CSR elements in GPT

For future applications of high-brightness electron beams, including the design of next generation FEL's, correct simulation of Coherent Synchrotron Radiation (CSR) is essential as it potentially degrades beam quality to unacceptable levels. However, the long interaction lengths compared to the bunch length, numerical cancellation, and difficult 3D retardation conditions make accurate simulation of CSR effects notoriously difficult. To ease the computational burden, CSR codes often make severe simplifications such as an ultra relativistic bunch travelling on a prescribed reference trajectory. Here we report on a new CSR model, implemented in the General Particle Tracer (GPT) code [1], that avoids most of the usual assumptions: It directly evaluates the Liénard–Wiechert potentials based on the stored history of the beam. It makes no assumptions about reference trajectories, while also taking into account the transverse size of the beam.

Table of Contents

1. Description

1.1 Introduction

Here we add a new alternative to the list of simulation codes for CSR effects. Our version is implemented in the GPT code, and we intentionally steered away from the approaches used by other codes such as Elegant [2,3]. The goal we had in mind was not to create the 'best' CSR code per-se; we wanted to make a model that could answer the question what the consequences are of the assumptions made by the already exiting codes. In order to do so, we created a code that is based on direct evaluation of the retarded Liénard–Wiechert potentials.

The main approximation used in this GPT element is that the bunch is modeled as a parametric curve, sliced in the direction of average propagation. This is typically a very good approximation because the dynamics of the underlying process are governed by the rest-frame properties of the bunch. In this frame, the bunch is elongated by the Lorentz factor compared to the length in the laboratory frame, resulting in a very narrow ribbon of charge. In this GPT model, the ribbon itself is allowed to meander through 3D space, capturing the largest amount of 3D effects possible within the parametric 1D framework. The chosen approach makes use of the fact that if all individual beam segments that have radiated in the past are stored, this information is sufficient to calculate the electromagnetic fields everywhere in 3D space, at any later point in time.

The result of the chosen approach is that changes in bunch shape between the point where radiation is emitted and the interaction point are properly taken into account. Another advantage is that the usual ultra relativistic assumptions are not needed. A conceptual simplification compared to some other codes is that the radiation is emitted based on the derivative of the momentum, regardless of the reason: A very strong electrostatic deflection field will give rise to the same emitted radiation as a bend-magnet or a heavily misaligned quadrupole as long as the trajectories are the same. Furthermore, because the element is based on electromagnetic fields and not 'effects', radiation fields and the fields of other beamline components can all be correctly added. This allows for beamline components very close to the exit of a bend magnet, where a strong CSR wake can be travelling almost parallel to the bunch over long distances, to be properly simulated without degrading the accuracy of the CSR calculations. Finally, both the transverse and longitudinal fields are calculated, taking into account not only energy changes but also transverse effects that might vary along the bunch causing projected emittance growth.

The implementation of this element consists of two main parts: A history manager that maintains the history of the bunch, and a field calculator that integrates the radiation fields based on this stored history. They will be described in the following sections. We end by showing first simulation results, followed by a conclusion.

1.2 History manager

The history manager of our new CSR code stores at discrete time intervals a 1D parameterization of the entire bunch, including basic information about transverse size. The first step is to switch to a new coordinate system, where **u** is the average direction of propagation, **v** lies in the bend-plane, and **w**=**u** x **v**. In this frame, an estimate of the current profile is obtained by slicing in **u**, and this in turn defines the non-equidistant charge quantiles of the beam. The recipe is shown schematically in Figure 1.

