electron beam current, $\beta_e c$ is electron velocity. These parameters are to be determined from transverse beam profile measurements (Fig. 2). For this distribution electric field can be obtained for each harmonic,

$$E_r(r, \theta) = \frac{I_e \xi_m}{4\pi \epsilon_0 \beta_e} \cos(m\theta + \delta_m) \begin{cases} -\frac{r^{m-1}}{r_e^m}, & r < r_e \\ r^{m-1} r_e^m, & r > r_e \end{cases} \quad (5a)$$

$$E_\theta(r, \theta) = \frac{I_e \xi_m}{4\pi \epsilon_0 \beta_e} \sin(m\theta + \delta_m) \begin{cases} \frac{r^{m-1}}{r_e^m}, & r < r_e \\ r^{m-1} r_e^m, & r > r_e \end{cases} \quad (5b)$$

On Fig. 3 electric field is shown for $m = 1, 2, 3$. To determine harmonic parameters we put origin into center-of-charge and divide space into angular slices. Then we count the number of particles in each slice, normalize this value and perform FFT to determine amplitudes and phases (Fig. 4). Transverse kicks for this model are,

$$\Delta p_x = -\Omega_e \Delta s \xi_m \frac{r^{m-1}}{r_e^m} \cos((m-1)\theta + \delta_m) \quad (6)$$

$$\Delta p_y = \Omega_e \Delta s \xi_m \frac{r^{m-1}}{r_e^m} \sin((m-1)\theta + \delta_m) \quad (7)$$

where,

- Ω_e is given by eq. (4)
- Δs – kick-drift subsection length [cm]
- ξ_m – harmonic's relative amplitude
- δ_m – harmonics phase
- r_e – electron beam radius [cm]

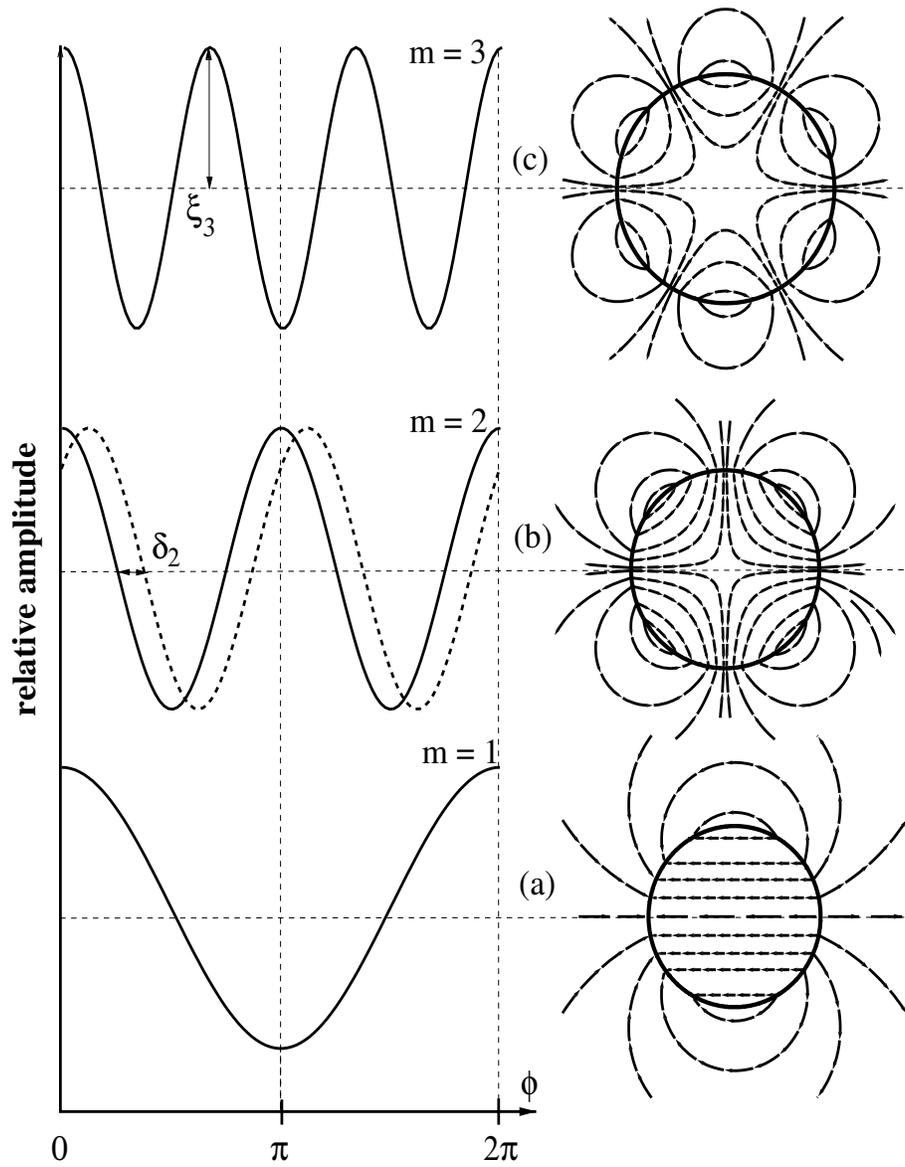


Figure 3: Angular distribution function and corresponding electric field. (a) $m = 1$ – dipole, (b) $m = 2$ – quadrupole, (c) $m = 3$ – sextupole.

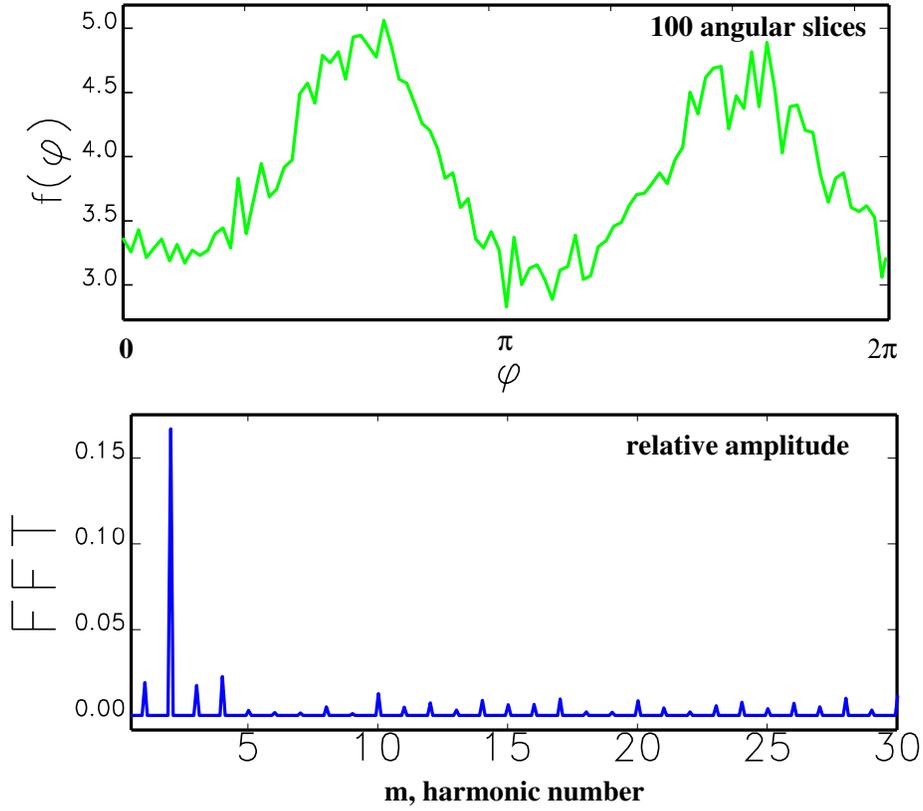


Figure 4: An example of angular distribution function and it's FFT for measured electron beam transverse profile. In ideal case angular distribution is constant. For this case only quadrupole harmonic $m = 2$ has large amplitude $\xi_2 = 17\%$, other harmonics' relative amplitudes are small. This is a typical picture for measured electron beam profiles ($\xi_2 = 15\% \dots 25\%$).

See more detains about data processing for angular model in Sec. 4.2.

1.6 HEBL model with radial imperfections

This model handles hollow electron beam radial imperfection, i. e. deviation of radial charge density from the ideal case. Having a 2D particles distribution (Fig. 2) it is possible to construct axisymmetric radial density distribution for non ideal beam transverse profile. It is done similar to the angular case but instead of dividing distribution into angular slices we use radial ones. This allows to obtain normalized radial density $g(r)$ which is connected to usual density as $\rho(r) = \rho_{max}g(r)$ (Fig. 5). Normalized electric field can be found from $g(r)$ by

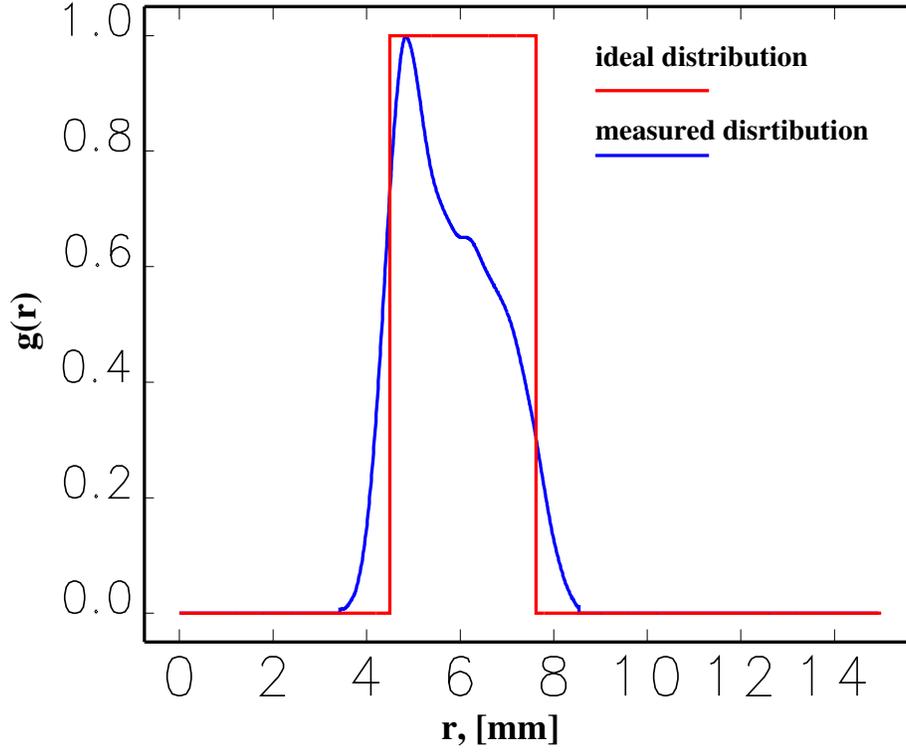


Figure 5: Normalized radial density.

numerical integration of $\frac{1}{r} \int_0^r g(r) r dr$ and further normalization (Fig. 6). Then positions are recalculated according to eq. (3) to obtain normalized density and electric field inside the main solenoid (Fig. 7). Transverse kicks are given by eq. (1) and (2) with $E_x = E_r \cos(\theta)$ and $E_y = E_r \sin(\theta)$. Radial electric field can be expressed as,

$$E_r(r) = E_{max} f(r)$$

The radius corresponding to the maximum field value is found from $f(r)$,

$$r_{max} : f(r_{max}) = 1$$

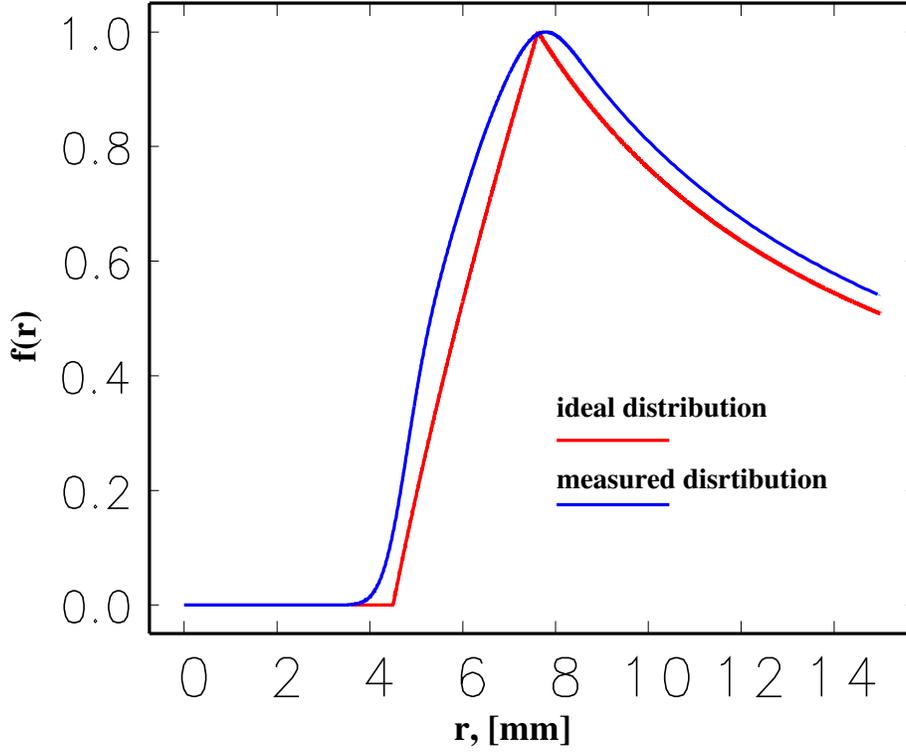


Figure 6: Normalized radial electric field

The maximal electric field is found to be,

$$E_{max} = \frac{1}{2\pi\epsilon_0} \frac{I_e}{\beta_e c} \frac{\kappa}{\eta} \frac{1}{r_{max}}$$

where,

- I_e is electron beam current
- $\beta_e c$ is electron beam velocity
- r_{max} radius value where $f(r) = 1$
- $\kappa = \int_0^{r_{max}} g(r) r dr$
- $\eta = \int_0^{r_{cut}} g(r) r dr$, r_{cut} is the edge of distribution $g(r > r_{cut}) = 0$

