Electron Cloud Lattice Element Manual v0.1

December 2, 2014

Contents

1 The Lattice Element **2**

2 Modes of Operation **2**
- 2.1 Self-Contained . . . 2
- 2.2 Read Density . . . 3
- 2.3 Snapshot . . . 3
- 2.4 Fully Simulated . . . 3

3 EM Field Calculations **4**
- 3.1 Self-Contained . . . 4
- 3.2 Read Density . . . 4
- 3.3 Snapshot . . . 5
- 3.4 Fully Simulated . . . 6

4 Bugs and Caveats **6**

5 Appendix **7**
- 5.1 Trends . . . 7
  - 5.1.1 Ecloud . . . 7
  - 5.1.2 Slopes . . . 7
  - 5.1.3 Intercepts . . . 7
- 5.2 Plot Section . . . 8
- 5.3 To Be Implemented . . . 8

1 The Lattice Element

The Electron Cloud Lattice Element is a custom EM field calculation method made for the *Bmad* accelerator simulation library to simulate the effects of an electron cloud on a beam of particles.

The Lattice Element had several goals throughout production, though is primary purpose is to affect the particles which pass through it as though they were passing through a continuous electron cloud in as many ways as *Bmad* will allow, it is also made to be faster than full scale simulations of the beam (as offered in *Cmad*), and flexible enough to adjust to small changes in beam settings[1] without requiring long, repeated simulations for each minor adjustment.
Beam effects of interest provided by the element:

- Tune Shifts
- Orbit Changes
- Bunch Deformations

2 Modes of Operation

The Lattice Element reads custom attributes from the lattice file. The first custom attribute is interpreted as the Mode of Operation, or *calc_method*. This is a number between (inclusive) 1 and 4 and are described in the subsections below. Different calculation methods make use of the remaining custom attributes in different ways. At the beginning of each subsection I will provide suggested names for the remaining attributes as well as a brief description. [only mode 3 currently operational]

Example of setting operation mode in the lattice file:

```
cloud: ... <make cloud element>
parameter[custom_attribute1] = &
  "CLOUD::CALC_METHOD"
parameter[custom_attribute2] = &
  "CLOUD::<OTHER NAME>"
cloud_element[calc_method] = ...
```

[1]The flexibility of the element to changes in the beam and lattice is subject to the mode of operation as well as user discretion, see Section 4 for details

2.1 Self-Contained

Description: Setting *calc_method* to 1 enables the self-contained operation mode. The purpose of this mode is to use no outside simulations as input while still providing accurate beam adjustments.

Table 1: Custom Attributes

Attribute	Name
1	calc_method
2	photoelectrons per particle
3	decay coefficient for cloud
4	cavity volume

Attribute 2: The number of photoelectrons per particle created as that particle passes through the cloud element. In reality this varies with the beam energy and material used in the walls of the accelerator, here you are asked to average those to make an estimate ignoring secondary yield electrons.

Attribute 3: The decay coefficient of the cloud changes the rate at which the electrons from the cloud return to the chamber walls. Since the calculation method here is unable to track trapping, low energy electrons which would normally get caught in magnetic traps will return to the walls of the chamber like their high energy counterparts. This mechanic, as well as the exact nature of the coefficient, will be changed and this description will be updated once fully implemented.

Attribute 4: The volume of the chamber is the V in the equation for the electron cloud density:

$$\rho_e = \frac{n_e}{V}$$

where n_e is the total number of photoelectrons.

Advantages: This method of operation allows for rapid testing. There is no need to run any lengthy simulations for input and changes to the attributes and beam govern the entire output of the element. It is therefore quite flexible and fast.

Disadvantages: However, it cannot deal with more complicated issues such as electron cloud distributions which arise from trapping and asymmetries. Likewise, it cannot yet calculate secondary yield elec-

tron emissions from high energy photoelectrons striking the walls.

