ACDIV-2018-04
March 2018

Development of a pass method for an insertion device with radiation

Oscar Jiménez Arranz, Zeus Martí

Abstract

A new pass method named GWigSymplecticRadPass, which simulates the trajectory of an electron in an insertion device -wiggler and ondulator- given the initial conditions, has been created. This method is a modification of GWigSym- plecticPass which did not take into consideration classical radiation loss.

The first part of the report consists on the presentation of the symplectic map used in the pass method. Then all the subroutines and functions added to the method are explained in detail. Finally, a numerical comparison between AT method, Elegant method and the analytic result is done.

Accelerator Division
Alba Synchrotron Light Source
c/ de la Llum, 2-26
08290 Cerdanyola del Valles, Spain

Development of a pass method for an insertion device with radiation

Author: Óscar Jiménez Arranz
Supervisor: Zeus Martí Díaz

March 15, 2018

1 Introduction

A new pass method named ***GWigSymplecticRadPass***, which simulates the trajectory of an electron in an insertion device -wiggler and ondulator- given the initial conditions, has been created. This method is a modification of *GWigSymplecticPass* which did not take into consideration classical radiation loss.

The first part of the report consists on the presentation of the symplectic map used in the pass method. Then all the subroutines and functions added to the method are explained in detail. Finally, a numerical comparison between AT method, Elegant method and the analytic result is done.

2 Insertion devices' symplectic map

According to Ref. [1], the second order symplectic map of a general s-dependent static magnetic field is:

$$\begin{aligned} M_2 = exp\left(: -\frac{(p_z-\delta)\Delta\sigma}{2} :\right) A_y exp\left(: -\frac{p_y^2\Delta\sigma}{4(1+\delta)} :\right) A_y^{-1} \\ \times A_x exp\left(: -\frac{p_x^2\Delta\sigma}{2(1+\delta)} :\right) A_x^{-1} \\ \times A_y exp\left(: -\frac{p_y^2\Delta\sigma}{4(1+\delta)} :\right) A_y^{-1} exp\left(: -\frac{(p_z-\delta)\Delta\sigma}{2} :\right) \end{aligned} \tag{1}$$

where we can distinguish three different Lie maps obtained from exactly solvable Hamiltonian. A detailed development and the solution of all three maps can be founded in Ref. [1].

3 New subroutines and functions

As said in the introduction, some new subroutines and functions have been added in order to implement the radiation loss. In this part of the report we will introduce this new part that have been added to the library ***gwigrad.c*** which is used instead of *gwig.c*.

3.1 *rad_loss* subroutine

This subroutine contributes to the determination of the radiation loss. *rad_loss* is the first subroutine to be explained since it is the one that fix us all the new information we need to determine. It has four arguments:

1. double *X: initial position of the electron and at the end of the subroutine it has the new coordinates.
2. double B2P: square of the component of the magnetic field which is perpendicular to the velocity.
3. double L: length of the insertion device.
4. double irho: inverse of the bending radius.

Taking all this into consideration we can determine the energy change with the following formula:

$$\delta_f = \delta_0 + d\delta \tag{2}$$

where:

$$d\delta = -C_{RAD}(1+\delta)^2|\vec{e}\times\vec{B}|^2\left(1+\frac{x}{\rho}+\frac{x'^2+y'^2}{2(1+\delta)^2}\right)dL \tag{3}$$

which is demonstrated in Ref. [2]. Not only do radiation loss affect the energy, it also echoes on the momentum with:

$$x_f'^i = x_0'^i\frac{\delta_f}{\delta_0}, \qquad for \quad i = x, y \tag{4}$$

We can easily see how $d\delta < 0$ and $|x_f'^i| < |x_0'^i|$.

3.2 *B2perp* function

The aim of this subroutine is to determine the square of the component of the magnetic field which is perpendicular to the velocity, that is, $|\vec{e}\times\vec{B}|^2$ where $\vec{e}$ is velocity unit vector. It has three arguments:

1. double bx: x component of the magnetic field.
2. double by: y component of the magnetic field.
3. double *X: position of the electron.

