Design of the OLYMPUS simulation

J. C. Bernauer[1] and A. Schmidt[1]

[1]*Laboratory for Nuclear Science, MIT, Cambridge, Massachusetts 02139, USA*

(Dated: February 18, 2013)

This paper is divided in two parts. First, we present our design for the OLYMPUS simulation. This design builds on the ideas that were discussed during collaboration meetings in the early phases of OLYMPUS. It extends the philosophy of the analysis framework, using plugins for modularity and a common framework to enable code reuse. In the second part, we will reflect on some basic principles of Monte Carlo simulations and how those principles guided our design. We introduce the Monte Carlo technique for integration and connect this to our goal of extracting a ratio in an efficient and accurate way.

I. SIMULATION DESIGN

A. Overall Data Flow

To gain an understanding of what is required of the simulation's design, it is worthwhile to revisit the structure of the current analysis. Figure 1 shows a chart representing the data flow. The experiment produces raw data, to which we apply track reconstruction software to generate a tree of "cooked" data. Physics analyses are applied to this cooked data to produce results.

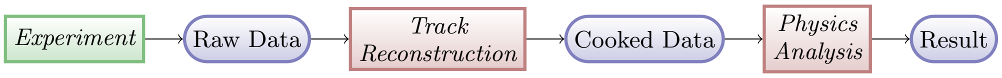

FIG. 1. The data analysis chain: rectangular red nodes represent a software process, while the rounded blue nodes represent some type of data object.

The OLYMPUS simulation must be situatied within this design. We start with an event generator that produces the initial conditions of each event. This generator needs to include radiative effects and must produce a lepton, a proton, and a (soft) photon. These initial conditions are fed into a module, which propagates the event through the detector system, calculating the effects of matter in the paths of the particles. In the initial iteration, these particles will be propagated using the Geant4 framework.

The end result of the propagation step should be a root tree which contains all information from the track propagation, such as true intersection points and times, energy losses and so forth. The tree already in the simulation may be used as the basis for this, but must be revisited to make sure all of the needed information is included.

The first application of the simulation is to test and extend our understanding of the detectors and reconstruction methods. For this, we need to *digitize* the simulation tree and produce a fake data tree with the same format as the raw data. Then the standard analysis chain can work on this data and a comparison with the true input parameters is possible. This is depicted as path I in Fig. 2.

By comparing the result from real data to that from the fake data, we can refine our understanding of the experiment. This process must be iterated until the simulation reproduces all features of the data. At this point, we can be confident that our model of the experiment is sufficiently sophisticated for reliable extraction of the form factor ratio.

In principle, this brings us to a state where the ratio may be extracted; the ratio from experimental data divided by the ratio from the simulation is a direct measure of the two-photon-exchange contribution. In this approach, we have already accounted for effects from beam position, slope, etc. and made an external application of radiative corrections superfluous.

The approach in path I is likely to be computationally expensive, making this analysis chain both time and resource intensive. However, it is possible to accelerate the simulation at least for a subset of the required studies, if not even for the full analysis. From the initial studies of detector performance and track reconstruction, it possible to emulate the behavior of this part of the chain with an appropriate detector response model and create faked cooked data directly. This will speed up the simulation data chain immensely. This latter approach is shown as path II in Fig. 2.

