14 TeV CMS Simulations and Webtools

8/5/2013

John Michael Farmer (Clemson University), SIST Intern

In collaboration with: Jake Callahan (Iowa State University)

Supervisors: Pushpa Bhat and Leonard Spiegel

Abstract

In 2015, the LHC will run at (pp) center of mass collision energy of 14 TeV and luminosity $10^{34}\,cm^{-2}\,s^{-1}$, an upgrade from its 2012 run at 7 TeV and $\sim 5 * 10^{33}\,cm^{-2}\,s^{-1}$. This will cause increased radiation load on the detector, and the electronics must be properly shielded. The radiation environment in the detector may be estimated through Monte Carlo simulations of proton-proton (pp) collisions occuring inside the CMS detector. To this end, we examine fluence and dose in the CMS detector using data from Monte Carlo simulations performed with the software package MARS by P.C. Bhat, A.P. Singh, and N.V. Mokhov in 2008 [1]. We have designed a suite of webtools for calculation and graphical display of fluence data from these simulations and have collected a much larger sample of data by running simulations using the Monte Carlo simulation package FLUKA.

Contents

List of Figures

List of Tables

1 USCMS Fluence/Dose Calculator

1.1 Introduction

The USCMS LHC Fluence/Dose Calculator was started in 2009 by D. Haznar and extended in 2011 by P. Sharma [2]. For this project, we have completely rebuilt the calculator, with a new codebase and a new interface. For details on the specifics of the code, refer to the software documentation in Appendix A of this paper; this lists all files, all scripts, and all MySQL tables, as well as details the changes and optimizations that were done to the original code.

The calculator loads live from `www.uscms.org/uscms_at_work_dmo/siTracker/lhcCalcForm.html`; a developer test page is located at `cmstrk.fnal.gov/radsim/lhc_DoseFlux_CalculatorTest.html` [3]. Upon navigating to the page, the user is presented with a single dialogue box that asks to specify an integrated luminosity in fb^{-1} (inverse femtobarns), which simulation program to draw data from (MARS [4] [5]. [6] or FLUKA [7] [8]), which volume to work with (full CMS detector, CMS tracker, or CMS calorimeters), and an operation

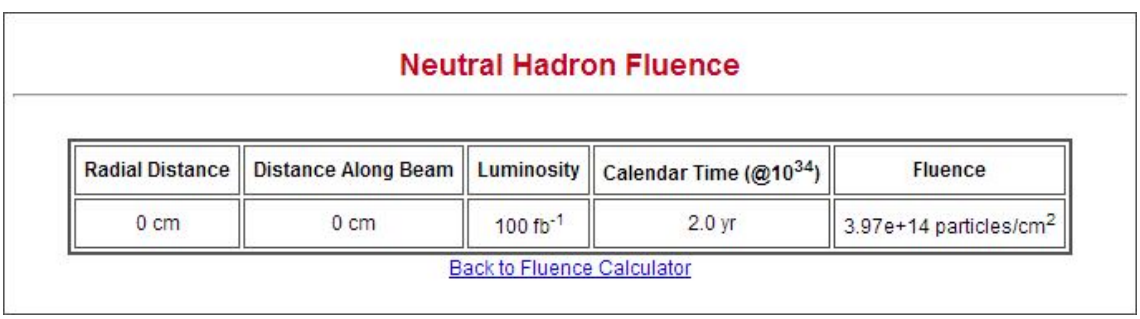
The calculator currently consists of three operations:

1. Point Calculator
2. Detector Maps (binned 2D histograms)
3. Slice Graphs

A detailed explanation of each of these operations follows. It is important to note that the MARS (and FLUKA) data assume ϕ symmetry, so understanding the complete geometry of the detector is possible through examining just R and Z. These coordinates are assumed to be in centimeters: the detector's full geometry spans from -2260 cm to +2260 cm in Z and from 0 cm to 750 cm in R.

1.2 Point Calculator

From the end-user perpsective, this function is unchanged from the original calculator version, though the code behind it has been completely renovated. It is the most simple of the four operations: in addition to the basic parameters, the user inputs an (R,Z) coordinate along the detector, and a small table of fluence data is calculated and displayed, shown below.



Neutral Hadron Fluence

Radial Distance	Distance Along Beam	Luminosity	Calendar Time (@10^{34})	Fluence
0 cm	0 cm	100 fb^{-1}	2.0 yr	3.97e+14 particles/cm^2

Back to Fluence Calculator

Figure 1: Table output by the point calculator, using MARS data.

1.3 Detector Maps (binned 2D histograms)

Development of this functionality was the cornerstone of the calculator project this summer. This operation makes use of the Fluence Package, a C++ program written by Charles Dietz [9] that uses CERN's statistical analysis framework, ROOT [10], to create histograms and projection maps. It reads in fluence data and CMS geometry from separate text files; there is one data file for each detector volume and simulation type.

We took one program in the Fluence Package, `plotFluence`, and modified it to suit the needs of the calculator. An example output graph can be seen below.

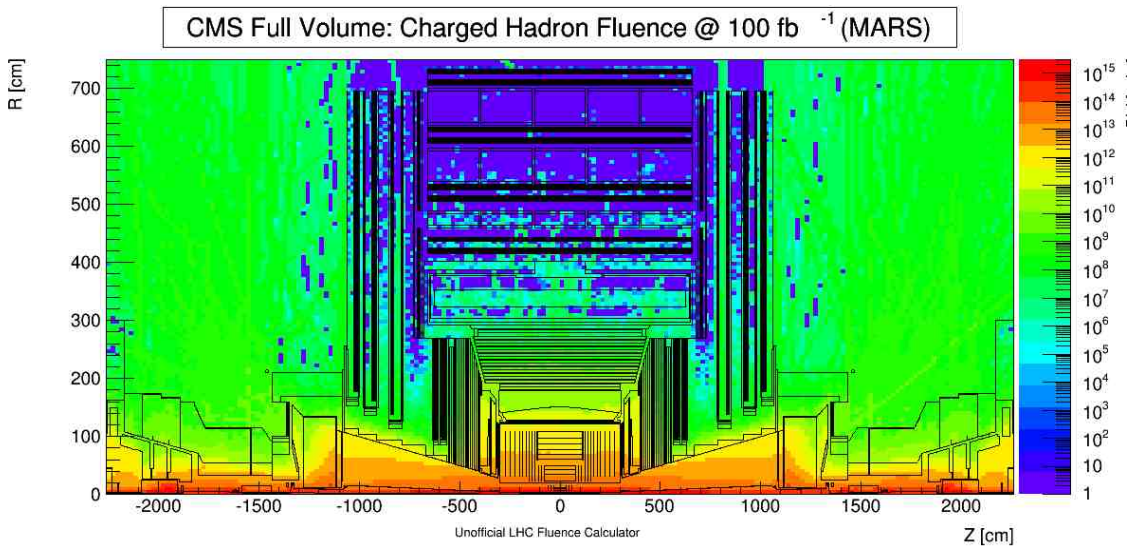


Figure 2: 2D binned histogram showing charged hadron fluence in the full CMS detector geometry, using MARS data.

In addition, we created an interactive interface for these maps using JavaScript. Hovering over the plot area will display the coordinates in (R,Z) of the current point beside the cursor, and clicking will retrieve the fluence data from the nearest point and display it at the bottom of the page in a table. This provides a quick, user-friendly alternative to using the calculator to individually fetch fluence data from the database. In addition, overplotting the geometry over the image allows for users to quickly navigate to points of interest, without having to remember the exact coordinates and input them into the calculator. The tracker, for instance, is clearly visible in the center of this map.

This plot, and all plots output by the calculator, may be downloaded as a .jpg or other common image format. In addition, we have included the option to download the image as a .root file; this allows end-users to modify the title, axes titles, and legends as they see fit. However, we do ask that the header at the bottom, "Unofficial LHC Fluence Calculator," remain intact in all circumstances.

1.4 Slice Graph

Detector maps are useful for observing general trends over the full geometry, and together with the calculator, offer an easy way to calculate a precise fluence value at a single point; however, from either the map or calculated value, it is difficult to understand precisely the evolution of fluence as R and Z

vary in the detector. To this end, we have developed a tool using PHP, C++, and ROOT that allows users to hold either R or Z constant and graph fluence versus the other coordinate.

1.4.1 Slices in R

It is useful to visualize fluence as a function of radial distance from the center of the CMS detector, at R=0 cm. Setting the Y axis to a logarithmic scale, it is expected to see a function that falls off approximately like $\frac{1}{R^2}$.