Let's start determining the cross product between the velocity unit vector and the magnetic field:

$$\vec{e}\times\vec{B} = \begin{vmatrix} \hat{x} & \hat{y} & \hat{z} \\ e_x & e_y & e_z \\ B_x & B_y & B_z \end{vmatrix} = (e_yB_z - e_xB_y, e_zB_x - e_xB_z, e_xB_y - e_yB_x) \tag{5}$$

We can easily see how:

$$|\vec{e} \times \vec{B}|^2 = (e_y B_z - e_x B_y)^2 + (e_z B_x - e_x B_z)^2 + (e_x B_y - e_y B_x)^2 \tag{6}$$

Finally, it is only necessary to determine the velocity. It can be proven that:

$$v^i = cx'^i \frac{\sqrt{E_{id}^2 - m^2c^4}}{E_{id}(1+\delta)}, \qquad for \quad i = x, y \tag{7}$$

where the demonstration can be found in Ref [3].

3.3 *GWigBx*, *GWigBy* and *GWigBz* subroutines

This both subroutines determine the magnetic field generated by an insertion device for a certain position. These are quite similar to *GWigAx* and *GWigAy* subroutines which determine the vector potential. Both of them have three arguments where in the case of *GWigBx* are:

1. struct gwig *pWig: structure that holds the parameters of the insertion device.
2. double *Xvec: position of the electron.
3. double *pbx: output argument which holds the value of the x component of the magnetic field.

The only difference we find at *GWigBy* and *GWigBz* is that the last argument is called *pby and *pbz, respectively. We shall remember that the parameters of an insertion device are presented like:

$$\begin{aligned} B = [\; & 1, \; C_1, \; k_{x1}, \; k_{y1}, \; k_{z1}, \; \theta_1; \\ & 2, \; C_2, \; k_{x2}, \; k_{y2}, \; k_{z2}, \; \theta_2; \\ & 3, \; C_3, \; k_{x3}, \; k_{y3}, \; k_{z3}, \; \theta_3; \\ & \ldots] \end{aligned} \tag{8}$$

where C_{mn} are the relative amplitudes of wiggler harmonics, k_w is the wiggler frequency and θ_n is the relative phase of the nth wiggler harmonic. All this parameters have to accomplish the following conditions: $k_{ym}^2 = k_{xl}^2 + k_{zn}^2$, $k_{zn} = nk_w$ and $k_w = \frac{2\pi}{\lambda_w}$ where λ_w is the wiggler period. With this given we can determine the magnetic field for an horizontal and vertical planar wiggler. If we start with the **horizontal wiggler**:

$$\begin{aligned} \frac{B_y}{B_0} &= -\sum_{m,n} C_{mn} cos(k_{xl}x) cosh(k_{ym}y) cos(k_{zn}z + \theta_n) \\ \frac{B_x}{B_0} &= \sum_{m,n} \frac{C_{mn}k_{xl}}{k_{ym}} sin(k_{xl}x) sinh(k_{ym}y) cos(k_{zn}z + \theta_n) \\ \frac{B_z}{B_0} &= \sum_{m,n} \frac{C_{mn}k_{zn}}{k_{ym}} cos(k_{xl}x) sinh(k_{ym}y) sin(k_{zn}z + \theta_n) \end{aligned} \tag{9}$$

For obtaining the magnetic field of the **vertical wiggler** we have to change x to y and y to $-x$ in the above expressions. If we do it we obtain:

$$\begin{aligned}
\frac{B_x}{B_0} &= \sum_{m,n} C_{mn} cosh(k_{xl}x)cos(k_{ym}y)cos(k_{zn}z+\theta_n) \\
\frac{B_y}{B_0} &= -\sum_{m,n} \frac{C_{mn}k_{ym}}{k_{xl}} sinh(k_{xl}x)sin(k_{ym}y)cos(k_{zn}z+\theta_n) \\
\frac{B_z}{B_0} &= -\sum_{m,n} \frac{C_{mn}k_{zn}}{k_{xl}} sinh(k_{xl}x)cos(k_{ym}y)sin(k_{zn}z+\theta_n)
\end{aligned} \tag{10}$$

More information about the parameters of the wiggler can be found in Ref [1] and Ref [4].

3.4 *wigg_radloss* subroutine

Finally, this subroutine is the one that unify all the subroutines and functions that were introduced before. It has three arguments:

1. struct gwig *pWig: structure that holds the parameters of the insertion device.
2. double *X: initial position of the electron and at the end of the subroutine it has the new coordinates.
3. double L: length of the insertion device.

