MONTE-CARLO SIMULATIONS OF THE (d,^{2}He) REACTION IN INVERSE KINEMATICS

By

Alexander B. Carls

A THESIS

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Physics - Master of Science

2019

ABSTRACT

MONTE-CARLO SIMULATIONS OF THE (d,^{2}He) REACTION IN INVERSE KINEMATICS

By

Alexander B. Carls

Charge-exchange reactions offer an indirect method for the testing of theoretical models for Gamow-Teller strengths that are used to calculate electron-capture rates on medium-heavy nuclei, which play important roles in astrophysical phenomena. Many of the relevant nuclei are unstable. However, a good general probe for performing charge-exchange reactions in inverse kinematics in the (n,p) reaction has not yet been established. The (d,^{2}He) reaction in inverse kinematics is being developed as a potential candidate for this probe. This method uses the Active-Target Time Projection Chamber (AT-TPC) to detect the two protons from the unbound ^{2}He system, and the S800 spectrograph to detect the heavy recoil. The feasibility of this method is demonstrated through Monte-Carlo simulations. The ATTPCROOTv2 code is the framework which allows for simulation of reactions within the AT-TPC as well as digitization of the results in the pad planes for realistic simulated data. The analysis performed on this data using the ATTPCROOTv2 code shows the techniques that can be done in experiment to track the scattered protons through the detector using Random Sampling Consensus (RANSAC) algorithms.

ACKNOWLEDGMENTS

The first person I have to thank is my advisor, Remco Zegers. Your patience for fielding my questions, whether naive or complex, and for focusing my efforts when I needed guidance made my experience doing research much less stressful. Your unwavering support as I struggled through classes is the reason I made it to the end. Thank you.

To my committee: Sean Liddick and Wolfgang Mittig. Thank you for your insightful commentary on my thesis and for asking the right questions to push the limits of my understanding.

To the faculty and post-docs of NSCL who helped along the way: Juan Zamora. Thank you for beginning my venture into these simulations and for drilling an understanding of the kinematics of the $(d, {}^{2}\mathrm{He})$ reaction into my mind. Yassid Ayyad. Thank you for the constant back-and-forth as I set out to understand the code down to the smallest detail and for the endless troubleshooting support. Daniel Bazin, Shumpei Noji, and Jorge Pereira. Thank you for being an ever present support and for the help you have given me along the way.

To my groupmates: Jaclyn Schmitt, Miles DeNudt, Rachel Titus, Felix Ndayisabye, and Cavan Maher. Thank you for reminding me I am not alone in the struggle of graduate life and for making work feel less like work sometimes.

To my friends from home: Kevin, Will, Alex, and Aaron. Thank you for making hundreds of miles not seem so far away.

To my friends in Michigan: The men and women of MSU Ultimate. Thank you for being my second family for the past few years and for forcing me to stay in shape when I would otherwise be sitting on my couch. Jared, Ryan, and Cody. Thank you for all the movie nights and for getting me to come out of my apartment every so often.

Finally, to my family: Mom. Thank you for updating the world every time I come home and for calling every week to make sure I am still alive. Dad. Thank you for keeping me up-to-date with all the best movies and shows and for walking me through some of the challenges of adulthood (taxes, insurance, etc.) that I promise I will learn to do myself one day. Jamie. Thanks for letting me hack into all of your streaming services to binge watch shows all this time. One day I will repay you. All of you have been a constant source for support and love without which I would not have succeeded.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

Introduction

1.1 Astrophysics

Astrophysics and nuclear physics are studies of nature separated by many orders of magnitude, yet in massive stars the concepts in each field are closely intertwined. Due to the intense gravitational pressure, fusion occurs in the core binding nuclei together in an exothermic reaction to form more massive elements. The thermonuclear energy produced in this process along with the degeneracy pressure due to high electron density in the core counteract the inward gravitational force [1]. This process begins with hydrogen burning, forming helium. Once the core exhausts its hydrogen as a fuel for fusion, the outward force due to internal pressures begins to decrease and the core contracts. Depending on the mass of the star M, the core will contract either until the electron degeneracy pressure balances the gravitational force or the core reaches the temperature required for helium fusion ($M > 0.4\, M_{\odot}$) [2].

Over the life-time of a massive star, this cycle of fuel burning, core contraction, and fusion ignition continues beyond helium creating a structure of ash shells. The average binding energy per nucleon increases to a peak near ^{56}Fe after which the decreasing binding energy corresponds to energy consumption for further fusion. Once the star reaches the silicon burning phase, an iron core develops which will not act as fuel for the outward force counteracting gravity. The core will continue to grow until its mass surpasses the

Chandrasekhar limit ($M \approx 1.4\ M_{\odot}$) [3] at which point the thermonuclear energy combined with the increasing degeneracy pressure is no longer sufficient to support the core against gravitational collapse. In the mass range $8\ M_{\odot} < M < 40\ M_{\odot}$ [4], this leads to the death of the star as a core-collapse supernova.

As core-collapse progresses, the pressure in the core increases to the point where further compression would cause electrons to occupy the same energy states, violating the Pauli exclusion principle. In order to reduce lepton number and decrease degeneracy, heavy nuclei undergo electron-capture reactions, $p + e^{-} \rightarrow n + \nu_e$, where a proton inside the nucleus and an electron fuse to produce a neutron and an electron neutrino. The decrease in electron degeneracy pressure along with the energy carried away by neutrinos accelerates the collapse [2]. Once the core reaches nuclear density, $\rho \approx 10^{14}$ g cm^{-3}, the matter can no longer be compressed and the outer layers bounce off the core. Initially, the resulting shock wave slows as it loses energy to the dissociation of nuclei in the surrounding shells, but it is thought that the wave is reinvigorated by the burst of neutrinos [5]. The shock wave propagates outward causing the outer layers to be ejected in an explosion.

Sensitivity studies have shown that electron-captures on neutron-rich nuclei in the $N = 50$ region contribute strongly to the change in electron fraction in the core of the star [6], which places importance on studying neutron-rich unstable nuclei for understanding core-collapse. This sequence of events, illustrated in Figure 1.1, ends with the astrophysical phenomenon known as a core-collapse supernova, which is believed to contribute to nucleosynthesis [7]. These events are also predicted to be sources for gravitational waves and neutrinos [8, 9, 10]. The role that core-collapse supernova play in stellar evolution make the astrophysical processes involved essential to understanding the evolution of the universe.

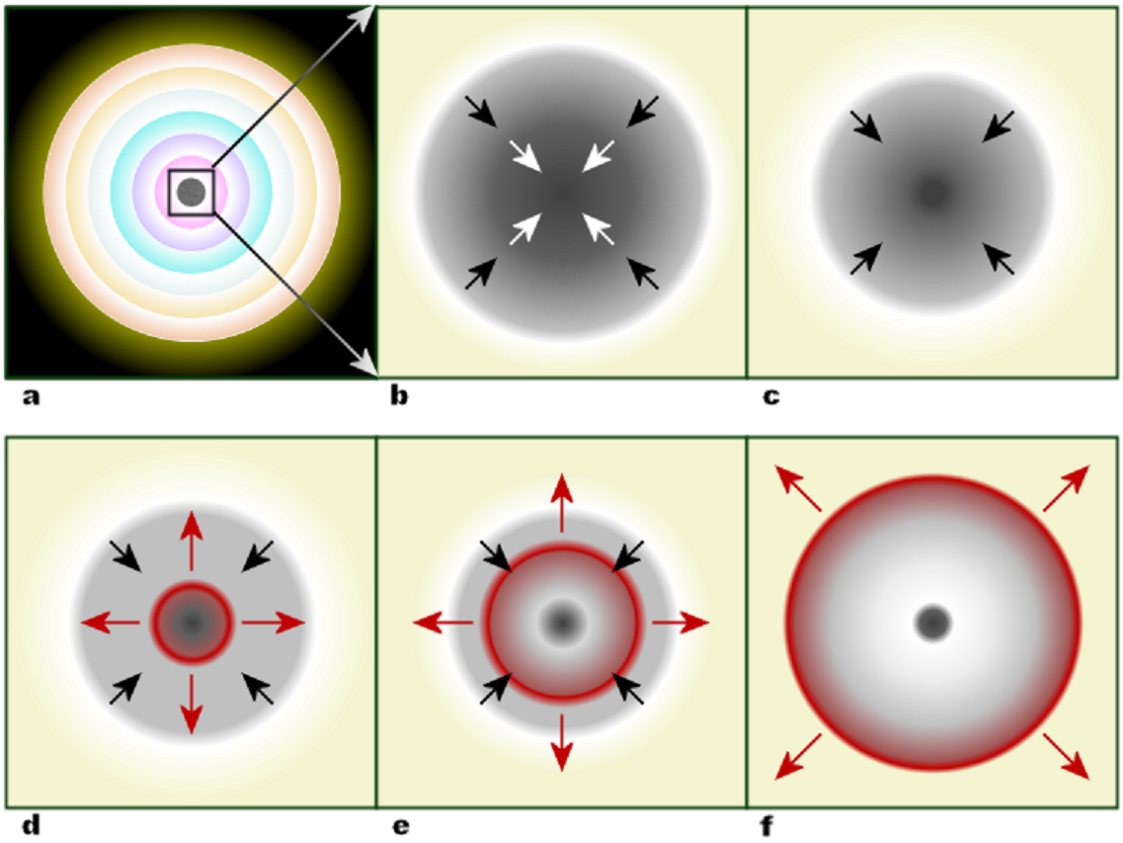


Figure 1.1: The inert iron core increases in mass as the silicon burning phase progresses (a) until it reaches the Chandrasekhar mass limit and the gravitational force overcomes thermonuclear energy and degeneracy pressures. (b) The core begins to contract and electron-capture occurs (c) until core density reaches $\rho \approx 10^{-14}$ g cm^{-3} and contraction quickly stops, (d) after which the outer layers bounce off the core creating an outward propagating shock wave. (e) The shock wave slows as it passes through the outer layers (f) but is reinvigorated by a burst of neutrinos. Figure credit [11].

1.2 Electron-Capture Rates

Electron-capture in these massive stars is one of the driving reactions behind the evolution of the inert iron core from growth to collapse. The rates at which these reactions occur are essential inputs in astrophysical simulations and, thus, need to be understood with good precision for a large range of energies and many nuclei.

Electron-capture reactions as illustrated in Figure 1.2a are mediated by the weak force with rates dominated by Gamow-Teller (GT) transition strengths and distributions [12, 13,

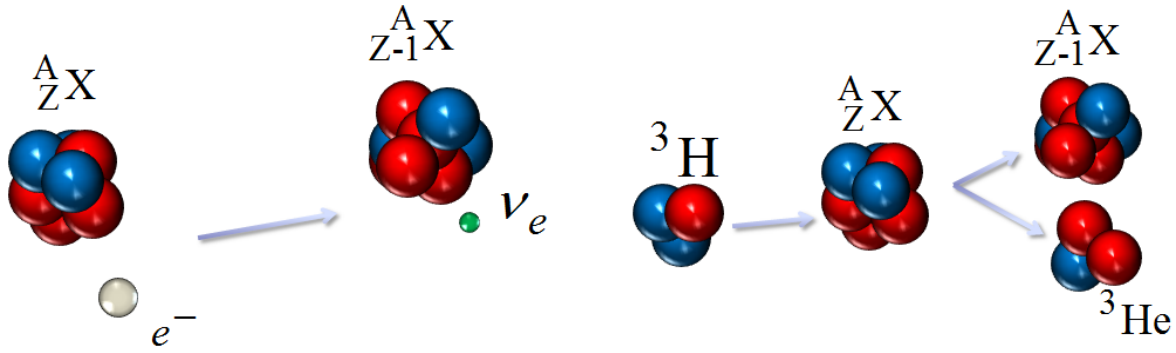
14, 15], from the initial to the final states. These transitions involve a transfer of spin ($\Delta S = 1$), with no change in angular momentum ($\Delta L = 0$). The total electron-capture rate for a nucleus is the sum of the of the individual electron-capture rates for each transition between parent nucleus state i and daughter nucleus state j, formulated as [12, 16]

$$\lambda_{EC} = \ln(2) \sum_{ij} f_{ij}(T, \rho, U_F) B(GT)_{ij}. \tag{1.1}$$

The Gamow-Teller strength for each $\langle i|j\rangle$ transition, $B(GT)_{ij}$, is weighted by a phase-space factor, $f_{ij}(T, \rho, U_F)$, which is calculable and dependent on the stellar temperature, T, density, ρ, and the electron chemical potential, U_F. The Gamow-Teller strengths can be studied experimentally but the direct measurement through electron-capture and β^+ decay studies is limited by the decay Q-value.

The Q-value of a reaction is the total energy absorbed or released by the reactants in creation of the products. For electron-capture and β decay, this Q-value sets a limitation on the allowed energies of the daughter nucleus as seen in Figure 1.3. Due to this limitation, from these measurements only Gamow-Teller transition strengths for a few levels can be extracted, providing an incomplete estimate of the electron-capture rate.

