Offline Software for the Mu2e Experiment

Robert K. Kutschke

Mailstop #234, Fermilab, P.O. Box 500, Batavia, IL, 60510, USA
E-mail: kutschke@fnal.gov

Abstract. The Mu2e experiment at Fermilab is in the midst of its R&D and approval processes. To aid and inform this process, a small team has developed an end-to-end Geant4-based simulation package and has developed reconstruction code that is already at the stage of an advanced prototype. Having these tools available at an early stage allows design options and tradeoffs to be studied using high level physics quantities. A key to the success of this effort has been, as much as possible, to acquire software and customize it, rather than to build it in-house.

1. The Mu2e Experiment

Within the Standard Model, muons decay almost 100% of the time to final states that conserve lepton family number (LFN); the very rare branching fractions to final states that violate LFN are much too small to be observed by any current or foreseen experiment. Therefore any observation of an LFN violating muon decay is direct evidence for physics beyond the Standard Model. One such decay mode may occur when a negative muon is bound to an atomic nucleus, forming a muonic atom: coherent, neutrino-less muon to electron conversion in the Coulomb field of a nucleus. The final state is a mono-energetic electron plus an unobserved, recoiling, intact nucleus, while all background processes produce electrons with a continuous energy spectrum. The Mu2e experiment\cite{1} will form muonic Aluminum, measure the energy spectrum of electrons from its decay and ask if an excess is observed at the conversion energy, about 105 MeV. The measured quantity in the Mu2e experiment is

\[ R_{\mu e} = \frac{\mu + Al \rightarrow e + Al \quad \mu + Al \rightarrow \text{all nuclear captures} }{ } \]  

The goal of the experiment is to achieve a single event sensitivity of \( R_{\mu e} = 5.4 \times 10^{-17} \), which is about 4 orders of magnitude better than the previous best upper limit of \( R_{\mu e} < 7 \times 10^{-13} \) at the 90% CL\cite{3}.

Figure 1 shows the major elements of the Mu2e apparatus, including the muon beamline and the detector elements. An 8 GeV proton beam enters from the right and strikes the production target, producing pions that decay to muons. The muon beamline is formed by a system of three graded-field solenoids and their associated collimators; the muon beamline captures the backwards going muons, reduces the flux of unwanted particles, and directs the muons onto the muon stopping target, a set of thin Al foils. Most muons that reach the stopping target range out in these foils and are captured to form muonic atoms. Electrons from the decay of the muonic atoms are directed by a graded magnetic field onto a tracking system and an electromagnetic calorimeter. The tracking system makes a precise \((\sigma_p/p \simeq 0.1\%)\) measurement of the momentum.
of charged particles that traverse the tracker. The calorimeter serves two purposes: together the calorimeter and tracker can reject several rare, but critical, background processes that cannot be rejected adequately by the tracker alone; the calorimeter also provides a redundant trigger for testing the primary, trigger-less data acquisition mode. Not shown in the figure are a system to monitor the extinction of the proton beam, a Cosmic Ray Veto system (CRV) and a system to measure the spectrum of X-rays emitted by muons as they cascade to the atomic K-shell; this last measurement is the primary input into the measurement of the denominator in $R_{\mu e}$.

2. Offline Software

The Mu2e offline software comprises code for simulation, calibration, reconstruction, analysis and event display; it also includes code to characterize the quality of results delivered by the reconstruction code when it is run on simulated events. At the present stage of development, there are only simulated events but the reconstruction code is being written so that it will work, as is, on experimental data, once that is available.

Studies undertaken to date include characterizing the joint spatial, temporal and momentum-space population of the particle fluxes entering the Detector Solenoid (DS), including the $\mu^-$ flux and the fluxes of all interesting background species; understanding how these fluxes depend on the designs of the magnetic field, the production target, and the collimators; computing realistic distributions of non-signal hits in the detector subsystems; computing the efficiency for reconstructing conversion electrons in the presence of the non-signal hits; computing the power of the detector to reject signal-like backgrounds produced by non-signal processes; computing the radiation and heat loads on elements of the apparatus; and redoing all of the above for variants on the design of the experiment.