First of all this subroutine determines the magnetic field and the vector potential but we need both of them for different purposes. On one hand, the magnetic field is determined since it is needed for determining the radiation loss in *rad_loss* subroutine. On the other hand, vector potential is needed since we need to apply radiation loss to the relativistic kinetic momentum, not to the whole momenta. We shall remember that the definition of the momenta p_x and p_y is:

$$p_{x,y} = \frac{[\vec{K} + e\vec{A}(x,y,z)]\vec{e}_{x,y}}{p_0} \tag{11}$$

where e is the electron charge, $\vec{K}$ is the relativistic kinetic momentum and $\vec{A}$ is the vector potential [5]. Thus, before applying the radiation loss we have to subtract the vector potential. Then *rad_loss* is applied and later we add again the vector potential.

4 AT vs Elegant vs analytic results

In order prove that this method has a proper functioning we have run some simulations to compare this method to the one used in Elegant code. We have considered a simple wiggler that has analytic solution for comparing our results to the analytic ones. The parameters of our wiggler are:

1. Peak of the magnetic field: $B_{max} = 0.55T$
2. Length of the wiggler: $L = 0.60m$
3. Length of a period: $\lambda = 0.10m$
4. $B = [\ 1,\ \ 1.0,\ \ 0.0,\ \ 1.0,\ \ 1.0,\ \ 0.0\ ]$

If we plot the magnetic field generated by a wiggler of this parameters we obtain something like this:

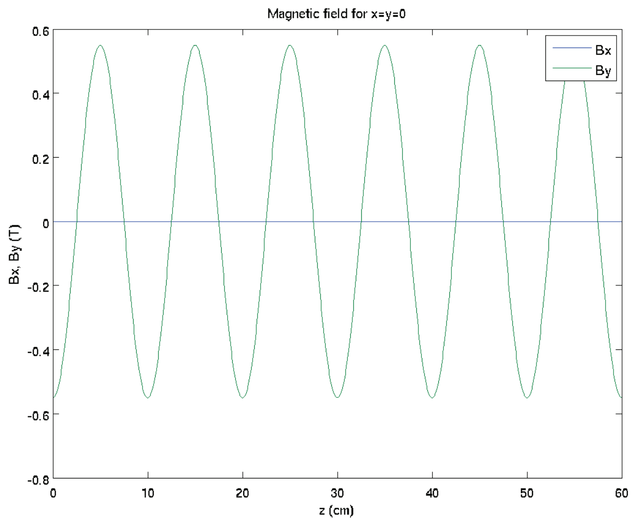


Figure 1: Magnetic field generated along the z axis.

We can clearly see how the magnetic field behaves as:

$$B_y(T) = -B_{max}\, cos\left(\frac{2\pi}{\lambda}z\right) = -0.55\, cos\left(\frac{2\pi}{0.10}z\right) \tag{12}$$

Since the equation of motion of an electron under paraxial approximation is given by:

$$\begin{aligned} \ddot{x} =& \frac{e}{\gamma m_0 c}(B_y - \dot{y}B_z) \\ \ddot{y} =& \frac{e}{\gamma m_0 c}(\dot{x}B_z - B_x) \end{aligned} \tag{13}$$

We can find that the trajectory of the electron will be:

$$\begin{aligned} x(z) = \frac{e}{\gamma m_0 c} B_0 \left(\frac{\lambda}{2\pi}\right)^2 \left[cos\left(\frac{2\pi}{\lambda}z\right) + 1\right] \\ y(z) = 0 \end{aligned} \tag{14}$$

where for obtaining this solution we have not included any change due to radiation loss.

4.1 Energy loss

Using eq. (4.13) of Sands [6] we can determine in an analytic way the energy loss. We will compare this value to the one obtained with AT and Elegant for the wiggler detailed before and for an electron moving along the ideal orbit. We can obtain:

