

Simulations and Testing of Retarding Field Analyzers for Electron Cloud Monitoring

Lee McCuller

07/02/09

Abstract

1 Forward

This work is done as part of the Lee Teng internship at Fermilab with the help of my supervisor Robert Zwaska. I am also getting support for my project from Kevin Duel who helped us decide on and draft the test vessel design, and from Cheng-Yang Tan who does e-cloud simulations and studies.

2 Introduction

My summer project is to assemble a test stand for new Retarding Field Analyzers (RFAs) to be used for electron cloud (ecloud) studies. Along with assembling this test stand I will be simulating the RFA response using the SIMION program and I will be doing some calibration and studies of the RFA response to test stand input. These RFAs to be studied are an important tool for understanding the ecloud in Fermilab's main injector.

The ecloud is a phenomenon in positively charged beamlines. It is a gas of electrons that is formed from a cascade of secondary emission electrons from the beam pipe. The process starts from the residual gas in the chamber. The beam passes through that and knocks off electrons from the gas. These electrons are attracted to the beam and so they are sucked into it and ejected from it at higher energy. These ejected electrons then impact the beam pipe with enough energy to eject more electrons. This process of heating electrons and ejecting more secondary emission electrons essentially causes an exponential generation of electrons, which maxes out once ecloud space charge effects cancel the acceleration that the beam is giving. All of these electrons push the beam around and lead to head-tail and multi-bunch instabilities.

This phenomenon is affected by several beam parameters. The heating of the electrons is sensitive to the flight time of the electrons and to the bunch lengths and spacings. The generation of electrons is sensitive to the energy gain and the surface properties of the the beam pipe. And finally the space charge limitations are affected by beam intensity. Since these factors affect an exponential process, small changes can have effects with orders of magnitude differences. The intensity dependence is particularly worrisome in this process,

since there is a risk that the ecloud limits the maximum intensity in the project X upgrades.

The ecloud has been seen at many of the high energy accelerators and it has been studied in several machines to help characterize it. The difficulty from these studies is that since it is such a nonlinear process which depends on the accelerator geometry as well as beam parameters, it behaves uniquely in each machine. Simulations have been done for the main injector but they can not replace real data.

2.1 RFA Assembly Design

The retarding field analyzers are sensors that can detect current from an beam of charged particles. They consist of a collecting plate behind a wire screen. The wire screen is set at a voltage so that the field that it creates serves as a high pass filter in the energy of electrons passing through it. The main injector has had an old RFA taken from an Argonne accelerator [] that has a few design differences to the model this paper focuses.

The new RFA's that have been designed for testing use a cupped shape around the wire grid for field shaping. This gives a natural-defocussing before, focusing after effect that should reduce the effects of secondary emission the collection current. Measurements done in the main injector using the previous RFA's had a substantial amount of noise to them. The new incarnations hope to prevent this with not only a better geometry, but they will also have a lowpass filter and amplification applied in the tunnel. The filter will reduce noise from the beam RF and the amplification will reduce line noise in transmission.

Tan's created the new RFA design and compared it to the old Argonne style RFA's that have been used.

3 Test Stand Setup

The primary focus of this project is to develop and use a test stand for the RFA. The design is to use an electron gun to provide a beam of electrons at a known energy and spread so that we can compare the gun emission current to the RFA collector current. The primary difficulty in this test stand will be preparing a vessel to house the gun and RFA in vacuum.

The vessel is designed to emulate the beam pipe effects on the ecloud, as well as facilitating measurements. It consists of a central pipe with end flanges, one for connecting the gun and the other for the RFA. In order, there are several components along the pipe. First is an extrusion to attach vacuum pump, second is an angled pipe with flange for a viewing window. This angled pipe meets at the vessel along with another pipe with a flange to attach a mount for a phosphor screen, to be viewed in the window. immediately after these connections is a screen visible in ???. This screen will also be present on the RFA mount to the beam pipe. Ideally the RFA would mount to a hole in the beam-pipe, but the screen is a compromise to reduce impedance. For the vessel the screen has the advantage that the electron gun should be calibrated using the light emitted from electrons off the screen, so the phosphor screen should not be a requirement.

3.1 Vacuum Setup

The vacuum system so far as I know will need to be a roughing pump and turbo pump with a pressure sensor and its electronics. Baking should not be necessary since 10^{-7} Torr should be as good a vacuum as we need.

3.2 Electron Gun Setup

This includes all that is necessary to run the Electron gun. Calibration is the most important using the auxiliary windows to hold a phosphor screen and developing a way to translate the screen image into beam parameters for the RFA. Other issues include setting up the filtering electronics on the RFA and getting a good readout from it.

3.3 Electronics

4 RFA Simulation

In order to develop a new RFA with characteristics suited for this application, the program SIMION was used to model the RFA geometry and fields and to develop a phenomenological model for the system. This program lacks features such as secondary emission electrons and certain data taking features that would be useful for this and future projects. I have added some features to more accurately model the RFAs using this program.