2.2 Read Density

Description: Setting *calc_method* to 2 enables the Read Density operation mode. The purpose of this mode is to use just a text file list of electron cloud average densities to apply to the beam. The cloud element manages EM field calculations based off the cloud density, but requires further simulations for changes to the beam which would alter the density or cause significant cloud heterogeneity (such as large bunch offsets).

Table 2: Custom Attributes

Attribute	Name
1	calc_method
2	reset time
3	cloud multiplier

Attribute 2: The reset time of the element defines the maximum time between particles before the element considers the new particle to be the head of a new train, thus returning to the first values for the cloud density as read from the file.
Example: If a particle is tracked through the cloud element with a Δt (between itself and the previously tracked particle) less than the element's reset time, it will experience an EM field from the electron cloud with a density extrapolated from the last two points in the file. If, however, the Δt is greater than the reset time, it will instead experience the cloud with a density as it was at time 0.
A recommended reset time for a single-turn series of cloud densities is approximately how long it would take the electron cloud to mostly decay from the passage of a train, but before the arrival of the next train (or $(\#bunches + 1) \times spacing$)

Attribute 3: Cloud Multiplier simply refers to a number by which to multiply all densities in the cloud. This is useful for shortening the element to maintain the same EM field effects (i.e. E over 2m = 2E over 1m).

Advantages: This method is able to deal with secondary yield electrons from simulations as well as simple manual changes (such as a constant cloud density).

Disadvantages: Requires cloud density output from simulations such as *Ecloud* to operate, still cannot deal with heterogeneities from trapping or offsets.

2.3 Snapshot

Description: Setting *calc_method* to 3 enables the Snapshot operation mode. It uses the EM fields gathered from a single bunch passing through an electron cloud and applies it to each bunch, scaled by the electron cloud density at that point in the train.

Table 3: Custom Attributes

Attribute	Name
1	calc_method
2	reset time
3	cloud multiplier

Attributes: All the attributes are the same as with the Read Density calculation method above.

Advantages: Since the simulation uses direct EM field values from the cloud, it can take into account the effects of heterogeneities provided in the simulation (such as trapping etc...). This can also take into account phenomenon from cases not yet studied which affect the cloud density or distribution in ways not taken into account in the generalized forms above.

Disadvantages: This method will require simulations be run for nearly every change made to the beam or lattice. The simulations required can take several hours, so making a series of small changes and testing each is not possible if you wish to preserve an accurate snapshot representation of the electron cloud and the resulting EM field.

2.4 Fully Simulated

Description: Setting *calc_method* to 4 enables the Fully Simulated method of operation. This reads the EM field from each bunch passage and applies it to the corresponding bunch in the train.

Table 4: Custom Attributes

Attribute	Name
1	calc_method
2	field multiplier

Attribute 2: The field multiplier is "added" to each component of the EM field before it is applied to the bunch.

Advantages: Since the simulation uses direct EM field values from the cloud, it can take into account the effects of heterogeneities provided in the simulation (such as trapping etc...). And unlike the Snapshot method, it can adjust for distribution changes over time. This can also take into account phenomenon from cases not yet studied as they evolve over the course of the bunch train.

Disadvantages: This method will require simulations be run for nearly every change made to the beam or lattice. Any abnormalities in the simulation will be applied to the bunch without hesitation, meaning if there is a perturbation in the simulation output which causes a bunch to be lost in Bmad (due to integration discontinuities or otherwise) it can cascade and affect every subsequent bunch in the train.

3 EM Field Calculations

This section will detail how calculations are performed and how to generate input files for the lattice element. The subsections below correspond to the calculation methods detailed in the section and subsections above.

Before describing the individual methods, I will first briefly describe how any custom EM field is implemented with, and used by, Bmad.