For these studies, Mu2e has used four different software packages. G4beamline and FastSim permitted a fast start to many critical studies but they do not have all of the features needed for development of hit based reconstruction code. The missing features are present in the art based Offline software, Mu2eSim and Mu2eReco, which have recently entered into wide use. Finally, MARS has been used, and will continue to be used, when precision information about low energy neutrons is required. Each package is described in its own sub-section, below.

---

1 The proton beam arrives in bunches with a full width at the base of about 200 ns; the experiment requires that, for every proton within a bunch, fewer than $10^{-10}$ protons arrive between bunches. This ratio is known as the extinction of the proton beam.
2.1. G4beamline

Most studies of the Mu2e muon beamline have been performed using G4beamline, “a single-particle tracking program based on the Geant4 simulation toolkit. It is specifically designed for the simulation of beamlines.”[1] The Mu2e experience has been that G4beamline provides, as advertised, an easy-to-learn interface that enables effective use of Geant4 by those who are not trained to use it in its native form.

The essence of G4beamline is that it provides a user friendly layer on top of Geant4, allowing the user to think in terms of beamline elements without needing to know the intricacies of Geant4. The input to G4beamline is a file that describes the elements of the beamline, including the placement of materials, the source of primary particles, the magnetic field maps, and a set of virtual detectors. The input file also controls the behaviour of Geant4 by choosing a physics list, the values of user cut-off parameters and the algorithm used to integrate the equations of motion in a magnetic field. The operation of G4beamline is to digest the input file, instantiate the Geant4 geometry, and instantiate Geant4 sensitive detector handlers for each virtual detector. Next, it runs the event loop: for each event it generates primary particles and uses Geant4 to track those particles and their descendants. G4beamline summarizes, using ROOT ntuple, the position, time, direction and particle type of selected particles that pass through each virtual detector. G4beamline can also be configured to record every G4Step of selected particles.

Using G4beamline, Mu2e physicists were able to make a quick start at studies of the properties of the muon flux arriving at the stopping targets and, equally important, at studies of the properties of the many backgrounds that accompany those muons. G4beamline has also been used for studies of the heat and radiation shield, an element of the Production Solenoid (PS) system that protects the cryogenic magnet coils from the particle spray produced when protons strike the production target.

There are several aspects of G4beamline that limit its applicability for detector studies and for more detailed studies of the muon beamline. The G4beamline geometry language does not provide access to the full set of solids defined by Geant4; for example boolean solids are not supported. In addition, G4beamline does not keep a detailed parent-child history of what happened inside Geant4, which complicates the understanding of some features of the simulated events. Nor does G4beamline provide the infrastructure needed for the creation of realistic hits. All of these features are available inside Mu2eSim, which is described in section 2.3.

In summary, G4beamline has enabled a fast start and a continued fast pace in the design of the muon beamline; this work was done by physicists who understand the physics well but who have no training in either C++ or Geant4.

2.2. FastSim

FastSim[5] is a fast, surface-based, simulation and reconstruction tool originally developed by the BaBar collaboration[4] for use when very large Monte Carlo samples were needed, for example, when validating high-dimensional, unbinned maximum likelihood fits. Among many other features, this package includes the battle-tested BaBar Kalman filter code. FastSim was also used by SuperB experiment and the Mu2e version of FastSim was ported from the SuperB version.

As used by Mu2e, FastSim processes each event in three steps. First it simulates the response of the detector to a signal track, producing simulated hits; next it overlays frames of background hits; finally it fits the simulated hits using the Kalman filter.