SIMION works in a very modular system. The entire simulation is encapsulated in a workbook, which holds the main coordinate system. Inserted into the workbook are Potential Array (PA) files which are each a grid of cells containing either electrodes or free space. The electrodes are set at a voltage and SIMION 'refines' them to generate the potential at each point in space using an iterative solution to the laplace equation. These PA files define the electrode geometry of the simulation and can have reflection or cylindrical symmetry applied to them. The workbook represents them in space with their symmetry applied as well as with arbitrary scaling and orientation transformations. Because solutions to the laplace equation can be superimposed linearly, SIMION allows connected electrode regions to be adjusted in real time which it calls fast adjusting the PA's. This method becomes useful in user programs.

The RFA simulation uses a workbook in this manner, but requires some caveats in order to simulate a realistic grid. This is necessary since the grid requires an extremely dense array to describe its geometry due to its thin wires. With the wires only 5 cells thick, and the grid only occupying 1/16th of the true grid area, the PA file is 100mb. The trick used is to transport electrons from an ideal grid region in the RFA PA to the smaller non-ideal grid, but to remember the electron's offsets so that they may be transported back. Toggling this transport can differentiate results with ideal and non-ideal grids.

My work was to enhance the user program to manage each particle and to add secondary emission capabilities. The modification design is highly coupled to how SIMION fly's particles through a workbook. The simion userprogram model allows the program to be as passive as possible. The user program is a script written in LUA that registers a series of callbacks that SIMION calls at various stages of the electrons' lifetimes. Difficulties arise because SIMION

treats particles in batches, even when it is actually simulating particles one by one, so advanced user programs need to have additional structure to account for this batch mode of operation. User program callbacks are called on a per particle basis and each callback type has variables registered that correspond to particle parameters such as location or velocity.

SIMION's order of operation is to first read a .fly2 file which describes the batch size of particles to fly including its size and starting location and velocity distributions. Then SIMION calls an initialize function on each of the particles. This function can modify energy or location and essentially override the characteristics that the .fly2 file had determined. Here one can check for the first particle and call a beginning of fly function or "global initialization" type function. The user program is then allowed to do a fast-adjust on the potentials before the fly (they can also be done during the fly but this simulation uses static fields). After all of the particles in the fly have been initialized, they go through a flying loop. SIMION internally adjusts their position and velocity using Runge-Kutta integration over the EM field forces. After that, it checks for collisions and calls the "other.actions" function. This is where the transport trick takes place, and also where secondary emissions are created upon impact with metal surfaces. At this time the user program can adjust the magnetic and electric fields seen by the particles.

After all of the particles have impacted a surface, SIMION calls "terminate" on each particle. Our program does not need this for each individual so it waits for the last particle and calls a global "fly completed" type function. After this is called SIMION checks if you want to begin another fly and repeats this process (so it uses the same .fly2 and the user program is responsible for changing particle parameters between flies).

4.1 Details of Added User Program Enhancements

The developed user program has added two abstract components to manage particles. It has a particleRun manager which stores all the different parameters that you want a datapoint to apply. This includes the magnetic field magnitude and direction, the electron energy, parameters related to secondary emission, and the number of particles to run at these parameters. This manager is generated and called directly by the SIMION callback functions to make adjustments. The particleRun class also stores a queue of future particles to generate from secondary emission. This queue is necessary since the batch mode operation does not allow one to generate SE electrons as they are created. They go into the queue and are created in the next fly. This class also records where particles impact and other statistics. When the current particleRun has taken enough data, control moves to the other abstracted component, the runManager. This class has the duty of recording the data stored in a particleRun to a file and it has the duty to start new runs with incremented parameters (such as particle kinetic energy). In implementation, the runManager class uses an iterator to feed it new particleRun instances.

The choice to use iterators to generate new runs allows one to control the parameters without needing to modify the particleRun or runManager classes themselves. The userprogram is given a script file to execute which gives it the iterator to use for the runManger, allowing a more automated control of data taking.

For individual particle data, the `particleRun` class also includes a list of all currently flying particles. This avoids the anonymity that the batch mode running enforces by allowing individual tagging of particles.

4.2 Details of Grid Transport Trick

As described above, the userprogram employs a trick in order to reduce the memory requirements of a fullsize realistic grid. To implement this trick, the workbook employs a series of "dummy" PA's. As a particle is flying, it knows which PA it is currently being influenced by. The userprogram checks this variable to see if the particle has entered the kickout PA. If so then it employs some modular arithmetic to determine where in the grid it should be transported to. This math effectively tiles the grid across the RFA. The offset is recorded to the particle once moved and the particle flies in the grid PA as normal. the grid PA is enclosed in a return PA, which the transporter checks for. Once the particle enters the this PA it is retranslated by its offset. In the flying view of SIMION, one can see that this does indeed preserve flight continuity. There are some catches to this trick though. Since the grid is of limited size. The return PA section must also check to make sure that the particle has not simply escaped the transverse boundaries of the grid PA. if it has, then it is transported around the grid and its offsets are adjusted (This oversight led to a reduction in the perceived efficiency of the RFA under a non-ideal grid). The other catch is that the transporter must also check that it is not returning the particle to a spot inside the RFA wall, since the walls are not present in the grid PA.