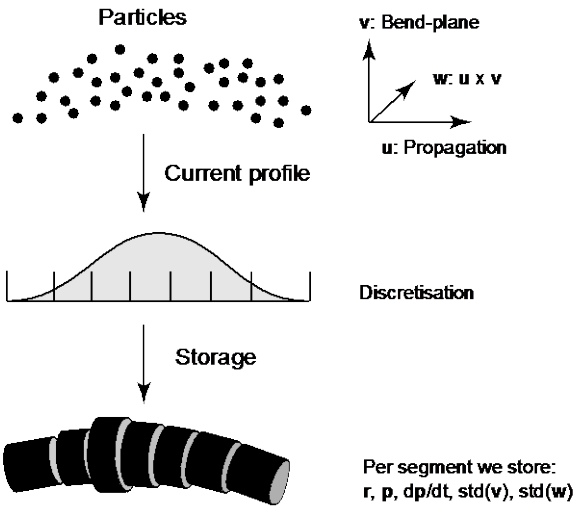

Figure 1: Schematic of the CSR history manager.

For each segment the total charge, average position, average momentum, average acceleration and transverse beam sizes are stored. As we will see, this information is all that is needed to calculate the CSR field at any position, at any later point in time.

When running simulations with large Lorentz factors and relatively smooth fields, it is possible that the average momentum changes direction over angles (much) larger than $1/\gamma$ during one timestep. This potentially results in cases where emitted radiation is partially missed due to relativistic beaming, also known as the headlight effect. In that case the history manager forces the Runge-Kutta integrator of GPT to reduce timesteps.

1.3 Field calculation

At each Runge-Kutta substep in the GPT simulations, the electromagnetic radiation emitted in the past is reconstructed at the current bunch position, as schematically shown in Figure 2. The first step in this process is making a 1D parameterization of the bunch based on non-equidistant charge quantiles of the beam, fully analogous to the history manager. Subsequently, the centers $\mathbf{r}_i$ of each slice *i* are calculated, thereby avoiding the need for a reference path or trajectory and allowing significant differences in trajectories between the head and the tail of the bunch.

The crucial step in the CSR field calculation is calculating the actual electromagnetic **E** and **B** fields at the current timestep at each of these $\mathbf{r}_i$, by summing the Liénard-Wiechert potentials [4] for all segments $\mathbf{r}_s$ that fulfil the retardation condition $|\mathbf{r}-\mathbf{r}_s|=c\,\Delta t$:

$$\mathbf{E}(\mathbf{r},t) = \sum_{segments} \frac{q_s}{4\pi\varepsilon_0}\left(\mathsf{K} + \frac{\mathbf{n}\times\left((\mathbf{n}-\boldsymbol{\beta})\times\dot{\boldsymbol{\beta}}\right)}{c(1-\mathbf{n}\cdot\boldsymbol{\beta})^3\sqrt{|\mathbf{r}-\mathbf{r}_s|^2+R^2}}\right)_{tr}$$

$$\mathbf{B}(\mathbf{r},t) = \sum_{segments} \frac{\mathbf{n}(t_r)}{c}\times\mathbf{E}(\mathbf{r},t)$$

where

$$\mathbf{n}(t) = \frac{\mathbf{r}-\mathbf{r}_s(t)}{|\mathbf{r}-\mathbf{r}_s(t)|}$$

with $\beta=v/c$, and where q_s is the charge of the segment and where Δt is the time difference between the current timestep and the timestamp of the stored profile. When only a fraction of a stored segment fulfils the retardation condition, only the corresponding fraction of the charge in that segment is used in the equations. The … in the given equation denotes the Coulomb term, that will be discussed later.

In order to include transverse effects, we model each emitting slice by four (or sixteen) independent points, located at plus or minus 1 standard deviation off-axis. This gives us 4 (or 16) times more emission points $\mathbf{r}_s$ than segments, and because the bunch travels on a curved trajectory the retardation conditions for these points have to be individually evaluated. We observed that with only four point we are able to capture relevant 3D effects at a

relatively low price and increasing this number to 16 did not show any significant differences for our initial test-cases.

The evaluated fields are singular when evaluated at $\mathbf{r}=\mathbf{r}_s$, but under normal circumstances this should not happen since a) the receiver's position $\mathbf{r}$ is on-axis and the emitting positions $\mathbf{r}_s$ are off-axis, and b) the bunch has moved forward since the last stored profile. Nevertheless, we add a tiny R in the denominator, analogous to the concept of Plummer spheres commonly used in the the astrophysical community, as safeguard.