Note that $r_{max} \neq r_{cut}$ in general, but the ratio $\frac{\kappa}{\eta}$ is still close to 1 (> 0.95 for measured profiles). Then transverse kick is,

$$\Delta p_r(r) = \frac{2\Omega_e \Delta s}{r_{max}} \frac{\kappa}{\eta} f(r)$$

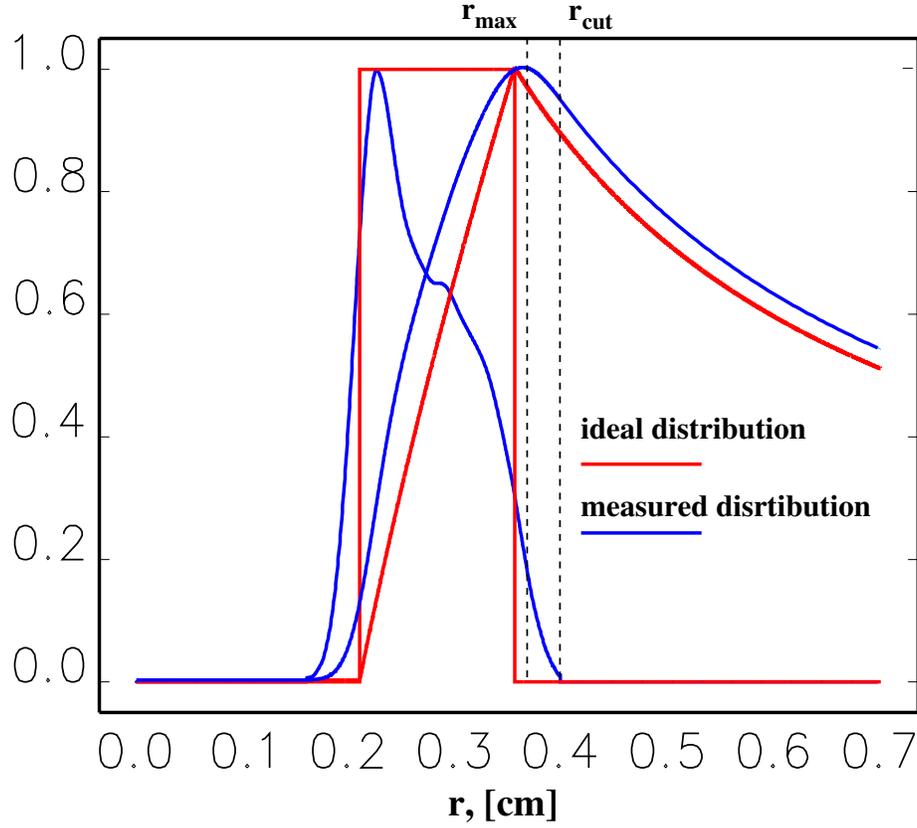


Figure 7: Normalized density and electric field inside the main solenoid.

where Ω_e is the same factor as in ideal or angular case. To obtain an analytical formula for transverse kick one need to interpolate $f(r)$ in the region of interest. But it is practically difficult to make this interpolation with one polynomial. From one hand it is known that $f(r) = \frac{r_{max}}{r}$ for $r > r_{cut}$. Then interpolation is required only for $0 < r < r_{cut}$. This region is divided into three pieces,

$$\begin{cases} f(r) = 0 & 0 < r < r_{initial} \\ f(r) = f_{p1} & r_{initial} < r < r_{middle} \\ f(r) = f_{p2} & r_{middle} < r < r_{cut} \end{cases}$$

See Fig. 8 where interpolation is shown. See Sec. 4.3 for more details about data processing for radial model. Routine is described in Sec. 3.2.

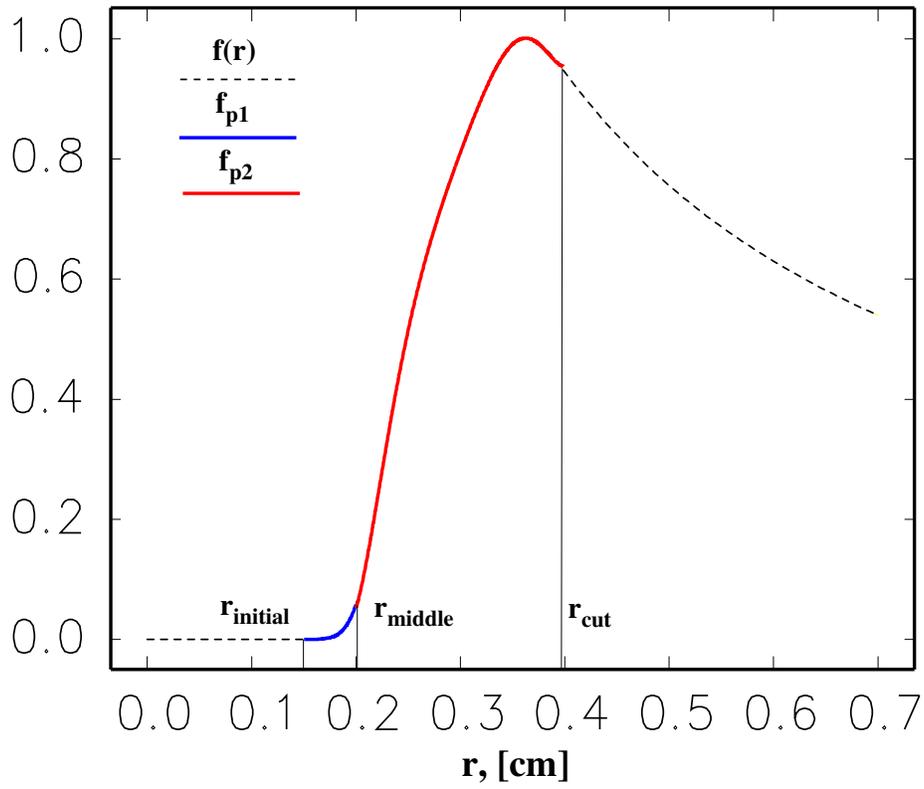


Figure 8: Interpolation of normalized electric field.

2 Euclidean Group in LifeTrack

In this section note patching of Euclidean Group into LifeTrack is discussed. Procedure for finite length external elements is developed first. Then locally correct Euclidean Group is introduced. The effect of ideal solenoid is developed for drift-kick element including beam alignment. Similar approach can be used to treat quadrupole magnets in IP region and other. Source code with comments and input description is given in 3 or track_ext.f90. Treatment is developed for reduced transverse phase space (x, p_x, y, p_y) , time-like variable is not changed within external element, but energy deviations are handled correctly.

2.1 Free space propagation

2.1.1 Hamiltonian z -parametrization

Relativistic Hamiltonian function for field-free propagation is known to be,

$$H(x, p_x, y, p_y, z, p_z; t) = c\sqrt{m^2c^2 + \vec{p}^2} \quad (8)$$

Here Hamiltonian is parametrized by time t in (x, p_x, y, p_y, z, p_z) phase space. However it is more convenient to use z -parametrization for local integration. It is possible if $z(t)$ is a monotonic function of t as it is the case in most accelerator elements. Then new phase space (x, p_x, y, p_y, t, p_t) can be introduced and new z -parametrized Hamiltonian,

$$K(x, p_x, y, p_y, t, p_t; z) = -\sqrt{\left(\frac{p_t}{c}\right)^2 - m^2c^2 - p_x^2 - p_y^2} \quad (9)$$

where $K = -p_z$ and $p_t = -H$. Now we introduce a reference particle, i.e. a particle with initial conditions $(0, 0, 0, 0, 0, -H_0; 0)$ and $H_0 = c\sqrt{m^2c^2 + p_0^2}$. Reference particle trajectory is simply a straight line. Particle with deviations has,

$$K(x, p_x, y, p_y, t, p_t; z) = -\sqrt{\left(\frac{H_0(1 + \delta)}{c}\right)^2 - m^2c^2 - p_x^2 - p_y^2} \quad (10)$$

where δ is energy deviation $H = H_0(1 + \delta)$. We scale (10) with respect to the reference momentum p_0 while keeping the same notation for K , p_x and p_y ,

$$K(x, p_x, y, p_y, t, p_t; z) = -\sqrt{\frac{H_0^2(1 + \delta)^2 - m^2c^4}{c^2p_0^2} - p_x^2 - p_y^2} \quad (11)$$

In relativistic case $H_0^2(1 + \delta)^2 \gg m^2c^4$ Hamiltonian (11) is reduced to,

$$K(x, p_x, y, p_y, t, p_t; z) = -\sqrt{(1 + \delta)^2 - p_x^2 - p_y^2} \quad (12)$$

Hamiltonian (12) is exactly solvable. It describes a field-free propagation of a particle with energy (momentum) deviation δ and $p_t = c(1 + \delta)$.

2.1.2 Exact Drift

Equations of motion for the field-free region with Hamiltonian (12) are,

$$\begin{aligned}
 \frac{dx}{dz} &= \frac{\partial K}{\partial p_x} = \frac{p_x}{p_z} \\
 \frac{dp_x}{dz} &= -\frac{\partial K}{\partial x} = 0 \\
 \frac{dy}{dz} &= \frac{\partial K}{\partial p_y} = \frac{p_y}{p_z} \\
 \frac{dp_y}{dz} &= -\frac{\partial K}{\partial y} = 0 \\
 \frac{dt}{dz} &= \frac{\partial K}{\partial p_t} = \frac{p_t}{p_z} \\
 \frac{dp_t}{dz} &= -\frac{\partial K}{\partial t} = 0
 \end{aligned} \tag{13}$$

These equations change only coordinates of a particle (x, y, t) . The action of this field-free map on initial phase point $(x_0, p_{x0}, y_0, p_{y0}, t_0, p_{t0})$ is,

$$\begin{aligned}
 x &= x_0 + z \frac{p_{x0}}{p_{z0}} \\
 p_x &= p_{x0} \\
 y &= y_0 + z \frac{p_{y0}}{p_{z0}} \\
 p_y &= p_{y0} \\
 t &= t_0 + cz \frac{1 + \delta}{p_{z0}} \\
 \delta &= \delta_0
 \end{aligned} \tag{14}$$

where,

$$p_{z0} = \sqrt{(1 + \delta_0)^2 - p_{x0}^2 - p_{y0}^2}$$

Usually instead of t the difference $\Delta t = \hat{t} - t$ is used in tracking, where \hat{t} is a time of flight of reference particle, $\hat{t} = \hat{t}_0 + zc$, then,

$$\Delta t = \frac{c\delta}{p_{z0}}$$

Equations (14) are exact solution for 6D-phase space field-free problem.

2.1.3 Paraxial Drift

In paraxial case we expand the square root in (12) and paraxial 6D Hamiltonian,

$$K(x, p_x, y, p_y, t, p_t; z) = -(1 + \delta) + \frac{1}{2(1 + \delta)} (p_x^2 + p_y^2) \tag{15}$$

New transverse positions are,

$$\begin{aligned}
 x &= x_0 + z \frac{p_{x0}}{1 + \delta} \\
 y &= y_0 + z \frac{p_{y0}}{1 + \delta}
 \end{aligned} \tag{16}$$

2.1.4 Drift element map

For now we care only about reduced phase space (x, p_x, y, p_y) . Exact solution for the field-free region is,

$$\begin{aligned} x &= x_0 + z \frac{p_{x0}}{p_{z0}} \\ p_x &= p_{x0} \\ y &= y_0 + z \frac{p_{y0}}{p_{z0}} \\ p_y &= p_{y0} \end{aligned} \tag{17}$$

where,

$$p_{z0} = \sqrt{(1 + \delta_0)^2 - p_{x0}^2 - p_{y0}^2}$$

Paraxial solution is given by,

$$\begin{aligned} x &= x_0 + z \frac{p_{x0}}{1 + \delta} \\ p_x &= p_{x0} \\ y &= y_0 + z \frac{p_{y0}}{1 + \delta} \\ p_y &= p_{y0} \end{aligned} \tag{18}$$

2.2 External Elements in LifeTrack

2.2.1 Tracking in Lifetrack

Schematically the tracking procedure in LifeTrack is shown on Fig.9. Ray prop-

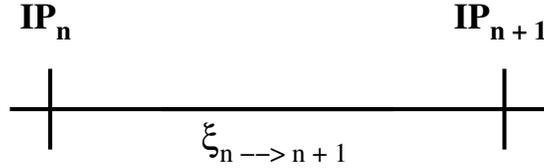


Figure 9: Tracking in LifeTrack. Ray propagates between zero lens IPs with map $\xi_{n \to n+1}$

agates with map $\xi_{n \to n+1}$ (this map is linear) from one IP to another. External elements are placed in IP-like manner and have zero length by default.

2.2.2 Finite length external element

To include a finite length external element one should use negative drifts $T_z(-L/2)$ (if $\xi_{n \to n+1}$ is a linear map) as it is shown on Fig.10. Ray initial coordinates come to the fiducial plane $z = L/2$, then propagated back in space for distance $L/2$ to obtain ray coordinates at the element fiducial entrance plane. External element propagates the ray with map $\xi_{1 \to 2}$ from $z = 0$ to $z = L$. The last step is to use a negative drift to move from $z = L$ to $z = L/2$. The negative drift is described with equations (17) or (18). Element EXT_EG00 can be used to make a finite length element (see 3.5).