This operation allows for users to specify a Z location and an R range; the calculator then generates a table of fluence data radially from that (R,Z) point, holding Z constant. It is possible to superimpose an arbitrary number of fluences in this fashion; for example, one can produce a plot that shows neutral hadron fluence, charged hadron fluence, and NIEL (Non-Ionizing Energy Loss) fluence. The result is optionally displayed in a table (which may be downloaded in plain text format), pictured below:

Beam Slice Graph

Luminosity: 100 fb^{-1}
Z = 0 cm

Download .txt

View Graphical Representation of the Data

Download .root

Radial Distance in cm	Neutral Hadron Fluence in particles/cm^2	Charged Hadron Fluence in particles/cm^2	NIEL Fluence in particles/cm^2
0	3.97433e+14	2.02045e+15	2.44971e+15
10	2.22641e+13	1.18628e+14	1.46834e+14
20	9.47592e+12	2.85282e+13	4.05589e+13
30	7.32664e+12	1.88271e+13	2.81530e+13
40	6.03628e+12	8.98744e+12	1.62089e+13
50	5.37035e+12	6.79769e+12	1.30168e+13
60	4.82087e+12	3.70474e+12	9.07369e+12
70	4.48762e+12	2.33363e+12	7.22221e+12
80	4.38306e+12	1.83519e+12	6.60181e+12
90	4.22206e+12	1.07351e+12	5.59591e+12
100	4.09780e+12	9.06152e+11	5.28458e+12

Back to Fluence Calculator

Figure 3: Data table as produced by the calculator, using MARS data.

A C++/ROOT executable then graphs these data into a simple line graph, as seen below. The step

size, here 10 cm, is user-defined. Users should be aware of the grid granularities used for the different detector simulations when selecting a step size; these granularities can be seen in Table 1, in Section 2.2. Selecting a step size that is less than the granularity of the data will result in repeated values, leading to distorted plots.

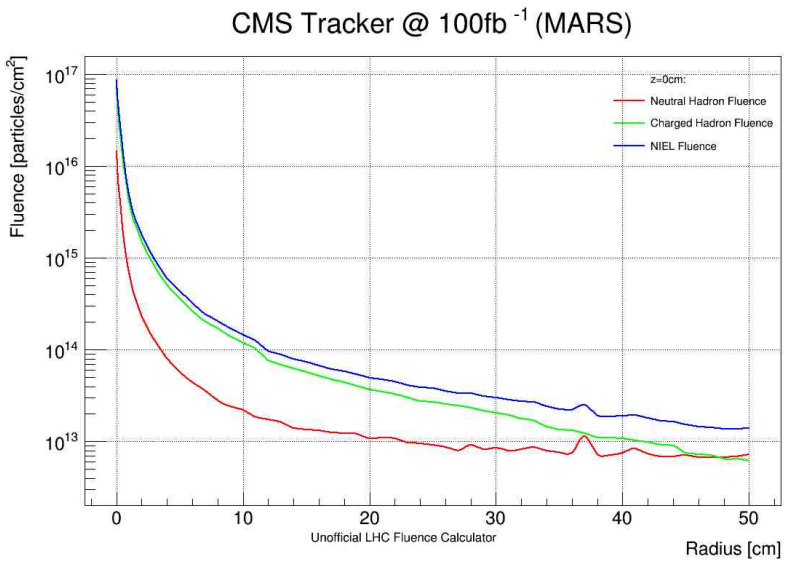


Figure 4: Slice graph overplot of neutral hadron, charged hadron, and NIEL fluence, using MARS data (Z=0 cm; R step size of 1 cm).

This tool also allows for the selection of multiple Z points at which to graph radial fluence. It is possible, for instance, to compare how charged hadron fluence falls off as a function of R in the tracker to how it falls off in the calorimeters. An example plot is shown in Figure 5.

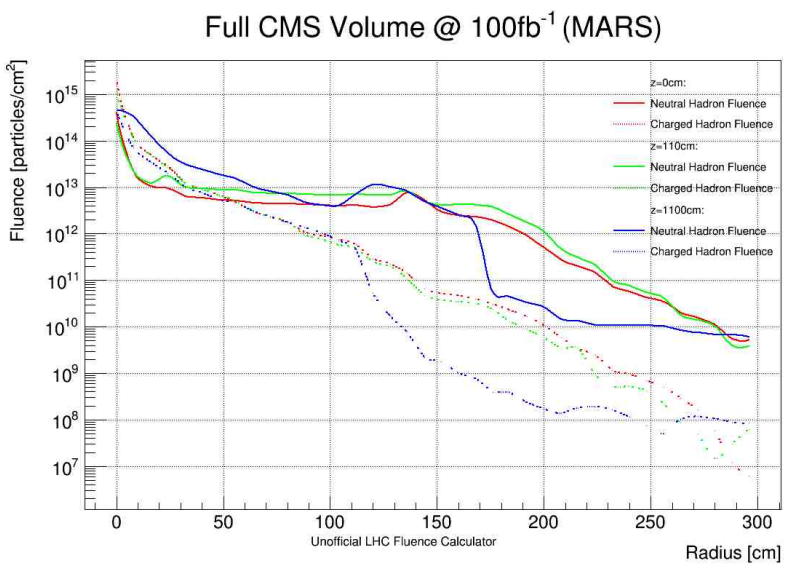


Figure 5: Graph showing fluences over a specified R range at z=0 cm, 110 cm, and 1110 cm overlayed, using MARS data (step size of 8 cm).

1.4.2 Slices in Z

In parallel with the R slice functionality, the Z slice operation allows users to hold R constant and examine how fluence varies as a function of Z over a portion of the detector. The user specifies an R point and a Z range, and fluence is plotted versus that range. In this fashion, it is easy to see the precise effect the detector geometry and material has on the evolution of fluences. Again, it is possible to superimpose an arbitrary number of fluences here and choose to output a data table which is downloadable as plain text. A simple example plot is shown in Figure 6.

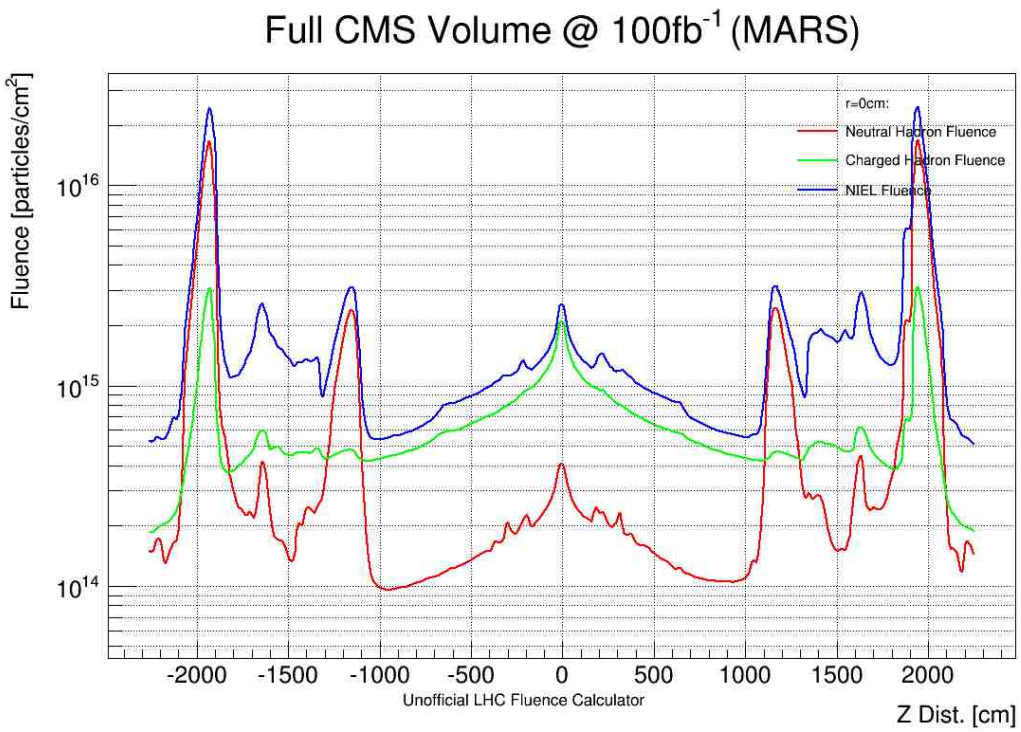


Figure 6: Figure showing Z slice at a single R over full geometry with neutral hadron fluence, charged hadron fluence, and NIEL fluence, using MARS data (R=0 cm; Z step size of 22 cm).

It is also possible to superimpose Z slices of various R values.