Charge-exchange reactions as illustrated in Figure 1.2b do not restrict the excitation of the daughter nucleus to a Q-value window. In these reactions, the target nucleus either exchanges a neutron for a proton from the beam particle, denoted a reaction in the (n, p) direction, or vice versa, denoted (p, n). While charge-exchange is mediated by the strong force, there exists a proportionality between the Gamow-Teller transition strength and the charge-exchange reaction cross section that allow charge-exchange reactions to be useful tools



(a) Electron-Capture (b) Charge-Exchange Reaction

Figure 1.2: (a) Electron-capture on a nucleus, decreasing atomic number by one while maintaining atomic mass and releasing an electron neutrino. (b) Example of a charge exchange reaction, $t(^A_Z X, {}^A_{Z-1}X)^3$He, in which a neutron from a triton is exchanged for a proton from the target nucleus, resulting in the release of a ^{3}He and a product nucleus with atomic number decreased by one but atomic mass unchanged

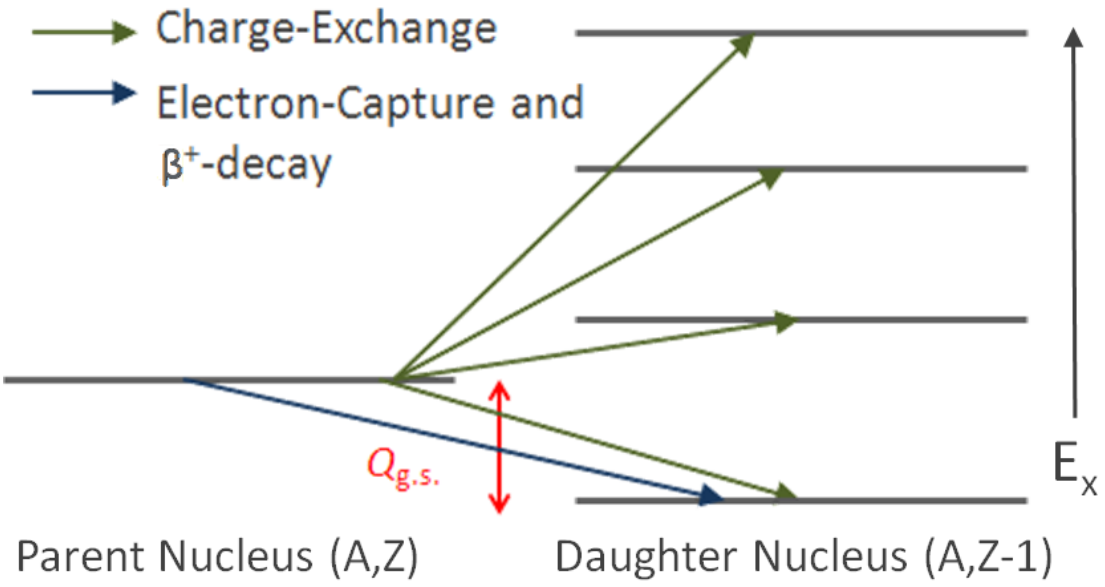


Figure 1.3: The excitation energies of the daughter nucleus are limited to states within the Q-value window, $Q_{g.s}$, for electron-capture and β decay reactions between the ground states of the nuclei. In charge-exchange reactions, the excitation energy of the daughter nucleus is not limited by the Q-value.

for extracting Gamow-Teller strength and calculating electron-capture rates [17, 18, 19, 20]. This relationship is taken as momentum transfer is reduced to zero ($q \rightarrow 0$) for Gamow-Teller transitions at intermediate energies, given by

$$\left(\frac{\mathrm{d}\sigma}{\mathrm{d}\Omega}\right)_{q=0} = \hat{\sigma} B(GT), \tag{1.2}$$

where the unit cross section, $\hat{\sigma}$, can be determined by calibrating cross sections against data from β decay experiments or, if β decay data is unavailable, it can be calculated using a mass-dependent relationship formulated by Zegers et al [20].

Since the charge-exchange reactions can populate states of the daughter nucleus up to high excitation, the electron-capture rates can be calculated using more individual Gamow-Teller transition rates, providing more accurate results. However, such measurements can only be performed for a limited set of nuclei. In addition, in the hot stellar environment, transition from excited states can occur, which cannot be measured in the laboratory. Therefore, one has to rely on theory to estimate Gamow-Teller strengths. These theories must be benchmarked and guided by the data obtained from charge-exchange experiments.

1.3 $(d,{}^2\mathrm{He})$ Reaction

Charge-exchange can be performed by a variety of reaction probes, each with its own sensitivities, to study the characteristics of nuclei. In forward kinematics, where a low-mass ion beam strikes the nucleus of a heavy-mass target, many charge-exchange experiments have been performed using probes like (p,n) [21] and $({}^3\mathrm{He},t)$ [22] for reactions in the (p,n) direction and (n,p) [23], $(d,{}^2\mathrm{He})$ [24, 25, 26, 27], and $(t,{}^3\mathrm{He})$ [22, 28] for reactions in the

(n, p) direction. Reactions in forward kinematics, however, are limited to targets made from stable isotopes, which is problematic since most of the nuclei relevant to this astrophysical study are unstable.

Experiments in inverse kinematics utilize a heavy-mass ion beam striking a low-mass nuclide target. Since the beam can travel at high velocities, the reaction can occur before unstable isotopes decay, allowing for charge-exchange reactions on nuclei in regions away from the valley of stability. Development of effective probes in inverse kinematics direction is then essential to studying nuclei, such as those in the high-sensitivity region around $N = 50$ described by Sullivan et al. The (p, n) reaction in inverse kinematics has been successfully developed and used to study unstable nuclei, including ^{56}Ni [29], ^{55}Co [30], ^{16}C [31], and ^{132}Sn [32]. In the (n, p) direction, the $(^{7}\text{Li}, ^{7}\text{Be})$ reaction has been employed for experiments in inverse kinematics [33, 34], but this probe is restricted to experiments involving low-mass nuclei ($A < 40$) and to excitation energies below the particle-decay threshold. Therefore, a better probe for (n, p)-type charge-exchange reactions is very desirable.

The $(d, ^{2}\text{He})$ reaction, illustrated in Figure 1.4, is a probe that has been proposed by the Charge-Exchange Group at the National Superconducting Cyclotron Laboratory (NSCL) for (n, p)-type experiments in inverse kinematics. The $(d, ^{2}\text{He})$ reaction has been studied in many experiments in forward kinematics [24, 25, 26, 27], but such an experiment has not been performed in inverse kinematics. In this reaction, the ^{2}He recoil presents a challenge since it is unstable and immediately decays into two protons. In order to extract information on the ejected beam, the momentum vector of the ^{2}He must be reconstructed from the tracks of the two protons. The goal of this thesis and the experiment it describes is to establish the $(d, ^{2}\text{He})$ reaction in inverse kinematics as a probe for charge-exchange reactions and provide the analytical techniques required to track the protons.

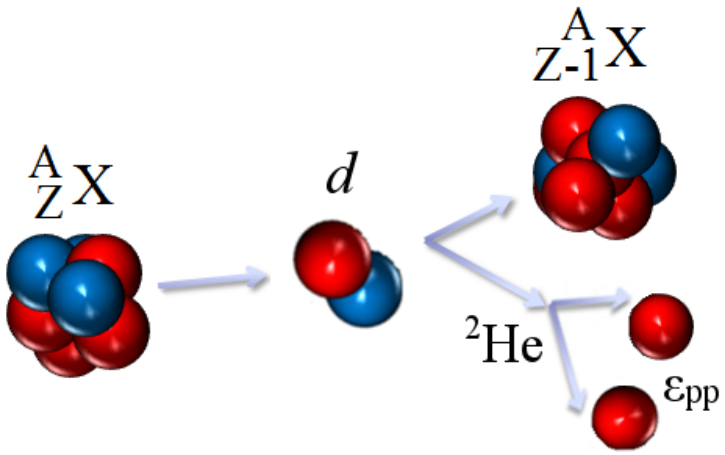


Figure 1.4: The $(d, {}^2\mathrm{He})$ reaction in inverse kinematics, where the unstable heavy nucleus is projected onto a deuterium target nucleus. The ${}^2\mathrm{He}$ recoil immediately decays into two protons with relative energy, ϵ_{pp}.

Chapter 2

Experimental Setup

2.1 Overview

The Charge-Exchange Group at the NSCL has proposed an experiment using the ^{14}O$(d, ^2$He) reaction in inverse kinematics as proof-of-principle for $(d, ^2$He) as a probe for measuring Gamow-Teller transition strengths of unstable nuclei. To measure the momentum vectors of both protons with good accuracy, the Active-Target Time Projection Chamber (AT-TPC) filled with deuterium gas will be used for tracking of charged particles within the detector. The AT-TPC will be used in conjunction with the S800 spectrograph [35], which serves to measure the heavy ejectile (^{14}N) or one of its decay products and to provide an event trigger for the AT-TPC. The experimental set up is illustrated in Figure 2.1.

In order to determine the excitation energy and center-of-mass scattering angle of the ^{14}N ejectile, a missing-mass calculation is performed with the ^{2}He particle whose momentum is reconstructed in an invariant-mass analysis from the two protons. The relative energy of the two protons is given by [36]

$$\epsilon_{pp} = \frac{1}{2}\left[E_1 + E_2 - 2(E_1 E_2)^{1/2}\cos\theta_{12}\right], \tag{2.1}$$

where E_1 and E_1 are the laboratory energies of the two protons and θ_{12} is the angle between the directions that each proton travels. In order to ensure that the spin-singlet $^1S_0$ pp state is

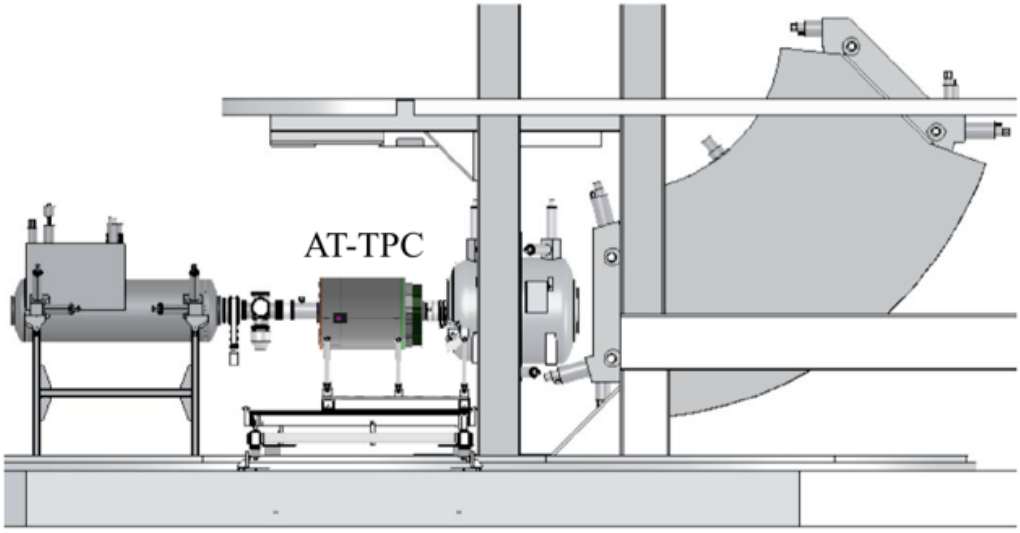


Figure 2.1: The AT-TPC is placed in front of the S800 spectrograph to utilize its ability to do particle identification, as well as to provide a trigger for events in the AT-TPC.

created and the reaction is exclusively of spin-transfer type, the data must be gated on events for which $\epsilon_{pp} < 1$ MeV, which corresponds to low-energy protons. The differential cross-sections for excitations of ^{14}N are calculated with the code "Adiabatic Coupled-Channels Born Approximation" (ACCBA) [37] and an example for the ^{14}O$(d, ^2$He) reaction can be seen in Figure 2.2.

The Gamow-Teller transition strengths from a nucleus follows a general sum rule: $S_{\beta-} - S_{\beta+} = 3(N - Z)$, where $S_{\beta\pm}$ corresponds to the summed Gamow-Teller strength in the $\beta^{\pm}$ direction. However, measurements only account for about $50 - 60\%$ of this strength [38]. Different mechanisms thought to contribute to this quenching of the Gamow-Teller strength [39, 40, 41]. To better understand the different contributions to the quenching, accurate measurements of the Gamow-Teller strength distributions are important. Charge-exchange reactions on light systems are a good way to obtain these distributions. In the case of ^{14}O, Gamow-Teller transitions in the β^- (or (p, n)) direction are Pauli blocked because the

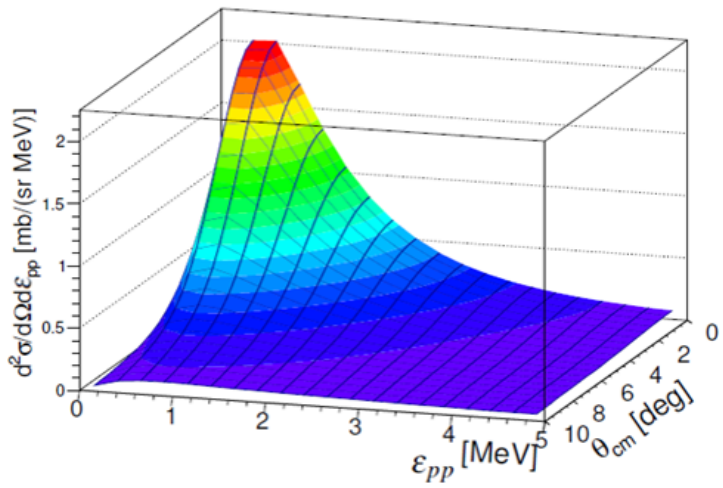


Figure 2.2: Double differential cross-section as a function of the center-of-mass angle (θ_{cm}) and the relative energy of the two protons (ϵ_{pp}) for the transition to the 3.95 MeV 1^+ ^{14}N state via the ^{14}O$(d, ^2$He) reaction. Calculations based off of the Adiabatic Coupled-Channels Born Approximation (ACCBA) code.

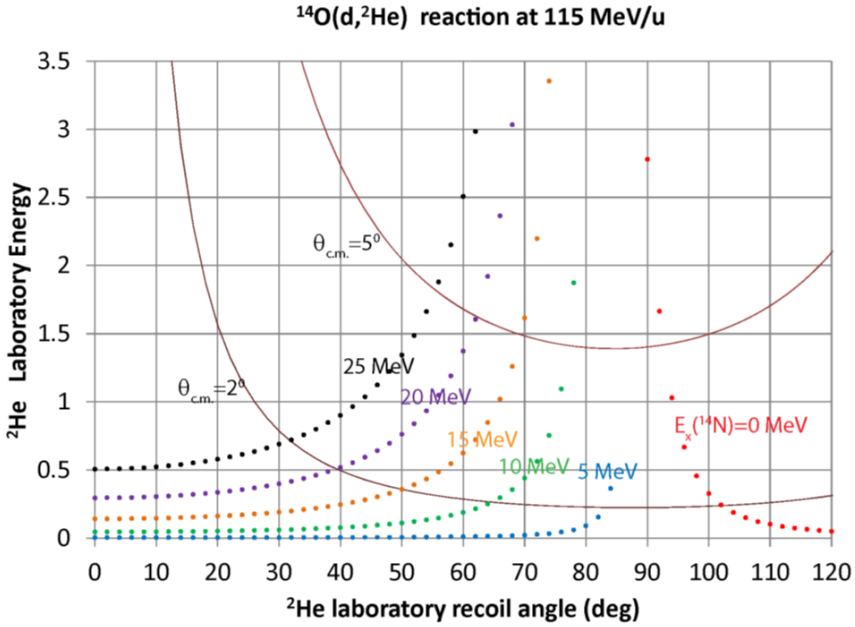


Figure 2.3: Kinetic energy versus the angle in the laboratory for the ^{2}He particle for the ^{14}O$(d, ^2$He) reaction in inverse kinematics at 115 MeV/u. The dotted colored lines indicate the excitation energy of 14N, in steps of 5 MeV. The brown lines indicate center-of-mass scattering angles (2° and 5°).

protons fill the p-shell. Therefore, $S_{\beta-} = 0$, and the sum rule can be assessed by a measurements of the β^+ Gamow-Teller transition only. Hence, in a single measurement, the Gamow-Teller sum rule can be evaluated. The strong transitions to the 3.95-MeV 1^+ and 2.31-MeV 0^+ states in ^{14}N make the ^{14}O$(d, ^2$He) reaction a good candidate for testing the basic properties of this new probe.