The main challenge writing this CSR model was to prevent interference problems between consecutive interpolations steps. This problem was solved by choosing low-order interpolations to prevent undesired oscillations and using quantiles instead of equidistant arrays to prevent artificial spikes in the Fourier transformed solution. What remains is the challenge that the integrand, tracing 'vertically' back in time, always has two large peaks of opposite sign that almost cancel when integrated. That however is the nature of CSR interactions when working directly with the LW-potentials.

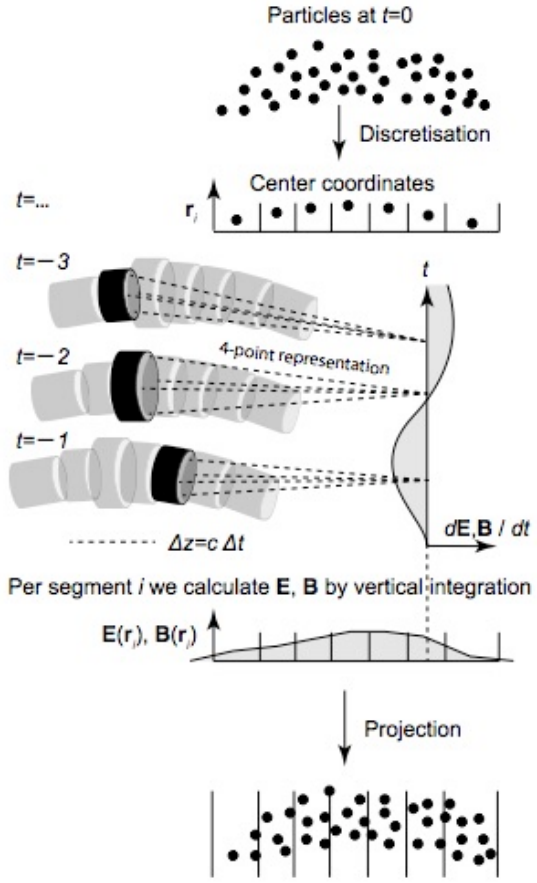

Figure 2: Schematic of field integration.

1.4 Limitations

Every simulation code has sweets spots in parameter space where it works best, gray areas preferably to be avoided, and no-go zones. In this section we describe the limitations of our code, and offer potential remedies and outlooks where applicable.

Slicing the bunch in the average direction of propagation is problematic in the case of roll-over compression. However, because the slicing problem only happens during very short timescales we are at this point not able to judge how severe the integrated error actually is.

The current implementation includes 3D effects for the emission process by having off-axis emission points. An analogous scheme is however not implemented for the receiving part, resulting in CSR fields that are independent on the transverse coordinate.
Adding the Coulomb term of the LW potentials to the interaction results in a r^{-2} singularity that sometimes causes numerical instabilities for bunches with very small transverse dimensions. The code has the option to switch off the Coulomb term in case this is needed or desired.

The current version does not allow for shielding effects. It is foreseen that this will be added using mirror bunches, at the expense of required computing power.

Due to relativistic beaming, the implemented method is not a good match for multi GeV beams and large deflection angles. It will work, but it will be slow.

2. Test-case

As test case for the new CSR model in GPT we used the settings as proposed at the DESY 2002 workshop about Coherent Synchrotron Radiation [5]. The chosen scenario consists of a 1 nC, 1 µm emittance, 500 MeV beam sent through a four-magnet chicane, while being compressed longitudinally from 200 to 20 µm. These parameters are still relevant for designs today, although the charge is arguably a bit high.