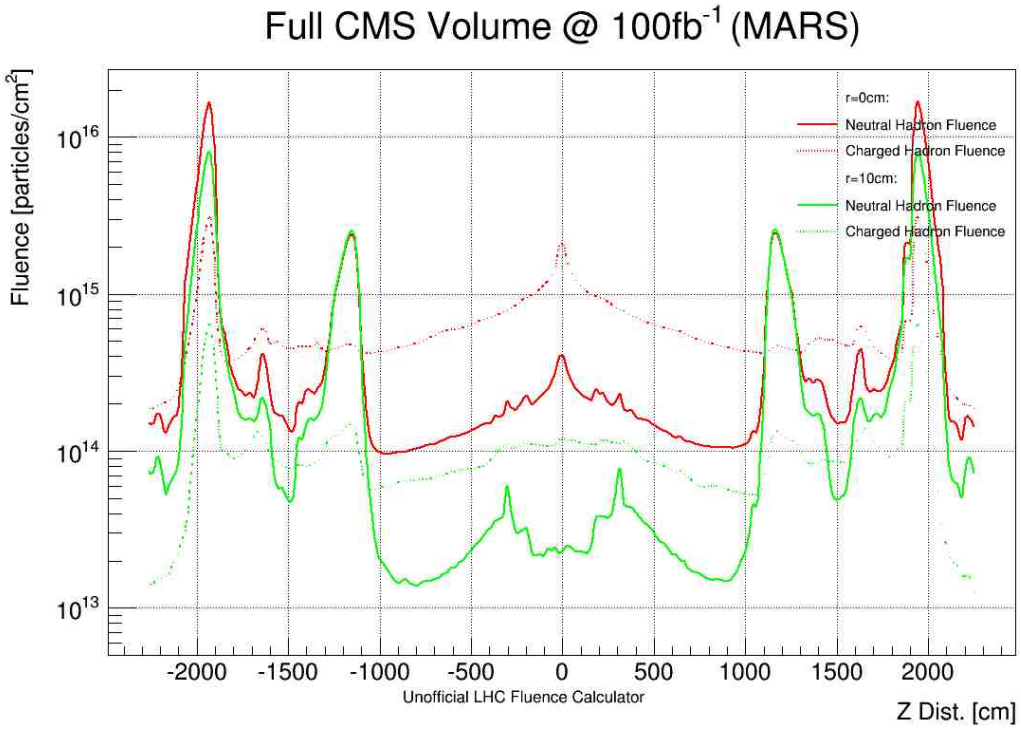


Figure 7: Figure showing Z slice at multiple R points over full geometry with neutral hadron fluence and charged hadron fluence, using MARS data (R=0cm, 10cm; Z step size 22 cm).

1.5 Future Work

Currently, the greatest weakness of the calculator is its inability to support multiple simultaneous users. This is because all graphs are output to an absolute path, overwriting the previous image. This is an intentional design choice to control storage on the server. However, this overwriting is not apparent until a user reloads the page; this could cause a user to download a different image than the one he requested the calculator generate. Currently, the calculator is a small enough project that this is not an issue, but should traffic increase in the future, it may necessitate a corrective action. For this reason, we will include instructions here.

Solving this problem is not particularly difficult, but requires some work on the server-side. It makes use of the PHP `uniqid()` function, which assigns a unique ID to each user. The idea is to concatenate this ID to the filename, so that no files will be replaced, and instead they will remain on the server indefinitely. In order to handle server clogging, a batch scheduler should be employed to schedule routine (i.e., daily or weekly) deletion of all plots generated by the calculator. JavaScript events `onunload`, `onbeforeunload`, and their jQuery equivalents present an ideal solution, but are not universally supported by all browsers, and thus cannot be trusted to accurately clean the filesystem of excess plots.

In addition, it might be useful to optimize the PHP and C++ code. Graph construction time, especially when examining Z-slice graphs with a small step size, can be quite high. This has been mitigated slightly by making HTML table construction optional, but given the amount of database querying and data processing the calculator software performs, execution time will naturally be high unless specifically optimized. Of particular usefulnes would be developing a more efficient algorithm to extract the relevant entries from the MySQL tables.

2 FLUKA Simulations

2.1 Introduction

FLUKA is a Monte Carlo simulation package for particle transportation and interaction, written in FORTRAN77 [7] [8]. It allows users to load in a desired geometry, using materials from the FLUKA material database, and simulate particle interactions. A graphical user interface for FLUKA, Flair, written in Python [11], was also used to prepare and examine the input file, as well as inspect the geometry and output.

Given an input file with CMS geometry loaded, it is relatively simple to configure the program to use any type of scoring desired. We used USRBIN scoring, read into units as seen in Table 7 in Appendix A, which imposes a grid on the detector for binning, and during simulation, traces and calculates the value of the desired quantity. We store the following quantities in the defined grid:

- HAD-NEUT (neutral hadron fluence, $\frac{particles}{cm^2}$)
- HAD-CHAR (charged hadron fluence, $\frac{particles}{cm^2}$)
- PHOTON (neutral EMS fluence, $\frac{particles}{cm^2}$)
- ELECTRON (charged EMS fluence, $\frac{particles}{cm^2}$)
- MUONS (muon fluence, $\frac{particles}{cm^2}$)
- NIEL-DEP (Non-Ionizing Energy Loss, GeV)
- ENERGY (Total Energy Deposition, $\frac{GeV}{g\,cm^2}$)
- DOSE (Dose, $\frac{GeV}{g}$)

NIEL Fluence is of particular interest in studying the radiation background, as it approximates radiation damage to the first order. It is important to note that in MARS simulations, the program did not track individual particle energies, and thus could not score NIEL directly. For MARS data, an approximation was made of NIEL fluence using a linear combination of other fluences:

$$\Phi_{NIEL} = \Phi_P + \Phi_\pi + \frac{1}{2}\Phi_\mu + \frac{1}{10}\Phi_{n>0.1\,MeV} + \frac{1}{100}\Phi_\gamma \text{ [1]} \tag{1}$$

Where the coefficients result from the specific interaction cross sections. However, FLUKA tracks NIEL through scoring NIEL-DEP. Similarly, MARS did not track dose directly; we obtained dose by performing scalar multiplication on the Total Energy Deposition:

$$\Phi_{Dose} = 128 \cdot \Phi_{TED} \tag{2}$$

After performing a few test runs on local machines, we ran jobs on the CMSLPC batch farm at FNAL using Condor. We wrote Python and bash shell scripts to run the jobs, analyze the data, and reformat into the two formats the Dose/Flux calculator needed. The details of this process can be found in Appendix B.

As with the MARS simulations, ϕ-symmetry of the detector is assumed.

2.2 Statistics

Aside from more accurate scoring of quantities MARS did not directly track, the main motivation for running these simulations were: (1) to increase the amount of statistics, up from the 13,000 collisions simulated with MARS, up to a minimum of 50,000 collisions; and (2) to obtain data with a finer granularity, particularly on the CMS full volume.

Simulation Data Statistics					
Volume	**Simulation**	**R Granularity (cm)**	**Z Granularity (cm)**	**Approx. # bins**	**# collisions**
Tracker	MARS	0.92	2.90	24000	13,000
	Fluka	1.009	2.003	34800	52,000
Calorimeters	MARS	2.00	2.00	34500	13,000
	Fluka	2.00	2.00	34500	53,225
Full Geometry	MARS	6.25	22.6	24000	13,000
	Fluka	3.004	7.535	150,000	78,750

Table 1: Table showing a comparison of granularity and data sampling between MARS (2008) and FLUKA (2013) simulations.

As seen in the table, as compared with the MARS data, the FLUKA results for the full geometry have generated around 6 times as many collisions and filled 6 times as many bins. An additional statistic worth examining is the transport threshold cutoff energy for various particles, beyond which the program no longer transports or scores particles. A table of these data can be found below.

Scoring Cutoff Thresholds		
Particle	MARS	Fluka
Charged Hadrons	1 MeV	1 MeV
Neutrons	100keV	10^{-5}eV
Photons	200 keV	30KeV
Electrons/positrons	200 keV	≤ 1MeV, most ≈ 100KeV*
Delta-ray production	100 keV	≤ 10MeV, most ≈ 100 KeV
*e+/e- and δ vary by region		

Table 2: Table showing the transport cutoff energies for the MARS and FLUKA simulation runs.

In general, a lower number leads to more accurate data, but not by much, as the particles dropped by the transport threshold system are of low energy. This comes at the cost of increased computing time. Electron/positron and delta ray thresholds vary by different regions in the geometry. Their highest value is listed foremost in the table, but these are outliers; most of the cutoff energies are much lower, more in the range of the MARS cutoffs. Charged hadron cutoffs are equal for both simulation runs. Photon cutoffs are slightly better (by one order of magnitude), and neutron transports are significantly better in the FLUKA simulation (by seven orders of magnitude).

2.3 Data Generation

For the tracker geometry, we ran Condor jobs on 10 CMSLPC nodes, each one running 15 primaries (that is, 15 pp collision events), at around 200 jobs per node. For the full CMS detector, we ran Condor jobs on 16 CMSLPC nodes, each one running 15 primaries, at around 350 jobs per node. In order to receive unique data for each job, it is necessary to adjust the random number seed used by FLUKA (RANDOMIZ input card); otherwise, all 200 jobs would return identical data. Jobs were submitted using our Python script `condorSubmit.py`, which created a directory for each node, a directory for each job inside the node directory, and a unique FLUKA input file for each job with a randomized seed. Any user with CMSLPC grid credentials and a proper FLUKA input files can easily submit additional jobs by using this script. For more information, see Appendix B.