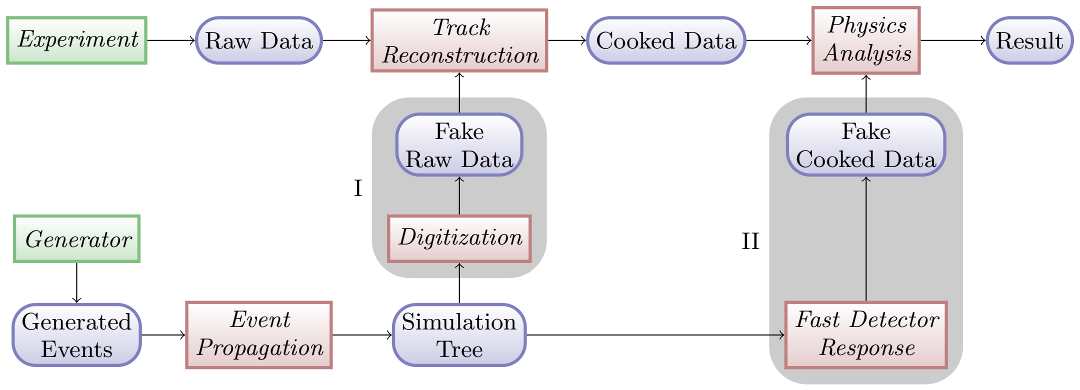

FIG. 2. Here's the data analysis chain including MC.

B. Task List

To realize this design, we identify tasks which must be completed. These tasks include those which must be executed by the individual detector groups. We limit ourselves here to only a coarse list—the individual tasks must be broken up by the coordinator of the responsible group into smaller problems suitable for an individual project.

- Radiative generator
- Simulation output tree
- For each group:
  - Accurate realization, e.g. geometry, in the simulation
  - Full digitization
  - If needed, emulation of the detector response
- Refine physics analysis

II. NOTES ON THE MONTE CARLO METHOD

A. What is a Monte Carlo simulation?

The word simulation is a bit misleading, since the general problem one has to solve is to describe the acceptance function of the detector system (including efficiencies, etc.). This problem leads to a multi-dimensional integral. While it might be possible with immense effort to solve the integral symbolically, a numerical approach is normally used.

There are very efficient numerical algorithms for integration in one dimension. However, for integration in multiple dimensions, all algorithms based on a grid tend to converge with $\mathcal{O}(n^{-1/d})$, where n is the number of calculations and d is the number of dimensions. For a typical case, d can be very large so that all of these algorithms converge very slowly.

Monte Carlo (MC) is a method of numerical integration employing a (pseudo) random sampling over the integration volume, which converges with $\mathcal{O}(n^{-1/2})$, albeit with a worse constant factor. Several techniques can be used to optimize this further (see the appendix), but for a higher number of dimensions, MC is almost always faster. Another advantage is that MC can be continuously refined.

B. Correction or Simulation?

The standard way to get a cross section is to divide the event count by the integrated luminosity and the detector acceptance A,

$$\frac{d\sigma}{d\Omega} \sim \frac{\Delta\sigma}{\Delta\Omega} = \frac{N}{\mathcal{L}_{\text{int.}} A}.$$

The result is the differential cross section averaged over the integration volume. A is typically found using the MC approach described above. Practically, these integration volumes are bins in a histogram. The difference between differential cross section and its average over that bin can be made arbitrary small by an appropriate histogramming choice.

This approach, however, neglects the effect of counts being sorted into the wrong bin due to non-zero detector and reconstruction resolution. This effect can be large if the cross section varies widely between bins; a small fraction of events from a high cross section bin can swamp a low cross section bin. But MC offers a natural solution to this problem. A realistic cross section and non-zero resolution can be included via MC into a simulation, now producing a simulated event rate. As long as the simulated rate and the measured rate are similar, their ratio yields information on how the cross section in the simulation differs from reality.

This philosophy of adding effects to a simulation for comparison with real data is extremely pertinent to the subject of "corrections." Historically, corrections for beam slope/position and radiation among others were made to measurement at the end of the analysis. Instead, these effects can be added to a simulation making a correction unecessary. The advantage is that a simulation will naturally account for the correlation between many different effects. If, say, the wire chamber acceptance correlates with beam position and beam slope, then a simulation will convolve those effects. Trying to make a manual correction that accounts for that convolution is tedious, especially if other effects are involved as well.

C. The relation between weights and cross section

We have had several discussions about unweighted and weighted simulations. The terms "weighted" and "unweighted" have often been used incorrectly and we feel that they obscure the important question: "Does the simulation include a cross section?" In this section we review how cross sections and weights are incorporated into simulations with the goal of establishing some consistent vocabulary.