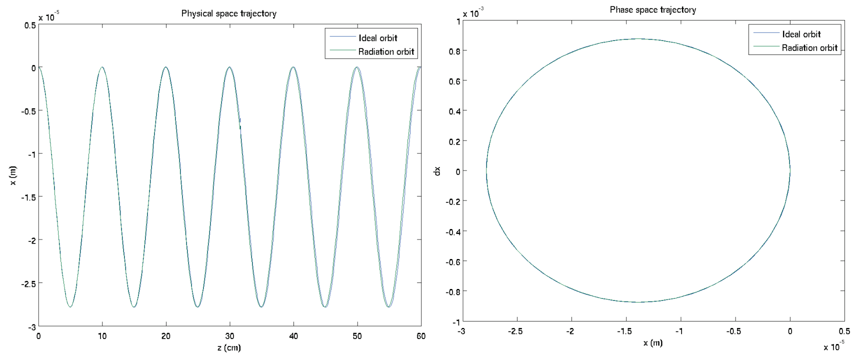
	Radiation loss			
	N$_{steps}$=2	N$_{steps}$=5	N$_{steps}$=15	N$_{steps}$=48
Sands	3.444335e-07	3.444335e-07	3.444335e-07	3.444335e-07
AT	6.965252e-07	3.444330e-07	3.444330e-07	3.444330e-07
Elegant	7.464168e-07	3.674669e-07	3.521557e-07	3.468925e-07

Table 1: Comparison of the energy loss determined by eq. (3) between different methods as function of the integration steps per period. In both methods the simulations have been run with a fourth order symplectic map.

We can clearly see how Elegant code has a slower convergence to the analytic result rather than AT method. This is due to the fact that Elegant code does more approximations than AT code. We shall remember that the analytic result has been obtained approximating the trajectory of the electron with the one that does not present radiation loss.

4.2 Physical space and phase space trajectory

Another way of proving the successful behaviour of AT method is comparing the physical space and phase space trajectory of an electron between this method and the analytic result. Of course there will be discrepancies since one of them does not consider radiation loss but in any way this could be a good test. Considering again an electron moving towards along the ideal orbit we obtain:



(a) Physical space trajectory (b) Phase space trajectory

Figure 2: Comparison between AT method and analytic results.

We can see how in both trajectories a small damping on the trajectory can be noticed when considering radiation loss in comparison to the one without it.

4.3 Initial condition dependence

Since Elegant code does more approximations than AT code it is interesting to compare both methods for different initial conditions. Some simulations have been run for the same wiggler than before and $Nsteps = 48$ in both cases. That is what obtained:

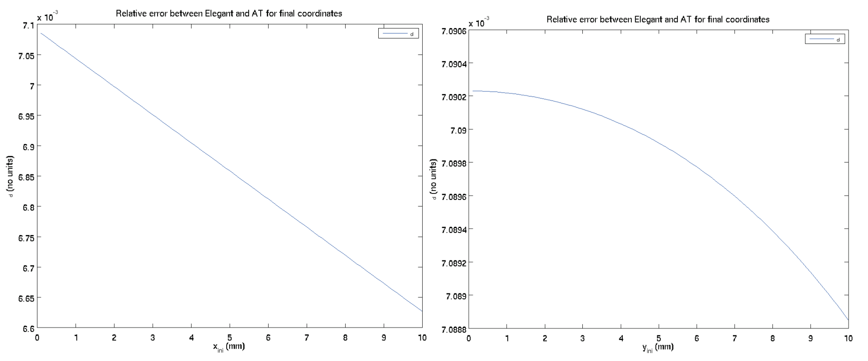


Figure 3: Energy relative error as function of different initial x and y coordinate positions.

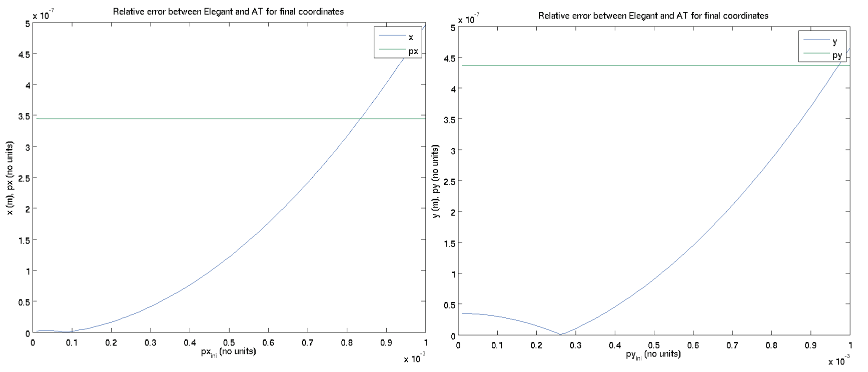


Figure 4: Position and momenta relative error as function of different initial x and y coordinate momenta.