2.1 GPT inputfile

Given below in listing 1 is the GPT inputfile **`CSRdemo.in`** for the DESY 2002 workshop consisting of four rectangular chicanes. We would like to mention the following:

- For performance reasons we add fringe fields to all magnets. The on-axis fields can be plotted by commenting the initial particle distribution and **`screen`** statemement, and uncommenting the **`setstartline`** and **`tout(0)`** at the very end of the inputfile.
- The initial particle distribution is defined in the workshop description using Courant-Snyder parameters at the entrance plane of the first magnet. This is problematic since starting particles halfway the fringe-field is not a good idea. The initial particle distribution is therefore analytically extrapolated 10 mm upstream the magnet using the **`setextrapolate`** keyword.
- A density oscillation is added using **`addzoscillation`** to show the microbunching capabilities of the code. The amplitude is set at **`amp`**, to be specified later on the command-line when running GPT.
- We add **`writedG`** so that it is conveniently plot the energy loss per particle as function of longitudinal position.

Listing 1: GPT inputfile for the CSRdemo.

```
# GPT inputfile for DESY 2002 workshop
#
# Author: Bas van der Geer

# CSR Interactions
csr1d("NoCoulomb") ;

# Description based on Twiss parameters
numberparticles = 8e6 ;

totalcharge = -1.0e-9 ;
meanEnergy = 0.5e9 ;
stdEnergy = 10e3 ;
aEnergy = 36 ;

betaX = 40;
betaY = 13;
alphaX = 2.6;
alphaY = 1.0;
epsX = 1e-6;
epsY = 1e-6;
sigma_s = 200e-6;
start_s = -10e-3 ;

# Settings
E0 = 5e9 ;
gamma = 1+|qe|*meanEnergy/(me*c*c) ;
gammabeta = sqrt(gamma^2 - 1) ;
beta = sqrt(1-1/gamma^2);

# Particles
setparticles("beam",numberparticles,me,qe,totalcharge);
setxdist("beam","g",0,sqrt(epsX*betaX/gamma),5,5);
setydist("beam","g",0,sqrt(epsY*betaY/gamma),5,5);
setzdist("beam","g",0,sigma_s,5,5);

addzoscillation("beam",amp,25/sigma_s) ;

```

```
setGdist("beam","g",gamma,|qe|*stdEnergy/(me*c*c),5,5);
setGBxdist("beam","g",0,gamma*beta*sqrt(epsX/betaX/gamma),5,5);
setGBydist("beam","g",0,gamma*beta*sqrt(epsY/betaY/gamma),5,5);
addxdiv("beam",0,-gamma*beta*alphaX/betaX);
addydiv("beam",0,-gamma*beta*alphaY/betaY);
addzdiv("beam",0,-(me*c*c/|qe|)*(gamma-1)*aEnergy) ;

# Start a little bit before the first magnet (how far is actually important)
setextrapolate("beam",start_s/c) ;

# --Geometry
Lb = 0.5 ; # Magnet length
L0 = 5.0 ; # Drift length B1-B2 and B3-B4
Li = 1.0 ; # Drift length B2-B3
Lf = 2.0 ; # Drift after B4
R  = 10.35 ; # Bend radius

zB1 = Lb/2 ;               # Center position B1
zB2 = zB1 + L0 + Lb ;      # Center position B2
zB3 = zB2 + Li + Lb ;      # Center position B3
zB4 = zB3 + L0 + Lb ;      # Center position B4
zf  = zB4 + Lb/2 + Lf ;    # Final screen

BfieldA = gammabeta *me*c/(|qe|*R) ;

dl = 0 ;
gap = 5e-3 ;
b1 = 2/gap ;
b2 = 0 ;

rectmagnet("wcs","z", zB1, 10,Lb, +BfieldA,dl,b1,b2) ;
rectmagnet("wcs","z", zB2, 10,Lb, -BfieldA,dl,b1,b2) ;
rectmagnet("wcs","z", zB3, 10,Lb, -BfieldA,dl,b1,b2) ;
rectmagnet("wcs","z", zB4, 10,Lb, +BfieldA,dl,b1,b2) ;

# --Output
zminmax("wcs","I",-1,15.001) ;
accuracy(6) ;
dtmax = Lb/c ;

screen("wcs","I",zf) ;
writedG() ;

# --Field profile
#setstartline("profile",100000, 0,0,-1, 0,0,zf+1) ;
#tout(0) ;
```

2.2 Running GPT with CSR

Typically, the CSR version of GPT runs on large Linux clusters using job schedulers such as PBSpro. A script is typically submitted using **`qsub CSRdemo.sh`** but we observe that every Linux cluster has its own taste for details. Please find below in listing 2 an example PBSpro submission script, where we would like to emphasise that *every* system is different and that modifications with local knowledge are always needed.