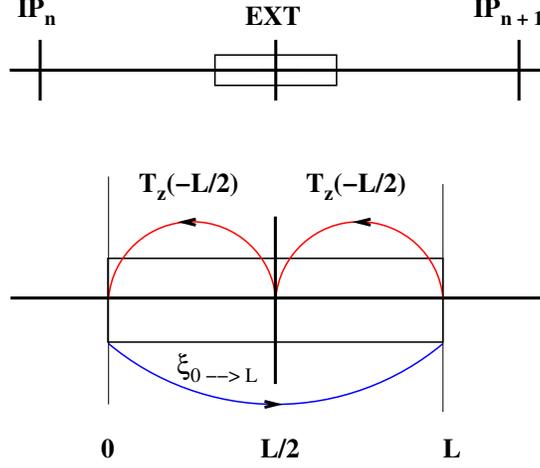


Figure 10: Finite length external element in LifeTrack. Ray propagates from $z = L/2$ to $z = 0$ with negative drift $T_z(-L/2)$. Then from $z = 0$ to $z = L$ with element map $\xi_{1 \rightarrow 2}$ and back to $z = L/2$ with $T_z(-L/2)$.

2.3 Locally correct Euclidean Group

Consider phase space $(x, p_x, y, p_y, t, p_t)(z)$ with $p_t = c(1 + \delta)$. Initial phase space point is propagated successively through each accelerator element from its entrance plane ($z = 0$) to exit plane ($z = L$ - element's length),

$$\begin{pmatrix} x^f \\ p_x^f \\ y^f \\ p_y^f \\ t^f \\ p_t^f \end{pmatrix} = M_{i \rightarrow f} \begin{pmatrix} x^i \\ p_x^i \\ y^i \\ p_y^i \\ t^i \\ p_t^i \end{pmatrix}.$$

When alignment is introduced coordinate transformations should be performed before and after applying element's map,

$$\begin{pmatrix} x^f \\ p_x^f \\ y^f \\ p_y^f \\ t^f \\ p_t^f \end{pmatrix} = P_{\vec{d}, \vec{\theta}}^{-1} M_{i \rightarrow f} P_{\vec{d}, \vec{\theta}} \begin{pmatrix} x^i \\ p_x^i \\ y^i \\ p_y^i \\ t^i \\ p_t^i \end{pmatrix}$$

Compositional maps $P_{\vec{d}, \vec{\theta}}$ and $P_{\vec{d}, \vec{\theta}}^{-1}$ can be decomposed into translations and rotations, for e.g.,

$$P_{\vec{d}, \vec{\theta}} = R_{\vec{\theta}} T_{\vec{d}} = R_x(\theta_x) R_y(\theta_y) R_z(\theta_z) T_z(d_z) T_y(d_y) T_x(d_x)$$

In this subsection we derive maps for translations $T_x(x)$, $T_y(y)$, $T_z(z)$ and rotations $R_x(\theta_x)$, $R_y(\theta_y)$, $R_z(\theta_z)$.

- T_x – x–translation
- T_y – y–translation
- T_z – z–translation
- R_x – rotation along x
- R_y – rotation along y
- R_z – rotation along z

T_z , R_x and R_y are "time coupled" transformations since they involve free-space propagation. All transformations are locally correct while $z(t)$ – monotonic holds. Transformations are time reversible, e.g. $T_z^{-1}(\theta_y) = T_z(-\theta_y)$.

2.3.1 Translations

First we deal with uncoupled translations T_x and T_y Fig. 11. These translations correspond to the case when element's plane (in blue) is shifted with respect to fiducial plane by amount dx or dy respectively. The coordinate transformation is simply,

$$x_2 = x_1 - dx \tag{19}$$

$$y_2 = y_1 - dy \tag{20}$$

where x_2 and y_2 correspond to particle coordinates in element's frame, x_1 and y_1 – fiducial frame. Formally transformations T_x and T_y applied successively even these translations commute, i.e $T_x T_y = T_y T_x$. T_z transformation (Fig. 11) is a drift in free-space (see subsection 1.) and coordinate transformations are given by equations (17) or (18). For exact drift,

$$\begin{aligned} x_2 &= x_1 + dz \frac{p_{x1}}{p_{z1}} \\ y_2 &= y_1 + dz \frac{p_{y1}}{p_{z1}} \end{aligned} \tag{21}$$

where,

$$p_{z1} = \sqrt{(1 + \delta)^2 - p_{x1}^2 - p_{y1}^2}$$

Inverse transformations are obtained by changing $(dx, dy, dz) \rightarrow (-dx, -dy, -dz)$.

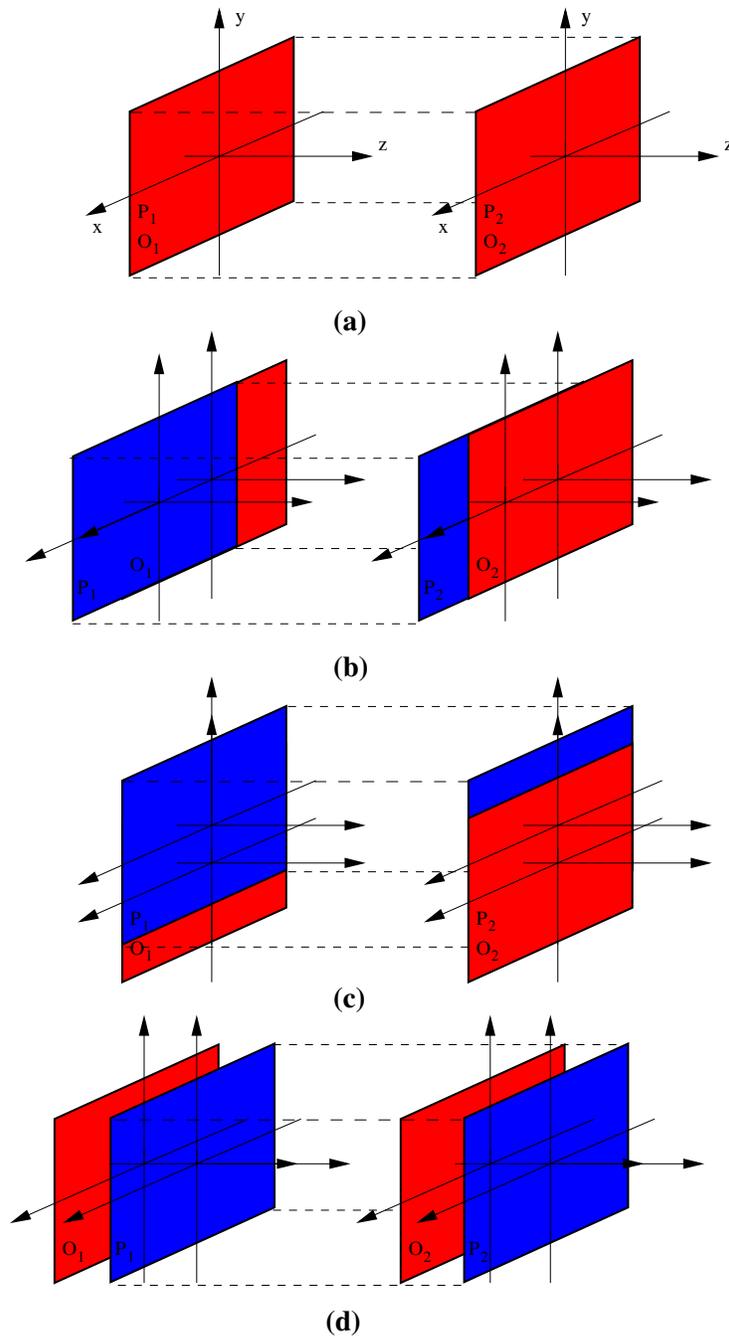


Figure 11: Translations. (a) – ideally positioned element, O_1 , O_2 – fiducial planes, P_1 , P_2 – actual element entrance and exit planes; (b) – translation along x -axes; (c) – translation along y -axes; (d) – translation along z -axes.

2.3.2 Rotations

We start with uncoupled R_z rotation. It's a rotation along z as it shown on Fig. 12 and is given by a usual rotation matrix,

$$\begin{pmatrix} x_2 \\ p_{x2} \\ y_2 \\ p_{y2} \end{pmatrix} = \begin{pmatrix} \cos(\theta_z) & 0 & -\sin(\theta_z) & 0 \\ 0 & \cos(\theta_z) & 0 & -\sin(\theta_z) \\ \sin(\theta_z) & 0 & \cos(\theta_z) & 0 \\ 0 & \sin(\theta_z) & 0 & \cos(\theta_z) \end{pmatrix} \begin{pmatrix} x_1 \\ p_{x1} \\ y_1 \\ p_{y1} \end{pmatrix} \quad (22)$$

This matrix describes a point rotation, i.e. integration parameter is not changed during this transformation. Inverse transformation is just $\theta_z \rightarrow -\theta_z$. Rotations along x and y axes (Fig. 12) are not so simple as eq. (22). We perform these

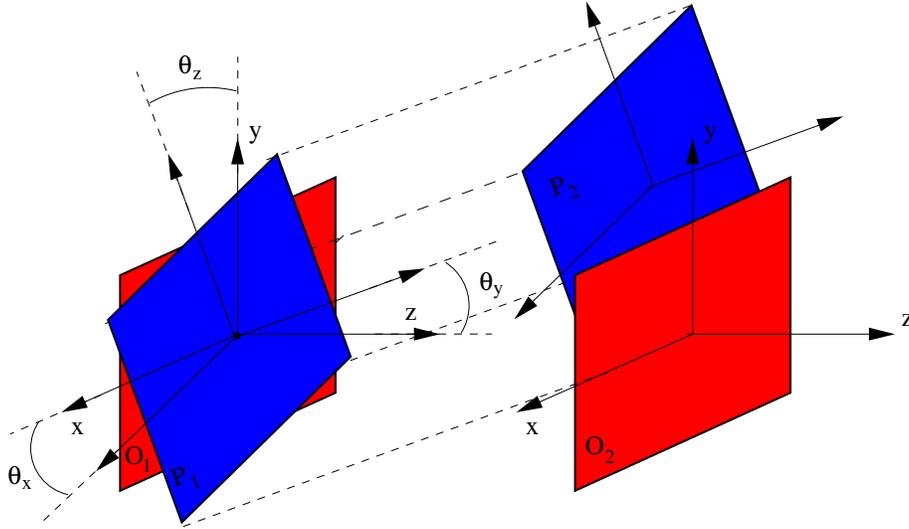


Figure 12: Rotations. O_1, O_2 – fiducial planes, P_1, P_2 – actual element entrance and exit planes; θ_x – rotation along x -axes; θ_y – rotation along y -axes; θ_z – rotation along z -axes.

transformations it two steps,

- point rotation (similar to eq. (22))
- z -propagation

The first step is equivalent to R_z case. Take for instance R_x rotation, then the first step transformation gives,

$$\begin{pmatrix} y_2 \\ p_{y2} \\ z_2 \\ p_{z2} \end{pmatrix} = \begin{pmatrix} \cos(\theta_x) & 0 & \sin(\theta_x) & 0 \\ 0 & \cos(\theta_x) & 0 & \sin(\theta_x) \\ -\sin(\theta_x) & 0 & \cos(\theta_x) & 0 \\ 0 & -\sin(\theta_x) & 0 & \cos(\theta_x) \end{pmatrix} \begin{pmatrix} y_1 \\ p_{y1} \\ z_1 \\ p_{z1} \end{pmatrix} \quad (23)$$

2.3 Locally correct Euclidean Group

where,

$$p_{z1} = \sqrt{(1 + \delta)^2 - p_{x1}^2 - p_{y1}^2}$$

In fact z_1 is arbitrary and can be set to 0. The next step is to propagate from $z = z_2$ to $z = 0$, where $z = 0$ is element entrance frame.

$$\begin{aligned} x_3 &= x_2 - z_2 \frac{p_{x2}}{p_{z2}} \\ y_3 &= y_2 - z_2 \frac{p_{y2}}{p_{z2}} \end{aligned} \quad (24)$$

To obtain inverse transformation one should add two more steps because of finite length of the element.

- $\theta_x \rightarrow -\theta_x$
- $z_2 \rightarrow -z_2$
- translation in y direction
- z - propagation

Third step is,

$$y_4 = y_3 - L \sin(\theta_x) \quad (25)$$

where L is element's length. And the fourth step,

$$x_5 = x_4 + L(1 - \cos(\theta_x)) \frac{p_{x4}}{p_{z4}} \quad (26)$$

$$y_5 = y_4 + L(1 - \cos(\theta_x)) \frac{p_{y4}}{p_{z4}} \quad (27)$$

The last two steps are obvious from geometrical considerations Fig. 13. Note that for point-like element (i.e. kick element) additional steps are redundant since $L = 0$. Rotation along y is similar. For more details see 3.7 or

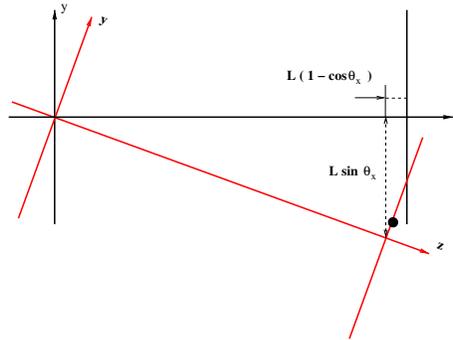


Figure 13: Additional steps in inverse transformation for coupled rotations.

track_ext.f90.