2.2 Active Target-Time Projection Chamber

Reconstruction of the ^{2}He recoil requires measuring each proton with good accuracy. Solid targets (e.g. CD_2) suffer the trade-off between resolution and luminosity. This option is not possible due to the poor luminosity that would result from a target thin enough to let the reaction products escape in combination with the relatively low beam intensities associated with the use of rare-isotope beams. A time projection chamber (TPC) contains a large volume of gas which is ionized as charged particles pass through. The chamber utilizes electric fields to guide the ejected electrons to a wire mesh or pad plane detection system. A three-dimensional reconstruction of the trajectory of the charged particles can be made using the signal readout from the incident electrons, where the X- and Y-coordinates are determined from the pad position and the Z-coordinate is determined from the timing of the signal and the drift velocity of electrons in the gas.

The Active Target-Time Projection Chamber (AT-TPC) [42] utilizes this TPC technology with the added functionality of the gas tracking medium working simultaneously as the reaction target. The concepts behind this detector eliminate the need to compromise between resolution and luminosity since the energy of the reaction products can be measured as they emerge from the detector, regardless of the thickness of the target medium.

The detector, illustrated in Figure 2.4, consists of a cylindrical active volume, 1 m in length and 29.2 cm in radius, which contains the deuterium gas target and tracking medium. A uniform electric field ($\sim 10^4$ V) is generated inside the active volume in the direction of the beam axis by creating a potential difference between the cathode near the beam entrance and an anode near the end of the detector. To ensure uniformity of the electric field, a field cage consisting of 50 concentric ring-shaped electrodes establishes evenly spaced equipotentials between each ring. As the charged protons travel through the detector, ionized electrons in the electric field drift down the length of the active volume to the sensor plane composed of 10240 Micromegas pads, which provide the signal read by the digital electronics on the back face of the detector. As seen in Figure 2.5, an aperture, 40 mm in diameter, was constructed in the sensor plane of the AT-TPC to allow the ejectile to continue into the S800 spectrograph. During the $(d, {}^2\mathrm{He})$ experiment, the particle detection in the S800 will serve as a trigger for reading out the event in the AT-TPC.

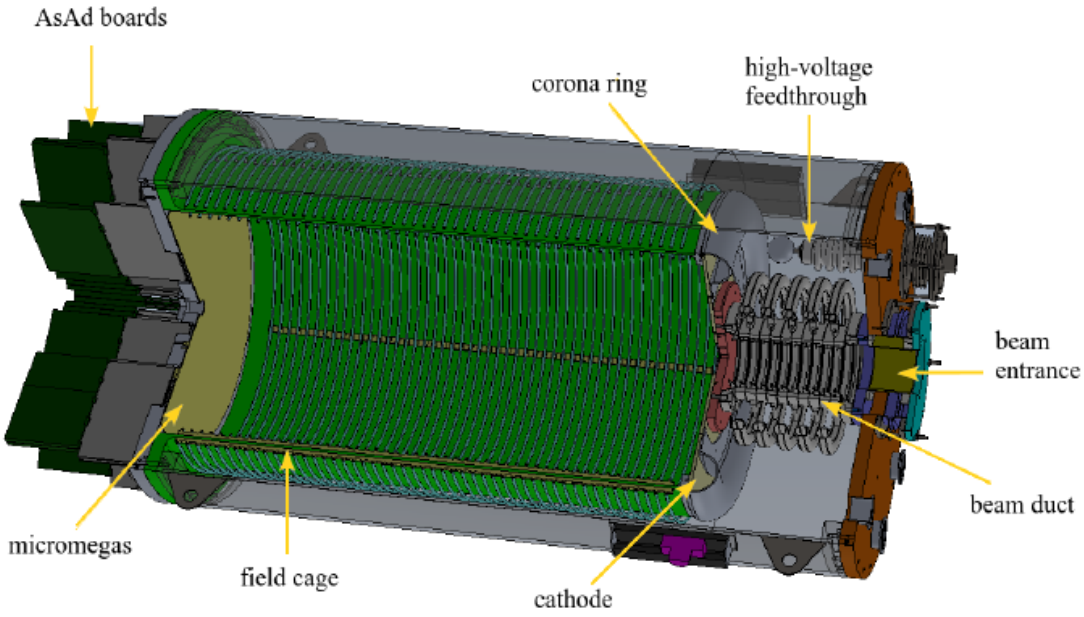


Figure 2.4: Illustration of the AT-TPC. A beam of ^{14}O particles enters the right side of the diagrammed detector. The active volume is filled with deuterium gas at 0.7 atm pressure, which reacts with the beam and produces two protons. The ejectile leaves the left side of the diagrammed detector and continues on to the S800 spectrograph.

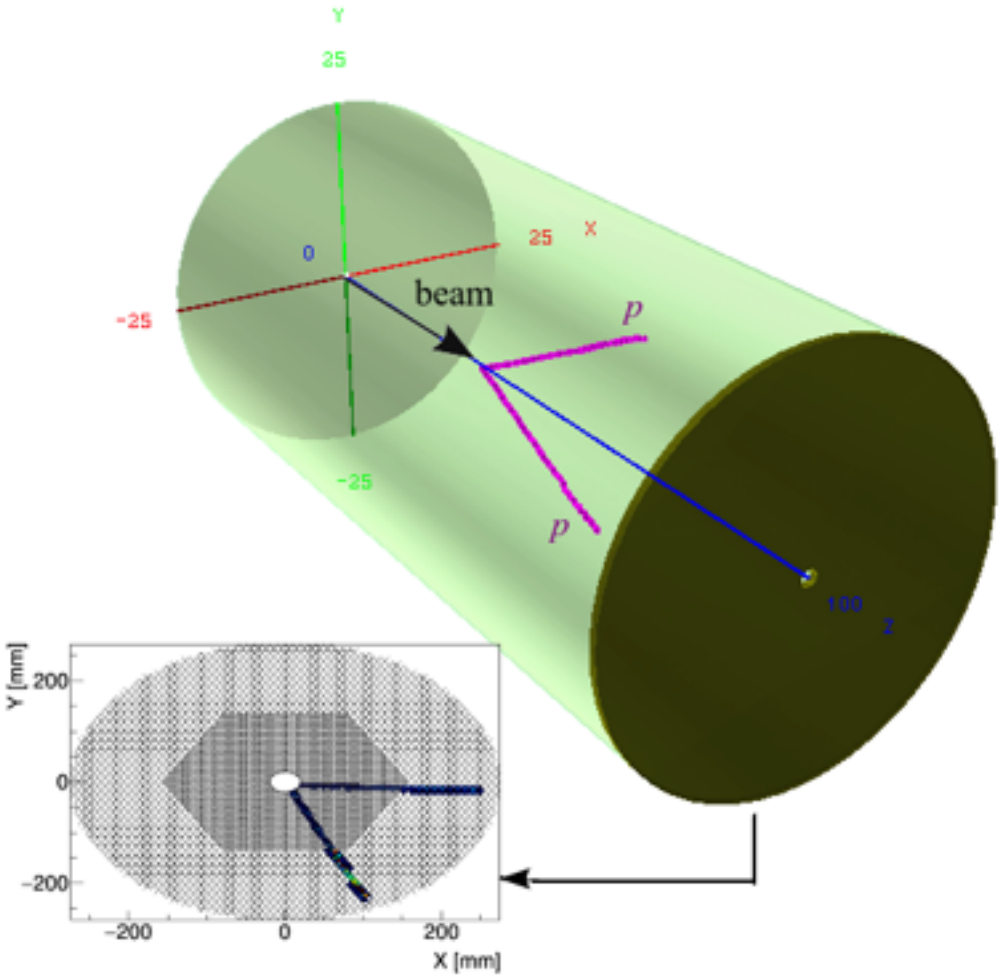


Figure 2.5: Visualization of the simulation of one $(d, {}^{2}\mathrm{He})$ event. Since the ejectile retains most of its momentum, it is carried through an aperture in the back of the detector into the S800 spectrograph. The two protons create tracks of ionized electrons in the gas as illustrated.

Chapter 3

Simulation

3.1 Framework

The ATTPCROOTv2 code this thesis describes is the framework for simulation and analysis of data from the AT-TPC as well as the Prototype AT-TPC [43]. The framework allows the end user to unpack and analyze the data, as well as perform realistic simulations based on a Virtual Monte Carlo (VMC) package. ATTPCROOTv2 allows the user to create a customized geometry to simulate the detector environment and to simulate the reaction on an event-by-event basis. Each generated event can be tailored to the reaction mechanism, whether it be, for example, fission or the $(d, {}^2\mathrm{He})$ reaction. After unpacking raw data or creating simulated data, the read out from the ATTPC pad planes can be processed by pulse shape analysis. The collection of points from each event can be analyzed using pattern recognition algorithms and the trajectories of particles in the detector can then be tracked.

The framework is built upon ROOT [44] and requires external libraries (FairSoft [45] and FairRoot [46]) to be compiled and executed, which are developed by other groups. Additional libraries for analysis are needed, like PCL [47], FLANN [48], and Eigen [49] and libraries like the HDF5 library [50] are needed for data formatting. The Monte Carlo simulations are handled by the Geant4 toolkit [51]. Details on installation of the code can be found in Appendix A.

FairRoot analyses and simulations are defined using ROOT macros and executed by ROOT. The processes involved in each macro are managed by a FairRoot class Run Manager. The FairRunSim class is used for simulation runs and the FairAnaSim class is used for analysis runs. The simulation or analysis is constructed by defining and adding components to the run manager. The parameters required by and created in simulation and used as inputs to analysis are managed by the FairRuntimeDb class (Parameter Manager). The input and output of data is organized by the FairRootManager class (IO Manager) which can provide access to or create branches in the input and output files. Figure 3.1 illustrates the flow of data types as input and output for the different processes involved in ATTPCROOTv2.

For simulations, the FairRoot classes FairRunSim, FairPrimaryGenerator, and FairMCApplication are used as they are with the user only calling functions on the classes. FairRunSim defines the experimental set up, including the Monte Carlo engine, detector geometries and event generators, through its member functions. FairPrimaryGenerator defines the primary beam conditions as well as any additional event generators that may be involved in the reaction. FairMCApplication is only used internally and serves as the interface between FairRoot and Geant4. The FairRoot classes that must be implemented by the user are FairModule, FairDetector, FairGenerator, and FairGenericStack. The FairModule are components of the geometry that particles do not interact with, such as the "Cave" in which all other components are anchored. The FairDetector are components of the geometry that constitute an active volume with which particles can interact. The FairGenerator creates particles for an specified event, like the instance of a beam particle or a two-body reaction, and puts the particle on the FairGenericStack. The FairGenericStack contains particles or tracks generated in an event. The processes involved after execution of a FairRoot simulation macro is illustrated in Figure 3.2.

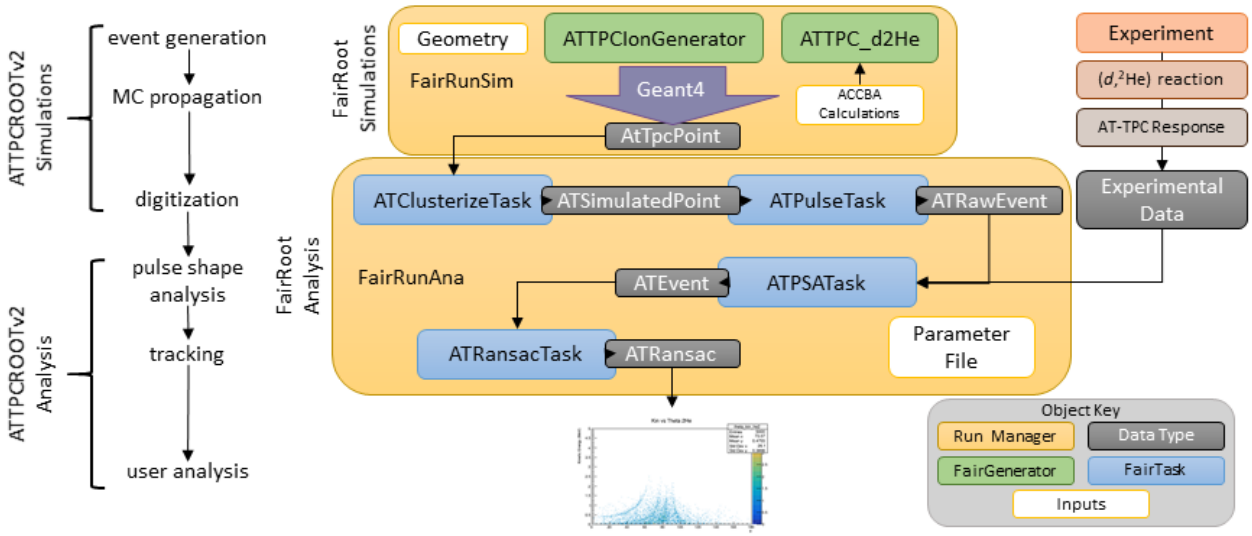


Figure 3.1: Flow of ATTPCROOTv2 data types at each step in the process from generation of data to user analysis. Example generators are those used in simulations of $(d, {}^2\mathrm{He})$ reactions. Each FairRoot analysis task requires a specific input data type and produces a different output data type as shown in the figure.

For analysis, the FairRoot class FairRunAna is used as it is with the user only calling its member functions. This class defines the analysis tasks and the order in which they are executed. The FairRoot class that must be implemented by the user is the FairTask class. This class connects with a specified type of data in the input file, performs analysis on that data, and registers the results in a branch of the output file. Subsequent tasks can connect to data in the output file registered by previous tasks. The processes involved after execution of a FairRoot analysis macro is illustrated in Figure 3.3.

The code shown in Figures 3.2 and 3.3 can be used as outlines that should be included in every FairRoot simulation and analysis macro for the $(d, {}^2\mathrm{He})$ reaction. The input for the arguments of the FairGenerators will be explained in more detail in Section 3.2.3. Each FairTask has parameters that need to be specified in the macro, which will be explained in more detail in Section 3.3.