- The **`-l`** option specifies the resources needed, such as number of nodes, sockets, cores, memory etc. Please see the PBSpro documentation on your system to see how resources need to be allocated. Different versions of PBSpro require a different syntax, and PBSpro's terminology in terms of cpus is what most people would describe as cores.
- The **`-A`** (for accounting) and **`-q`** (for queue specification) options might be essential on your system, but on other systems they can be omitted.
- Often the modules environment must be loaded, even if it is default present on the interactive nodes. On every system the required line is different, here it is: **`source /usr/share/Modules/init/bash`**
- For CSR simulations it is best to launch the same amount of GPT executables as cores reserved. This might be the default, but here we force this by using the **`-np 64`** option in **`mpirun`**. Please note that each GPT executable runs single-core, enforced using the **`-j 1`** option.
- We switch off the Coulomb term using the **`NoCoulomb`** parameter since the demo involves relatively high energy and relatively small transverse dimensions.

- The additional parameter **amp=0.02** is added to the command-line, since it is used in the **CSRdemo.in** inputfile but not set so far.

Listing 2: PBSpro submission script **CSRdemo.sh**

```
#!/bin/bash

#PBS -l select=4:ncpus=16
#PBS -c n
#PBS -A ??
#PBS -N CSRdemo
#PBS -q ??
#PBS -l walltime=24:00:00
#PBS -o CSRdemo.log
#PBS -j oe
#PBS -m abe

cd $PBS_O_WORKDIR
rm CSRdemo.log
echo "================"
echo "PBS queue:" $PBS_QUEUE
cat $PBS_NODEFILE
echo "================"

export GPTLICENSE=?????????
export GPTHOME=/u/home/svdgeer/gptrelease

# Load modules
source /usr/share/Modules/init/bash
source ${GPTHOME}/modules.sh
module list
echo "================"

PATH=${GPTHOME}/bin:$PATH
time mpirun -np 64 gpt -j 1 -v -o CSRdemo.gdf CSRdemo.in amp=0.02
```

2.3 Results

The main result, the longitudinal phase space at the exit plane, is shown in Figure 3. It was obtained with 8M macro-particles and took 3.5 hours wall-clock time using 64-cores on a Linux cluster. It shows the well-known CSR-signature, where particles at the front are slightly accelerated. The average energy loss is 0.34%, being reasonably close to Elegants value of 0.42%. The main message of this paper is not a detailed comparison between GPT and other simulation codes: It is merely a demonstration that the method is available and is ready for testing.

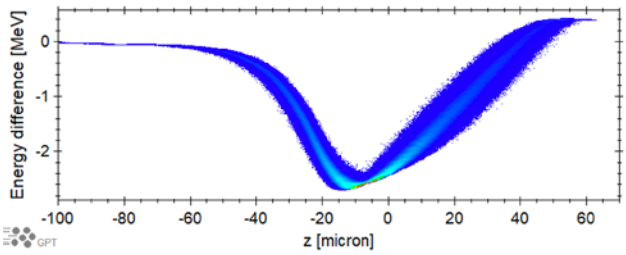

Figure 3: Raw GPT simulation result at the target plane at z=15 m. Head of the bunch is at the right.

To test the capabilities of the model in terms of numerical instabilities, we varied the initial density profile. Shown in figure 4 are the results of 1%, 2% and 5% initial density variation at fixed k_z=25/sigma. Because CSR is sensitive to the derivative of the current profile the expected result is a fluctuation in final energy [6]. Figure 4 clearly shows this effect, where we would like to emphasize that our from-first-principles approach yields this result without numerically taking the derivative of the current profile and thereby avoiding all related numerical issues.