2.4 EG in solenoid

2.4.1 Ideal solenoid

z -parametrized Hamiltonian for ideal solenoid is,

$$H = -\sqrt{(1 + \delta)^2 - (p_x + \frac{b_z}{2}y)^2 - (p_y - \frac{b_z}{2}x)^2} \quad (28)$$

where $b_z = \frac{B_z}{|B\rho|}$. Without proof the solution for ideal solenoid is given by,

$$\begin{pmatrix} x \\ p_x \\ y \\ p_y \end{pmatrix} = \begin{pmatrix} \frac{1+c}{2} & \frac{s}{b_z} & \frac{s}{2} & \frac{1-c}{b_z} \\ \frac{sb_z}{4} & \frac{1+c}{2} & \frac{b_z(c-1)}{2} & \frac{s}{2} \\ -\frac{s}{2} & \frac{c-1}{b_z} & \frac{1+c}{2} & \frac{s}{b_z} \\ \frac{b_z(1-c)}{4} & -\frac{s}{2} & \frac{sb_z}{4} & \frac{1+c}{2} \end{pmatrix} \begin{pmatrix} x_0 \\ p_{x0} \\ y_0 \\ p_{y0} \end{pmatrix} \quad (29)$$

$$t = t_0 + cz \frac{1 + \delta}{H} \quad (30)$$

with,

$$c(z) = \cos\left(-\frac{zb_z}{H}\right)$$

$$s(z) = \sin\left(-\frac{zb_z}{H}\right)$$

See ?? or track_ext.f90 for routine description.

2.4.2 Kick element inside ideal solenoid

One can convince himself that the map for ideal solenoid (29) doesn't produce change of phase-space variables if $z = 0$, i.e. zero-length solenoid. Then similar to a drift-kick split a "solenoid-kick" split can be performed.

2.4.3 Alignment inside ideal solenoid

We have derived local Euclidean group for elements inside a free-space. It is valid if time-like variable is monotonic. In the case of an element inside ideal solenoid propagation in a rotated time-coupled frame is not trivial, since transverse fields are modified. Transverse x and y translations and z rotation are remain unaffected. Here we treat the case of a kick-element. Propagation along z axes is given by ideal solenoid map (29). z translation is a solenoid with length dz . To perform transformation for coupled rotations we first propagate along z axes by Δ (Fig. 12). This step is performed by propagation through a solenoid which length depends on particle transverse positions at the fiducial entrance plane (see 3.9 where routine is described). Then point rotations are made for x and y planes. Inverse transformations start with inverse point rotations for coupled planes. After these rotations inverse solenoid (solenoid is time reversible) is applied. The rest inverse transformations are the same as in the regular free-space case except inverse z translation that is an inverse solenoid.

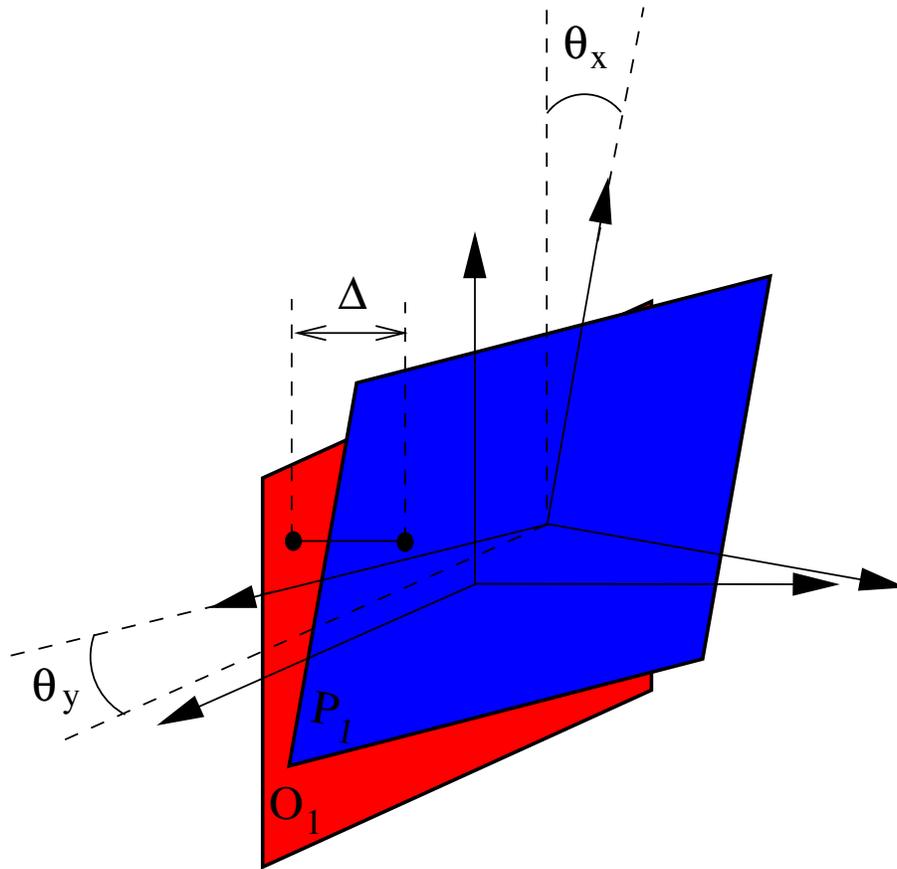


Figure 14: Propagation step. $\Delta = dz + x \tan \theta_y + y \tan \theta_x$

3 HEBL Routines for LifeTrack

This section is a list of LifeTrack routines for HEBL. Brief description of input parameters, element usage and nesting is presented. All HEBL models are "one-kick" elements, this is done in order to simplify code itself and also this leads to a greater element nesting capabilities, i.e. several "one-kick" elements can be placed at the same position. One can make several kicks by nesting "one-kick" elements with drift element. Elements with several kicks are also available but not described here. Only reduced phase space (x, p_x, y, p_y) is treated but with correct δ dependents in drifts and solenoids. Modify PRE_TRACK_EXT and TRACK_EXT in track.ext.f90.

3.1 Ideal HEBL lens : EXT_EL00

Detailed description of HEBL ideal model is given in Sec. 1.3. Place HEBL element into the element list in LifeTrack input file. Set values for EXT_EL00, (use EXT_EL10 for several kick-elements separated by paraxial drift)

```
# <name>: EXT_EL00
Value_1: 1.0
! skip parameter,
! when set to 1 element is on each turn,
! e.g for 6 element is on every 6th turn
Value_2: 1.0
! # of slices, needed for correct kick magnitude calculations
! if more then one kick element used in a row
Value_3: 200.
! element length [cm]
Value_4: 0.9E-10
! omega, see below, this value is for I=0.5[A], beta=0.2
Value_5: 0.24
! r_in[cm], inner HEBL radius
! usually r_in = N sigma, where N is number of sigmas to match,
! sigma is the largest transverse beam size (sigma_y = 0.06 [cm] for Tev TEL2)
Value_6: 0.4064
! r_out[cm], outer HEBL radius, see below
```

Value_4 is dimensionless parameter given by,

$$\Omega = 0.3 \times 10^{-7} \frac{I_e[A]}{p_b[GeV/s]} \gamma_e \frac{1 \pm \beta_e \beta_b}{\beta_e \beta_b} \quad (31)$$

Index "e" refers to electron beam, "b" - to circulating beam. Take "+" sign if $\vec{v}_e \vec{v}_b < 0$ and "-" if $\vec{v}_e \vec{v}_b > 0$. Value_5 is calculated assuming that electrons obey,

$$\frac{r_m}{r_g} = \sqrt{\frac{B_g}{B_m}}$$

For HEBL that was installed on Tevatron beam parameters at the gun: $r_{in}^g = 0.45$ [cm] and $r_{out}^g = 0.762$ [cm]. Then for fixed Value_5,

$$r = r_{in} \frac{r^g}{r_{in}^g} = N \sigma \frac{r^g}{r_{in}^g} \quad (32)$$

Setting $r^g = r_{out}^g$ gives the value of outer radius r_{out} inside the solenoid. This formula can be used to obtain density distribution or electric field of electron beam at solenoid location if density distribution or other is given at the gun.

3.1 Ideal HEBL lens : EXT_EL00

```
SUBROUTINE PRE_TRACK_EXT (type, pref, param)
CASE('EXT_EL00')
! param(1) - skip
! param(2) - # of slices
! param(3) - length[cm]
! param(4) - omega
! param(5) - r1[cm]
! param(6) - r2[cm]
param(4)=2._8*param(4)*param(3)/param(2)
param(7)=param(5)**2
param(8)=param(6)**2-param(7)

SUBROUTINE TRACK_EXT (type, param, coord, pwght, nturn, rflag)
REAL(8) :: r,t

CASE('EXT_EL00')
IF(MOD(nturn,INT(param(1)))==0 .AND. nturn/=0) THEN
  r=SQRT(coord(1)**2+coord(3)**2)
  t=ATAN2(coord(3),coord(1))
  IF(r<param(5)) THEN
    r=0.0
  ELSEIF(r>param(6)) THEN
    r=param(4)/r
  ELSE
    r=param(4)*(r**2-param(7))/(r*param(8))
  ENDIF
  coord(2)=coord(2)+r*COS(t)
  coord(4)=coord(4)+r*SIN(t)
END IF
```

3.2 HEBL with radial imperfections : EXT_EL01

Detailed description of HEBL model with radial imperfections is given in Sec. 1.6. Place HEBL element into the element list in LifeTrack input file. Set values for EXT_EL01 (use EXT_EL11 for several kick-elements separated by paraxial drift),

```
# <name>: EXT_EL01
Value_1: 1.0
! see EXT_EL00
Value_2: 1.0
! see EXT_EL00
Value_3: 200.
! see EXT_EL00
Value_4: 0.972E-10
! omega, see below (see EXT_EL00)
Value_5: 0.15
! r_initial[cm], initial point of interpolation
Value_6: 0.22
! r_middle[cm], middle point of interpolation
Value_7: 0.4176
! r_final = r_cut[cm], final point of interpolation, see below
Value_8: 0.3579
! r_max[cm]/(kappa/eta), f(r_max)=1
! first interpolation function, up to order 4 (5 terms)
! fp1(r_initial < r < r_middle) = a0+a1 r + ... + a4 r^4
Value_9: 44.58
! a0
!...
Value_13: 49592.6
! a4
! second interpolation function, up to order 6 (7 terms)
! fp2(r_middle < r < r_final) = b0 + b1 r + ... + b6 r^6
Value_14: 7.73546
! b0
!...
Value_20: 112676
! b6
```

See Sec. 4.3 for details about data processing for EXT_EL01 element. Value_4 in this routine is Ω that is given by eq. (31) times the normalization coefficient $\frac{\kappa}{\eta}$ where,

$$\kappa = \int_0^{r_{max}} g(r)rdr \quad \text{and} \quad \eta = \int_0^{r_{cut}} g(r)rdr$$

$g(r)$ is normalized radial charge distribution, r_{max} is a radius value where electric field is maximum, r_{cut} is the edge of distribution.

```

SUBROUTINE PRE_TRACK_EXT (type, pref, param)
CASE('EXT_ELO1')
! param(1) - skip
! param(2) - # of slices
! param(3) - length[cm]
! param(4) - omega
! param(5) - r_initial[cm]
! param(6) - r_midle[cm]
! param(7) - r_final[cm], r_cut
! param(8) - r_max[cm]/(kappa/eta), f(r_max)=1
! param(9) - param(13) - first poly
! param(14) - param(20) - second poly
param(4)=2._8/param(8)*param(4)*param(3)/param(2)

SUBROUTINE TRACK_EXT (type, param, coord, pwght, nturn, rflag)
REAL(8) :: r,t
CASE('EXT_ELO1')
IF(MOD(nturn,INT(param(1)))==0 .AND. nturn/=0) THEN
  r=SQRT(coord(1)**2+coord(3)**2)
  t=ATAN2(coord(3),coord(1))
  IF(r .LE. param(5))THEN
    r=0.0
  ELSEIF((r .GT. param(5)) .AND. (r .LE. param(6)))THEN
    r=param(4)*(param(9)+param(10)*r+param(11)*r**2+&
param(12)*r**3+param(13)*r**4)
  ELSEIF((r .GT. param(6)) .AND. (r .LE. param(7)))THEN
    r=param(4)*(param(14)+param(15)*r+param(16)*r**2+&
param(17)*r**3+param(18)*r**4+param(19)*r**5+param(20)*r**6)
  ELSE
    r=param(8)*param(4)/r
  ENDIF
  coord(2)=coord(2)+r*COS(t)
  coord(4)=coord(4)+r*SIN(t)
END IF

```

3.3 HEBL with angular imperfections (1) : EXT_EL02

Detailed description of HEBL model with angular harmonics is given in Sec. 1.5. This model is build to include electron beam profile angular imperfections, i.e. $\frac{\partial \rho}{\partial \theta} \neq 0$. Electron beam is replaced with δ -cylinder which charge distribution that can be decomposed into harmonics. This routine models a single arbitrary harmonic with parameters which come from measurements of electron beam profile (inside the cylinder only). To include several harmonics see the last subsection of this note. For details about harmonic parameters calculation see Sec. 4.2. Place HEBL element into the element list in LifeTrack input file. Set values for EXT_EL02 (use EXT_EL22 for several kick-elements separated by paraxial drift),

```
# <name>: EXT_EL02
Value_1: 1.0
! see EXT_EL00
Value_2: 1.0
! see EXT_EL00
Value_3: 200.
! see EXT_EL00
Value_4: 0.9E-10
! see EXT_EL00
Value_5: 1.
! rb[cm], cylinder-beam radius, see discussion below
Value_6: 2.
! harmonic number, e.g. m=2 -- quad, m=3 -- sext.
! m=0 (1/r for r>r_b) is not included
Value_7: .2
! harmonic relative (to average) amp,
! 0.2 means that harm's amp is 20% of the average density value
! for HEBL test stand typical values m=1, amp=10...15%;
! m=2, amp = 15...25%, other, amp < 5%
Value_8:
!harmonic phase
```

Harmonics parameters do not depend on radius r_b . But kicks do depend. r_b can be chosen arbitrary, for e.g. if one wants to study effects on the beam core. When analyzing such "rescaled" model one should remember that results of this simulations are valid for $r < r_m$, where r_m is given by (32) with r^g set to be cylinder radius at gun position. Harmonics amplitudes should be rescaled to keep kick value fixed,

$$\left. \frac{\xi_m}{r_b^m} \right|_{new} = \left. \frac{\xi_m}{r_b^m} \right|_{old}$$

where m – harmonic number, ξ_m – harmonic amplitude, r_b – cylinder radius.