The most basic example is an unweighted simulation without a cross section included. In an unweighted MC, every event contributes the same amount to the bin it gets sorted into. Every event is on an equal footing. To add a cross section to this simulation, the simplest approach would be to calculate a cross section for every event, and apply this cross section as a weight. Now the MC is weighted; an event adds its weight to the bin it gets sorted into. A single bin will be filled by events with different weights.

It is also possible to include a cross section without using weights. In this method events are drawn from a suitable non-uniform distribution. In this method, more samples will be drawn from a high cross section region than from a low cross section region.

As a final example, it is sometimes practical to introduce weights into an MC even when a cross section is not included. If one were to simulate the acceptance function of a detector in center-of-mass coordinates, but was interested in the result in lab coordinates, then one would calculate the jacobian of the transformation for each event and apply it as a weight.

The current plan for the radiative generator will include a cross section and will make use of a mixture of both weighted and unweighted methods to incorporate it. For variables over which we want to integrate, like photon momentum or angle, we will draw from an appropriate distribution, while the dependence on the lepton angles will be modeled using weights. The rational behind this is given in the appendix.

D. Radiative generator

The classical approach to handle the physics beyond the simple Born picture is to correct the intermediate result by means of one of the various radiative correction formulas, depending on an ill-defined cut in the radiative tail. However, this is suboptimal for several reasons. As noted above, this does not take any correlation with other effects into account. Furthermore, all these formulas need a well defined cut-off, which is hard for us to accomplish. It is therefore much better to actually use a radiative generator, which produces the correct cross section and kinematics including radiated photons from internal bremsstrahlung.

E. Time varying parameters

It is clear that the beam parameters like position and slope correlate with beam current and vary over time, even within single runs. This is probably true for many other corrections as well, since many of them correlate with beam current. MC is the ideal way to handle the complicated convolution of these effects efficiently and correctly; still, we must account for the parameter variation over time. There are two principal ways to organize this:

1. We chop up runs and find short time periods where the parameters are similar enough that we can use a common simulation setup: constant beam offset, slope, etc.

2. We extend the simulation to vary the simulated beam position etc. during the generation of events to follow their variation over time as given in the data.

It is obvious that 1 is tedious at best. What is "similar enough?" As we find a new effects, we would have to chop up the runs again. So, approach 2 seems superior. As a remark: it does not matter whether we run one big simulation with the complete development over time included, or a do a simulation for every run. The total number of calculated samples will be the same in either case. For the latter, we would just distribute the allocated sample count according to the integrated luminosity of each run. In any case, the number of MC samples should be large enough that we can neglect any error from the stochastic aspect of the MC integration.

For ease of use, we will follow the approach to simulate each run individually. This has the benefit that we can combine different sets of runs at will without having to rerun the simulation, saving time when doing these kind of systematic tests.

As described above, we will have to reach a point where simulation and experiment match. Then, for the final extraction of the ratio, we will build a double ratio of experiment and simulation,

$$R = \frac{\sum_i y_{\mathrm{exp}}^{e^+,i} / \sum_i y_{\mathrm{sim}}^{e^+,i}}{\sum_i y_{\mathrm{exp}}^{e^-,i} / \sum_i y_{\mathrm{sim}}^{e^-,i}},$$

with y the experimental or simulated yield in a given Q^2 bin. If the simulation includes everything except TPE, we get the result we are after: a measurement of the TPE contribution. If we include a model calculation of TPE, the result will tell us how good this particular model is at describing our data.

F. Modular design

We think it was very helpful to split up functionality of the analysis into plugins. It was rare that broken code in one plugin prevented people from using other parts. By contrast, the current simulation is monolithic, which makes independent development by the individual groups complicated. Given our good record using plugin architecture, we plan to integrate the simulation into the cooker framework, using plugins to realize the generator, the propagation, the digitization (including smearing) and the emulation modules described above. This has the added benefit that we can make use of all the cooker functionality and can possibly avoid duplicating code in the existing plugins.