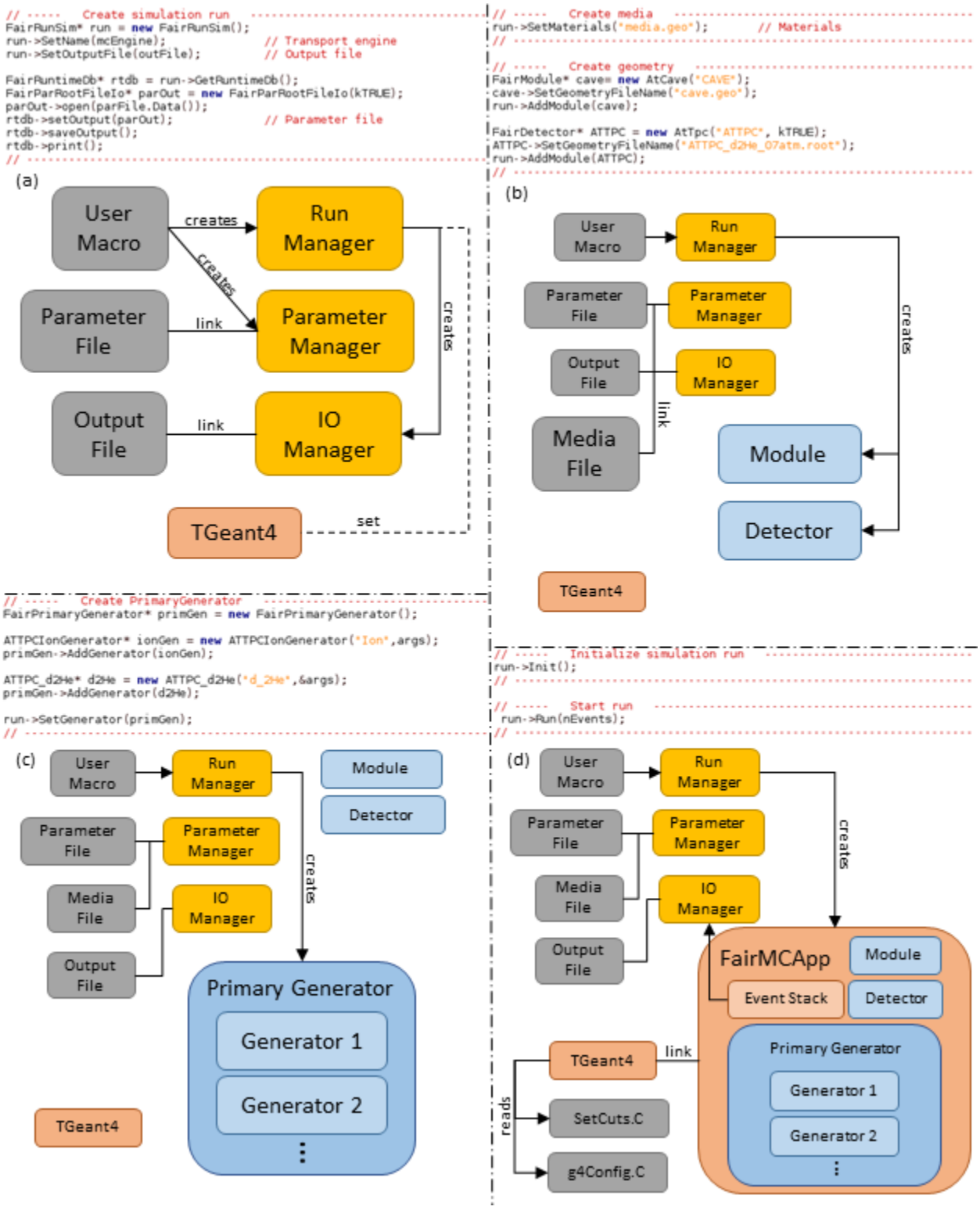


Figure 3.2: Workflow of FairRoot simulation, including the outline of a simulation macro. Illustrations adapted from [52].

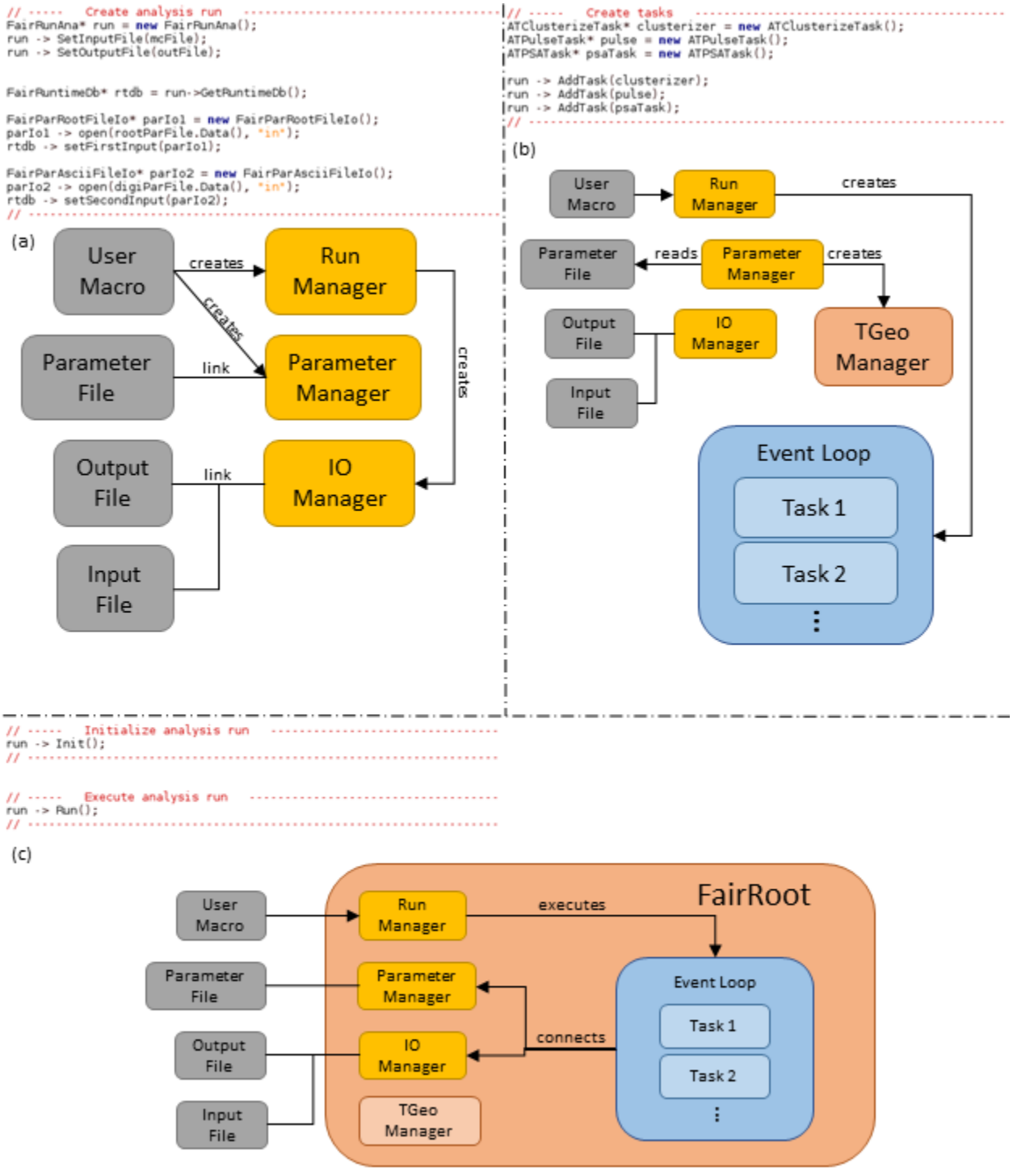


Figure 3.3: Workflow of FairRoot analysis, including the outline of an analysis macro. Illustrations adapted from [52].

3.2 Data Generation

There are four components involved in creating accurate simulation data for reactions within the AT-TPC: the geometry, the parameter file, the event generators, and the ROOT simulation macro. For new reactions, each will need to be adjusted. The following examples are used to simulate the $^{14}\mathrm{O}(d, ^{2}\mathrm{He})\,^{14}\mathrm{N}$ reaction in inverse kinematics, using a 115 MeV/u beam and a deuterium gas target at 0.7 atm.

3.2.1 Geometry

Some of the inputs to the FairRunSim needed for propagation of the particles through space by Monte Carlo engine are the geometry position of detectors and the media involved in the reaction environment. In the ATTPCROOTv2/geometry directory, there are macros for creating these geometries that utilize TGeoManager object and accompanying classes from the ROOT geometry package.

To construct the detector, the geometry file contains a list of volumes characterized by a number of parameters. The geometry will consist of a top volume which contains all of the other modules that make up specific detector components. To describe a module, the following requirements need to be met:

- Construct all necessary materials
- Define shapes/solids required to describe the geometry
- Construct and place volumes of the detector geometry
- Define sensitive detectors and identify detector volumes with which to associate them
- Define visualization attributes for the detector elements

The media.geo file acts a dictionary for materials that are commonly used in simulations since there are no predefined materials in FairRoot. Each medium is defined by several numerical parameters as seen below and described on the FairRoot HowTo page [46]. The following parameters are required:

- **int ncomp** - number of components in the material (ncomp= 1 for a basic material and < 1 or > 1 for a mixture. If ncomp > 0 the array wm contains the proportion by weight of each material in the mixture. If ncomp < 0 the array wm contains the proportion by number of atoms of each kind.)
- **float aw[ncomp]** - atomic weights A for the components
- **float an[ncomp]** - atomic numbers Z for the components
- **float dens** - density DENS in g/cm^{-3}
- **float wm[ncomp]** - weights WMAT of each component in a mixture (only for a mixture)
- **int sensflag** - sensitivity flag ISVOL
- **int fldflag** - fieldflag IFIELD
- **float fld** - maximum field value FIELDM in kilogauss
- **float epsil** - boundary crossing precision EPSIL
- **int npckov** - number of values used to define the optical properties of the medium. This variable is 0 for all media except some special media used for the Rich where the tracking of the Cherenkov photons is necessary. These media have additional parameters
  - **float ppckov[npckov]** - photon momentum in eV
  - **float absco[npckov]** - absorption length in case of dielectric and of absorption probabilities in case of a metal
  - **float effic[npckov]** - detection efficiency
  - **float rindex[npckov]** - refraction index for a dielectric, rindex[0]=0 for a metal

  Remark: In the present program version a mixture may contain a maximum of 5 components. If this is not sufficient one has to change MAXCOMP in hgeomedium.h.

The following parameters are normally not read. The default values are -1 and the real values are automatically calculated by Geant. If the user wants to set these values manually, they must type the keyword AUTONULL in the media file. After this keyword all media must contain these additional 4 parameters before the Cherenkov (int npckov).

- **float madfld** - maximum angular deviation TMAXFD due to field
- **float maxstep** - maximum step permitted STEMAX
- **float maxde** - maximum fractional energy loss DEEMAX
- **float minstep** - minimum value for step STMIN

The ATTPCROOTv2/geometry/ directory is the location of the geometry macros as well as the geometry files that they create. These macros are basically comprised of three functions. The main function, which ought to have the same name as the file name, builds the geometry using the TGeo ROOT geometry package and writing it to a file. The create_materials_from_media_file() function loads the parameters defining the materials in an external dictionary of materials called media.geo. The create_detector() function is what defines the volumes of the detector, their dimensions, and the material of which each component is made.

A working macro to create a basic version of the AT-TPC geometry for use with a $(d,^2He)$ reaction is ATTPCROOTv2/geometry/ATTPC_d2He.C, which can be run using the command `root -l ATTPC_d2He.C`. In general, the main function will not need to be edited. If new materials need to be used (e.g. same gas at new pressure), they must first be defined in the ATTPCROOTv2/geometry/media.geo file if they are not already. Examples of media definitions following the parameters outlined above are shown in Figure 3.4. If new volumes need to be added, simply create a new TGeoVolume in the create_detector()

function in a similar manner to those already defined. Then changes needed in the macro include:

1. Add its predefined name as a constant TString for later use (after included packages at beginning of macro)
2. Load the new material as FairGeoMedium objects in the create_materials_from_media_file() function
3. Create TGeoMedium objects of the material for use in the TGeoVolume definitions

Two ROOT files are created, like ATTPC_d2He.root and ATTPC_d2He_geomanager.root as in the example macro above. The first file is the geometry file that contains the TGeoVolume which consists of the top volume (vacuum) and nodes corresponding to the components of the detector and target added in the macro. A visualization of this geometry is shown in Figure 3.5. The second file created is the TGeoManager file which contains a list of the media, volumes, and their positions defined in the macro.

```
Air             3 14.01 16. 39.95.  7.  9.  18. 1.205e-3  .755  .231  .014
                0 1 3.  .001
                0

Aluminum        1  26.98  13.  2.7
                0  1  20.  .001
                0

TargetD2_07     2  2.0141  2.0141  1.  1.  0.1143e-3  0.5  0.5
                1  1  20  .001
                0

AUTONULL

HeCO2           3  4.0026  12.01  15.999  2.  6.  8.  3.2716e-4  0.9  0.0333  0.0667
                1  1  20.  .001
                99.  0.001   0.0001  0.0000001
                0
```

Figure 3.4: Examples of media definitions for air (1 atm), aluminum metal, deuterium gas (0.7 atm), and $HeCO_2$ gas. Note that since the **ncomp** parameter (first row, first number) for each mixture is positive, the **wm** parameters (first row, last ncomp numbers) are listed by weight proportion.

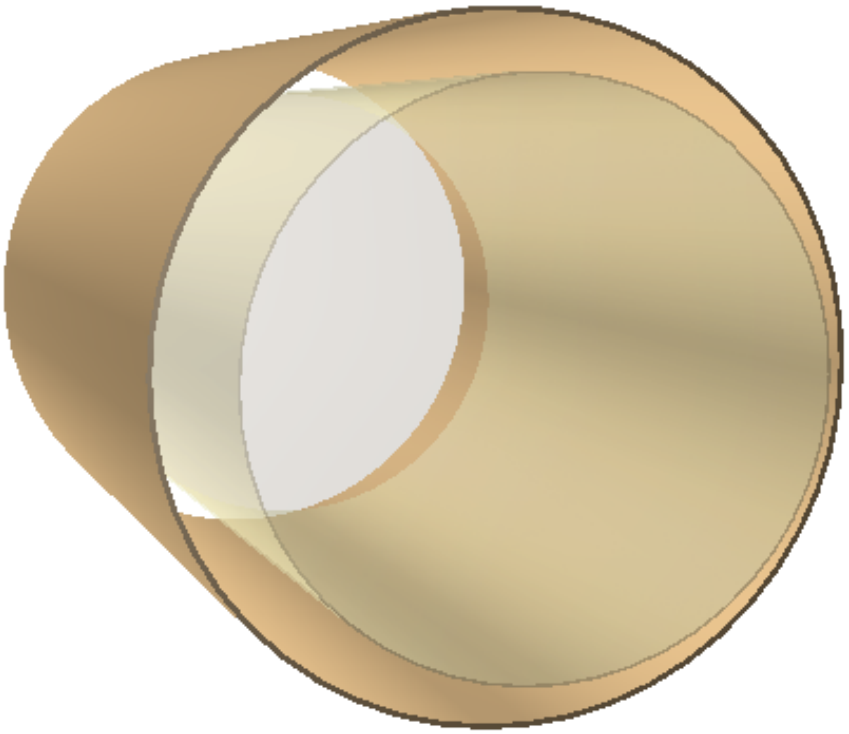


Figure 3.5: A very basic geometry of the AT-TPC with a drift volume of deuterium gas.