3.3 HEBL with angular imperfections (1) : EXT_ELO2

```
SUBROUTINE PRE_TRACK_EXT (type, pref, param)
CASE('EXT_ELO2')
! param(1) - skip
! param(2) - # of slices
! param(3) - length[cm]
! param(4) - omega
! param(5) - rb[cm]
! param(6) - harm number
! param(7) - relative amp
! param(8) - harm phase
param(4)=param(4)*param(3)/param(2)*param(7)/param(5)**param(6)
param(6)=(param(6)-1._8)/2._8

SUBROUTINE TRACK_EXT (type, param, coord, pwght, nturn, rflag)
REAL(8) :: r,t
CASE('EXT_ELO2')
IF(MOD(nturn,INT(param(1)))==0 .AND. nturn/=0) THEN
  r=param(4)*(coord(1)**2+coord(3)**2)**param(6)
  t=2._8*param(6)*ATAN2(coord(3),coord(1))+param(8)
  coord(2)=coord(2)-r*COS(t)
  coord(4)=coord(4)+r*SIN(t)
END IF
```

3.4 HEBL with angular imperfections (2): EXT_EL03

See EXT_EL02 description. This routine models a single arbitrary harmonic with parameters which come from measurements of electron beam profile (inside and outside the cylinder). Place HEBL element into the element list in LifeTrack input file. Set values for EXT_EL03 (use EXT_EL13 for several kick-elements separated by paraxial drift),

```
# <name>: EXT_EL03
Value_1: 1.0
! see EXT_EL00
Value_2: 1.0
! see EXT_EL00
Value_3: 200.
! see EXT_EL00
Value_4: 0.9E-10
! see EXT_EL00
Value_5: 1.
! see EXT_EL02, see below
Value_6: 2.
! see EXT_EL02
Value_7: .2
! see EXT_EL02
Value_8:
! see EXT_EL02
```

For this element rescaling is not of a great interest. Value_5 is fixed by choosing cylinder radius at the gun position.

```
SUBROUTINE PRE_TRACK_EXT (type, pref, param)
CASE('EXT_ELO3')
! param(1) - skip
! param(2) - # of slices
! param(3) - length[cm]
! param(4) - omega
! param(5) - rb[cm]
! param(6) - harm number
! param(7) - relative amp
! param(8) - harm phase
param(10)=param(4)*param(3)/param(2)*param(15)/param(7)**param(6)
param(11)=param(4)*param(3)/param(2)*param(15)*param(7)**param(6)

SUBROUTINE TRACK_EXT (type, param, coord, pwght, nturn, rflag)
REAL(8) :: r,t
CASE('EXT_ELO3')
IF(MOD(nturn,INT(param(1)))==0 .AND. nturn/=0) THEN
  r=SQRT(coord(1)*coord(1)+coord(3)*coord(3))
  t=ATAN2(coord(3),coord(1))
  IF(r .LE. param(5)) THEN
    r=param(10)*r**(param(6)-1._8)
    t=t*(param(6)-1._8)+param(8)
  ELSE
    r=param(11)*r**(-param(6)-1._8)
    t=t*(param(6)-1._8)+param(8)
  ENDIF
  coord(2)=coord(2)-r*COS(t)
  coord(4)=coord(4)+r*SIN(t)
END IF
```

3.5 Field-free propagation routine : EXT_EG00

Detailed description of drift element is given in Sec. 2.1. This element is used to create HEBL models with several kicks. It is also used with negative length to propagate coordinates from reference plane $z = L/2$ to the element's entrance plane $z = 0$ and back to $z = L/2$, i.e for any finite length element, not only HEBL. This drift also include energy deviations. See last subsection for more information about nesting. Set values for EXT_EG00,

```
# <name>: EXT_EG00
Value_1: 1.0
! skip, see EXT_EL00
Value_2: 1.0
! 1=exact, 0=paraxial
Value_3: 200.
! drift length[cm]

SUBROUTINE PRE_TRACK_EXT (type, pref, param)

CASE('EXT_EG00')
! param(1) - skip
! param(2) = 1 - exact, 0 - paraxial
! param(3) - length [cm] (negative for inverse)

SUBROUTINE TRACK_EXT (type, param, coord, pwght, nturn, rflag)
REAL(8) :: pz(3), a(4)

CASE('EXT_EG00')
IF(MOD(nturn,INT(param(1)))==0 .AND. nturn/=0) THEN
  IF(INT(param(2))==1) pz(1)=SQRT((1._8+coord(6))**2-coord(2)**2-coord(4)**2)
  IF(INT(param(2))==0) pz(1)=1._8+coord(6)
  param(3)=param(3)/pz(1)
  coord(1:3:2)=coord(1:3:2)+param(3)*coord(2:4:2)
END IF
```

3.6 Euclidean group direct routine : EXT_EG1A

Detailed description of EG direct transformation is given in Sec. 2. This routine handles misalignment of external element. Set values for EXT_EG1A,

```
# <name>: EXT_EG1A
Value_1: 1.0
! see EXT_EL00
Value_2: 200.0
! length [cm]
Value_3: 1.
! dx[mm] shift, x_new = x - dx
Value_4: 1.
! dy[mm] shift, y_new = y - dy
Value_5: 1.
! dz[mm] shift,
! entrance plane of the element is
! shifted forward by dz
Value_6: 1.
! az[deg], rotation along z (x--y plane)
! when az=90 new x axes is in opposite direction
! then old y axes
Value_7: 1.
! ay[deg], rotation along y (x--z plane)
! when ay=90 new z axes is in opposite direction
! then old x axes
Value_8: 1.
! ax[deg], rotation along x (y--z plane)
! when ax=90 new z axes is in opposite direction
! then old y axes
```

Note that EG is split into two routines. This is done to create a better nesting and save input parameters for external element. Both direct and inverse routine should be used to include misalignment correctly, i.e in element list file DIRECT TEL2 INVERSE. EXT_EG1A and EXT_EG1B are valid for arbitrary element with arbitrary map. Also note that EG routines use exact drift for propagating. Paraxial approximation appears to give bad numerical accuracy for finite length elements. One can add paraxial case by changing $p_z(1)$ in source code according to,

```
p_z(1)=1+coord(6)
```

or

```
p_z(1)=1+coord(6)-1/(2(1+coord(6)))(coord(2)**2+coord(4)**2)
```

3.6 Euclidean group direct routine : EXT_EG1A

```
SUBROUTINE PRE_TRACK_EXT (type, pref, param)
REAL(8) :: pi = 3.14159265358979_8
! direct transformation
CASE('EXT_EG1A')
! param(1) - skip
! param(2) - length [cm]
! param(3) = dx [mm]
! param(4) = dy [mm]
! param(5) = dz [mm]
! param(6) = az [deg]
! param(7) = ay [deg]
! param(8) = ax [deg]
! convert [mm] to [cm]
param(3:5)=param(3:5)/10._8
! convert [deg] to [rad]
param(6:8)=param(6:8)*pi/180._8
param(10:15)=&
(/COS(param(6)),SIN(param(6)),COS(param(7)),SIN(param(7)),COS(param(8)),SIN(param(8))/)

SUBROUTINE TRACK_EXT (type, param, coord, pwght, nturn, rflag)
REAL(8) :: pz(3), a(4)
CASE('EXT_EG1A')
IF(MOD(nturn,INT(param(1)))==0 .AND. nturn/=0) THEN
! x,y - translation
coord(1:3:2)=coord(1:3:2)-param(3:4)
! z - translation
pz(1)=SQRT((1._8+coord(6))**2-coord(2)**2-coord(4)**2)
param(5)=param(5)/pz(1)
coord(1:3:2)=coord(1:3:2)+param(5)*coord(2:4:2)
! z - rotation
a(1:2)=coord(1:2)*param(10)-coord(3:4)*param(11)
a(3:4)=coord(3:4)*param(10)+coord(1:2)*param(11)
coord(1:4)=a
! y - rotation
! step 1
pz(1)=SQRT((1._8+coord(6))**2-coord(2)**2-coord(4)**2)
pz(2)=pz(1)*param(12)-coord(2)*param(13)
pz(3)=-coord(1)*param(13)
a(1)=coord(1)*param(12)
a(2)=coord(2)*param(12)+pz(1)*param(13)
coord(1:2)=a(1:2)
! step 2
pz(3)=-pz(3)/pz(2)
coord(1:3:2)=coord(1:3:2)+pz(3)*coord(2:4:2)
! x - rotation
! step 1
```

3.6 Euclidean group direct routine : EXT_EG1A

```
pz(1)=SQRT((1._8+coord(6))**2-coord(2)**2-coord(4)**2)
pz(2)=pz(1)*param(14)-coord(4)*param(15)
pz(3)=-coord(3)*param(15)
a(3)=coord(3)*param(14)
a(4)=coord(4)*param(14)+pz(1)*param(15)
coord(3:4)=a(3:4)
! step 2
pz(3)=-pz(3)/pz(2)
coord(1:3:2)=coord(1:3:2)+pz(3)*coord(2:4:2)
END IF
```

3.7 Euclidean group inverse routine : EXT_EG1B

Detailed description of EG direct transformation is given in Sec. 2. This routine handles misalignment of external element. Set values for EXT_EG1B,

```
# <name>: EXT_EL1B
Value_1: 1.0
! see EXT_EG1A
Value_2: 200.0
! see EXT_EG1A
Value_3: 1.
! see EXT_EG1A
Value_4: 1.
! see EXT_EG1A
Value_5: 1.
! see EXT_EG1A
Value_6: 1.
! see EXT_EG1A
Value_7: 1.
! see EXT_EG1A
Value_8: 1.
! see EXT_EG1A
```

See discussion for EXT_EL1A routine.

3.7 Euclidean group inverse routine : EXT_EG1B

```
SUBROUTINE PRE_TRACK_EXT (type, pref, param)
REAL(8) :: pi = 3.14159265358979_8
! inverse transformation
CASE('EXT_EG1B')
! param(1) - skip
! param(2) - length [cm]
! param(3) = dx [mm]
! param(4) = dy [mm]
! param(5) = dz [mm]
! param(6) = az [deg]
! param(7) = ay [deg]
! param(8) = ax [deg]
! convert [mm] to [cm]
param(3:5)=param(3:5)/10._8
! convert [deg] to [rad]
param(6:8)=param(6:8)*pi/180._8
param(10:15)=&
(/COS(param(6)),SIN(param(6)),COS(param(7)),SIN(param(7)),COS(param(8)),SIN(param(8))/)

SUBROUTINE TRACK_EXT (type, param, coord, pwght, nturn, rflag)
REAL(8) :: pz(3), a(4)
CASE('EXT_EG1B')
IF(MOD(nturn,INT(param(1)))==0 .AND. nturn/=0) THEN
! x - rotation
! step 1
pz(1)=SQRT((1._8+coord(6))**2-coord(2)**2-coord(4)**2)
pz(2)=pz(1)*param(14)+coord(4)*param(15)
pz(3)=coord(3)*param(15)
a(3)=coord(3)*param(14)
a(4)=coord(4)*param(14)-pz(1)*param(15)
coord(3:4)=a(3:4)
! step 2
pz(3)=-pz(3)/pz(2)
coord(1:3:2)=coord(1:3:2)+pz(3)*coord(2:4:2)
! step 3
coord(3)=coord(3)-param(2)*param(15)
! step 4
pz(1)=SQRT((1._8+coord(6))**2-coord(2)**2-coord(4)**2)
coord(1:3:2)=coord(1:3:2)+coord(2:4:2)*param(2)*(1._8-param(14))/pz(1)
! y - rotation
! step 1
pz(1)=SQRT((1._8+coord(6))**2-coord(2)**2-coord(4)**2)
pz(2)=pz(1)*param(12)+coord(2)*param(13)
pz(3)=coord(1)*param(13)
a(1)=coord(1)*param(12)
a(2)=coord(2)*param(12)-pz(1)*param(13)
```

3.7 Euclidean group inverse routine : EXT_EG1B

```
coord(1:2)=a(1:2)
! step 2
pz(3)=-pz(3)/pz(2)
coord(1:3:2)=coord(1:3:2)+pz(3)*coord(2:4:2)
! step 3
coord(1)=coord(1)-param(2)*param(13)
! step 4
pz(1)=SQRT((1._8+coord(6))**2-coord(2)**2-coord(4)**2)
coord(1:3:2)=coord(1:3:2)+coord(2:4:2)*param(2)*(1._8-param(12))/pz(1)
! z - rotation
a(1:2)=coord(1:2)*param(10)+coord(3:4)*param(11)
a(3:4)=coord(3:4)*param(10)-coord(1:2)*param(11)
coord(1:4)=a
! z - translation
pz(1)=SQRT((1._8+coord(6))**2-coord(2)**2-coord(4)**2)
param(5)=-param(5)/pz(1)
coord(1:3:2)=coord(1:3:2)+param(5)*coord(2:4:2)
! x,y - translation
coord(1:3:2)=coord(1:3:2)+param(3:4)
END IF
```