Custom EM field calculations are held in a file named *em_field_custom.f90* and must be compiled with the executable. The electron cloud additions come in a separate file called *ecloudmod.f90* and are used by the other file. When a lattice element is told to use a custom EM field calculation Bmad will make requests to *em_field_custom* to find a path along which to integrate and add momentum each particle. Bmad will make a request by sending lattice information along with the x, y, z, and t values for a given particle. *em_field_custom* responds with the Ex, Ey, Ez, Bx, By, and Bz values at that given location and time. Below I describe how each calculation method finds the EM field components given the x, y, z, and t values of the particle in question:

3.1 Self-Contained

[This method is not yet implemented]
The Self-Contained calculation for the EM field operates as a mixture of a mathematical equation derived from trends seen in the output of Ecloud, and an algorithmic component to keep track of passing particles. As each particle passes, an internal density variable (ρ_e) is incremented. The EM field across an axis η (x or y) is given by

$$E_\eta = E_{\eta_0} + \alpha_\eta \rho_e \eta + \Lambda$$

Λ takes the place of smaller coupled terms which can be ignored when focusing on orbit changes and tune shifts, but are responsible for bunch deformation.
For details on how this trends came to be found, see Appendix 5.1

3.2 Read Density

The calculations performed in this method are similar to those performed in the Self-Contained calculation method, which the exception of the algorithmic component. Instead, the Read Density method makes use of a electron cloud density file containing a header and two columns of data. The data must be presented in a file called *qdens.data* as follows:

```
! qdens.data
! Number of qdens datapoints:
n
! time       qdens
t1           q1
...          ...
tn           qn
```

The lines preceded by exclamation points (!) are comments and may be omitted or added at will.

When Bmad requests the EM field at a given time the electron cloud density ρ_e must most likely be interpolated or extrapolated based on the discreet values on either side. If the requested particle's t is between two defined t values in the file, the line between the two will be weighted to give the approximate ρ_e at a given t. The equation is simply:

$$\rho_e(t) = \frac{\rho_e(t_b) - \rho_e(t_a)}{t_b - t_a}(t - t_a) + \rho_e(t_a)$$

If the particle time, t, is not between two points then the same equation is used but with the two closest points (two first or more likely two last points).

If a new particle (at time t_0) is further than the parameter *reset time* seconds away from the last tracked particle, than instead of comparing all new particles to time=0, a new base time is chosen at this particle's time. Meaning all new incoming particles (at t_α) are treated as though they arrived $t_\alpha - t_0$

This $\rho_e(t)$ is then used to calculate E using the same equations from the Self-Contained calculation method.

3.3 Snapshot

The Snapshot calculation method is, perhaps, the most difficult to understand, though hopefully will be clear after this section. The idea behind it is to take a sample bunch's experience as it passes through the electron cloud, and apply it to all other bunches. Since the cloud grows in strength as particles pass, the fields are scaled by the cloud density at each time. The Snapshot method uses the same *qdens.data* file, plus two files called *bunchhead.data* and *snapshot.data* given below:

```
! bunchhead.data
n
! Bunch start times
t1 t2 ... tn
```

The *snapshot.data* is more complicated and contains the Ex and Ey field values along a grid spanning the central beam area. Comment lines cannot be placed after the indicated line (lots of equals signs), but otherwise the same rules apply.

```
! snapshot.data
! Beam grid size:
m
! Number of time slices
n
! Beam grid values:
x1 x2 ... xm
y1 y2 ... ym
! ==================
t1
Ex(x1, y1, t1) ... Ex(xm, y1, t1)
...
Ex(x1, ym, t1) ... Ex(xm, ym, t1)
Ey(x1, y1, t1) ... Ey(xm, y1, t1)
...
Ey(x1, ym, t1) ... Ey(xm, ym, t1)
t2
[...]
...
tn
[...]
```

Production of this file can be simplified using *plotSection* tool, it takes Ecloud output and converts it to snapshot files (see Appendix 5.2 for usage and examples).