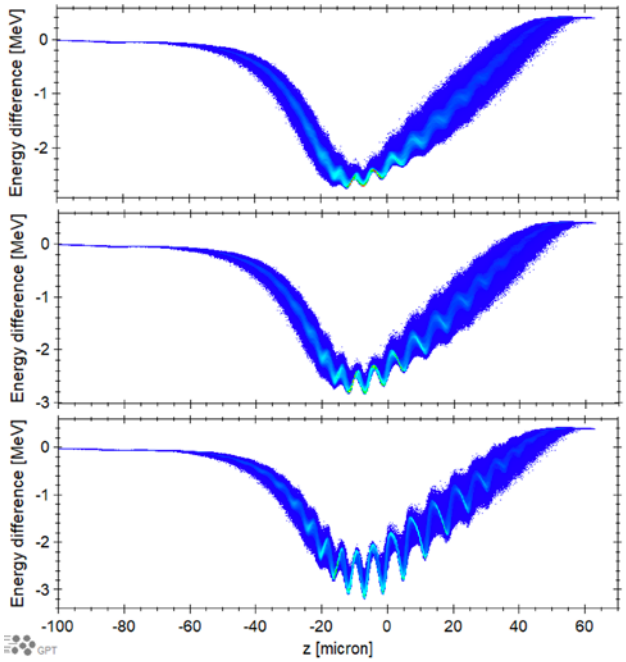

Figure 4: GPT simulation results based on the DESY 2002 workshop settings, with initial density fluctuations of 1% (top), 2% (middle) and 5% (bottom).

2.4 Conclusion

We created a new CSR model in the GPT code, based on evaluating retarded Liénard-Wiechert potentials from-first-principles. The model does not assume a reference trajectory, allows the head and tail of the bunch to be on different tracks, takes into account the transverse size of the bunch for the emission process, makes no ultrarelativisitc approximations, and correctly takes into account changes in the current profile between emission and interaction with emitted radiation. The code runs on MPI Clusters and shows consistent results for the testcase of the DESY 2002 workshop about CSR. Adding a small density fluctuation to the initial particle distribution shows that the code has sufficient numerical precision for microbunching simulations.

3. Technical reference

3.1 CSR1D

`csr1d([options,...]) ;`
Include 1D tail-to-head CSR fields based on retarded Liénard–Wiechert potentials.

`options` A list of options that affect the model

This element includes the fields of the retarded Liénard–Wiechert potentials, evaluated using the stored history of the beam. The basic simplification this model makes is that the interaction is not calculated between all particles individually, but between longitudinal segments. The segments themselves are stored with full 3D coordinates, and also the fields are all calculated in 3D. Four (or sixteen) off-axis emission points are used to take into account the transverse size of the bunch.

Included in the model are:
- Radiation passing through a series of beamline components.
- High/low-energy tails of a bunch traveling on a different trajectory.
- Correct retardation conditions based on actual trajectories.
- Multiple off-center emission points to estimate transverse effects.
- Coulomb fields with correct retardation conditions.
- Transverse forces (both B and E perpendicular).
- Change in current profile between emission and reception of radiation. (no rigid bunch approximation).
- Low energy beams (no ultrarelativistic approximations).

Not included in the model are:
- Transverse dependence of the CSR field.
- Shielding or pipe effects.

The code is implemented using a combination of openMP and MPI and performs well on both multi-core and/or multi-node machines. Because the model calculates electromagnetic *fields*, not forces, the normal time output of GPT can be used to plot these fields. The implementation consists of almost two almost independent parts:
- The history manager that stores the history of the beam
- The field evaluator that calculates the fields in the current timestep based upon the stored history

Settings and options for these parts of the code are listed below.