3.8 Ideal solenoid (1) routine : EXT_EGS0

Detailed description of ideal solenoid is given in Sec. 2.4. This routine can be used to model HEBL inside ideal solenoid, assuming HEBL is a drift-kick. Then instead of EXT_EG00 use EXT_EGS0. But only to propagate forward. Procedure of moving coordinates from reference plane $z = L/2$ to entrance plane $z = 0$ is based on EXT_EG00. Set values for EXT_EGS0,

```
# <name>: EXT_EGS0
Value_1: 1.0
! see EXT_EL00
Value_2: 200.
! length [cm]
Value_3: 3.E-8
! bz[cm^-1] = Bz/|B rho[cm]|

SUBROUTINE PRE_TRACK_EXT (type, pref, param)
! ideal solenoid
CASE('EXT_EGS0')
! param(1) - skip
! param(2) - length [cm]
! param(3) - bz[cm^-1]=Bz/|B rho|

SUBROUTINE TRACK_EXT (type, param, coord, pwght, nturn, rflag)
REAL(8) :: pz(3), a(4)

CASE('EXT_EGS0')
IF(MOD(nturn,INT(param(1)))==0 .AND. nturn/=0) THEN
  pz(1)=-SQRT((1._8+coord(6))**2-(coord(2)+coord(3)*param(3)/2._8)**2-&
    (coord(4)-coord(1)*param(3)/2._8)**2)
  pz(2)=COS(-param(2)*param(3)/pz(1))
  pz(3)=SIN(-param(2)*param(3)/pz(1))
  a(1)=(1._8+pz(2))/2._8*coord(1)+pz(3)/param(3)*coord(2)+&
    pz(3)/2._8*coord(3)+(1._8-pz(2))/param(3)*coord(4)
  a(2)=pz(3)*param(3)/4._8*coord(1)+(1._8+pz(2))/2._8*coord(2)+&
    param(3)*(1._8-pz(2))/4._8*coord(3)+pz(3)/2._8*coord(4)
  a(3)=-pz(3)/2._8*coord(1)+(pz(2)-1._8)/param(3)*coord(2)+&
    (1._8+pz(2))/2._8*coord(3)+pz(3)/param(3)*coord(4)
  a(4)=param(3)*(1._8-pz(2))/4._8*coord(1)-pz(3)/2._8*coord(2)+&
    pz(3)*param(3)/4._8*coord(3)+(1._8+pz(2))/2._8*coord(4)
  coord(1:4)=a
END IF
```

3.9 Ideal solenoid (2) routine : EXT_EGS1

This routine Sec. ?? can be used to model misalignment for kick–drift HEBL inside an ideal solenoid. See last subsection for details. Set values for EXT_EGS1,

```
# <name>: EXT_EGS1
Value_1: 1.0
! see EXT_EL00
Value_2: 200.
! length [cm]
Value_3: 3.E-8
! bz[cm^-1] = Bz/|B rho[cm]
Value_4: 1
! 1 - direct transformation,
! 0 - inverse transformation
Value_5:
! ay[deg], see EXT_EG1A
Value_6:
! ax[deg], see EXT_EG1A
```

Length of this solenoid L depends on alignment angles α_x , α_y and particle coordinates x , y . When used for direct transformation (Value.4: 1),

$$L = x \tan \alpha_y + y \tan \alpha_x$$

For inverse transformation (Value.4: 0),

$$L = L_{element} - x \tan \alpha_y + y \tan \alpha_x$$

3.9 Ideal solenoid (2) routine : EXT_EGS1

```
SUBROUTINE PRE_TRACK_EXT (type, pref, param)
! ideal solenoid with coord. dependent length
CASE('EXT_EGS1')
! param(1) - skip
! param(2) - length [cm]
! param(3) - bz[cm^-1]=Bz/|B rho|
! param(4) = 1 - direct, 0 - inverse
! param(5) - ax
! param(6) - ay
param(5)=TAN(param(5))
param(6)=TAN(param(6))

SUBROUTINE TRACK_EXT (type, param, coord, pwght, nturn, rflag)
REAL(8) :: pz(3), a(4)

CASE('EXT_EGS1')
IF(MOD(nturn,INT(param(1)))==0 .AND. nturn/=0) THEN
  IF(INT(param(2))==1) param(2)=coord(1)*param(6)+coord(3)*param(5)
  IF(INT(param(2))==0) param(2)=param(2)-coord(1)*param(6)-coord(3)*param(5)
  pz(1)=-SQRT((1._8+coord(6))**2-(coord(2)+coord(3)*param(3)/2._8)**2-&
    (coord(4)-coord(1)*param(3)/2._8)**2)
  pz(2)=COS(-param(2)*param(3)/pz(1))
  pz(3)=SIN(-param(2)*param(3)/pz(1))
  a(1)=(1._8+pz(2))/2._8*coord(1)+pz(3)/param(3)*coord(2)+&
    pz(3)/2._8*coord(3)+(1._8-pz(2))/param(3)*coord(4)
  a(2)=pz(3)*param(3)/4._8*coord(1)+(1._8+pz(2))/2._8*coord(2)+&
    param(3)*(1._8-pz(2))/4._8*coord(3)+pz(3)/2._8*coord(4)
  a(3)=-pz(3)/2._8*coord(1)+(pz(2)-1._8)/param(3)*coord(2)+&
    (1._8+pz(2))/2._8*coord(3)+pz(3)/param(3)*coord(4)
  a(4)=param(3)*(1._8-pz(2))/4._8*coord(1)-pz(3)/2._8*coord(2)+&
    pz(3)*param(3)/4._8*coord(3)+(1._8+pz(2))/2._8*coord(4)
  coord(1:4)=a
END IF
```

3.10 Element nesting tips

- Finite length element

As it is already mentioned initial coordinates are given at $z = L/2$ plane. This position is final for previous element that is a linear map from IP to EXT element. Since that map is linear then using inverse drift coordinates of a particle can be found at the element entrance plane. Then particles propagate with EXT element which particular form map be arbitrary. The last step is to use a negative drift to move from $z = L$ to $z = L/2$. In elements list one put,

... DR1 TEL2 DR1 ...

and DR1 is # DR1: EXT_EG00 with Value.3: -L/2.

- Combining different kick models

If LEN1 and LEN2 are two different and comparable (i.e. valid in the same range of parameters, e.g. two harmonics with the same radius). Then they can be placed at the same position. In the elements list

... LEN1 LEN2 ...

Any number of different kicks can be nested in this way.

- Dividing element into several kicks

... D(-L/2) K1(L/n) D(L/n) ... Kn(L/n) D(L/n) D(-L/2) ...

where D is a drift, K – kick (or several kicks), n – number of kicks, L – elements length.

- Handling general misalignment

... EGDIR LEN EGINV ...

where EGDIR – direct transformation, EGINV – inverse transformation, LEN – any external element.

- Drift–kick HEBL inside ideal solenoid

D(-L/2) K1(L/n) S(L/n) ... Kn(L/n) S(L/n) D(-L/2)

where D– drift, K – kick (or several kicks), S – ideal solenoid

- Misalignment for drift–kick HEBL inside ideal solenoid

Direct and inverse transformation should be performed after each kick.

D(-L/2) DIR K(L/n) INV S(L/n) ... DIR K(L/n) INV S(L/n) D(-L/2)

where,

DIR = S(dz) S(1,ax,ay,L) EGDIR(dx,dy,ax,ay,az,L=0)

First solenoid of length dz for z translation. Then coordinates propagate to the element reference frame. Then usual EG is performed for zero lens element. The inverse procedure is similar.

INV = EGINV(dx,dy,ax,ay,az,L=0) S(0,ax,ay,L) S(-dz)

4 HEBL models scripts

This section mostly describes scripts for HEBL parameters generation. Parameters are used for HEBL angular (1.5) and radial models (1.6). Several post processing scripts are also presented. They are used to obtain particles loss rate and lost particles positions inside initial beam distribution. Below you can find current list of scripts for HEBL,

- AngularDistribution.sh
- RadialDistribution.sh
- LossRate
- LostParticles

All scripts are written with shell and SDDS.

4.1 2D particles distribution

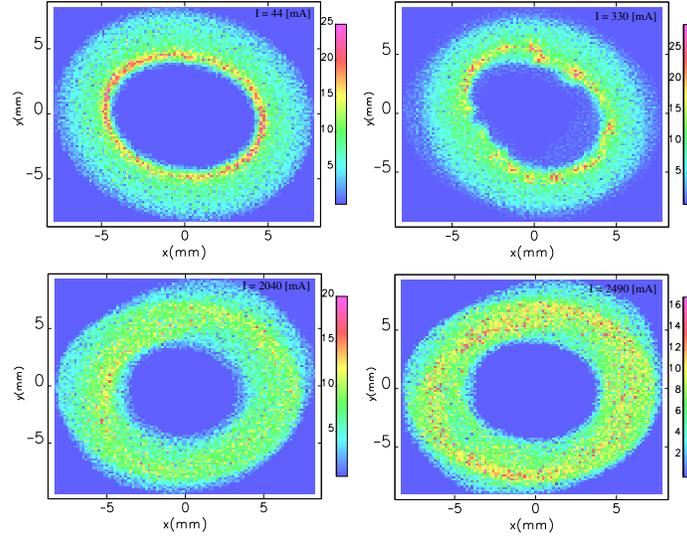


Figure 15: 2D electron beam histogram for different beam currents.

4.1 2D particles distribution

This script convert input 2D transverse electron beam distribution into SDDS format and plots 2D histogram of particles distribution (Fig. 15). It also calculates distribution center of mass and moves zero into it. One also can specify center offset and range of particles radius (r_{min}, r_{max}). This options are used in angular model to monitor how harmonics parameters depend on the beam offset (dipole harmonic) and how different regions of electron beam contribute into harmonics amplitudes (Fig. 16).

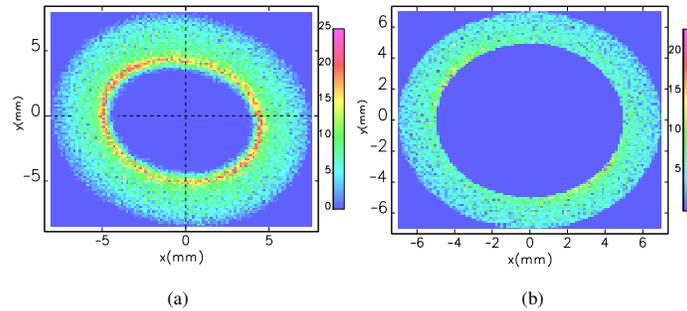


Figure 16: (a) – off centered beam $(x, y)_{off} = 0.5$ [mm], (b) – filter by radius 5...6 [mm].

4.2 Angular model script

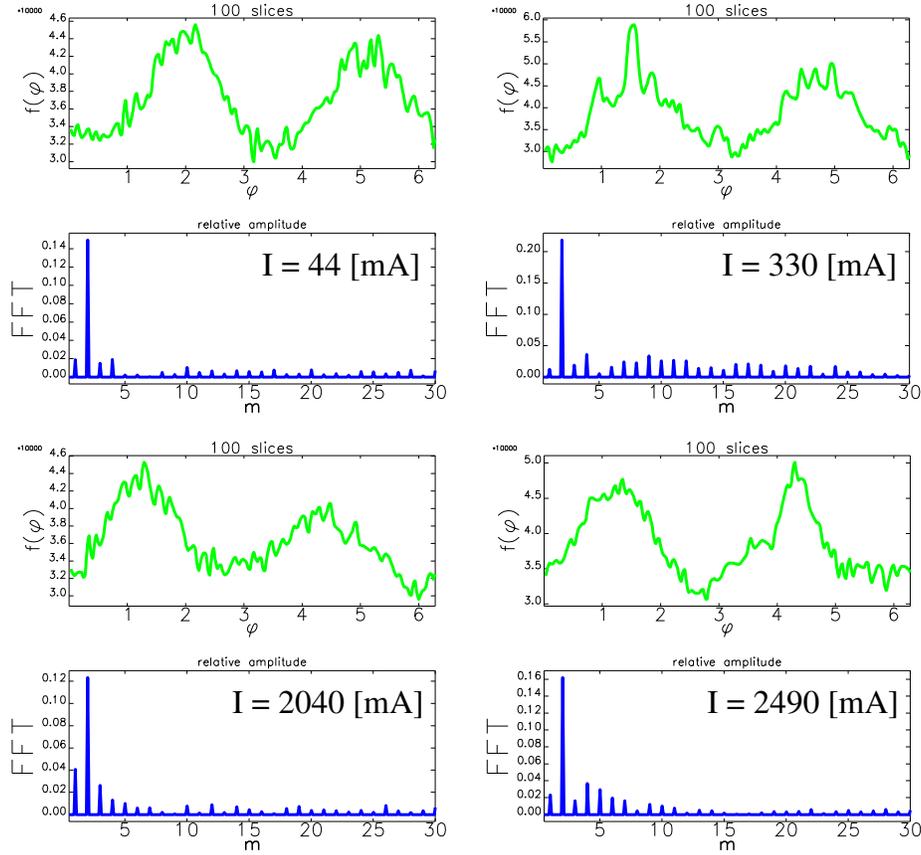


Figure 17: Angular distribution function and relative harmonics amplitudes.

4.2 Angular model script

This script is used to obtain angular distribution function and harmonic parameters (relative amplitudes and phases) for transverse electron beam profile. To run this script use command:

```
./AngularDistribution.sh
```

You also may want to change script parameters such as input distribution, center of charge offset, particle filter, number of slices, number of harmonics and other (see comments for details). Script creates file "harmonics.txt" where specified number of largest harmonic parameters are saved. Template for angular routine is saved into "angular.txt". Harmonic parameters can be rescaled to a large cylinder radius (see comments). This is done in order to study effects on the beam core and to save some computation time. But results of such simulations are only valid for the region $r < r_b$, where r_b is original cylinder radius.

Printout for SDDS file results/angular.naff

m	fFrequency	fAmplitude	fPhase
1.000883e+00	1.592955e-01	1.893044e-02	-1.206049e+00
2.001751e+00	3.185886e-01	1.493936e-01	2.406676e+00
3.002622e+00	4.778821e-01	1.492338e-02	1.460827e-01
4.003503e+00	6.371773e-01	1.917637e-02	-1.669732e+00
8.007018e+00	1.274357e+00	5.053672e-03	-2.667851e-01
9.007868e+00	1.433647e+00	2.558125e-03	2.977856e-01
...			
5.204551e+01	8.283300e+00	4.380327e-03	-2.902914e+00

Figure 18: Harmonics parameters.