The simulation step includes transport in a (possibly non-uniform) magnetic field, multiple scattering, energy loss, and bremsstrahlung in the detector materials; the models of scattering and energy loss include non-Gaussian tails. When a simulated track passes through a

---

2 The BaBar collaboration has given the Mu2e collaboration permission to use FastSim.
measurement surface, FastSim creates a hit, which is smeared by a function with a Gaussian
core plus non-Gaussian tails. A model of hit inefficiency can be applied at this time. Hits for
background frames are generated in a similar way. The result of the first two steps is an event
containing hits from both the signal and background sources.

The reconstruction step begins by using the Monte Carlo truth to identify the hits that
belong on the signal track. FastSim then looks along the true trajectory to identify hits from
background sources that lie close to the true trajectory; it has a model, tuned to \textit{BaBar} data, of
how often an incorrect hit from this sample should be added to the track and how often it should
replace a correct hit. The set of all hits believed to be on the track is then passed to the Kalman
filter which is configured to reject outlier hits. The result of this process is an ensemble of fitted
signal tracks that have a resolution function described by a Gaussian core plus non-Gaussian
tails; those tails contain contributions from multiple scattering, energy loss, bremsstrahlung, the
spatial resolution of the straw hits and from an important class of pattern recognition errors.

FastSim played a critical role early in the experiment, when it was used to optimize the gross
layout of the tracker and calorimeter. Recently it has been used to optimize the gross layout of
an alternate calorimeter design.

Finally, FastSim was used to study how errors in the modeling of the magnetic field might
give rise to biases and tails in the momentum resolution function. For this study, FastSim
integrated, along the trajectory, the shift in the momentum caused by the error in the field.
This technique produces a robust estimate of effects that are too small to be seen directly
by looking at the difference between generated and reconstructed quantities. This study was
performed to compute the required precision with which the magnetic field must be mapped
within the tracking volume.

\textit{Mu2e} made two major improvements to FastSim. First, the \textit{BaBar} Kalman filter did not
correctly deal with tracks that turn through multiple arcs of their helix; this has been fixed. This
bug affected only \textit{Mu2e}-like geometries, not geometries like those of \textit{BaBar} or SuperB. Second,
the non-Gaussian tails in the energy loss and scattering distributions were tuned to \textit{BaBar} data;
however some of the materials in the \textit{Mu2e} detector are much thinner than those in the \textit{BaBar}
detector and the models were discovered to be outside of their range of validity. These models
were improved to work correctly on thin materials.

FastSim does not have support for creation of realistic hits, including drift times in the straw
gas and time division to measure position along the straw; these features are present in \textit{Mu2eSim}
and studies that require detailed hit level simulations will be done using \textit{Mu2eSim}.

2.3. \textbf{Art Based Offline Software: \textit{Mu2e Sim} and \textit{Mu2eReco}}

When the \textit{Mu2e} effort at Fermilab began, the collaboration requested that the Fermilab
Computing Division (CD) supply and support infrastructure software for use by the
collaboration. The term \textit{infrastructure software} includes a framework\footnote{Here the word \textit{framework} refers to the entity that loads the dynamic libraries needed to do the requested work and then drives the event loop; other functions are not part of the framework proper.} an event-data model,
a persistency mechanism, run-time configuration, management of the state of random number
ingines, message logging, configurable responses to exceptions, and tools for the management
of singleton-like entities such as geometry and conditions data. It also includes the tools to
manage access to data bases, file catalogs and the GRID workflow management tools. \textit{Mu2e}
required one system that could be used for all purposes, from the lowest level non-real-time
task in the trigger/DAQ system through to final analysis; the steps in between include software
triggers, reconstruction, calibration, filtering/skimming of data sets, and simulation. Moreover,
the system had to be very analysis-friendly.

A team from CD recommended that \textit{Mu2e} adopt a fork of the CMS infrastructure software
(CMSSW), a choice that would leverage CD’s considerable investment in the development of
CMSSW. Some of the important ideas retained from CMSSW are the notions of modules, services, data products, the ROOT-based persistency mechanism, parameter sets for run-time configuration, the message logger and the patterns by which these entities interact with each other. Mu2e anticipates that its conditions data will be sufficiently simple that adequate functionality can be delivered using a service based model; therefore EventSetup was removed. Most other features of CMSSW were retained. Additional details are available [6][7].