A particle's time is found in respect to the head of the bunch of which it is a part. The time with respect to the head of the bunch is then compared to the *snapshot.data* file, whereas the time with respect to the head of the train is compared to the *qdens.data* file. With no E variation on the Z axis, the problem is now interpolating in 3 dimensions (x, y, t). To do so we follow the cubic interpolation procedure seen in figure 1. This method takes the 8 points is a cube most closely encapsulating the particle position (seen in blue). It then interpolates to 4 points along the t axis (in red), then to 2 points along the y axis (in green), then to 1 point along the x axis (red point). This is done to find the E field (x and y components) and interpolation is performed to find the electron cloud density as well. The fields are then scaled by the density and returned to *Bmad*.

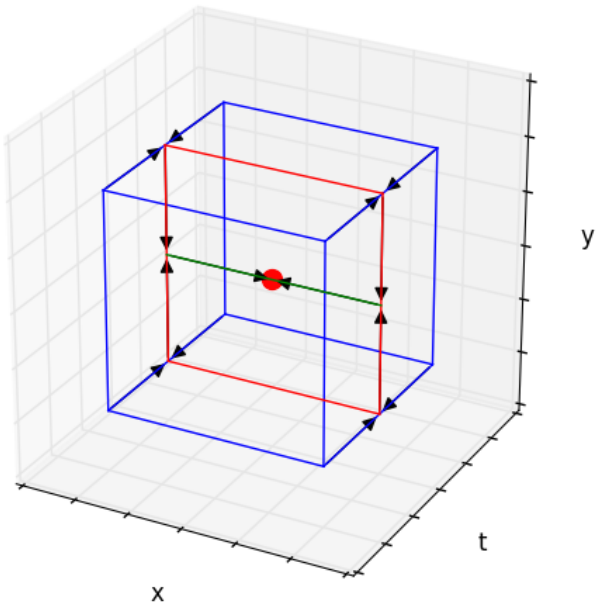

Figure 1: Cubic interpolation

3.4 Fully Simulated

[This method is not yet implemented]

This method uses just a *snapshot.data* file and uses the method of cubic interpolation to track every particle in each bunch through its respective E field as defined in the file. There's is little to say for this method thus far.

4 Bugs and Caveats

Most methods are not yet implemented, only the snapshot calculation procedure is fully functional. This also means it is difficult to see the current limitations of the models. Otherwise, the two major areas of concern are strong E fields and large offsets:

Strong E field: With density multipliers greater than 1.0, *Bmad* will often claim it causes non-symplectic orbits. These are caused by the kick contributions from the cloud being greater than that which the magnets can correct leading to a runaway amplitude growth. *Bmad* cannot compute Twiss parameters in this case and will refuse to run.

Large offsets: Should a bunch pass by an area with a large offset, it will significantly change the way it produces and interacts with the electron cloud This will be difficult to account for since the Self-Contained method cannot (current) calculate non-homogenous cloud densities, and the Fully Simulated version would requite a simulation input for each potential offset of the bunch. Possible solutions are:

- A hybrid Input/Math wherein a mathematical trend is found for the E field change caused by the offset.
- A Self-Sufficient model which calculates some degree of density on each side and uses a different algorithm to calculate the E field based off of the difference between densities.

5 Appendix

5.1 Trends

This is a conglomeration of information from the previous *reports* page and new data. In cases where the relationship between x and y is symmetric (changing all xs with ys and vice versus is still true) I will use η to denote a coordinate and $\bar{\eta}$ to denote the orthogonal coordinate

5.1.1 Ecloud

The simulation engine used to explore the trends and effects on the electron cloud was *Ecloud*. When a simulation is run, Ecloud is told how many filled and empty bunches pass through an area (with its own set of parameters such as magnetic field type etc...). Ecloud outputs 2D slices of the Ex and Ey components of the electric field at 15 time intervals throughout the passing bunch. In addition to a large grid covering the whole pipe (see figure 2) it also outputs a smaller grid of just the local beam area (up to 3.4σ). This local area was the focus of our studies for the models. By plotting the Ex and Ey components as E vectors in the region (see figure 3). All output is multiplied by a "macro particle scaling factor" of -5.7×10^{12} which is why the fields appear so large in the figures (on the order of 10^{11} $V.m^{-1}$).