4.2.1 Angular distribution

To create an angular distribution function transverse profile is divided into angular slices. Number of particles in each slice is then calculated and normalized with respect to the number of angular slices. Then FFT is performed to find harmonic parameters. Amplitudes of harmonics are then normalized with respect to the signal average value. On Fig 17 angular distribution and it's FFT is shown for several transverse electron beam profiles. One can also offset distribution center to monitor harmonics behavior. It is also possible to filter particles by radius in order to see how harmonics parameters alter within the beam. For off set and filtered case output is shown on Fig. 19.

4.2.2 Harmonic parameters

On Fig. 18 script output with harmonics parameters is shown. Program searches for specified number of largest harmonics and then sort them by m – harmonic number. Harmonic parameters can be rescaled to a large cylinder radius. It is useful if one care only about effects on the beam core. Then to keep kick constant new amplitude is calculated (see comments). Template for one specified harmonic is save into "angular.txt" (Fig. 20, 21, 22).

4.2 Angular model script

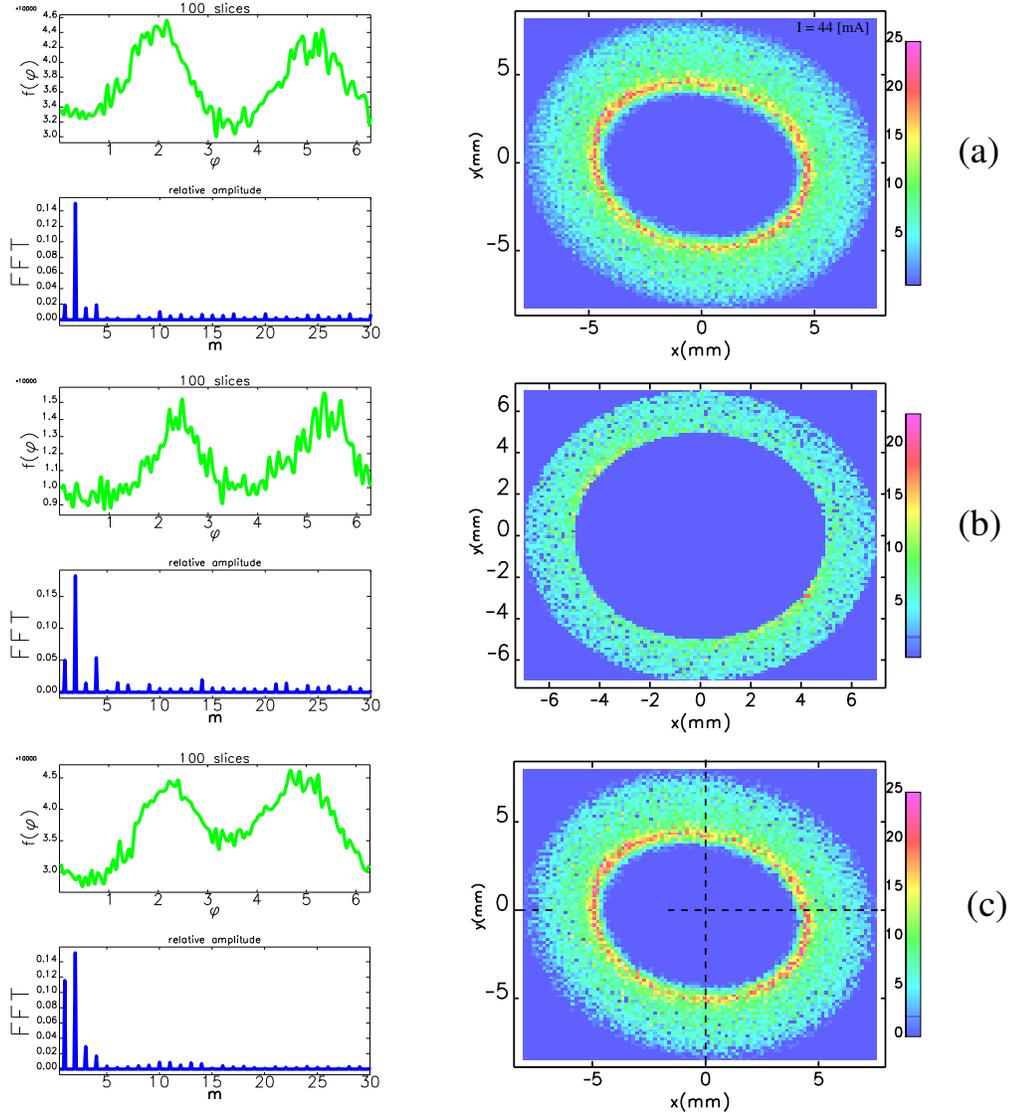


Figure 19: (a) – usual case, (b) – filtered case, (c) – offset case.

```
Value_1: 1.0 ! skip
Value_2: 1.0 ! # slices
Value_3: 200.0 ! length[cm]
Value_4: 1.0E-10 ! omega, i=0.4, b=0.135
Value_5: 0.21 !rb[cm]
Value_6: 2 ! harmonic #
Value_7: 0.149393571407428 ! harmonic amp
Value_8: 2.406675646657053e+00 ! harmonic phase
```

Figure 20: Template for quadrupole harmonic with cylinder radius $r_b = 3.5\sigma$

```
Value_1: 1.0 ! skip
Value_2: 1.0 ! # slices
Value_3: 200.0 ! length[cm]
Value_4: 1.0E-10 ! omega, i=0.4, b=0.135
Value_5: 1. !rb[cm]
Value_6: 2 ! harmonic #
Value_7: 3.38760932896663 ! harmonic amp
Value_8: 2.406675646657053e+00 ! harmonic phase
```

Figure 21: Template for quadrupole harmonic and amplitude rescaled to cylinder radius $r_b = 1$ [cm].

```
Value_1: 1.0 ! skip
Value_2: 1.0 ! # slices
Value_3: 200.0 ! length[cm]
Value_4: 1.0E-10 ! omega, i=0.4, b=0.135
Value_5: 0.21 !rb[cm]
Value_6: 3 ! harmonic #
Value_7: 0.0149233755097962 ! harmonic amp
Value_8: 1.460826894915697e-01 ! harmonic phase
```

Figure 22: Template for sextupole harmonic.

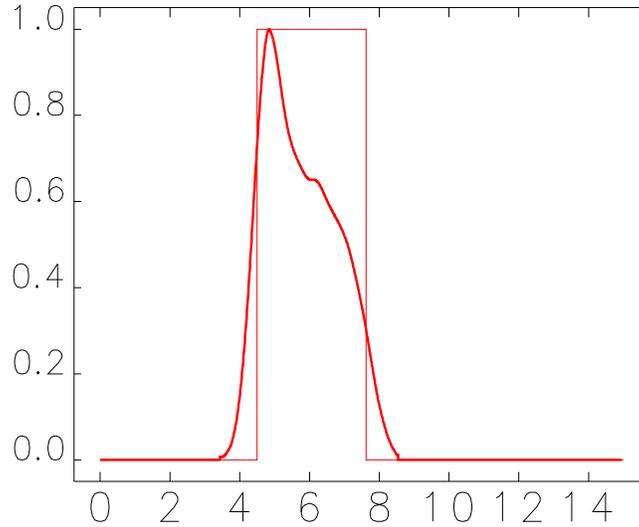


Figure 23: Normalized beam radial distribution and ideal distribution at the gun position.

4.3 Radial model script

This script is used to simulate HEBL element with non ideal radial profile. Non ideal radial profile can be contracted from electron beam measurements or simulations. Script creates polynomial interpolation of normalized radial electric field and calculates normalization parameters (see ??). The run command is,

```
./AngularDistribution.sh
```

One can start with 2D particles distribution, normalized radial density or normalized radial electric field. Interpolation order and regions can be modified inside the script. By default electric field is divided into two parts r_1, r_2 and r_2, r_3 and two corresponding polynoms are found. r_1 is chosen such that electric field for $r < r_1$ is negligible. r_3 is the distribution cut off, i.e. $\rho(r > r_3) = 0$ and the electric field is proportional to $1/r$. r_2 should be chosen reasonably to provide a better interpolation. By default interpolation order is 4 for the first region and 6 for the second one. This particular chose is done because the number of free parameters is limited. One can use smaller interpolation orders but not lager then default values. It is also possible to change matching parameters, i.e. the position in sigmas and sigma size itself.

4.3.1 Normalized radial distribution from 2D

First normalized radial beam density is calculated (Fig. 23). 2D beam distribution is divided into radial slices, then the number of particles is calculated for each slice.

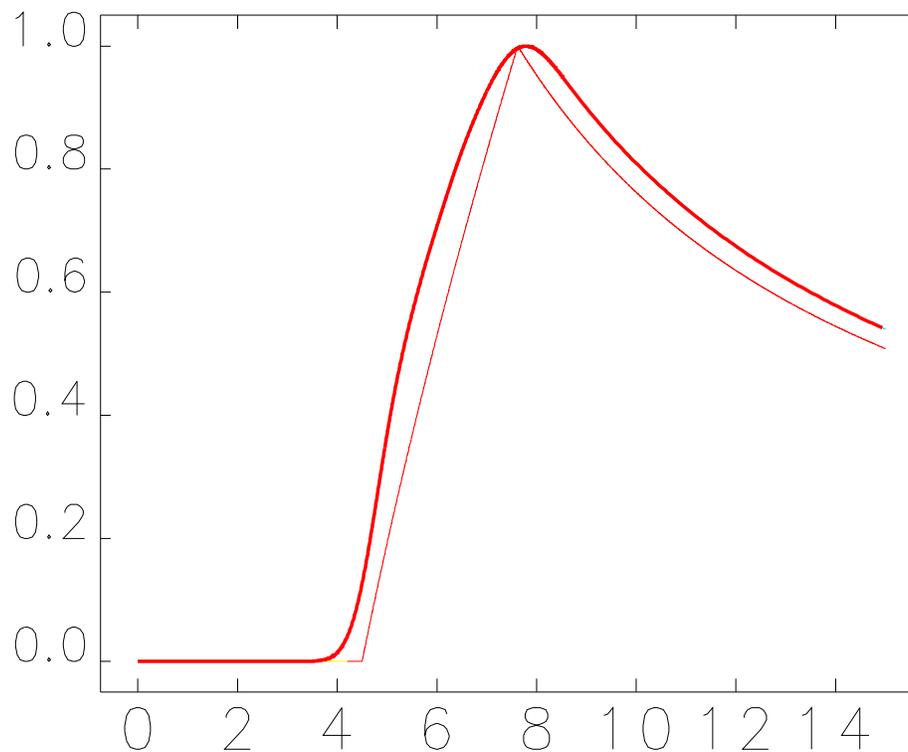


Figure 24: Normalized electric field and ideal HEBL field at the gun position.

4.3.2 Normalized electric field

At this stage normalized electric field is calculated (Fig 24). We also find position where electric field has it's maximum. Transverse kick is inverse proportional to this radius value. For non ideal profiles r_{max} is larger then the ideal maximum location position. Then maximum kick is smaller.

4.3.3 Transport of density and electric field

Obtained normalized density and electric field are given in some position (gun or other) outside the main solenoid. Then we should transport these distributions and match to the desired numbers of sigma(Fig. 24).

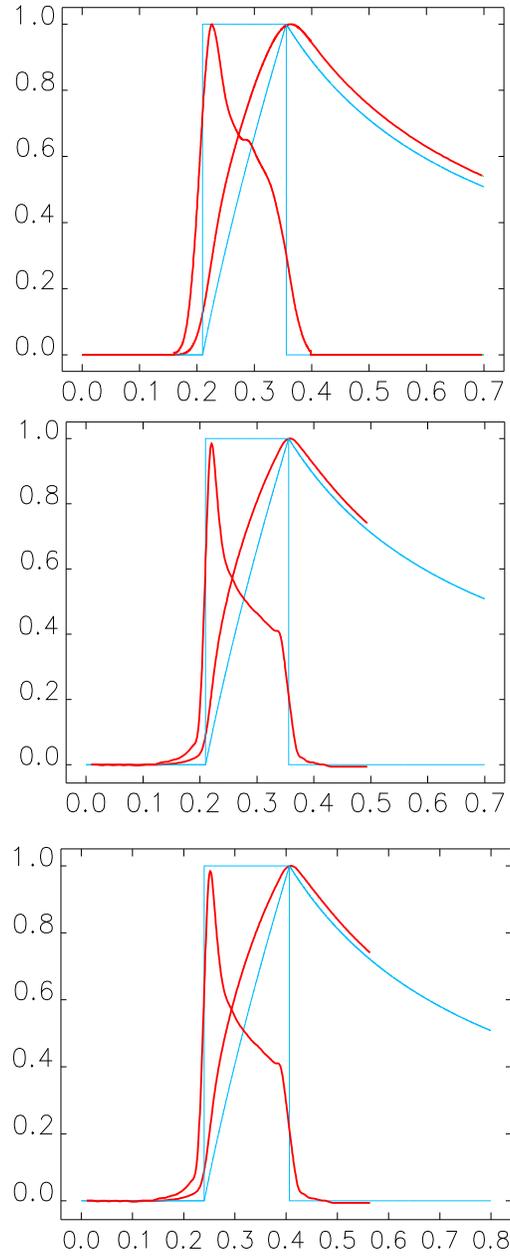


Figure 25: (a) – profile from 2D matched to 3.5σ , (b) – profile from warp matched to 3.5σ , (c) – profile from warp matched to 4σ .