3.2.2 Parameters

FairRoot analysis of simulation results requires numerical parameters to process the data. Some information required by the analysis tasks can include geometry positions of detectors or digitization parameters related to the experimental set up or detector response specifications. The FairRuntimeDB class is a repository for these parameters used in FairRoot for simulation and analysis. This class contains a list of parameter containers as well as a list of runs performed and the different versions of the parameters used in each run. Inputs to this class can either be ROOT files or ASCII files.

```
##########################################################################################################
# Parameters used throughout ATTPCROOT software                                                          #
# Format:                                                                                                #
# parameter_name:parameter_type(Int_t-integer, Double_t-double) parameter_value                          #
# ATd2HePar                                                                                              #
# Description of parameters                                                                              #
##########################################################################################################
[ATDigiPar]
PadPlaneX:Int_t                 864   # mm
PadPlaneZ:Int_t                1344   # mm
PadSizeX:Int_t                    8   # mm
PadSizeZ:Int_t                   12   # mm
PadRows:Int_t                   108   #
PadLayers:Int_t                 112   #
AnodeWirePlaneY:Double_t      -4.05   # mm
GroundWirePlaneY:Double_t      -8.1   # mm
GatingWirePlaneY:Double_t       -14   # mm
EField:Double_t               50000   # V/m
NumTbs:Int_t                    512
SamplingRate:Int_t             12.5   # MHz
DriftVelocity:Double_t         2.55   # cm/micros
DriftLength:Double_t           1000   # mm //ATTPC : 1000 - Prototype : 500
YDivider:Int_t                 1000   # Full drift length will be divided by this number when finding cluster
GasFile:Int_t                     2
PadPlaneFile:Int_t                3
PadShapeFile:Int_t                4   # might not be needed because the shape is simply rectangle
BField:Double_t                 0.0   # Tesla
TiltAng:Double_t                0.0   # Degrees
TB0:Int_t                        98   # Time Bucket reference for micromegas position
LorentzAngle:Double_t           0.0   # Lorentz Angle
TBEntrance:Int_t                512   # Beam position at detector entrance in TB
ZPadPlane:Double_t           1000.0   # Position of the micromegas pad plane
Density:Double_t             0.1143   # Gas density (kg/m3) (0.7 atm deuterium)
ThetaPad:Double_t               0.0   # AT-TPC pad plane rotation
ThetaRot:Double_t               0.0   # Second Lorentz Angle
GasPressure:Double_t            532   # Gas pressure in torr
EIonize:Double_t               13.6   # Ionization energy of gas in eV
CoefL:Double_t                0.025   # Longitudal coefficient of diffusion [cm^-0.5]
CoefT:Double_t                0.025   # Transverse coefficient of diffusion [cm^-0.5]
Gain:Double_t                 100.0   # Average gain of micromegas
MaxRange:Double_t              1000   # Beam energy
##########################################################################################################
```

Figure 3.6: Parameter file (ATTPCROOTv2/parameters/ATTPC.d2He.par) for use in FairRoot analysis of simulation data.

FairRoot simulations automatically create a parameter container that contains information on the geometry used and the runtime database outputs the information to a ROOT file which can be used later as input in analysis. Additional parameters are needed for analysis that the user has to specify which are defined in ASCII files in the ATTPCROOTv2/parameters/ directory. The example shown in Figure 3.6 is used in the simulation and analysis that follows.

These files contain a number of parameters related to the AT-TPC design (outlined with red in the figure), like specifications on the Micromegas pad plane and the location of the anode and cathode wires, which are independent of the experiment and should not be changed.

Additional parameters pertain to the configuration of the AT-TPC in the experimental set up (outlined with blue in the figure), which could change between experiments. Of the remaining parameters, NumTbs and SamplingRate are parameters used for digitization and should not be changed. The rest of the parameters are dependent on the properties of the gas used in the AT-TPC. Eionize and Density will change depending on the isotope used as the target and the density will depend on the pressure defined by GasPressure. The DriftVelocity parameter refers to the drift velocity of electrons through the gas and is dependent on the magnitude of the electric field in the AT-TPC defined by the EField parameter and the pressure of the gas [53].

3.2.3 Generators

Another input to the FairRunSim is provided using particle generators classes with inherited members from the FairGenerator class. These generators create single particles or multiple particles involved in a reaction, calculating the kinematics involved, given the initial conditions of the particles. Each contains the ReadEvent() function that adds tracks to the stack in the PrimaryGenerator. Predefined generators can be found in the ATTPCROOTv2/ATGenerators/ directory.

The FairPrimaryGenerator class handles the MC input of the event generation, allowing the user to set various aspects of the beam, target, and number of events. This object is only instantiated once but several generators can be registered to it depending on the reaction being simulated. The primary generator contains a FairGenericStack which holds information on every particle generated in a list of "tracks", which describes the particle type, momentum, and position.

Reactions in the AT-TPC must be simulated using two events. In simulations involving

solid targets, the beam prior to the reaction does not need to be considered. However in the case of a gas target like in the AT-TPC, the gas has an effect on the energy and trajectory of the beam in the moments leading up to the reaction. This means that an event must be generated for both the beam and the reaction itself so that the physics engine can propagate the particles before and after the reaction. An example of the declaration of generators for the simulation of the $^{14}\mathrm{O}(d,{}^{2}\mathrm{He})\,^{14}\mathrm{N}$ reaction in inverse kinematics is shown in Figure 3.7.

Each simulation macro in the AT-TPC will first register an ATTPCIonGenerator object to the primary generator. It is important to note the units of the arguments used in the constructor of this class. The momenta must be defined in units of GeV per nucleon and the beam energy, beam mass, and nominal energy must be defined in units of GeV. The nominal energy is the amount of energy that the beam would lose to the gas if it passed through the entire active volume of the AT-TPC without reacting. The generator randomly chooses an energy between zero and the nominal energy. This determines the reaction vertex location in the detector as it would correspond to the amount of energy lost up to that location.

For the $(d,{}^{2}\mathrm{He})$ reaction, the ATTPC_d2He object is registered as the second event generator. When this class generates an event, it calculates the relativistic kinematics of the two-body reaction as well as the kinematics for the two protons produced. The argument for its constructor takes an array with an entry corresponding to each particle considered in the reaction: (1) beam (^{14}O), (2) target (d), (3) ejectile (^{14}N), (4) recoil (^{2}He), and (5,6) each proton. The arguments of this class require different units, namely momenta and excitation energy must be defined in units of MeV and the mass must be defined in atomic mass units. An additional input to this generator is the data from the ACCBA calculations of the double differential cross-sections for this reaction. The macro shown in Figure 3.7 reformats the resulting data file from these calculation for use in the generator.

```
// -----   Create PrimaryGenerator   --------------------------------------
FairPrimaryGenerator* primGen = new FairPrimaryGenerator();

Double_t unitmass = 931.494; // MeV per amu

// Beam Information
Int_t z = 8;     // Atomic number
Int_t a = 14;    // Mass number
Int_t q = 0;     // Charge State
Int_t m = 1;     // Multiplicity
Double_t kBeam = 115.0; // Beam energy per nucleon
Double_t BExcEner = 0.0;
Double_t Bmass = 14.008596359*unitmass/1000.0;  // Mass in GeV
Double_t NomEnergy = 0.0858025;                 // Nominal Energy of the beam
Double_t px = 0.000/a;  // X-Momentum per nucleon
Double_t py = 0.000/a;  // Y-Momentum per nucleon
Double_t pz = sqrt( pow(kBeam * a / 1000.0 + Bmass,2) - pow(Bmass,2) )/a; // Z-Momentum per nucleon

ATTPCIonGenerator* ionGen = new ATTPCIonGenerator("Ion",z,a,q,m,px,py,pz,BExcEner,Bmass,NomEnergy);
primGen->AddGenerator(ionGen);
// Variables for 2-Body kinematics reaction
std::vector<Int_t> Zp;  // Zp
std::vector<Int_t> Ap;  // Ap
std::vector<Int_t> Qp;  // Electric charge
Int_t mult = 6;         // Multiplicity
std::vector<Double_t> Pxp;  // Px momentum X
std::vector<Double_t> Pyp;  // Py momentum Y
std::vector<Double_t> Pzp;  // Pz momentum Z
std::vector<Double_t> Mass; // Masses
std::vector<Double_t> ExE;  // Excitation energy

// ---- Beam (1) ----                  // ---- Target (2) ----          //--- Scattered (3) -----
Zp.push_back(z);                       Zp.push_back(1);               Zp.push_back(7);
Ap.push_back(a);                       Ap.push_back(2);               Ap.push_back(14);
Qp.push_back(0);                       Qp.push_back(0);               Qp.push_back(0);
Pxp.push_back((a*1000.0)*px);          Pxp.push_back(0.0);            Pxp.push_back(0.0);
Pyp.push_back((a*1000.0)*py);          Pyp.push_back(0.0);            Pyp.push_back(0.0);
Pzp.push_back((a*1000.0)*pz);          Pzp.push_back(0.0);            Pzp.push_back(0.0);
Mass.push_back(Bmass*1000.0/unitmass); Mass.push_back(2.01410177812); Mass.push_back(14.00307400443);
ExE.push_back(0);                      ExE.push_back(0.0);            ExE.push_back(0.0);

// ---- Recoil (4) -----               // ---- proton 1 (5) -----      // ---- proton 2 (6) -----
Zp.push_back(2);                       Zp.push_back(1);               Zp.push_back(1);
Ap.push_back(2);                       Ap.push_back(1);               Ap.push_back(1);
Qp.push_back(0);                       Qp.push_back(0);               Qp.push_back(0);
Pxp.push_back(0.0);                    Pxp.push_back(0.0);            Pxp.push_back(0.0);
Pyp.push_back(0.0);                    Pyp.push_back(0.0);            Pyp.push_back(0.0);
Pzp.push_back(0.0);                    Pzp.push_back(0.0);            Pzp.push_back(0.0);
Mass.push_back(2.0*1.0078250322);      Mass.push_back(1.0078250322);  Mass.push_back(1.0078250322);
ExE.push_back(0.0);                    ExE.push_back(0.0);            ExE.push_back(0.0);
Double_t ThetaMinCMS = 0.0;
Double_t ThetaMaxCMS = 180.0;
Int_t N_cross = 6560;
std::vector<Double_t> Arr1(N_cross), Arr2(N_cross), Arr3(N_cross);
Double_t col1, col2, col3;

// Read ACCBA cross-section data
string filename= "all2_140.dat";
ifstream  inputfile;
inputfile. open(filename.c_str());
if(inputfile.fail() ){
cerr << "Error opening "<< filename << endl;
exit(1);
}

for(Int_t i=0;i<N_cross;i++){
inputfile >> col1 >> col2 >> col3 ;
Arr1.at(i) = col1;
Arr2.at(i) = col2;
Arr3.at(i) = col3;
}
inputfile.close();

ATTPC_d2He* d2He = new ATTPC_d2He("d_2He",&Zp,&Ap,&Qp,mult,&Pxp,&Pyp,&Pzp,&Mass,&ExE, &Arr1, &Arr2, &Arr3, N_cross);
primGen->AddGenerator(d2He);

run->SetGenerator(primGen);
```

Figure 3.7: Example macro code showing input required in definition of ATTPCROOTv2 generators, ATTPCIonGenerator and ATTPC_d2He, for simulation of a $(d, {}^2\mathrm{He})$ reaction.

3.2.4 Monte Carlo Simulation

The Geant4 platform is the software toolkit that handles the simulation of particles through matter using Monte Carlo methods. Some of the relevant capabilities of this software include [51]:

- *Geometry Management* - handles the physical layout of the detector including the various media involved and considers the effects these materials have on the path of particles.
- *Tracking* - propagates particles through matter and considers possible interactions and decay processes.
- *Detector response* - records when a particle passes through the detector volume and approximates how a real detector would respond.
- *Run management* - records the details of each run (a set of events), as well as setting up the experiment in different configurations between runs.

While the generators define the initial conditions of the beam and the reaction, Geant4 determines what happens to the particles in between the two events and afterwards, taking into consideration the energy loss that occurs during transport and the interactions that may occur with each material in the geometry. In order to carry the information of the vertex location and energy loss between the event generators, an object called ATVertexPropagator is created at the beginning of the simulation and its member variables (e.g. momenta, energy loss, and position) are set. Each generator can then read this information to initialize the properties of the particles it generates.

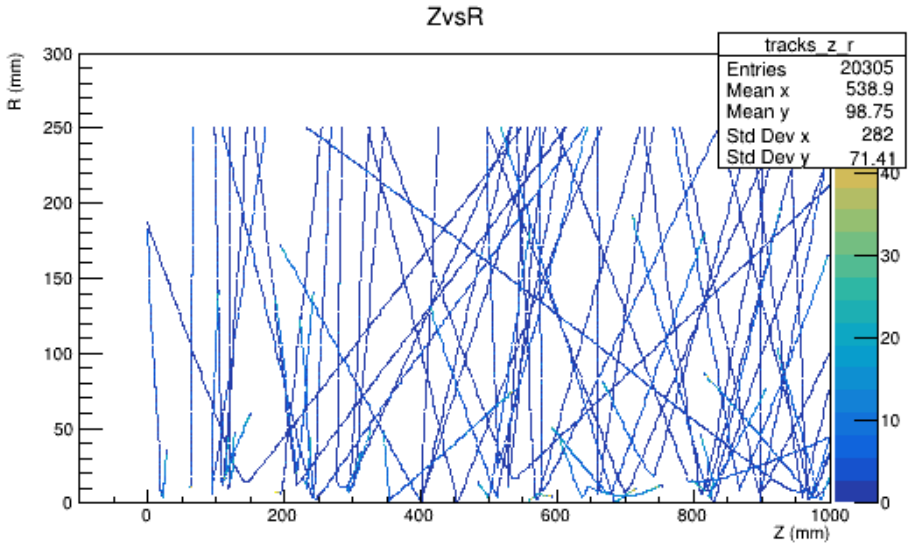
Predefined ROOT macros for FairRoot simulations are found in the ATTPCROOTv2/-macro/Simulation/ directory. For simulation of the ^{14}O$(d,{}^{2}$He$)\,^{14}$N reaction in inverse kinematics at 115 MeV/u beam energy and 0.7 atm deuterium gas pressure, data is produced using the d_2He_sim_14O.C macro in the ATTPCROOTv2/macro/Simulation/d2He/ direc-

tory. This file reads data from the ACCBA calculations stored in all2_14O.dat in the same directory. Additionally, it requires the geometry file ATTPC_d2He_07atm.root to be present in the ATTPCROOTv2/geometry/ directory which can be created using the ATTPC_d2He.C macro in the geometry folder.