Mu2e agreed to this plan and, in early 2009, started to use the first generation fork from CMS[4]. In this environment Mu2e learned how to use the combination of Geant4 and the framework, developed its first generation of data products to hold the outputs produced by Geant4, developed its first analysis modules and delivered the first, preliminary physics studies. The interface to Geant4 uses the recommended technique of inheriting from G4RunManager and overriding the BeamOn method; in this way the framework drives the event loop and Geant4 behaves like just another module that processes one event at a time.

Mu2e made a design decision that Mu2e code should do the minimal Mu2e-specific work inside Geant4; whenever possible, Mu2e specific work, such as hit formation, should be done in framework modules. The minimal Mu2e-specific work is to extract information from Geant4, repackage it and export it to the framework event object. The exported information includes an object that summaries the completion status of G4 for each event and a parent-child history of every particle tracked by Geant4. It also includes a summary of the G4Step information for every step that was taken in every sensitive detector, including all virtual detectors; the Mu2e G4Step summary class is named StepPointMC. Optionally the full trajectory information can be exported for every G4Track; this last feature permits the event display to draw complete simulated events.

Hit formation is done in framework modules, one for each detector type. These modules are scheduled to run after Geant4 has completed each event. The hit making codes read StepPointMC objects and combine one or more of them to form objects that represent data-like hits; the Monte Carlo truth information for the hits is stored in separate data products. This pattern obeys the design decision that objects that can exist when processing experimental data must not contain any Monte Carlo truth information.

The Mu2e geometry is rich in virtual detectors; in addition, many non-traditional elements, such as the stopping target foils, are instrumented as Geant4 sensitive detectors. Together these elements allow one to perform detailed studies at the Monte Carlo truth level and they provide the information needed to develop, debug and characterize the reconstruction algorithms. Mu2e is aware of the issues caused in the CMS PbWO4 calorimeter by the nuclear counter effect; to address this, the Mu2e geometry explicitly represents the readout devices attached to each calorimeter crystal and instruments them as a separate class of sensitive detector. The calorimeter hit making code will produce a saturated readout value if a readout device is hit directly by a particle.

By spring 2011, the CD team had evolved the first generation fork into the product now called art[6][7], which is used today by Mu2e, NOvA, ArgoNeut, MicroBoone and the muon g-2 experiment. Averaged over these two years, the level of effort going into art was less than 2 FTEs and the level of effort going into the Mu2e code was about 1 FTE. Even with this low level of effort, the following were available by spring 2011, a crude but complete hit based simulation and two event displays, one using the Geant4 display technology and the other a ROOT based display that draws geometric information plus many of the data products that can be found in an event.

Soon after the first release of art the art team added several new features. For example, art supports persistable pointers that reside in an object in one data product and point to another

---

4 This fork was informally named CMS-lite, which is not to be confused with a very different product of the same name within CMSSW.
object in an arbitrary data product. This technology, `art::Ptr<T>`, is now in wide use within Mu2e data products; it is used to record Monte Carlo truth information and the calorimeter cluster reconstruction code uses it to record which hits are members of each cluster. `art` also supports a data product type whose purpose is to describe relationships between objects in other data products, `art::Assns<T1,T2,D>`, where T1 and T2 are two data product classes and where D is an optional, user defined class that contains information about the quality of the relationship, perhaps a $\chi^2$ or other information. This feature will be used for recording which tracks from the tracker match to which clusters of energy in the calorimeter; it will also be used for expressing matches between simulated and reconstructed particles. The technology supports one:one, one:many, many:one and many:many matches.