We can clearly see how the differences in energy are lower than 0.1% and regarding position and momenta their relative error is lower than 10^{-5}%.

In addition, another pass method named ***GWigSymplectic3GeVRadElegPass*** has been created which works under the same approximations that Elegant does.

5 Conclusions

As a sum, it has been proven that from now on AT code has a pass method for an insertion device with radiation. The difference between this method and the one implemented in Elegant holds on the several approximations done in this last one.

References

[1] Y. K. Wu, E. Forest, and D. S. Robin. Explicit symplectic integrator for s-dependent static magnetic field. *Physical Review E 68, 046502*, 2003.

[2] Y. Cai. Simulation of synchrotron radiation in an electron storage ring. *SLAC-PUB-7793*, 1998.

[3] O. Jiménez and Z. Martí. Non-paraxial drift pass method. 2017.

[4] Y.K. Wu. Explicit symplectic integrator for generic insertion devices: implementation notes. 2001.

[5] K. Ohmi, K. Hirata, and K. Oide. From the beam-envelope matrix to synchrotron-radiation integrals. *Physical Review E 49, number 1*, 1994.

[6] M. Sands. The physics of electron storage rings: an introduction. 1970.

6 Annex

6.1 *rad_loss*: the code

```
void rad_loss(double *X, double B2P, double L, double irho, double E0)
{   /*This subroutine determines the radiation loss of the wiggler*/
    double TWOPI = 6.28318530717959;
    double CGAMMA = 8.846056192e-05;     /* m GeV-3 */
    double CRAD = CGAMMA*E0*E0*E0/(TWOPI*1e27);    /* m */
    double c = 299792458;    /* m s-1 */
    double e = 1;
    double dp_0, dDelta, Brho;

    dp_0 = X[4];
    Brho = E0/(e*c);
    dDelta = - CRAD*SQR(1+X[4])*B2P*(1 + X[0]*irho +
             + 0.5e0*(SQR(X[1])+SQR(X[3]))/SQR(1+X[4]))*L/SQR(Brho);
    X[4] = X[4] + dDelta;
    X[1] = X[1]*((1+X[4])/(1+dp_0));
    X[3] = X[3]*((1+X[4])/(1+dp_0));
}
```

6.2 *B2perp*: the code

```
double B2perp(double bx, double by, double bz, double *X, double E0)
{   /*Calculates sqr(|e x B|) , where e is a unit vector in the direction
of velocity*/
    double coef,vx,vy,vz,gamma,E,vmod;
    double mc2=0.510998928e6;      /* eV */
    double c=299792458;          /* m s-1 */
    E=E0*(1+X[4]);
    gamma=E/mc2;
    vmod=c*sqrt(1-1/(gamma*gamma));
    coef=(c*sqrt(E0*E0-mc2*mc2))/(mc2*gamma);
    vx=X[1]*coef;
    vy=X[3]*coef;
    vz=sqrt(vmod*vmod-vx*vx-vy*vy);
    return((SQR(bz*vy-by*vz) + SQR(bx*vz-bz*vx) + SQR(bx*vy-by*vx))/SQR(vmod));
}
```

6.3 *GWigBx*: the code

```
void GWigBx(struct gwig *pWig, double *Xvec, double *pbx)
{
  int       i;
  double x, y, z, Cmn;
  double kx, ky, kz, tz;
  double sx, shy, cz;
  double chx, cy;
  double bx, BMAX;

  x = Xvec[0];
  y = Xvec[2];
  z = pWig->Zw;
  BMAX = pWig->PB0;
  bx    = 0e0;

  /* Horizontal Wiggler: note that one potentially could have: kx=0 */
  for (i = 0; i < pWig->NHharm; i++) {
    Cmn= pWig->HCw_raw[i];
    kx = pWig->Hkx[i];
    ky = pWig->Hky[i];
    kz = pWig->Hkz[i];
    tz = pWig->Htz[i];

    sx  = sin(kx * x);
    shy = sinh(ky * y);
    cz  = cos(kz * z + tz);
    bx  = bx + BMAX*(Cmn*kx/ky)*sx*shy*cz;
  }
  /* Vertical Wiggler: note that one potentially could have: ky=0 */
  for (i = 0; i < pWig->NVharm; i++ ) {
    Cmn= pWig->VCw_raw[i];
    kx = pWig->Vkx[i];
    ky = pWig->Vky[i];
    kz = pWig->Vkz[i];
    tz = pWig->Vtz[i];

    chx = cosh(kx * x);
    cy  = cos(ky * y);
    cz  = cos(kz * z + tz);
    bx  = bx + BMAX*Cmn*chx*cy*cz;
  }
  *pbx    = bx;
}
```