3.1.1 Combined options

Both the history manager and the field evaluator work with discretized 1D-profiles of the beam. The internal settings to create these profiles are shared between the two parts of the code, and their common settings are given below:

`"MeshNfac",Nfac` Default 1.
`"MeshNpow",Npow` Default 1/3
The total number of segments N_{seg} in the 1D profile is given by the equation N_{segs}=**`Nfac`** * $N^{\texttt{Npow}}$ where N is the total number of particles. It is tempting to try to work with more line segments in an attempt to get better spatial resolution, but this comes at a price in terms of more numerical noise. Because this noise can be amplified due to the nature of CSR interactions there is not much room for improvement.

`"MeshAdapt",adapt` Default 0.5
The length of the 1D-segments is adjusted to 1D line charge density raised to the power **`adapt`**. Consequently, setting **`adapt`** to **`0`** creates segments of equal length while a value of **`1`** gives segments that are two times longer when the current is half. The default value of **`0.5`** seems to work reasonably well over a large parameter regime,

but when CSR effects in the low-density tails of the distribution are important it can be necessary to reduce **adapt** in order to create relatively smaller segments in the tails.

3.1.2 History manager

The history manager stores position, velocity and acceleration information for all segments at all timesteps. This information is subsequently used in the field evaluator to calculate the fields. A few options can be given to affect when and how the information is stored.

"Mincurvature",value No default
When the **"NoCoulomb"** option is given, see below, there is no need to store the beam information at straight sections. This **value** defines the threshold [m] for the radius of curvature for historical beam information to be stored. If this option is not specified, history information is always stored.

"CSRprecision",value Default 3
This option enforces that the beam is stored minimally **value** times during the time that the beam makes a direction change equal to the opening angle of the radiation cone. Especially at high energies this is essential because the Ruge-Kutta integrator would by default take far larger stepsizes resulting in crucial information not stored. The default value works reasonably well over a large parameter range, but it might be needed to increase this value for microbunching simulations.

"Points",value Default 4
Off-axis beam information is stored by positioning **value** emission points per line-segment around the beam-center. The transverse position of these four points corresponds to the rms-width of the beam. Currently the only **values** implemented are 4 and 16. The idea is that if 4 versus 16 gives very different results, the 1D model is not applicable.

3.1.3 Field evaluator

The field evaluator has a number of options that to switch on and off certain parts of the Liénard–Wiechert potentials. Default all components are evaluated.

"NoCoulomb"
Do not include the Coulomb term of the LW potentials.

"NoRadiation"
Do not include the Radiation term of the LW potentials

"NoTransverseFields"
Only calculate the electric field component in the direction of the velocity of the receiving particles. Please note that switching off the magnetc field and only calculating one component of the Electric field does not offer any performance gain, it's for debugging purposes only.

"FieldFactor",value
Multiply all fields with the given **value**. This option can be used if for some reason the total charge cannot be adjusted, while it is nevertheless desirable to affect the overall amplitude of the fields.

3.2 writedG

cwritedG() ;
Writes energy difference to the outputfile.

Adding this element to the GPT inputfile writes additional per-particle information to the GPT outputfile for each **tout** and **screen** output. Written is the following column:

dG Difference in Lorentz factor between the initial particle distribution and the current value.

References

[1] General Particle Tracer (GPT) code, Pulsar Physics, http://www.pulsar.nl/gpt

[2] M. Borland. *Elegant: A flexible SDDS-compliant code for accelerator simulation*. Proceedings of ICAP'00, Darmstadt, Germany, 2000.

[3] E. L. Saldin, E. A. Schneidmiller, and M. V. Yurkov. *On the coherent radiation of an electron bunch moving in an arc of a circle*. Nucl. Instrum. Meth. A, 398(2):373 – 394, 1997.

[4] J. D. Jackson. Classical Electrodynamics. Wiley, 1962

[5] http://www.desy.de/csr/

[6] E. L. Saldin, E. A. Schneidmiller, and M. V. Yurkov. *Longitudinal space charge-driven microbunching instability in the TESLA Test Facility linac*. NIM-A, 528(1):355, 2004.