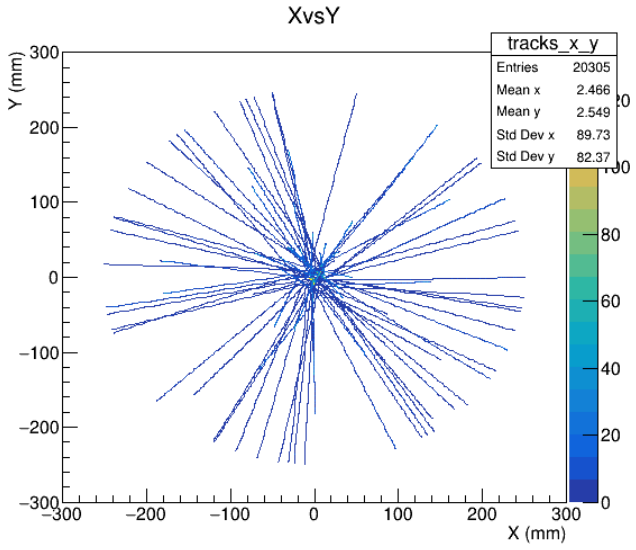
The output of ATTPCROOTv2 simulations is a ROOT file which contains a branch of AtTpcPoint objects and a ROOT file which holds the parameter containers that are used later in FairRoot analysis macros. Each AtTpcPoint corresponds to a particle and is designated by an event ID, track ID, and the volume it occupies as well as the particle's position, momentum, and initial energy and angle. Since the simulation run is defined by an ion generator and a $(d, {}^2\mathrm{He})$ reaction generator, each even event ID corresponds to a beam event and each odd event ID corresponds to a reaction event. Track ID numbers and their corresponding particles are: (0) ^{14}O, (1) ^{14}N, and (2/3) protons.

The kinematic parameters can be studied using the macro d2He_ana_14O_07atm.C in the ATTPCROOTv2/macro/Simulation/d2He/Analysis_d2He/ directory. This analysis file creates three sets of histograms stored in branches of a ROOT file called "Parameters", "He2_reconstr", and "He2_reconstr_resol". The histograms in the "Parameters" directory display information taken directly from each AtTpcPoint in the simulation run being analyzed with such plots as "Z vs R" and "X vs Y" as seen in Figure 3.8. The histograms in the "He2_reconstr" directory provide information on the reconstruction of each ^{2}He particle with calculations based off of the information taken directly from the AtTpcPoints in each event. Histograms such as the θ_{cm} distribution and "$\mathrm{KE}_{\mathrm{lab}}$ vs θ_{lab}" are display in Figure 3.9. The histograms in the "He2_reconstr_resol" directory contain many of the same histograms as in "He2_reconstr" except the information on the reconstruction of each ^{2}He particle is calculated based off of a graphical fit of each proton track, which is then used to determine

the proton energy and angles needed for reconstruction. The results of this are displayed in Figure 3.10 and can be compared to those from "He2_reconstr".

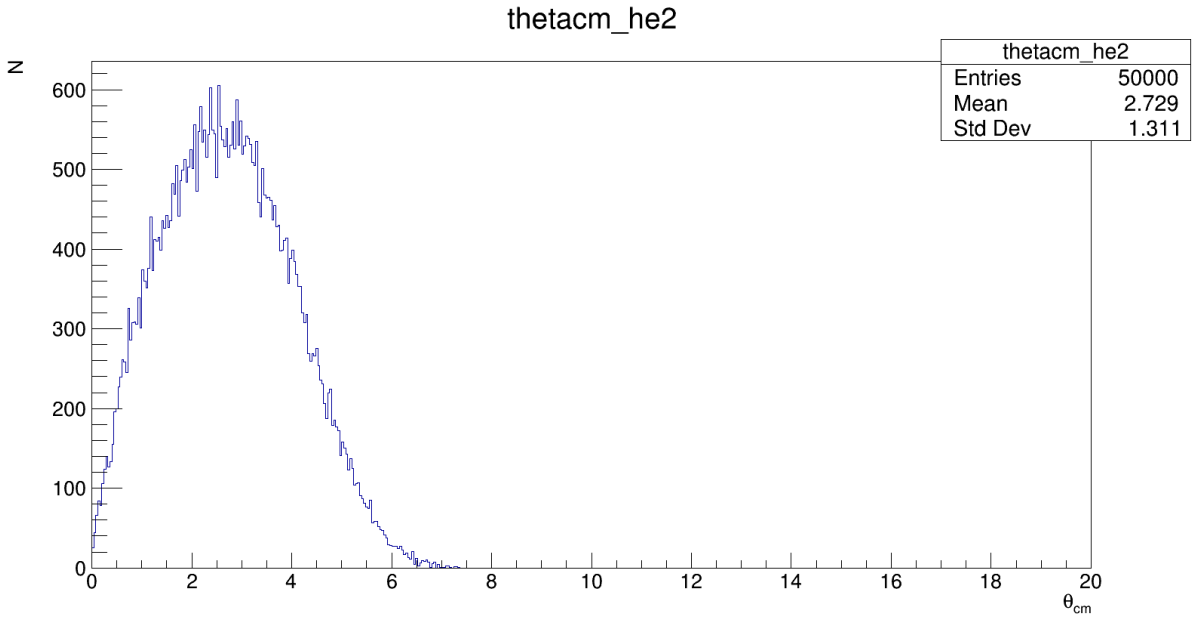


(a)

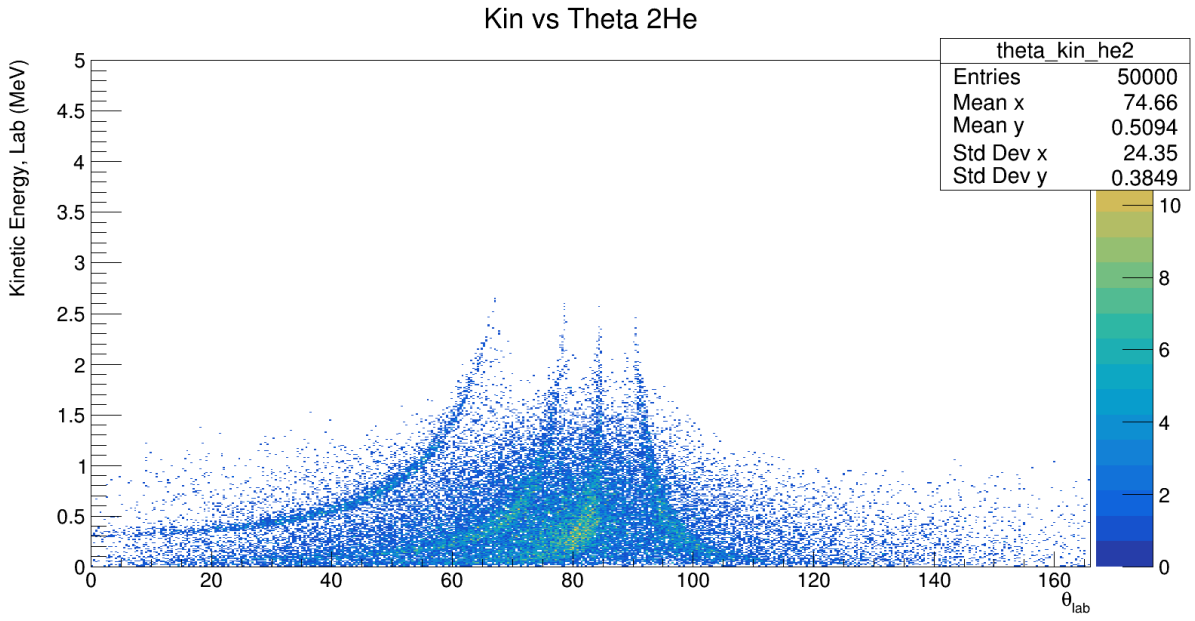


(b)

Figure 3.8: A total of 20305 AtTpcPoints were created in a 100 event simulation of the ^{14}O$(d, ^{2}$He$)\,^{14}$N reaction. Each point holds the position, momentum, and energy of a particle at one moment in its trajectory. These positions are plotted in the histograms shown. (a) Radius versus Z-component of tracks made by protons in the AT-TPC. (b) Y-component versus X-component.

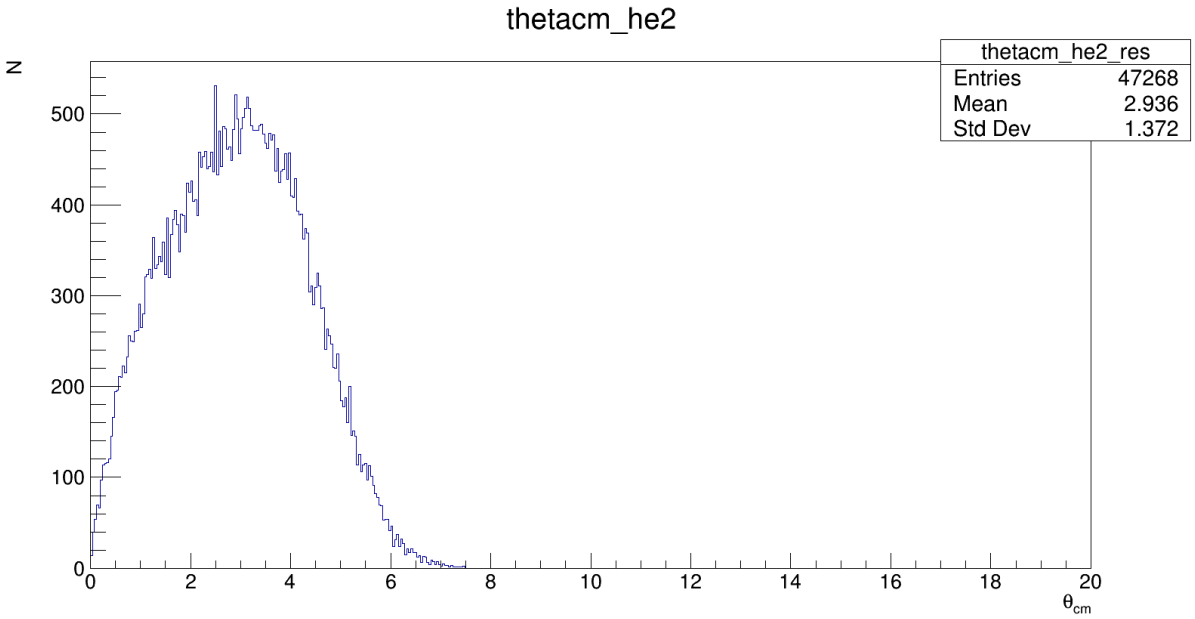


(a)

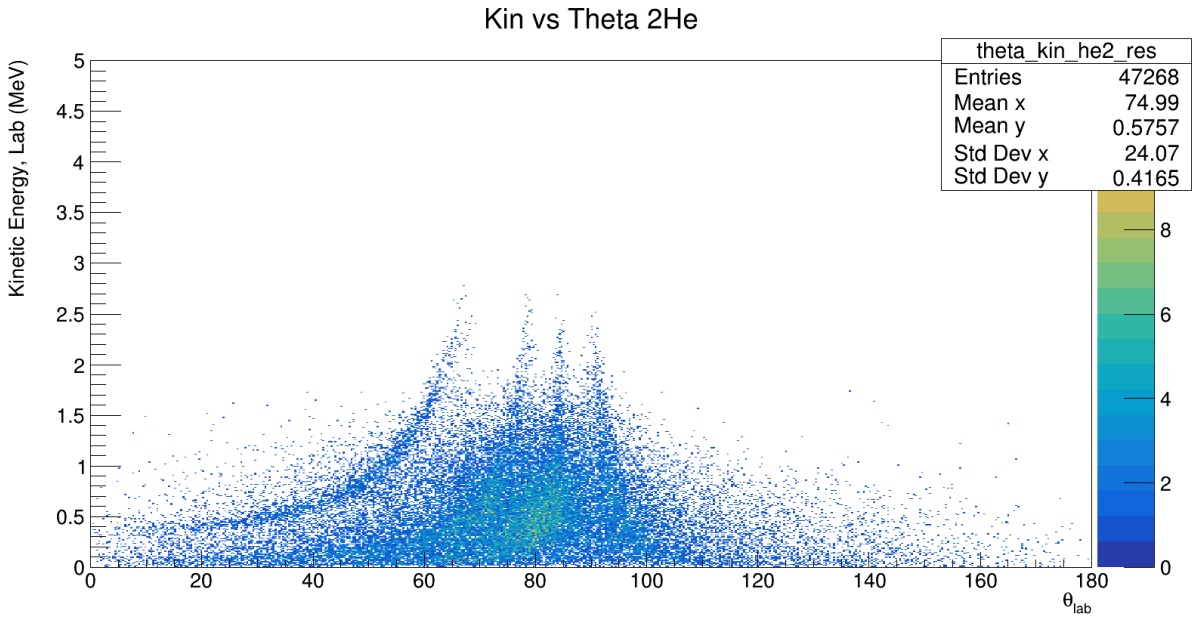


(b)

Figure 3.9: Results of invariant mass calculation for a 10000 event simulation calculated from AtTpcPoint parameters. (a) Angular distribution of the ^{2}He particle center-of-mass. (b) Kinetic energy versus angle in the laboratory for the ^{2}He particle.



(a)



(b)

Figure 3.10: Results of invariant mass calculation for a 10000 event simulation calculated graphical fit of each proton to reconstruct the ^{2}He particle. (a) Angular distribution of the ^{2}He particle center-of-mass. (b) Kinetic energy versus angle in the laboratory for the ^{2}He particle.