Data were combined by running FLUKA's `usbsuw.f` FORTRAN routine; we used a bash script to automate this process to sum up first the output of all jobs from each node, and then again to sum up the results from all nodes into a single binary output file. We then ran FLUKA's `usbrea.f` to obtain unformatted ASCII output, which was organized into various formats by another Python script, `readFluka.py`. Each job completed in around an average of six to ten hours, depending on the speed of the node.

These data have been inserted into the calculator database and are now accessible by selecting `FLUKA` for the `Simulation` option. Figure 8 shows a plot of the FLUKA data that exhibits its fine granularity.

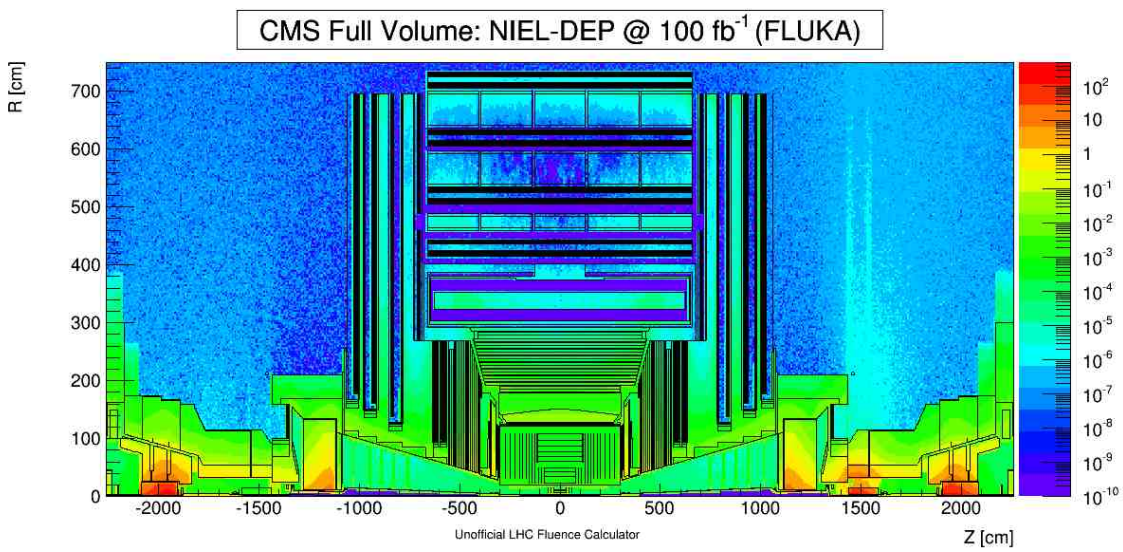


Figure 8: 2D histogram showing NIEL-DEP over the CMS full volume, using FLUKA data.

The asymmetry around $z = \pm 1500\,cm$ comes from the CASTOR calorimeter, which begins at $Z = 1430\,cm$ [12].

2.4 Comparison Graphs with MARS Data

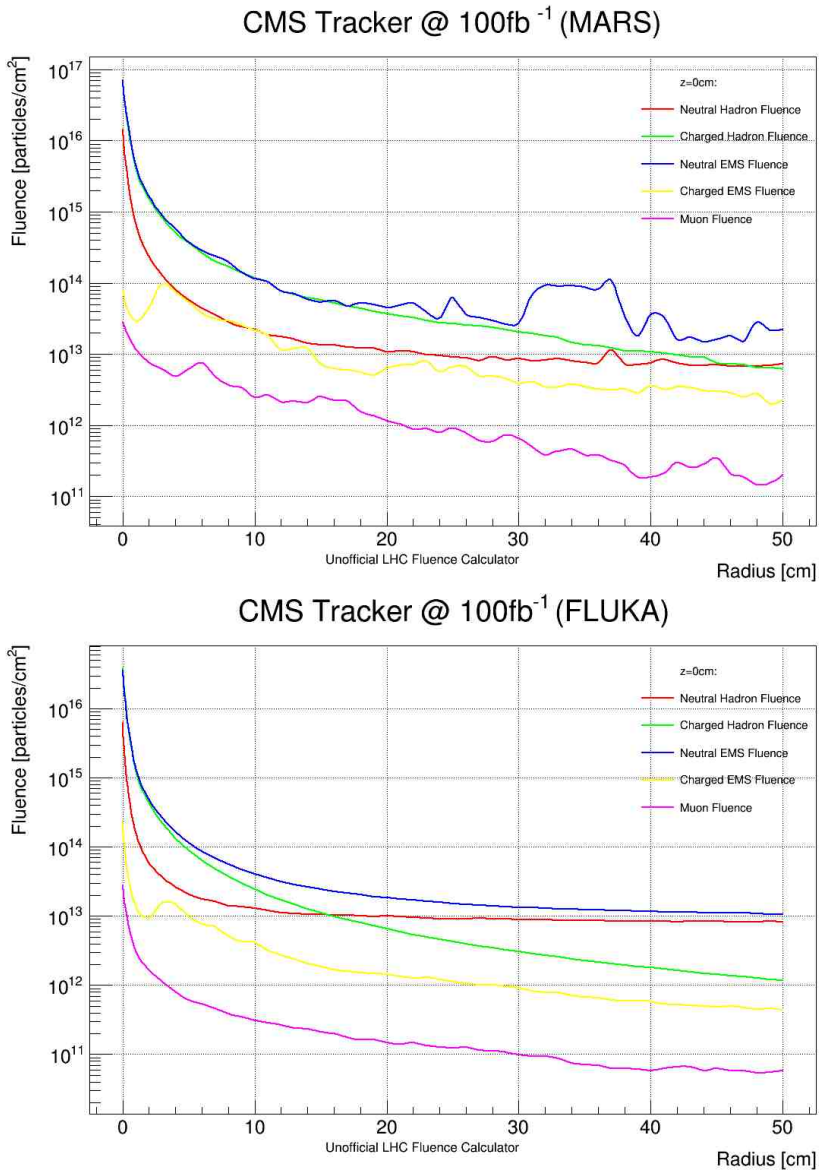


Figure 9: Multigraph comparison of basic fluences between MARS and FLUKA.

As expected, the plots in Figure 10 look quite similar, but the FLUKA data is much smoother, and does not have the consistent spikes around R=36 cm that are present in the MARS data. The general trend, following an approximate $\frac{1}{R^2}$ law on a logarithmic Y-axis scale as R increases, seems to hold; however, the curves do evolve a bit differently, most notably affecting their point of intersection and the fluences closest to the beam area. Analyzing and reviewing this data will be an important task in upcoming months, as FNAL conducts a review on radiation simulation results.