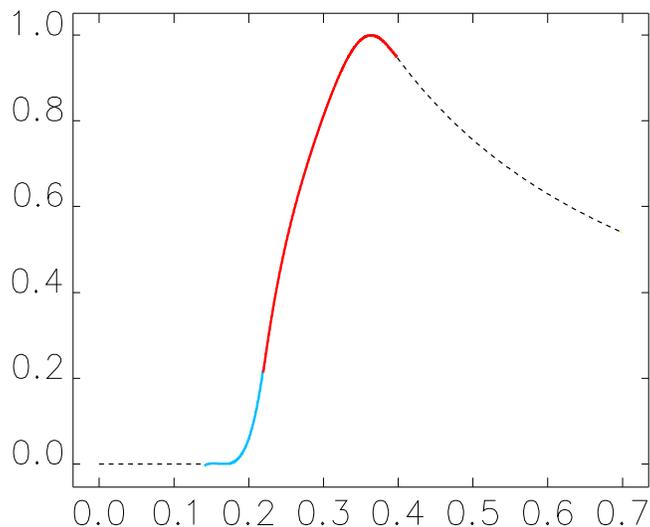


Figure 26: Electric field interpolation.

4.3.4 Electric field interpolation

Electric field then interpolated to obtain analytical expression (Fig. 26). Interpolation is done for two regions (shown in different colors). Other simulation parameters are written into "poly.txt" file (see below).

```
radial model parameters
nsigma= 3.50 -- numbers of sigma
rmax_cut= 3.984746127526733e-01 -- rho(r>rmax_cut=0)
param(8)=rmax/frac=0.377728621808966 -- rmax/(kappa/eta)
rmin= 0.14 -- interpolation-1 start
rmax= 0.2193333333333333 -- intrpoalition-1 end
interpolation order= 5 -- interpolation order
f(r) -- first poly
Printout for SDDS file results/fit1.sdds
ErSddspfitlabel = Er = -1.56034 +14.1873*r +99.0892*r$a2$n
-1370.34*r$a3$n +3613.19*r$a4$n
rmin= 0.2193333333333333 -- interpolation-2 start
rmax= 3.984746127526733e-01 -- interpolation-2 end
interpolation order= 7 -- interpolation order
f(r) -- second poly
Printout for SDDS file results/fit2.sdds
ErSddspfitlabel = Er = 28.2834-835.716*r +8870.45*r$a2$n-
46438.9*r$a3$n +130594*r$a4$n-189364*r$a5$n +111241*r$a6$n
```

4.4 Lost particle distribution script

This set of scripts is used to obtain particles loss rate due to the presents of HEBL element and lost particle positions in the initial beam distribution.

4.4.1 Loss rate

Here beam intensity is normalized with reference intensity (without HEBL). And then slopes (loss rate) are calculated (Fig. 27).

4.4.2 Histograms

Several histograms are used to monitor lost particles positions within the initial distribution. On Fig. 28 2D histograms of lost particles are shown. Number of lost particles with respect to 4D radius ($r^2 = x^2 + y^2 + p_x^2 + p_y^2$) is shown on Fig. 29.

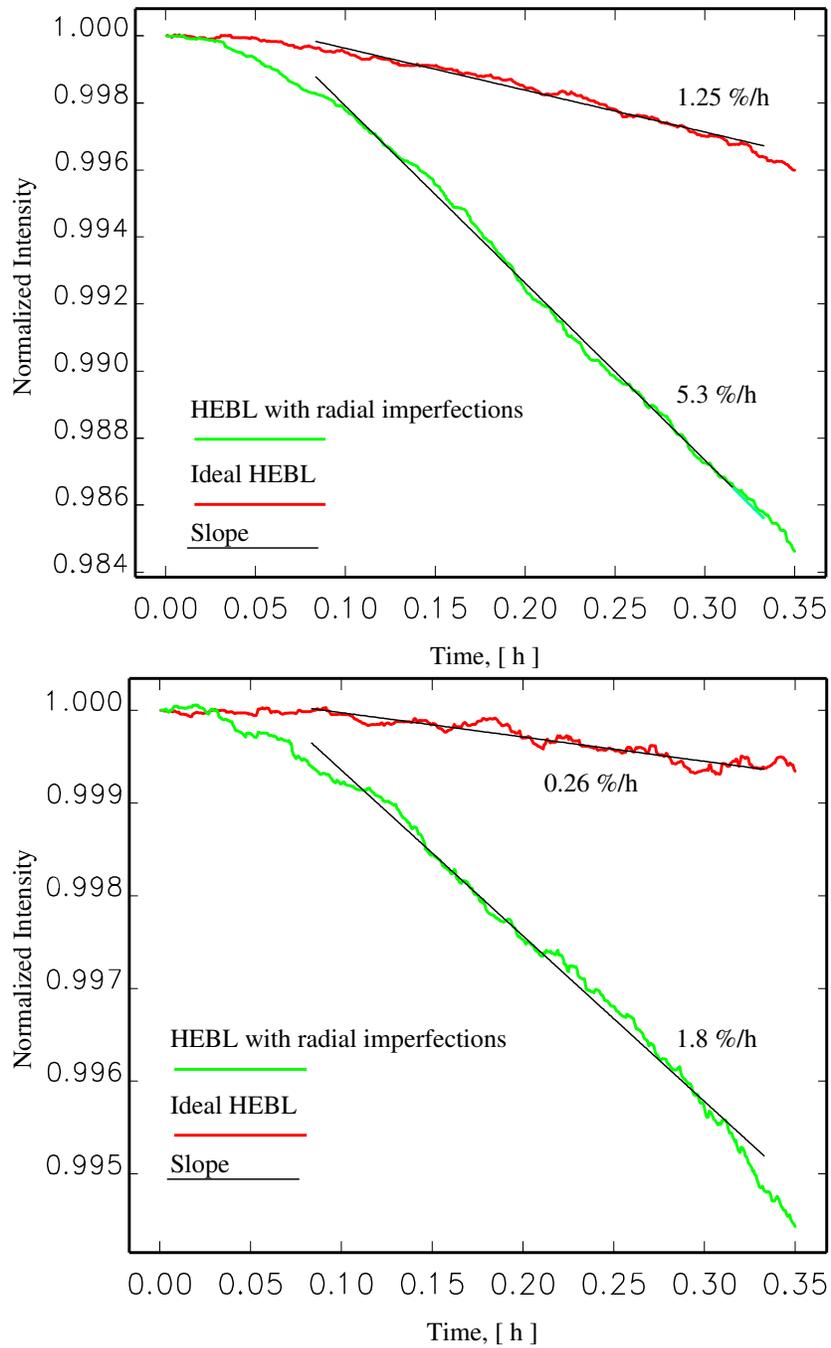


Figure 27: Particles loss rate for 3.5σ (top) and 3.75σ (bottom).

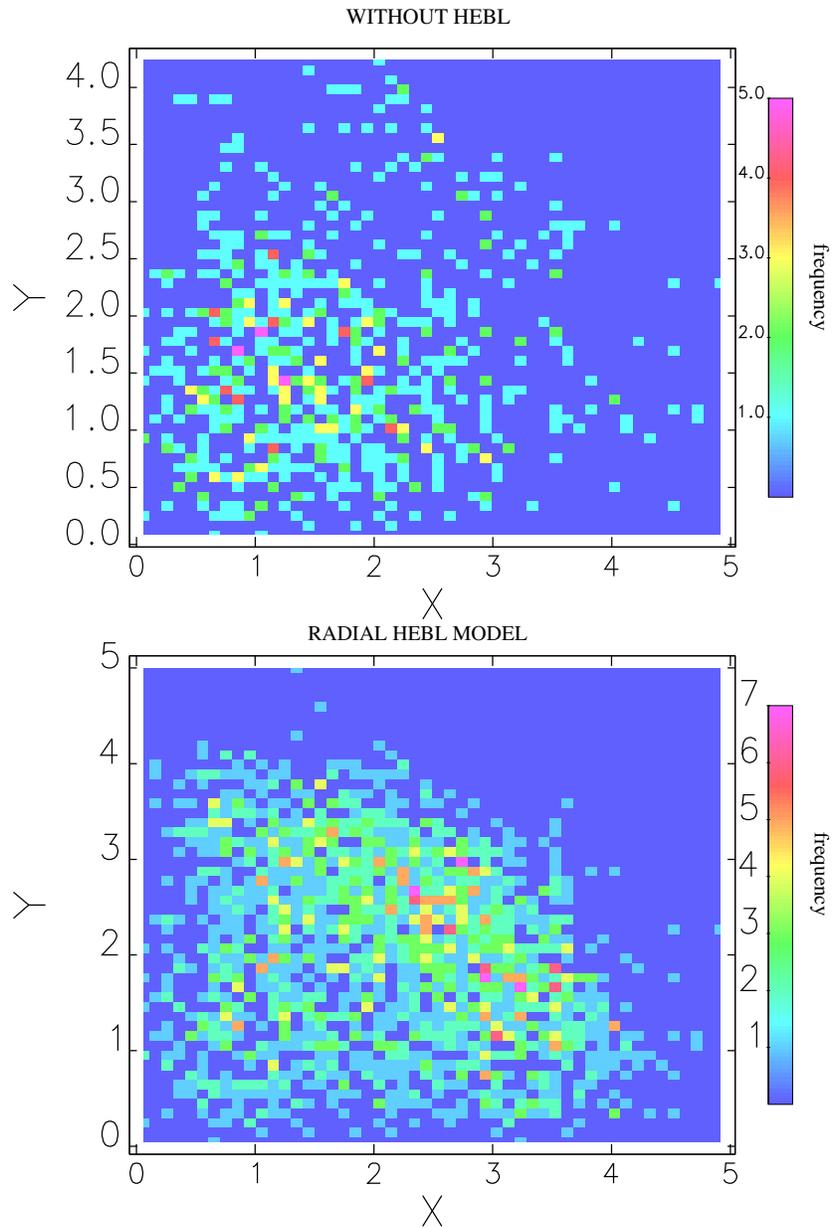


Figure 28: Lost particles positions inside initial distribution; without lens (top), radial model with 3.5σ (bottom). $X = \sqrt{x^2 + p_x^2}$ and $Y = \sqrt{y^2 + p_y^2}$

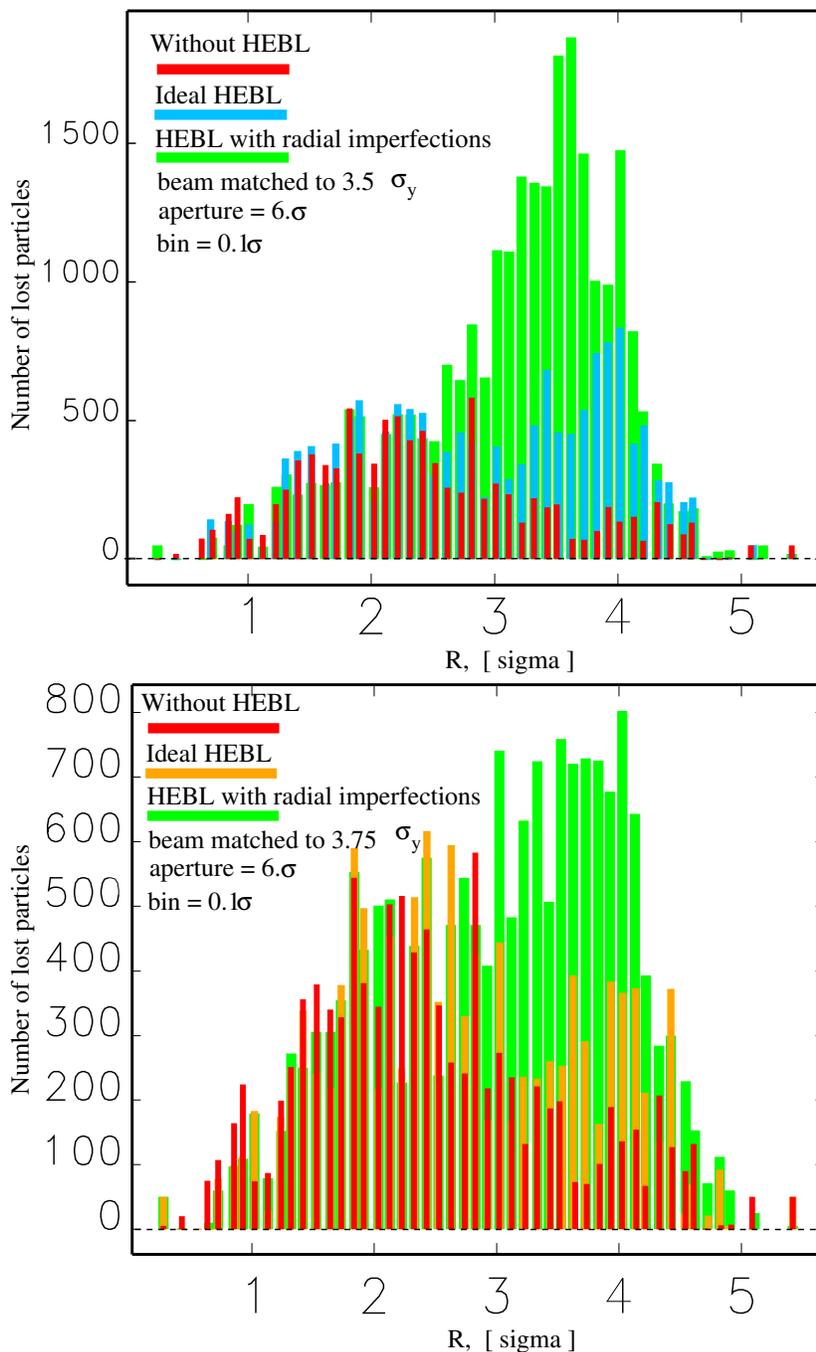


Figure 29: Number of lost particles as a function of 4D radius.

5 HEBL simulations

5.1 Harmonics

5.2 Aligment

5.3 Experiment