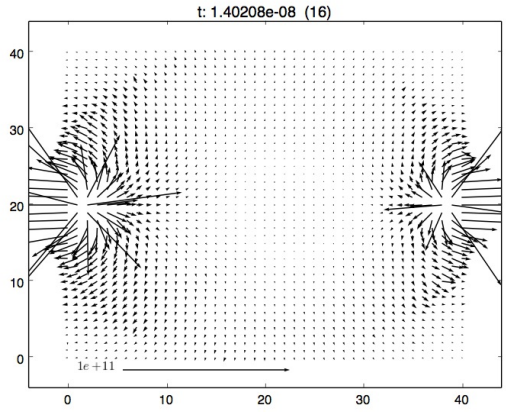

Figure 2: E fields in the pipe

5.1.2 Slopes

Plotting E_η with respect to η gave a linear relationship with an offset ($E_\eta = E_{\eta_0} + \alpha_\eta \eta$). The slope, here written as α_η, is equal and opposite to its counterpart for the other coordinate, $\alpha_{\bar{\eta}}$.[2]
The slope can be compared to the average cloud density to find another linear relationship (assuming the cloud is produced by bunches with no offset).

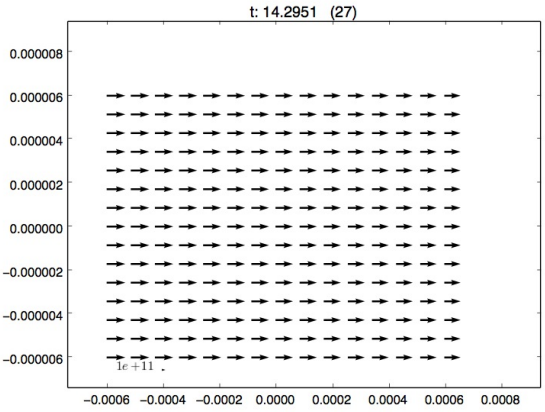

Figure 3: E fields in the beam area

5.1.3 Intercepts

The reason that figure 3 would not lead you to believe that $\alpha_\eta = -\alpha_{\bar{\eta}}$ is because $\|E_{\eta_0}\| >> \|E_{\bar{\eta}_0}\|$. These

[2]The slope is important for the tune shift experienced by the bunch, a negative slope focuses the bunch and increases the tune while a positive slope defocuses.

intercepts are not as predictable as the slopes and are still being studied to try to find a mathematical model.

5.2 Plot Section

The PlotSection python file in included in the BunchAnalyzer package in the *wakefields* repository. The file can also be downloaded directly from *this link*.

```
Usage: python plotSection.py job_dirs
  -s snapshots : make snapshot files
  -p plots : make timeslice plots
  -k keys and sims : not implemented
  -g no-general : Do not make info files
  job_dirs : list of job directories (or glob)
```

PlotSection is primarily a tool to analyze trends for ecloud jobs, but it can also produce snapshot files for those jobs. By using the flags "-g" and "-s" only the snapshots will be generated. All input files must be included in a "snapshot" directory with the same parent as the lattice file (or otherwise accessible by *Bmad*).

5.3 To Be Implemented

2 out of the 4 methods of calculation have yet to be implemented, these are:

- Self-Contained
- Fully Simulated

Self-Contained: I primarily need to find out how to count the number of individual particles passing through a lattice element. This cannot be accomplished by just counting the absolute number of requests due to the use of Runge-Katta integration (which make multiple requests per particle). Furthermore more study of the EM field produced from a mostly homogenous cloud is needed.
Fully Simulated: This requires a rewrite of the snapshot parsing to deal with multiple bunches. This shouldn't prove difficult if it becomes a priority. The main problem is on the side of the Ecloud output which often includes discontinuities.