Working closely with Mu2e, the `art` team developed event mixing code to overlay pileup events on top of signal events. In Mu2e, a typical signal event is accompanied by a few hundred events from a half dozen or so separate pileup streams. Although most pileup events make only one or two hits in the detector, it is critical that the spatial and temporal structure of these hits be well modeled; therefore the pileup events are generated by long runs of Mu2eSim using the appropriate background event generators. The solution to the pileup problem was factorized into two pieces: an `art`-supplied mixing module class template, which understands persistency, and a user-supplied detail class, which understands the structure of the data products. Each mixing module is an instantiation of the mixing template with a detail class as the template argument; methods in the detail class are called by the mixing template to perform data-product-specific tasks. With this solution, the authors of the mixing template do not need to know anything about the internal structure of the data products and the authors of the detail class do not need to know anything about the internal details of persistency. This solution places no limit on the number of input mixing streams or on the number of pileup events read from each mixing stream — the only limits are those imposed by the available memory on the computer on which `art` is running. The Mu2e mixing detail class knows how to update all `art::Ptr<T>` objects to be consistent with the new data product geometry of the mixed event; it can also shift events in time.

It was mentioned above that Mu2e exports StepPointMC objects from Geant4 and forms hits in a separate `art` module. When event mixing is performed, the mixing is done at the StepPointMC level and hit formation uses the ensemble of the many StepPointMCCollections. In other words, event mixing is done at the analog level, not at the level of digitized hits.

Mu2e has ported the FastSim Kalman filter (originally the BABAR Kalman filter) to run in the `art` environment; this operation took only a few weeks and it exploits the many person years of effort that have gone into the development of that code. Over the course of the past year, Mu2e collaborators have developed track pattern recognition code that operates on events with pileup; once a track candidate has been found, it is passed to the Kalman filter for final fitting, including outlier rejection and the possible addition of hits that are near to the trajectory but not included in the initial hit list. Current work has been focused on the improvement of the left-right ambiguity resolution in the straws. All of this work uses only information that will be available in data; the Monte Carlo truth is carefully segregated and is only used for quality control after the reconstruction is complete.

Figure 2 shows the results of a simulation for events that contained one conversion electron, plus the anticipated pileup. The pattern recognition and Kalman filter were run on these events. For each reconstructed track, the measured momentum was determined at the upstream face of the tracking system. After the fit was completed, the Monte Carlo truth momentum at the upstream face of the tracker was extracted from one of the virtual detectors. Figure 2 shows

---

5 Currently the FastSim Kalman filter lives in its own source code repository and is treated by Mu2eReco as an external product; this has worked well for studies to date. Integration of the Kalman filter into the Mu2e code will be a big project that the software team has just started to work through.
the difference, measured-true momentum, evaluated at the upstream face of the tracker. The tails are asymmetric because electrons that undergo bremsstrahlung in the tracker material are preferentially reconstructed with a low measured momentum. Figure 3 shows the same quantity for slightly tighter track quality cuts; this significantly reduces the high side tail at the expense of reducing the efficiency from 46% to 38%\textsuperscript{6}. While the efficiency is not yet as high as one would like, these plots serve as a proof of principal that, despite the unusual Mu2e tracker geometry, an adequate track reconstruction and track fitting algorithm exists. Moreover, there are many promising avenues, yet to be explored, for improving the efficiency without increasing the high side tails.

At this time efforts are underway to form clusters in the calorimeter, to match tracks to clusters and to extrapolate tracks through the inhomogeneous magnetic field from the tracker to the stopping targets.

In summary, the track reconstruction code is now at the level of an advanced prototype and the first generations of the other reconstruction codes are expected soon. The two major acquisitions, art and the BaBar Kalman filter, have allowed most of the Mu2e physicist effort to go towards physics software, not infrastructure software. In this sense the art based software has been a great success. In the future many studies will be moved from G4beamline to Mu2eSim in order to exploit the greater level of detail recorded by Mu2eSim.