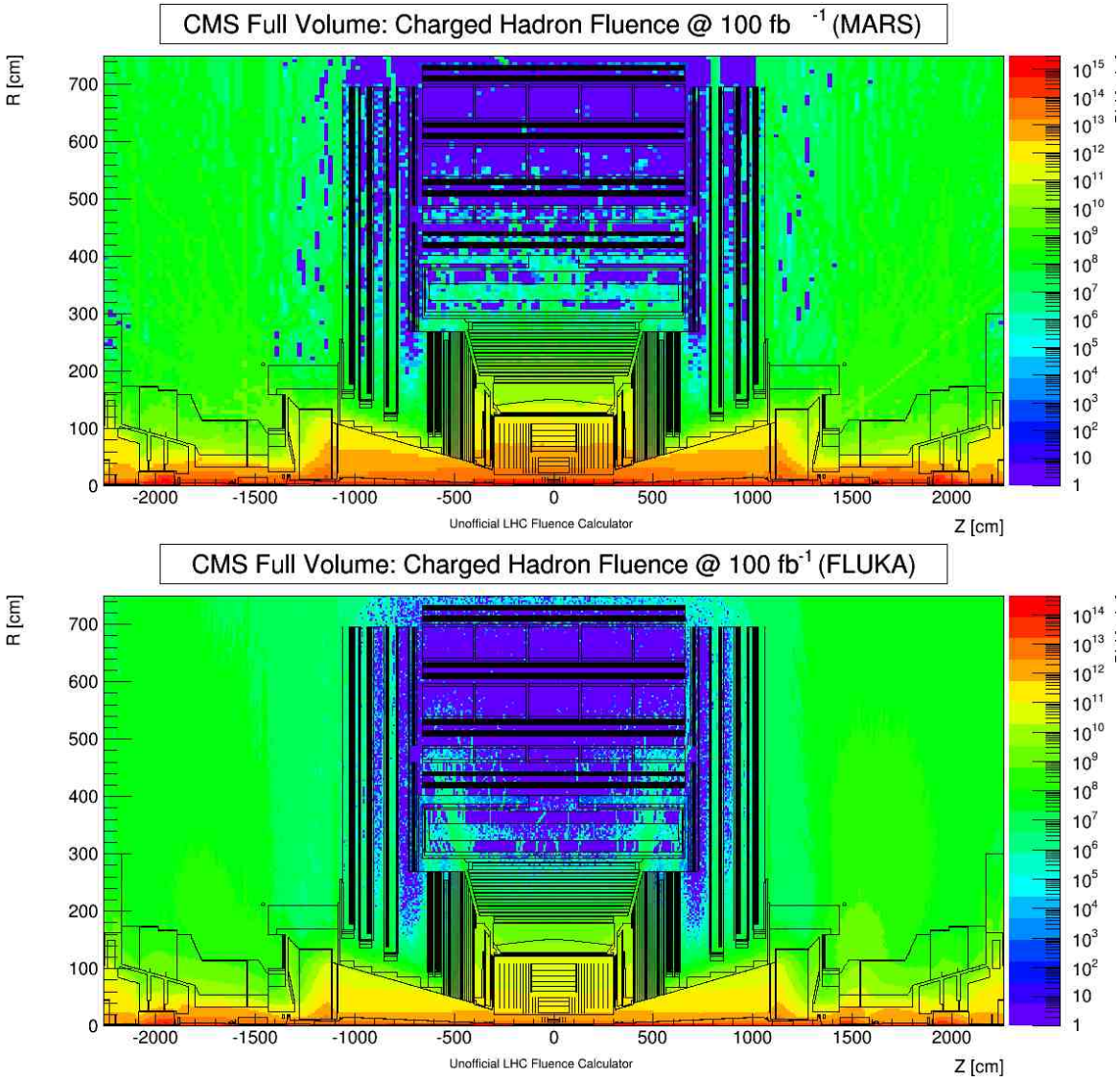


Figure 10: 2D Histogram comparison of charged hadron fluence between MARS and FLUKA.

The data between the two sets are fairly consistent, and again we see the difference in fluence exacerbated nearest to the beam. The enhanced detail in the FLUKA map is a consequence of the finer data granularity and an enhanced number of simulated pp collisions.

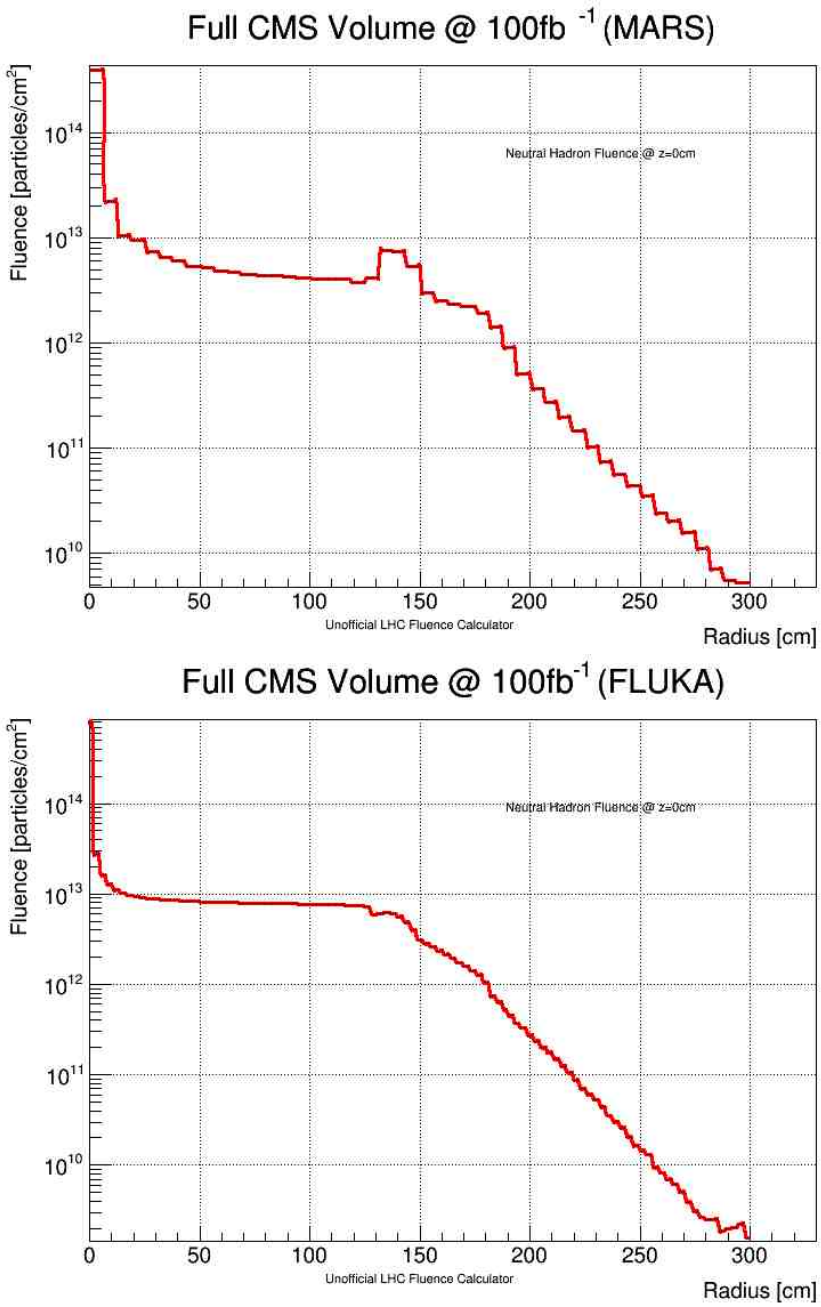


Figure 11: Comparison of radial slice graphs between MARS and FLUKA with 1 cm step size.

The plots in Figure 11 exhibit the usefulness of a finer data granularity. The step size of both graphs is 1 cm, but the MARS data granularity is around 6 cm, meaning that the data at R=0–5 cm is the same for every point. Therefore, as a consequence of the 1 cm step size, each data value is repeated six times, creating a graph that is rather difficult to follow. FLUKA's finer granularity allows for a much smoother plot with a smaller step size. In addition, the MARS data has a spike around 135 cm, which likely represents a shift into the electromagnetic calorimeters, that is not reproduced as strongly in the FLUKA data. This difference has to be further investigated.

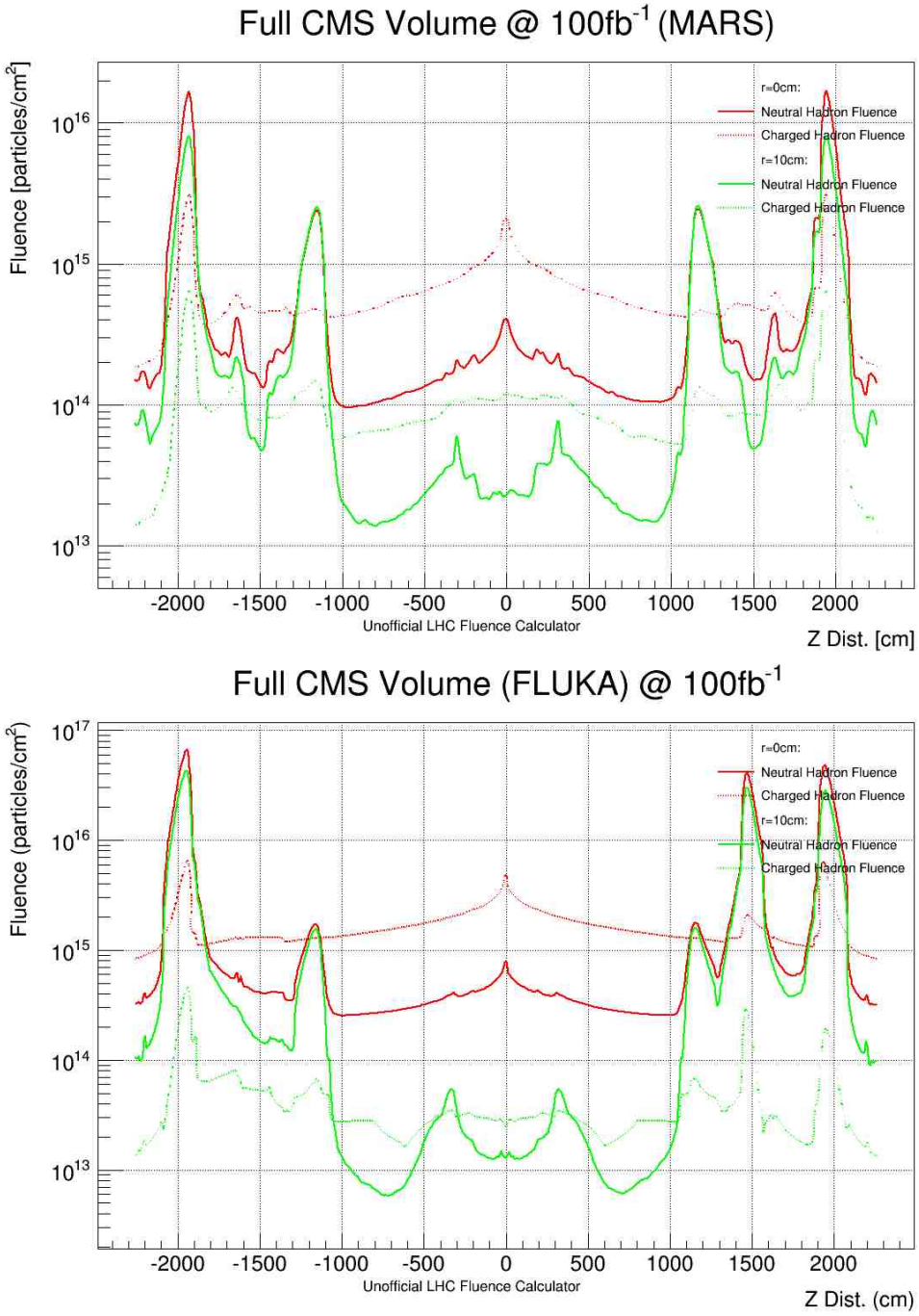


Figure 12: Comparison of Z-slice graphs between MARS and FLUKA data with 8 cm step size.