6.4 *GWigBy*: the code

```
void GWigBy(struct gwig *pWig, double *Xvec, double *pby)
{
  int      i;
  double x, y, z, Cmn;
  double kx, ky, kz, tz;
  double cx, chy, cz;
  double shx, sy;
  double by, BMAX;

  x = Xvec[0];
  y = Xvec[2];
  z = pWig->Zw;
  BMAX = pWig->PB0;
  by    = 0e0;

  /* Horizontal Wiggler: note that one potentially could have: kx=0 */
  for (i = 0; i < pWig->NHharm; i++) {
    Cmn= pWig->HCw_raw[i];
    kx = pWig->Hkx[i];
    ky = pWig->Hky[i];
    kz = pWig->Hkz[i];
    tz = pWig->Htz[i];

    cx  = cos(kx * x);
    chy = cosh(ky * y);
    cz  = cos(kz * z + tz);
    by  = by - BMAX*Cmn*cx*chy*cz;
  }
  /* Vertical Wiggler: note that one potentially could have: ky=0 */
  for (i = 0; i < pWig->NVharm; i++ ) {
    Cmn= pWig->VCw_raw[i];
    kx = pWig->Vkx[i];
    ky = pWig->Vky[i];
    kz = pWig->Vkz[i];
    tz = pWig->Vtz[i];

    shx = sinh(kx * x);
    sy  = sin(ky * y);
    cz  = cos(kz * z + tz);
    by  = by - BMAX*(Cmn*ky/kx)*shx*sy*cz;
  }
  *pby   = by;
}
```

6.5 *GWigBz*: the code

```
void GWigBz(struct gwig *pWig, double *Xvec, double *pbz)
{
  int     i;
  double x, y, z, Cmn;
  double kx, ky, kz, tz;
  double cx, shy, sz;
  double shx, cy;
  double bz, BMAX;

  x = Xvec[0];
  y = Xvec[2];
  z = pWig->Zw;
  BMAX = pWig->PB0;
  bz    = 0e0;

  /* Horizontal Wiggler: note that one potentially could have: kx=0 */
  for (i = 0; i < pWig->NHharm; i++) {
    Cmn= pWig->HCw_raw[i];
    kx = pWig->Hkx[i];
    ky = pWig->Hky[i];
    kz = pWig->Hkz[i];
    tz = pWig->Htz[i];

    cx  = cos(kx * x);
    shy = sinh(ky * y);
    sz  = sin(kz * z + tz);
    bz  = bz + BMAX*(Cmn*kz/ky)*cx*shy*sz;
  }
  /* Vertical Wiggler: note that one potentially could have: ky=0 */
  for (i = 0; i < pWig->NVharm; i++ ) {
    Cmn= pWig->VCw_raw[i];
    kx = pWig->Vkx[i];
    ky = pWig->Vky[i];
    kz = pWig->Vkz[i];
    tz = pWig->Vtz[i];

    shx = sinh(kx * x);
    cy  = cos(ky * y);
    sz  = sin(kz * z + tz);
    bz  = bz - BMAX*(Cmn*kz/kx)*shx*cy*sz;
  }
  *pbz    = bz;
}
```

6.6 *wigg_radloss*: the code

```
void wigg_radloss(struct gwig *pWig, double *X, double L)
{       /*This subroutine applies the radiation loss*/
        double B2P,bx,by,ax,ay,axpy,aypx,irho;
        double E0 = pWig->E0*1e9;       /* eV */
        double c = 299792458;       /* m s-1 */
        double e = 1;
        GWigBx(pWig, X, &bx);
        GWigBy(pWig, X, &by);
        GWigAx(pWig, X, &ax, &axpy);
        GWigAy(pWig, X, &ay, &aypx);
        B2P=B2perp(bx, by, X, E0);
        irho=e*c*sqrt(B2P)/(E0*(1+X[4]));
        X[1] -= ax;
        X[3] -= ay;
        rad_loss(X, B2P, L, irho, E0);
        X[1] += ax;
        X[3] += ay;
}
```