By realizing the generator and propagation modules as cooker plugins, it becomes possible to use the slow control information from the raw data files to directly inform the simulation about the time-evolution of the experimental conditions. This can be augmented with information in the init files or secondary root files for parameters which can not be extracted easily.

III. SUMMARY

MC is an efficient method for properly accounting for many correlated systematic effects. By modeling the experiment as closely as possible, including the time variation of experimental parameters, we will improve the accuracy of our result. Our radiative generator will incorporate a cross section by using a combination of weighting and cleverly chosen probability distributions, minimizing the computational cost. Building on the experiences with the cooker framework for the analysis, we present a modular and flexible design for the OLYMPUS simulation, which will be adequate for all of our needs, from tuning the reconstruction to producing the final result. The modular design decouples expensive tasks from those that must be repeated often, and facilitates parallel code development. We list the tasks needed to realize this design and stress that it is the responsibility of the individual groups for developing the digitization and emulation of their respective detectors.

Appendix A: Error Analysis in MC

First, we would like to decouple the concept of single MC event from the concept of a scattering event in an experiment. In a simulation, the problem being solved is an integral, so the "events" that are simulated are, more precisely put, samples. We will refer to them as such in the Appendix.

A second point of clarification has to do with the uncertainty in MC integration. It is easy to get this uncertainty confused with the statistical uncertainty of the experiment itself (for which we typically associate a relative error of $1/\sqrt{N}$). The uncertainty of an MC integral reflects the deviation from the true value of that integral and is driven by the numerical and stochastic properties of the Monte Carlo method.

In the special case of constant weights and good (pseudo) random numbers, the expected relative error has an upper bound of $1/\sqrt{N}$ where N the number of events in a bin. This is number is only an upper bound because the knowledge of the total number of events generated reduces the statistical uncertainty. However the upper bound is a good estimate if both the number of bins and N are large.

If the events have non-constant weights, the error is enlarged, because the bin content now depends on the variance of the weights in a bin. A good upper bound for the expected relative error is $\sqrt{\sum w^2}/\sum w$. This increase in uncertainty can be demonstrated by an extreme example: suppose a bin is filled with 1000 events of weight 1, and three events with weight 3000. In this case, the first 1000 events make up only 10% of the complete bin content. Thus, the error of the bin in this example is dominated by the uncertainty of only a few events.

Since the error in a bin increases with the variance of the weights in a bin, it is desirable to have all the weights be the same, i.e. to make the variance of the weights zero. In order to make the weights constant, one needs to draw samples from a distribution proportional to the cross section. For a radiative generator, this is a hard task because it requires forming a cumulative distribution function and inverting it. Rather than attempting this on the actual cross section, one can choose an approximate distribution that is easy to invert. The weight of a sample is then given by the ratio of true to approximate cross section, which has a small variance. This approach was used in the generator for the Mainz proton scattering experiment with great success.

For OLYMPUS, this method does not solve all of our problems: if we generate samples according to a function having a similar lepton-angle dependence like the real cross section, we will have a simulation error with a similar shape as the data errors, rapidly increasing with larger angles. We would therefore waste a lot of computing time generating events for the low-angle bins in order to reach the desired precision for the high-angle bins. This can be mitigated by having a less strongly falling angular dependence and applying weights. While the weights would be quite different for the smallest and the largest angle, the weights in any given angular bin would be closer together. We would need to find a balance to get optimal efficiency. A different approach is to split up the angular range in regions and to combine the result from these separate simulation runs. A decision on which method is better can be made when the generator is in a working state.

There is one interesting technique to speed up the convergence of an MC. Instead of using pseudo-random numbers, one uses “quasi-random” numbers (also called low-discrepancy sequences), which are number sequences that tend to fill a region without “clumping.” The benefit is faster convergence, while the cost is greater difficulty in making an analytic estimate of the error. It is possible to mix pseudo- and quasi-random numbers for different dimensions of the integration.