The plots in Figure 12 show the similarity in the MARS and FLUKA data when slicing along Z. We can see that the spikes in the data due to geometry effects are very similar in the two data sets.

2.5 Future Work

It is simple to add collisions from additional FLUKA runs into the existing data set, as long as we keep the original data and run a similar number of runs on each node, each run performing a similar number of primaries. However, changing the number of bins requires building an entirely new dataset. Still, given the relative ease of running FLUKA jobs on the grid, it is not unfeasible that a later project could produce an accurate dataset of the detector's full geometry with a binning grid of $(1, 1)$, especially if running CRAB jobs rather than Condor jobs. This would require filling 3395271 bins and would significantly increase server load for slice graph and 2D histogram construction. We very seriously doubt the need for this large-scale of an operation, however, unless there is a specific need for it; a slight change in the CMS geometry would require modification of our input file and re-running of all our data. In addition, it may be useful in future runs to use lower electron/positron and delta ray thresholds, to at least match the MARS data in this respect.

Users wishing to perform similar FLUKA runs at FNAL should see Appendix B.

3 Appendix A: Software Manual

3.1 Web

3.1.1 LHC_DoseFlux_CalculatorTest.html

Uses: lhcCalc.php, lhcGraph.php

The main index page for the calculator tool. Original design (using HTML tables) scrapped; now features an entirely CSS-driven layout with JavaScript powering hidden displays.

This page is a single form with multiple submit buttons, each calling a PHP script. To generate plots, PHP scripts execute C++ code and source the resulting image.

3.1.2 lhcCalc.php

Uses: lineGraph

Originally three scripts; all were combined into one object-oriented script, and modular, object-oriented coding cut away 100 lines of code while greatly increasing readability. The code has since been extended greatly.

This PHP script handles three features of the Dose/Fluence Calculator: the point calculator, the table/linegraph feature, and the slice graph feature.

NIEL fluence for MARS data cannot be directly measured, as it requires information (energies of individual particles) that MARS does not store. Instead, the value of NIELS fluence is approximated, using Equation 1 in Section 2.1.

3.1.3 lhcGraph.php

Uses: jQuery, plotFluence, FetchData.php

Reads in values from the HTML form (the values input by the user on the webpage) and executes plotFluence. Once the interactive histogram is created, the image is sourced and an event is attached to the Javascript event `onmousemove` to display the image x-y coordinates at mouse position, conditional upon hovering over the plot area.

On click, an AJAX request is submitted to the server and JavaScript is used to display the fluence at the mouse position in the table located below the image, using `FetchData.php`.

JQuery is used to here to get the offset of the image from the top and left of the screen; this allows for the interactive image to be centered.

In order to determine where the mouse cursor is on the plot area, some calculations (depending on the image size in px) are necessary. A ratio for each axis, scalex and scaley, is defined as the ratio of the axis length in px (a fixed quantity fo rall images) to the axis length in units of CM (determined by the user imput for rMin and rMax). A few simple simple addition/subtraction operations are then performed to account for the portion of the image that is 'dead' (ie, is not the plot area) and the offset from the left and top side of the webpage.

3.1.4 fetchData.php

PHP file executed by the AJAX request submitted by lhcGraph.php. Connects to the appropriate MySQL database and returns the appropriate data to display in the table.

3.1.5 CSS/lhcCalcStyleTest.css

Master stylesheet for all web content. See table below. All divisons marked with (h) are hidden by default and shown through JavaScript.

Division	Description
wrap1	contains topbox, tablebox, calcbox, and graphbox
topbox	main window
helpbox	governs layout of help.html
tablebox	(h) shown when table is selected
calcbox	(h) shown when calculator is selected
graphbox	(h) shown when interactive graph is selected
slicebox	(h) shown when slice graph is selected
noGraph	(h) shown when datatable graph style is 0
singGraph	(h) shown when datatable graph style is 1
multiGraph	(h) shown when datatable graph style is 2-5
sliceform	(h) shown when slicebox options are selected
wrap2	used by lhcGraph.php and lhcCalc.php
mapTable	fixs displace of interactive graph table
mapImage	fixes display of interactive graph
footer	governs footer on index

Table 3: Table showing descriptions of divisions in the Calculator's Cascading Style Sheet.

Height and width values are given in fixed sizes (using units of ems), to allow for uniformity of appearance on all machines.

`help.html` and `lhcGraph.php` also use small amounts of inline styling, set by the HTML style attribute.

3.1.6 help.html

Help page for the calculator; links to the README and gives some information about data statistics.

3.2 Executables

These programs are called through PHP and ran on the cmstrk server.

3.2.1 plotFluence

Uses: Geometry.dat

Originally written by Charles Dietz from National Taiwan University; we took one executable, plotFluence, and modified it to suit our needs. Takes a range of Z/R values, luminosity, detector volume, fluence type, and several boolean switches and creates a ROOT TH2DHisto (_proj) or ROOT TP2Dhisto (_histo) in .jpg format.

Currently all graphs are output at the same fixed size (1,196px 572px), regardless of Z-R ranges given submitted by the HTML form. The output image size can be easily adjusted by modifying the canvas size in BasicPlots.cc. If image size is modified, then the JavaScript in `lhcGraph.php/FetchData.php` must be reparameterized.

plotFluence reads its data from a set of .txt files containing radiation simulation data. The files are as follows, inside cmstrk at `/var/www/html/radsim/`:

Data Set	**File**
MARS (full)	mars1.txt
MARS (trk)	FluencePackage2/mars2.txt
MARS (cal)	FluencePackage2/mars_new.txt
FLUKA (full)	Fluka_Full.txt
FLUKA (trk)	FluencePackage2/Fluka_Track.txt
FLUKA (cal)	FluencePackage2/Fluka_Cal.txt

Table 4: Table showing plaintext data files accessed by `plotFluence`.

If it is necessary to update the FLUKA data files (for instance, to include more statistics, or to do another run with a finer granularity), this can be accomplished by running the script `readFluka.py`. Details about this process can be found in Appendix B.

plotFluence consists of three main scripts:

1. plotFluence.cc: processes arguments, sets up variables, creates objects, and calls methods. Returns 1 on success and 0 on fail.
2. BasicPlots.cc: sets up binning, granularity, and graph options and plots the histograms. Most any modifications to the graphing system will be done on this script, unless changing the way data is read in.
3. HandleFluenceData.cc: reads in data from the plain text files on the server; stores it in a vector of vectors that is used by BasicPlots.cc.
4. Geometry.cc: loads in CMS geometry from Geometry.dat and plots in on the detector maps.

Each file also has an associated header file. `Geometry.cc` and `HandleFluenceData.cc` are unchanged from the original FluencePackage. `BasicPlots.cc` has been modified marginally, and `plotFluence.cc` has been completely rewritten.

3.2.2 lineGraph

Written by John Farmer and Jake Callahan.

Reads in a space-delimited text file of fluences (ignoring the first row, used for human-readable column headings).

Example text file:

```
Radius Neutral Hadron Fluence  Neutral EMS Fluence
10 28767535200000 1.61388792E+14
```

```
15 13372756800000 71725039200000
20 13367640000000 51217200000000
25 13367640000000 51217200000000
30 10513843200000 38998773600000
35 10076553600000 35433840000000
40 8818470240000 31359883200000
45 8191012800000 28209410400000
50 8191012800000 28209410400000
```

When read, `lineGraph` sees two columns of fluence data and will create two superimposed linegraphs.

Functionality has been extended to include slice graphs: this overlays, over the same R value, several fluence plots from different Z values along the detector.

The web interface allows the user to select three options:

1. spline or curve fit
2. show data points or no data points
3. linear Y scale or logarithmic Y scale
4. Data table or no table

Selecting spline will cause ROOT to draw a straight line from each data point to the next. Selecting curve fit will fit a function to the graph. In most cases, a curve-fit logarithmic scale without data points is recommended.

3.3 Data

With the completion of the FLUKA simulations, FLUKA data is now the recommended set. It has more statistics and a finer granularity than the MARS data.

Data for plotFluence is organized into text files as mentioned above. This section details the SQL tables on the cmstrk MySQL database.