4.3 Details of Secondary Emission Implementation

All of the abstractions added above were implemented so that secondary emission could be modeled with SIMION. SE is difficult to model in SIMION due to several program limitations. SIMION's representation of the electrode surfaces is the minimum necessary to do electric field calculations. It does not store metal surface properties and, more importantly, it has no knowledge of the surface normal. When a particle impacts a surface, our userprogram detects this, and knows the location and velocity of the impact. A phenomenological SE yeild model is employed to determine the number of expected particles generated.

The SE yeild model follows

We use poisson statistics to determine the actual number of SE particle to create. Each one created is colored differently for inspection purposes. For each SE electron to generate, we need to know its outgoing energy, and its outgoing direction. The energy is determined by another model (which currently is just to use 1/9 the incoming energy). The outgoing direction has a uniform distribution over the 2π arcadian hemisphere away from the surface. this distribution is generated in two parts, determining the surface normal, and applying that distribution to the normal.

Since SIMION represents its surfaces in a grid, generating a realistic surface normal is a nontrivial process. From boundary conditions, The electric field off of the surface is always normal to the surface. This fact is used to approximated the normal in this finite element model. SIMION's scripting language LUA exposes a lot of particle data but does not by default have an interface for finding the electric field in PA's. Our program uses a current SIMION beta release and

activates "early access mode" to gain this interface in our LUA userprogram. Each PA has a unique grid size, which sets a scale across which the electric field should be averaged to get the normal direction. Our computation works in PA local coordinates to naturally work across this scale. It takes the electric field in the 27 cell, 3 by 3 cube of PA grid cells centered at the impact point. It ignores those cells that are electrodes and the remaining values are averaged. This vector is then normalized to give the normal. Sign checks are made since the electric field lines can be pointing inside or outside the surface. The Runge-Kutta flying algorithm often has particles that impact just below the electrode surface, in which case the SE particle can't escape. Checking for this is most convenient in these methods while PA coordinates are available, so the surface normal computation can also flag to toss out SE for this impact.

Once the normal, N is found, the distribution around it must be implemented. This is done by creating a uniform distribution of vectors in the positive X axis hemisphere, and then rotating these so that the X-axis points in the direction of the normal. To generate the hemispherical distribution, I used spherical coordinates and the inverse PDF method for generating θ and ϕ . The differential area element for integrating a spherical surface in these coordinates is $r^2 \sin(\phi)$, and $r = 1$ for a unit sphere. With the inverse PDF method, uniformly distributed numbers from 0 to 1 are passed through $\arccos()$ to get the proper distribution for ϕ . θ can be generated from a uniform distribution from 0 to 2π .

$$V = \begin{bmatrix} \cos(\phi) \\ \cos(\theta)\sin(\phi) \\ \sin(\theta)\sin(\phi) \end{bmatrix}$$

These parameters are plugged into this vector formula to get their linear coordinates. I use a rotation matrix to point the x-axis through the direction of the normal. To generate this rotation matrix I arbitrarily choose a vector orthonormal to N and then use a cross product to get a final orthonormal basis set. I can then construct an orthogonal matrix from this set. In reality, such an orthogonal matrix has two vector degrees of freedom, a "looking" direction which we give, and an orientation direction. Here the orientation direction is ignored since the components that it affects have a random orientation anyway.

5 Simulation Data

Here will go all of the simulation data. The plots for this are being processed.

6 Measurements

Here I will put the results of the test stand measurements that I have taken.

7 Drawings and Figures

Here I'm just putting all the figures until I figure out what to do with them.

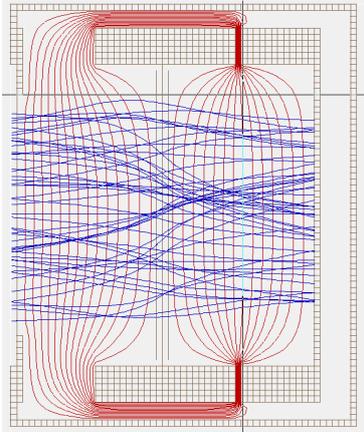


Figure 1: Here is a simulation with electrons in blue and contour lines of chamber voltage in red

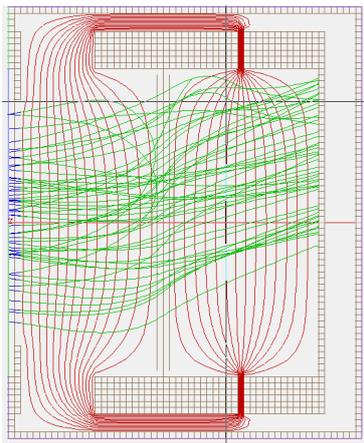


Figure 2: here is a simulation of electrons in a magnetic field of 5 Gauss into the page

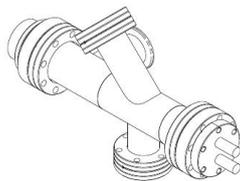


Figure 3: Here is a side view of the fully assembled vessel



Figure 4: Here is a test RFA used by C.Y. Tan