3.3 Digitization

The data that comes from prior simulation is not very useful for analysis because it is not formatted similarly to the data that will be taken during the experiment. The experimental data is blind to the type of particle producing each track detected in the pad plane and provides no information on the momentum, energy, or trajectory angle for each point. This information must be extracted from a collection of pulses created in the pad plane by using the location of the each pad and the timing of the pulse to construct the tracks in three dimensions.

The digitization of the simulated data is performed using the rundigi_sim.C macro found in the ATTPCROOTv2/macro/Simulation/d2He/Analysis_d2He/ directory. This FairRoot analysis requires two files as input to the runtime database, which are the parameter file created by the simulation macro and the parameters described in Section 3.2.2.

3.3.1 Clusterization

The ATClusterizeTask class found in the ATTPCROOTv2/digi/ directory is a FairTask that accepts a branch of AtTpcPoint objects as input. The name of this task is simply refers to the process of creating groups of electrons between each point along the proton trajectories. This task loops through every point in each event and calculates the number of electrons (with some Gaussian fluctuation dependent on the Fano factor) ionized by a particle passing through the detector's gas using the energy loss since the previous point. The distance between the current point and the previous point is divided into equal spaces such that one of the ionized electrons can be placed at a point in each one. Each electron's position is calculated with some Gaussian fluctuation using the longitudinal and transverse

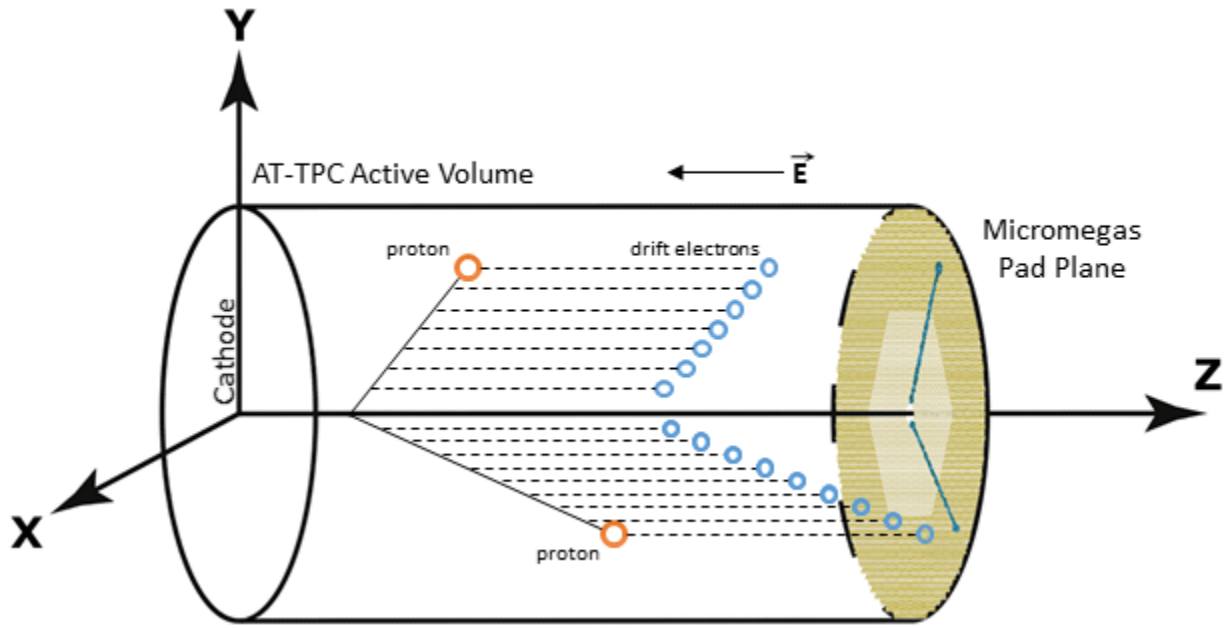


Figure 3.11: Illustration of the ionization of the deuterium gas as the two protons from a $(d, {}^2\mathrm{He})$ reaction travel through the AT-TPC volume. Ionized electrons drift toward the pad plane in the presence of the electric field and undergo a Gaussian diffusion through the deuterium gas, adding width to the tracks detected in the pads. Figure adapted from [42].

coefficients of diffusion found in the parameter file to account for dispersion. The drift time is determined by dividing the Z-component of each electron's position by the drift velocity found in the parameter file. The task produces a branch of ATSimulatedPoints each of which contain the X- and Y-components of an electron as well as the drift time. An illustration of the process that is simulated by this task can be seen in Figure 3.11.

3.3.2 Pulse Creation

The ATPulseTask class found in the ATTPCROOTv2/digi/ directory is a FairTask that accepts a branch of ATSimulatedPoint objects as input. The task produces a branch of ATRawEvent objects each of which contain the collection of ATPad objects. An AtTpcMap object is created which can be used to map the X- and Y-components of each ATSimulat-

edPoint to one of 10240 pads in the pad plane. This task loops over each event, creating an ATPad for every new X- and Y-coordinate and adding hits to existing ATPads when multiple electrons are mapped to the same pad. Each hit is registered with a time stamp corresponding to the electron's drift time. Once all ATSimulatedPoints in the event have been registered to a pad, the task loops over the pads and creates signals according to a pad response function. Multiple hits result in a superimposition of pulses. The binning parameters and gain in ATTPC.d2He.par are needed for this task. A visualization of the pulses is shown in Figure 3.12 created using the runeve.C macro in the analysis directory, which produces an interactive pad plane that displays the hits and their pulses for each event.

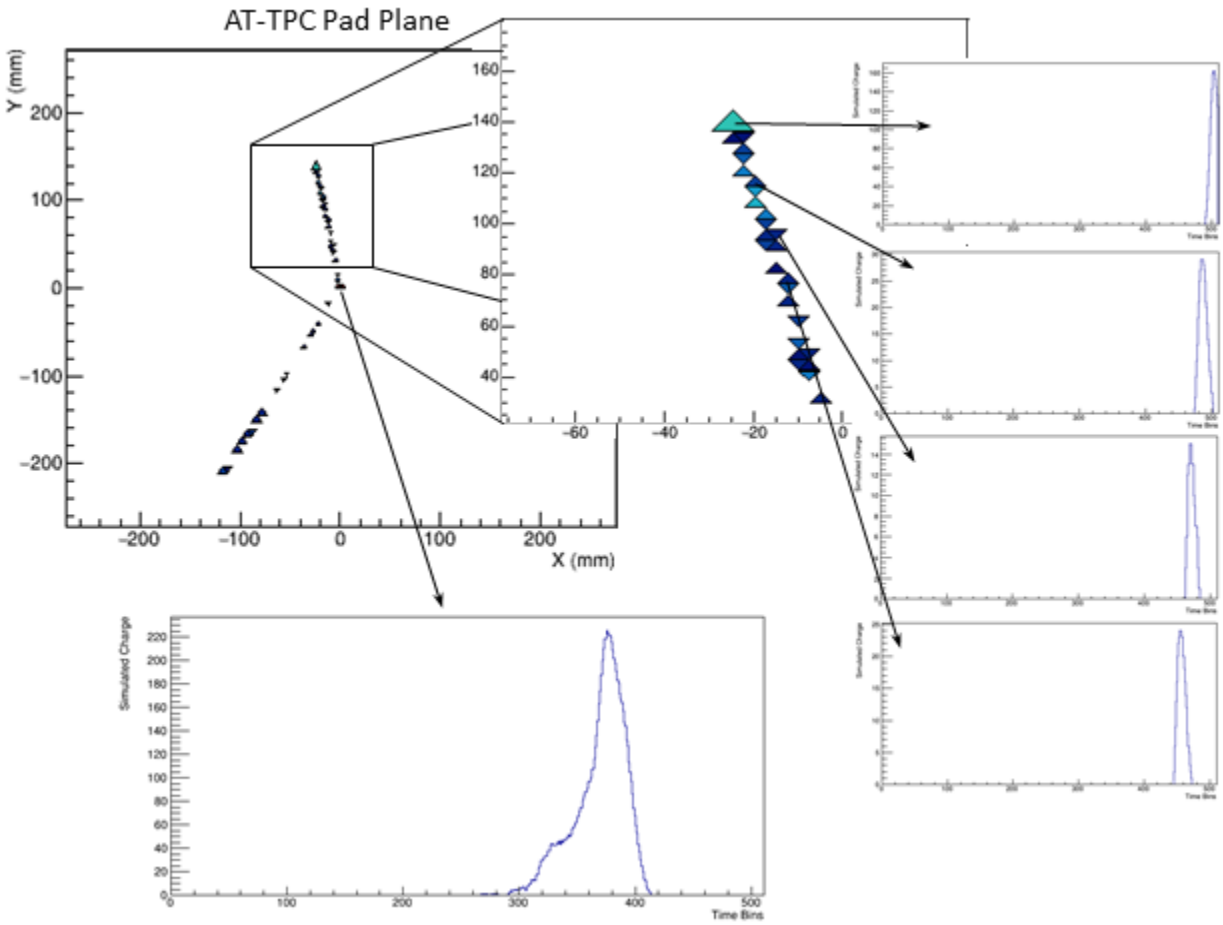


Figure 3.12: Visualization of the digitized data displayed as a map of the AT-TPC pad plane. Each pad in the interactive map can be selected to view the pulse produced by the ATPulseTask task. One the right side of the figure, four pads are selected showing the change in the timing of the pulse for each subsequent point in the proton's trajectory. The plot on the bottom shows the pulse created in a pad where electrons arrived to the same pad at different times, resulting in multiple response functions superimposed.

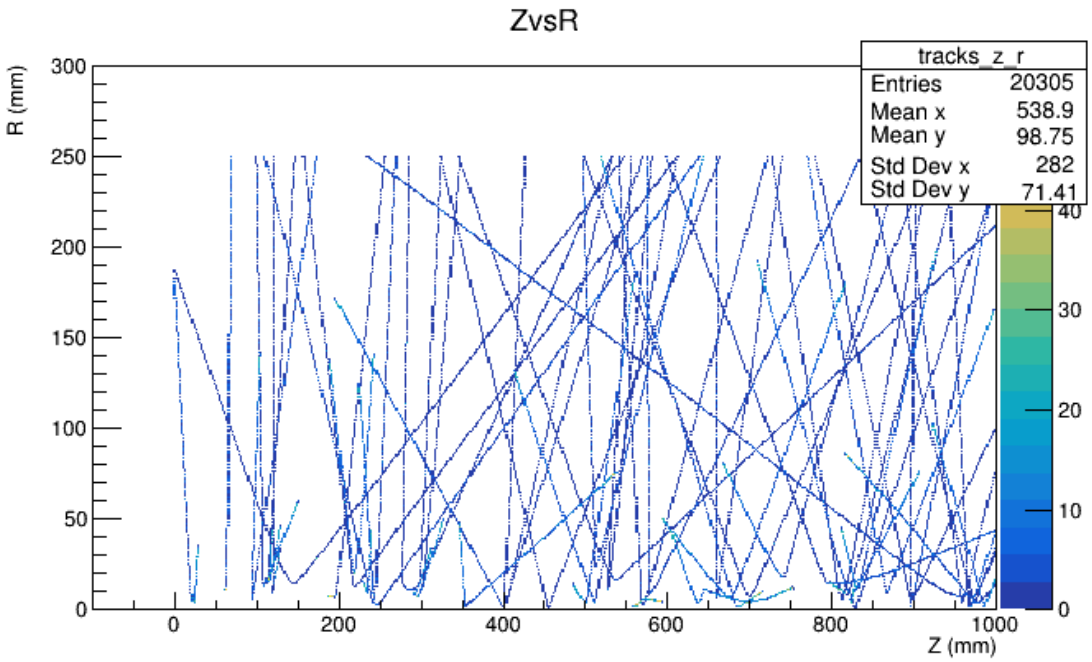
Chapter 4

Analysis

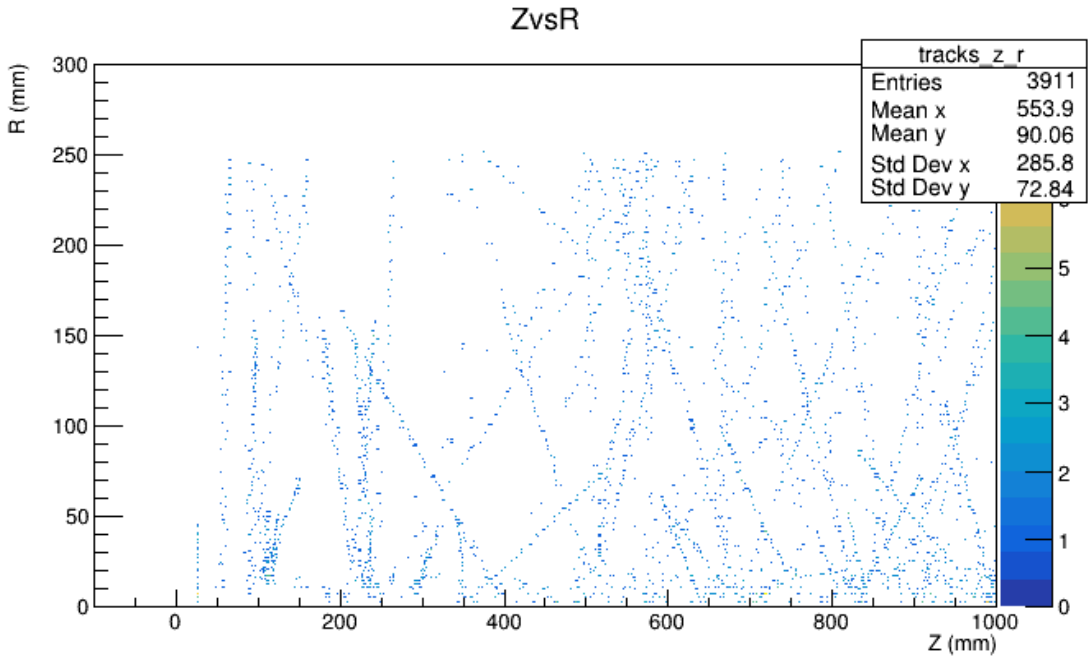
The data up to this point came from simulation of charge-exchange reactions. The analysis in the following discussions pertains to both data created in simulation and data recorded in experiment. After the simulated data undergoes digitization, it is formatted similarly to experimental data and is an accurate representation of what will be observed. Therefore, the techniques proposed in this chapter and executed on simulation data serve as proof of their viability for analyzing experimental data.