3.3.1 MARS

MARS SQL Data Tables					
Geometry	Database	Table	R Range(cm)	Z Range(cm)	Granularities (R, Z)
Full Volume	mars	mars1	[0, 750]	[-2260, 2260]	(6.25, 22.6)
Tracker	mars	mars2	[0, 110]	[-290, 290]	(0.92, 2.90)
Calorimeters	mars_new	mars_new	[0, 300]	[0, 560]	(2.00, 2.00)

Table 5: Table showing name, granularity, and detector span of MARS SQL data tables.

3.3.2 FLUKA

FLUKA SQL Data Tables					
Geometry	Database	Table	R Range(cm)	Z Range(cm)	Granularities (R, Z)
Full Volume	FLUKA	Fluka_Full	[0, 750]	[-2260, 2260]	(3.004, 7.535)
Tracker	FLUKA	Fluka_Trk	[0, 110]	[-290, 290]	(1.009, 2.0003)
Calorimeters	FLUKA	Fluka_Cal	[0, 300]	[0, 560]	(2.00, 2.00)

Table 6: Table showing name, granularity, and detector span of FLUKA SQL data tables.

3.4 Misc.

3.4.1 Geometry.dat

The data file `Geometry.dat` is populated with the point-by-point coordinates of the CMS Detector's physical components. When read by `Geometry.cc` in `plotFluence`, the outline of the detector is constructed, scaled appropriately, and written to the histogram.

3.4.2 Other

Most of the scripts and programs use an absolute path to direct to `cmstrk.fnal.gov:/var/www/html/radsim` or `cmstrk.fnal.gov:/var/www/html/radsim/FluencePackage2`. In order to update the website, all that is required is to copy `LHC_DoseFlux_CalculatorTest.html`, `CSS/lhcCalcStyleTest.css`, and `help.html` to the desired live location.

4 Appendix B: Submitting FLUKA Jobs on the LPC Batch System

4.1 Prerequisites

To access the user analysis farm, a connection with the LPC cluster must be established with a valid FNAL ID through Kerberos. Any Linux machine at FNAL or personal computer with gssapi and Kerberos is capable of connecting to the cluster once both CMS and LPC permissions have been enabled to the unique user ID. To set up permissions for LPC or CMS, a service request can be submitted to the FNAL service desk if an FNAL account has already been set up.

From any FNAL onsite Linux machine, a user can access the LPC Cluster by secure-shell to `username@cmslpc-sl5.fnal.gov`. From here, each user has access to the shared Blue Arc Mounted Data areas. It is important to take note of the amount of storage space alloacted in each area; while a user recieves roughly 3TB of data, only 2GB of storage is available on the `/uscms/home` directory for each user. For more storage space when working with sizeable amounts of data, the `/uscms_data` areas are accessible and provide a substantial amount of storage. However, running batch jobs that access data from a shared filesystem is not recommended since it increases the load heavily on user nodes.

For especially large jobs, it may be necessary to use the LPC batch production farm or CRAB Systems. Both types of jobs require an up to date grid certificate from CERN and a VO membership.

4.2 FLUKA

In order to download FLUKA, users must register with FLUKA and obtain a FLUKA User ID (FUID). If manually compiling FLUKA, g77 will be necessary. FLUKA 2011.2b.2 has already been installed in `/uscms/physics_grp/lpcfluka/Fluka2013`; however, group privileges with lpcfluka may be necessary to write to this directory.

Creating Fortran Input files for FLUKA is a daunting process, but usingFlair, the user-developed GUI for FLUKA, simplifies it and eliminates the potential for syntax errors. As of this paper, only cmslpc42 is set up to run Flair.

With Flair, it is easy to insert an external geometry file and view it with the geoviewer. Additionally, users can use a preconstructed input file with the geometry already added. For example, an input file with the CMS Detector's geometry pre-loaded can be found at: `/uscms/physics_grp/lpcfluka/Fluka2013/CMSpp/CMSpp.inp`. This file is pre configured for 7TeV pp collisions in the CMS Detector.

Once FLUKA is unpacked and installed, a directory will be created housing serveral of the pre-packaged FLUKA scripts and executables executables. A detailed explanation of all the executables can be found in the FLUKA manual or the README file in the FLUKA directory. Here we will describe the basic ones we used:

- rfluka: this is the command for running FLUKA; arguments may be viewed using the -h option. The user must set the FLUPRO environmental variable to the working directory; on CMSLPC, this is the Fluka2013 directory. Data from these individual FLUKA runs must then be combined and processed.
- `usbsuw.f`: USRBIN processing script. Combines the results of the FLUKA "fort" output files for USRBIN scoring based on the number of collisions. This execuatable can be used for multiple jobs and includes the standard deviations. It is important to note that the output of this executable will be a binary file that must converted to a standard ASCII output with `usrbrea.f`.

- `usbrea.f`: USRBIN processing script. Converts the `usbsuw.f` binary output to the standard FLUKA ASCII output.

For more information, see the FLUKA Course lecture on estimators and scoring.

Next, we will give a basic description of Flair/FLUKA features. To start building an input file, either load a pre-built input file using Flair or create a new one. To new users, it is often confusing that there are two important files associated with running FLUKA through FLAIR: the input file, which FLUKA uses, and the .flair project, which Flair produces. On the left hand window some important elements for running jobs are as follows:

- General → RANDOMIZ: This card sets the random number seed for the estimators. If this number is not changed between each job on the batch system, the results produced by every job will be identical. The CRAB System automatically creates a unique random number seed for each job. Our script `condorSubmit.py` handles seed randomization for Condor submision.
- Primary: Listed in this section are the beam details, such as energy, position, and particle type, as well as the number of primaries (collisions). This, combined with SPECSOUR, gives our beam properties.
- Transport → EMFCUT, THRESHOL, PART-THR; PHYSICS → DELTARAY: Threshold transport cut-offs for various regions.
- Scoring: determines what outputs we receive. We used USRBIN scoring to receive binned fluence/dose outputs. The outputs from each scoring card will be written to the unit specified in the card; specifying unit 41, for instance, will write the USRBIN output from that card to a file ending in "_fort.41". The units we used are shown in Table 7; all our processing scripts are set up for these units, so a simialr setup is suggested, but said scripts could be modified for any unit setup by someone with a basic grasp of Python.

Unit	Scoring
41	HAD-NEUT
42	HAD-CHAR
43	PHOTON
43	ELECTRON
45	MUONS
46	NIEL-DEP
47	ENERGY
48	DOSE

Table 7: Table showing unit binning setup for the FLUKA run.

- Run: allows the user to run small scale jobs on the current machine. The user may specify a number of cycles for each primary, which changes the random number seed and runs the simulation under the same conditions. This is only recommended for small test runs, as the processing time required to generate a useful amount of statistics is far too much for a single machine.

The user may also plot the results of the simulation using the the built in PLOT tool on the left toolbar.

4.3 Condor Configuration

To submit a job to the LPC batch farm, users must create a condor submission file. A basic submission file follows the form:

```
## universe type
universe = vanilla
Executable = /uscms/physics_grp/lpcfluka/Fluka2013/flutil/rfluka
Should_Transfer_Files = YES
WhenToTransferOutput = ON_EXIT
## Any environmental variables
## When running FLUKA, specify FLUPRO
environment = FLUPRO=/uscms/physics_grp/lpcfluka/Fluka2013;
notify_user = YourEmail@YourEmailProvider
## Input Arguments
Arguments = -N0 -M1 /uscms/home/jakec/CMSSW_5_3_2/FlukaGridFull
## Number of Jobs to Run (always 1 if using condorSubmit.py)
Queue
```

Some options to consider are the requirements of the computer the job will be run on, whether to output errors for troubleshooting, and the location of the `Condor-Scratch_Directory`. If the grid production farm needs to be used, the universe should be changed to either globus or grid and the grid resource must be specified. A proxy must also be generated.

Once the job configuration file has been created, the file can be submitted from the working directory with the command `condor_submit "your_file"`. All jobs running on a particular node can viewed by using the command `condor_q`. This is useful for apporximating the current load of the node and deciding where to run the jobs so they complete in a timely manner. To just view the individually submited jobs, run the command `condor_q "your_usr_name"`. Jobs are marked as either idle/"waiting", running, or held. Held jobs typically have errors in the submission file and will be removed. Jobs may be removed from condor at anytime with the `condor_rm` command. Either specify the username after the command to delete all jobs by a particular submitter or specify the job number to delete a particular job. Job output is always returned to the working directory unless the `$_Condor_Scratch_Directory` options are modified.

Condor jobs, when not specifically directed otherwise, will output job results into the current working directory. Jobs run from any Blue Arc shared space such as `/uscms_data/` and `/uscms` can slow down the system for all users, especially when utilizing lots of open file desciptors. As a result, large jobs that require an immense amount of CPU time may be held or even deleted by the Tier 1 team. A good alternative to submitting the default condor file on the Blue Arc Space is to use the `\char36\_Condor_Scratch_Directoryvariable` to redirect the output data to a dCache space.