2.4. MARS

The authors of MARS describe their code as "...a Monte Carlo code for inclusive and exclusive simulation of three-dimensional hadronic and electromagnetic cascades, muon, heavy-ion and low-energy neutron transport in accelerator, detector, spacecraft and shielding components in the energy range from a fraction of an electron volt up to 100 TeV."\textsuperscript{8}. Mu2e uses MARS configured so that it uses the MCNP\textsuperscript{9} package for low momentum neutrons. It is understood throughout the HEP community that MARS plus MCNP is the best calibrated code for simulation of neutrons, in particular neutrons with low kinetic energy.

Two MARS experts work on Mu2e and have used MARS to study radiation levels throughout the experimental hall. These studies inform the requirements for shielding, for radiation safety and for the radiation hardness of electronics. The team is currently computing the neutron flux through the cosmic ray veto counters; if this flux is safely low, then it will be possible to retain the baseline design of a plastic scintillator based CRV system; if the flux is too high, the backup plan is to use gas based detectors, which are less sensitive to neutrons. MARS was also used

\textsuperscript{6} There is an important class of backgrounds, muon decay in orbit, for which controlling the high side tail is the critical mitigation.
for the critical calculation of the heat and radiation load in the coils of the PS; this is used to study variants on the design of the heat and radiation shield. This calculation was sufficiently important that it was done with both MARS and G4beamline; this was done as a cross-check on the underlying physics models and cross-section tables.

The most critical calculations will also be done using Mu2eSim. Because these three codes were developed independently, the three way cross-check will help expose any pilot error that has escaped other controls.

Until recently, security requirements imposed by the MARS and MCNP license issuers restricted the running of these codes on the GRID, which was regarded as an insecure platform. Working with the MARS team, Mu2e worked through the security issues and pioneered a procedure for running MARS and MCNP on GP FermiGrid. This procedure is now used by many Fermilab Intensity Frontier experiments.

3. Geometry Issues
One weakness of the present system is that each package has its own geometry language that is inconsistent with that of the other packages. To address this, a plan has been developed to generate all of the geometry descriptions from a common authoritative source.

4. Summary and Conclusions
In the early days of Mu2e, the design of the experiment got off to a fast start by using G4beamline and FastSim: using these packages required mostly just run-time configuration and little, or no, code development. In parallel with these efforts, a small team has made rapid progress on the development of full-featured simulation and reconstruction code: an end-to-end simulation and an advanced prototype track reconstruction are already in wide use; calorimeter and cosmic ray veto reconstruction codes are in progress. This rapid progress was enabled by acquiring two major components of the offline software suite, art and the BABAR Kalman filter code. MARS has also been used from the early days of Mu2e and will continue to be used for calculations involving neutrons.

Acknowledgments
The author would like to thank the Fermilab management and staff for their strong support of the Mu2e experiment. This work is supported in part by the Office of Science of the U.S. Department of Energy, the U.S. National Science Foundation, INFN in Italy, and the U.S.-Japan Agreement in High Energy Physics.

References
[1] The Mu2e Conceptual Design Report 2012
http://mu2e-docdb.fnal.gov/cgi-bin/ShowDocument?docid=1169
[2] The Mu2e Collaboration home page: http://mu2e.fnal.gov
[3] Bellgardt U et al (SINDRUM Collaboration) 1988 Nucl. Phys. B299, 1
[4] http://www.muonsinc.com
[5] Andreassen R et al 2011 J. Phys.: Conf. Series 331 032038
[6] https://cdcvs.fnal.gov/redmine/projects/art
[7] Green C, Kowalkowski J, Paterno M, Fischer M, Garren L and Lu Q. "The art Framework", in these proceedings (abstract # 354).
[8] Mokhov N V 1995 The Mars Code System User’s Guide, Fermilab-FN-628; Mokhov N V and Striganov S I 2007 AIP Conf. Proc. 896, pp. 50-60; http://www-ap.fnal.gov/MARS/
[9] MCNP-A General Monte Carlo N-Particle Transport Code, Version 5 Los Alamos National Laboratory Reports LA-UR-03-1987, LA-CP-03-0245, LA-CP-03-0284 (2005); http://mcnp-green.lanl.gov