Running jobs on the LPC Batch System can be a great way to run large, time-consuming jobs on the grid, when done correctly. To set up a job with a large number of primaries, it's important to break-up the job into smaller partitions accordingly. DO NOT submit a job with a large number of primaries. Jobs with low numbers of primaries can be combined for the same results as a job with a large number of primaries and will finish much more quickly. Large Jobs will take a long time to complete and a lot of computer resources since the job is run from a single cluster; indeed, it is likely that large jobs will be killed for excess run duration. Instead, break a job into pieces. For example, submit 100 jobs on 5 nodes with 30 primaries each instead of one job with 15,000 primaries on a single node. How to perform this will be outlined later on.

For jobs that are simply too large for the LPC Batch Farm and need access to the cmsenv, CRAB may be a good alternative. The Crab system is theoretically faster than the LPC Batch Farm and as

a result may be better for jobs with very close deadlines.

4.4 Scripts and Data Processing

Several scripts have been created to make submitting the jobs on the LPC Batch Farm faster and more user friendly.

`condorSubmit.py` provides the user with a quick and easy way to submit and organize a large number of jobs on multiple nodes for the LPC Batch System. This script, when placed in a directory with the FLUKA input file and the condor submission file, will create a directory of directories to store the results of the all the submitted jobs. In addition it creates a new input file for each job with a new RANDOMIZe card. The user simply enters the following command to run the script: `python condorSubmit.py Arg[1] Arg[2] Arg[3]`. The first argument is the stem of the FLUKA input file: for example `FlukaInput` for `FlukaInput.inp`. The second argument is the number of jobs to run and the third is the node. The node selection has no bearing on the execution of the job; it is just for organizational purposes, as all the data from each condorSubmit will be neatly contained in a directory named Argv[1]+Argv[3]. This directory will be populated with directories numbered for each job and filled with the output data and the unique input file. The user will need to edit the script for his or her condor submission file name (by default `sleep_condor`).

Ex: `python condorSubmit.py FlukaGridTest 200 25`

`suwRun.sh` descends into the specified directory and runs the FLUKA executable usbsuw on all all fort files ending in 41-48 and puts the output in a user created `out` directory. This script can be modified to:

1. change the executable for a different type of scoring
2. change the bin numbers for as many scoring options as selected
3. change the output directory

This executable allows the user to combine the results of the jobs and place them in an easy to manage directory.

Ex: `bash suwRun.sh FlukaGridTest25`

`suwComp.sh` is run within the output directory the results of `suwRun.sh` were placed in. It compiles the output files based on the bin number argument (41, 42, 43, etc...) and produces one summed binary file for all of the jobs. `usbrea.f` can then be run on each of these newly created files to get the standard ASCII output complete with averages and standard deviations.

Ex: `bash suwComp.sh 41`

`readFluka.py` is a job-specific python script that organizes the data from `suwComp.sh` into two text files: one unaltered for a database, such as MySQL, and another with appropriate scaling factors. Currently readFluka.py reads eight unformatted Fluka ASCII files produced by `usbrea.f` for neutral hadron fluence, charged hadron fluence, ems fluence, charged ems fluence, muon fluence, niel fluence, total energy deposition, and absorbed dose. It then creates one tab-delimited and one space-delimited text file. It can easily be modified based on the types of scoring for a variety of simulations.

Ex: `python readFluka.py 0 110 110 -290 290 290`

4.5 MySQL

MySQL offers an easy way to store large amounts of tabular data. A MySQL table populated with the MARS and FLUKA simulation results is available on `root@cmstrk.fnal.gov`.

Once MySQL is installed, databases and tables can easily be created to manage data for easy access from server side scripts like PHP for web applications.

Following is an example script of how to connect to the MySQL database using PHP:

```
CREATE DATABASE DATABASE_NAME;
```

To create a table, enter (from within the database):

```
CREATE TABLE TABLE_NAME (column1 VARCHAR(20), column2 VARCHAR(20), column3
VARCHAR(20));
```

To load data from a simulation into a MySQL table, ensure the data is tab-delimited and organized in the same format as the MySQL table, then enter:

```
load data local infile '/path/to/file/FlukaOutput.txt' into table MyTable;
```

`readFluka.py`, when given a FLUKA output file generated by `usbrea.f`, will output a file structured for input into SQL.

To connect to a MySQL dataset from PHP, enter:

```
$link = mysql_connect('localhost', 'user', 'mysqlpassword');
mysql_select_db("my_database") or die(mysql_error());

if (!$link)
{
  die('Could not connect: ' . mysql_error());
  exit();
}

$query = "SELECT * FROM My_table ORDER BY ABS(column1-($var1)),
ABS(column2-($var2)) LIMIT 1";
$result = mysql_query($query) or die('Query failed: ' . mysql_error());

if($result)
{
  $row = mysql_fetch_array($result, MYSQL_ASSOC);
  $id = $row['run_id']
}
else
{
  printf("Could not retrieve records: %s\n", mysqli_error($mysqli));
}
mysqli_free_result($result);
mysqli_close($link);
```

4.6 CRAB jobs

While CRAB jobs may be more difficult to set up, they allow for access to a much larger and more powerful grid. Indeed, in order to view many of the twiki help pages for CRAB and register for the CMS VO through CERN, users must apply for a CERN account.

Depending on the computing environment, it may be neccesary to source the environmental data to the machine you are currently working on. The process is outlined in depth on the Twiki pages; however, if you are using an SL5 machine, a CMSSW enviornment is neccesary to run CRAB as well as the following commands: `source/uscmst1/prod/grid/gLite\_SL5.csh`, `source/uscmst1/prod/grid/CRAB/crab.(c)sh`

Before jobs can be run, the user must take his or her grid certificate and export it as a .p12 file, copy the file to cmslpc-sl5.fnal.gov and run the openssl commands to generate usercert and userkey. Then the user can generate the neccesary proxies to run CRAB with by running voms-proxy-init -voms cms.

The crab -create and crab -submit commands are used to create a CRAB submission directory for each job. Details for this process can be found on the twiki.

5 Bibliography

[1] P.C. Bhat, N.V. Mokhov, A.P. Singh, and S.B. Beri. Simulation Studies of the Radiation Environment in the CMS Detector for pp Collisions at the LHC. CERN-CMS-NOTE-2013-001, 2013.

[2] D. Haznar and P. Sharma. CMS Radiation Simulations Data (MARS15), 2011. `cmstrk.fnal.gov/radsim/LHC_DoseFlux_Calculator.html`.

[3] J. Farmer and J. Callahan. Radiation Simulation Data @ 14TeV, 2013. `www.uscms.org/uscms_at_work/dmo/siTracker/lhcCalcForm.html`, dev. test page at `cmstrk.fnal.gov/radsim/LHC_DoseFlux_CalculatorTest.html`.

[4] N.V. Mokhov. The MARS Code System User's Guide. Fermilab-FN-628 (1995).

[5] N.V. Mokhov and O.E. Krivosheev. MARS Code Status. Proc. Monte Carlo 2000 Conf., p. 943, Lisbon, October 23-26, 2000; Fermilab-conf-00/181 (2000).

[6] N.V. Mokhov, K.K. Gudima, and C.C. James et al. Recent Enhancements to the MARS15 Code. Fermilab-Conf-04/053 (2004); `http://www-ap.fnal.gov/MARS/`.

[7] G. Battistoni, S. Muraro, P.R. sala, F. Cerutti, A. Ferrari, S. Roesler, A. Fasso', and J. Ranft. The FLUKA code: Description and benchmarking. Proceedings of the Hadronic Shower Simulation Workshop 2006, Fermilab 6–8 September 2006, M. Albrow, R. Raja eds., AIP Conference Proceeding 896, 31-49, (2007).

[8] A. Fasso' A. Ferrari, P.R. Sala and J. Ranft. FLUKA: a multi-particle transport code. CERN-2005-10 (2005), INFN/TC_05/11, SLAC-R-773.

[9] Charles Dietz. Fluence package. Cern TWiki: `https://twiki.cern.ch/twiki/bin/vewauth/CMS/FluencePackage`.

[10] Rene Brun and Fons Rademakers. ROOT - An Object Oriented Data Analysis Framework. Proceedings AIHENP'96 Workshop, Lausanne, Sep. 1996, Nucl. Inst. & Meth. in Phys. Res. A 389 (1997) 81-86. See also `http://root.cern.ch`.

[11] V. Vlachoudis. FLAIR: A Powerful But User Friendly Graphical Interface For FLUKA. Proc. Int. Conf. on Mathematics, Computational Methods & Reactor Physics (M&C 2009), Saratoga Springs, New York, 2009.

[12] CASTOR Forward Calorimeter. CERN Twiki: `https://twiki.cern.ch/twiki/bin/viewauth/CMS/CASTOR`.

5.1 Additional Acknowledgements

- Jake Callahan from Iowa State University, coworker; cowrote the code and contributed the original draft of Appendix B.
- Pushpa Bhat and Leonard Spiegel, project supervisors.
- Patrick Gartung, for technical support.
- Sudeshna Banerjee, for comments on the paper draft.
- The SIST (Summer Internships in Science and Technology) Committee.