4.1 Pulse Shape Analysis

The ATPSATask class found in the ATTPCROOTv2/reco/ directory is a FairTask that accepts a branch of ATRawEvent objects as input. The task produces a branch of ATEvent objects each of which contain the collection of ATHit objects. Each ATHit object contains the X-, Y-, and Z-coordinates of a particle. An ATHit is created for each ATPad in the ATRawEvent where the X- and Y-coordinates correspond to the position of the pad in the plane. The Z-coordinate is determined by analyzing the pulse in the ATPad in one of two ways: (1) finding the maximum value of the pulse or (2) fitting the pulse and and finding peaks. This provides the time of the hit which can be used with the drift velocity to find the Z-coordinate. An comparison of the data before and after digitization and pulse shape analysis can be seen in Figures 4.1 and 4.2.

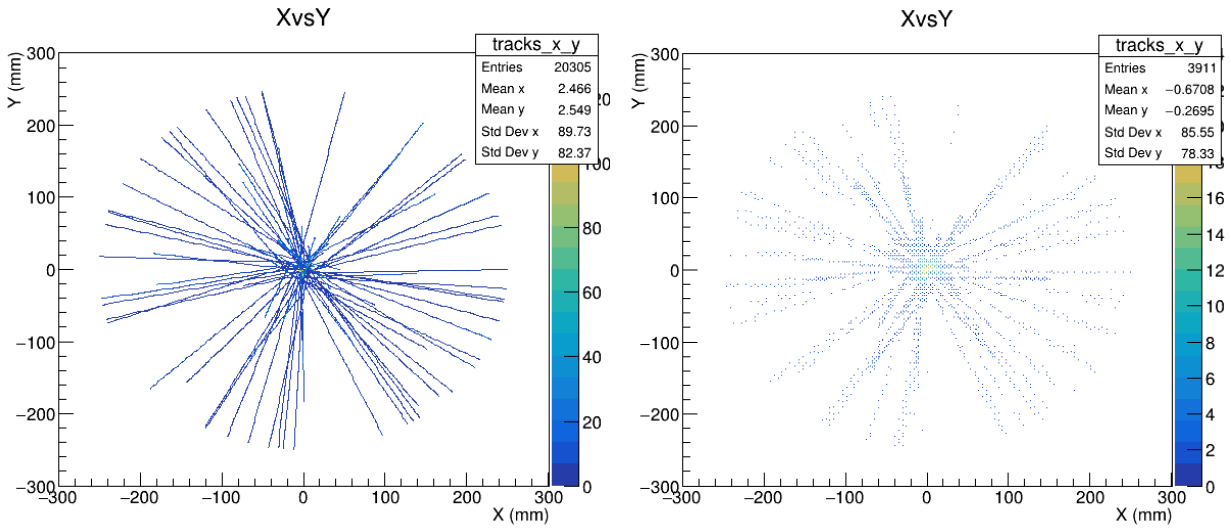


(a)



(b)

Figure 4.1: The 20305 AtTpcPoints created in a 100 event simulation of the $^{14}\mathrm{O}(d,{}^{2}\mathrm{He})\,{}^{14}\mathrm{N}$ reaction resulted in 3911 ATHits after digitization and pulse shape analysis. It is important that the spatial components of the tracks remain the same before and after this process. The radius versus Z-component for the tracks are plotted for data (a) just after simulation as in Figure 3.8a (b) and after digitization and pulse shape analysis.



(a) (b)

Figure 4.2: The 20305 AtTpcPoints created in a 100 event simulation of the $^{14}\mathrm{O}(d,{}^{2}\mathrm{He})\,{}^{14}\mathrm{N}$ reaction resulted in 3911 ATHits after digitization and pulse shape analysis. It is important that the spatial components of the tracks remain the same before and after this process. The Y-component versus X-component for the tracks are plotted for data (a) just after simulation as in Figure 3.8b (b) and after digitization and pulse shape analysis.

4.2 RANSAC Tracking

Unlike in simulation, particle tracks observed in experiment are often obscured by noise from external sources and other reaction mechanisms. Furthermore, there is no method for directly differentiating the tracks made by each proton in the detector. Classical fitting techniques, such as least squares, include the entire data set in pursuit of optimizing a fit to the model, meaning noise and multiple tracks can skew a single linear fit.

4.2.1 RANSAC Algorithm

The Random Sampling Consensus (RANSAC) algorithm is a technique for determining the subset of a data set that includes outliers which best fits a given model and providing

parameters for that fit. When applied iteratively to a data set, removing the subset of data best fitting the model after each iteration, multiple features in the data can be extracted each with its own parameters for the given model.

Suppose the problem involves a data set P where the features of the data are defined by a model M that requires a minimum of n points to ensure a unique fit. The general implementation of the algorithm is outlined as follows [54]:

(i) Select a random subset of n points from P.

(ii) Fit the subset to a model M_{test}.

(iii) Test all points in P against M_{test} according to some loss function. Points within a certain threshold are considered inliers and form a subset S_{inliers}.

(iv) Repeat steps (i) through (iii) until S_{inliers} contains a sufficient number of points and the best model, M_{best}, for a single feature has been found.

(v) Return S_{inliers} as an extracted feature with parameters given by M_{best}.

(vi) Remove S_{inliers} from P and repeat steps (i) through (v) to obtain additional features in P.

The efficiency of this process relies on two assumptions on the data set: the noise in the data will not consistently agree with a given model, meaning there are few outliers compared to the data in the features, and the features are prominent enough to agree on a good model, meaning there are few missing data.

The RANSAC algorithm leaves three parameters to be specified: (1) the threshold for agreement with the model, (2) the number of iterations to find one feature, and (3) a standard for what constitutes a good model (i.e., how many inliers is sufficient?).

The tracks left by protons in the AT-TPC can be modeled by a straight line in three dimensions. The minimum number of points for the initial subset of inliers found in step (i) is then $n = 2$. For a linear model, an simple loss function is the distance from the data

point to the line, which leaves the first parameter to be a minimum distance threshold. An appropriate threshold to capture much of the data in a track should be on the order of the standard deviation of the diffusion of the electron Gaussian cloud.

Since the tracks are presumed to contain the most data points, the third parameter does not need to be considered if the number of iterations in the second parameter is large enough. In this case, returning the model which fit the most inliers after all iterations must yield the most populous track. In order to guarantee this case, the number of iterations would have to be large enough to consider every two-element permutation of the data set but this is still an arbitrary number since the total number of data points is still unknown. In practice for simulations, good events have around 50 pad plane hits. Thus, $\binom{50}{2} \approx 10^3$ should be sufficient to yield a track. An additional parameter for the least squares error can be added when a track is fit by many models corresponding to different pairs of the track's inliers.

4.2.2 RANSAC Implementation

The method for implementing the RANSAC algorithm used in the ATTPCROOTv2 framework utilizes the Point Cloud Library (PCL) [47]. This library is useful for creating objects of three-dimensional data points called point clouds which can be modeled. The library has options for modelling using a RANSAC algorithm that returns the parameters for a line in three-dimensions.

The ATRansacTask class found in the ATTPCROOTv2/reco/ directory is a FairTask that accepts a branch of ATEvent objects as input. For each ATEvent, the task converts the positions contained by the ATHit objects into individual points in a point cloud. The PCL RANSAC algorithm is applied to the point cloud and a set of parameters and the inliers they correspond to are extracted. These parameters and points are registered to an ATTrack

object. The inliers are removed from the point cloud and additional tracks are extracted as the algorithm reiterates until the number of points has reduced to 10% of the original point cloud. The task results in an ATRansac object which contains the stack of ATTrack objects. This task, like the pulse shape analysis task, can be added to the FairAnaRun in rundigi_sim.C to analyze simulation data from clusterization to RANSAC analysis in one run.

4.3 Reconstruction of ^{2}He

The code d2He_ana_ransac.C in the ATTPCROOTv2/macro/Simulation/d2He/Analysis_d-2He/ directory performs the invariant-mass analysis to calculate the He momentum vector. For each event where the RANSAC algorithm found more that one track, this macro uses the parameters to construct the momentum vector for each proton. The parameters found by RANSAC are the four coefficients to the parametric form of a three-dimensional line. A point along this line will have coordinates of the form

$$\begin{bmatrix} p[0] + p[1] * t \\ p[2] + p[3] * t \\ t \end{bmatrix}$$

where p is the parameter vector given by RANSAC. To find a vector in the direction of the line, subtract the vectors for two coordinates along the line. Take the points for $t = 0$ and $t = 1$ to get a general form for the vector in the direction of the track:

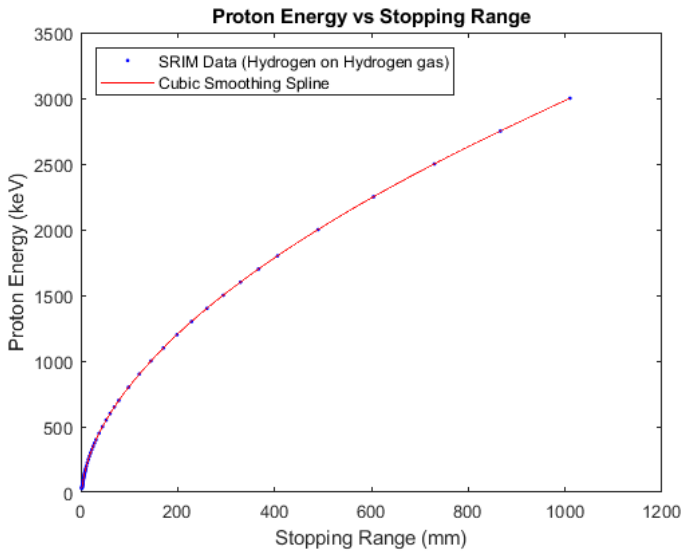


Figure 4.3: Data calculated by the SRIM program [55] for the stopping range of the hydrogen ion in deuterium gas at 0.7 atm pressure. Data is modeled by a cubic smoothing spline function.

$$\begin{bmatrix} p[0]+p[1] \\ p[2]+p[3] \\ 1 \end{bmatrix} - \begin{bmatrix} p[0] \\ p[2] \\ 0 \end{bmatrix} = \begin{bmatrix} p[1] \\ p[3] \\ 1 \end{bmatrix} \tag{4.1}$$

which can be used to determine the angles of each proton, ϕ and θ, as well as the vertex of the reaction. The ϕ-angle is the azimuthal angle about the beam axis (Z-axis) and the θ-angle is the angle with respect to the forward direction of the beam axis.

The energy loss of each proton, E_{loss}, depends on the range that it travels in the detector. The Stopping and Range of Ions in Matter program [55] calculates the stopping range for a set of energies. Fitting this data provides a method for determining the energy loss given the stopping range. The data calculated for a hydrogen ion in deuterium gas at 0.7 atm pressure is shown in Figure 4.3 and is fit using a cubic smoothing spline. The parameters of this fit are stored in the files, energyRangeCoefs.txt and energyRangeKnots.txt, in the analysis

directory. The momentum of each proton, $\vec{p}$, can then be calculated using the relativistic momentum

$$\vec{p}_{\text{proton}} = \sqrt{E_{\text{loss}}^2 - (m_{\text{proton}}c^2)^2} * \begin{bmatrix} \sin\theta\cos\phi \\ \sin\theta\sin\phi \\ \cos\theta \end{bmatrix} \tag{4.2}$$

where $m_{\text{proton}}c^2$ is the rest mass of the proton. The reconstructed ^{2}He momentum vector is given by

$$\vec{p} = \vec{p}_{\text{proton},1} + \vec{p}_{\text{proton},2}. \tag{4.3}$$

The ATd2HeAnalysis class found in the ATTPCROOTv2/reco/ATAnalysis/ directory has a function that performs the relativistic kinematics calculations needed to find the excitation energies of ^{14}N and the center-of-mass scattering angles.

Chapter 5

Conclusions and Outlook

Electron-capture rates for unstable nuclei play an important role in the evolution of core-collapse supernovae. The well-established proportionality between the Gamow-Teller transition strengths, on which the electron-capture rates strongly depend, and the cross-section of charge-exchange reactions makes such reactions powerful tools for estimating electron-capture rates. Establishing $(d, {}^2\mathrm{He})$ reaction in inverse kinematics as a viable probe for studying unstable nuclei will enable studies of nuclei in the region around $N = 50$ described by Sullivan et al for which the change in electron number is highly sensitive in core-collapse simulations.

The experiment described in Chapter 2 is expected to run at the NSCL in 2020. The results of this experiment will serve as the first test of the $(d, {}^2\mathrm{He})$ reaction in inverse kinematics as a new probe in the (n, p) direction for unstable nuclei. The ATTPCROOTv2 framework provides a useful tool for the simulation and analysis of reactions involving the AT-TPC detector. The RANSAC algorithm together with the concepts behind the AT-TPC will enable the accurate measurement of proton tracks in experiment. The code described in this thesis needs further optimization to ensure the simulation and future analysis yields the most realistic results and the tracking algorithm properly estimates the range and energy of each proton. A manual for installation and use of the code as a simulation and analysis tool for a broader range of experiments in the AT-TPC was developed as part of this work.

APPENDICES

Appendix A

ATTPCROOTv2 Manual

This manual is a guide to installation of and implementation of the ATTPCROOTv2 code, which is the framework based on FairRoot for data analysis and simulations of reactions in the Active-Target Time-Projection Chamber (AT-TPC). While this thesis describes use of this code specifically for the $^{14}\mathrm{O}(d, ^{2}\mathrm{He})\,^{14}\mathrm{N}$ reaction in inverse kinematics, use of the framework code can be extended to any experiment that may involve the use of the AT-TPC. This manual will describe the general building blocks for simulations using this broader range of reaction mechanisms.

A.1 Installation

The framework is built upon ROOT [44] and requires external libraries (FairSoft [45] and FairRoot [46]) to be compiled and executed, which are developed by other groups. Additional libraries for analysis are needed, like PCL [47], FLANN [48], and Eigen [49] and libraries like the HDF5 library [50] are needed for data formatting. The Monte Carlo simulations are handled by the Geant4 toolkit [51].

For NSCL users, log into fishtank. The dependent packages are already installed on the server. The modules will simply need to be loaded at each login. Change directories to the one where the analysis and simulations will be performed and data will be stored. Input the following commands:

```
module load gnu/gcc/6.5

module load fairroot/18.00

git clone -b develop https://github.com/ATTPC/ATTPCROOTv2.git

cd ATTPCROOTv2

mkdir build

cd build

cmake ../

make -j8
```

This should install the ATTPCROOTv2 libraries and headers. The config.sh file that is now in the build/ directory loads the environment variables but, in this process, destroys some of the FairRoot settings. After sourcing the configuration file, the FairRoot and compiler modules need to be loaded again.

```
source /path/to/ATTPCROOTv2/build/config.sh

module load gnu/gcc/6.5

module load fairroot/18.00
```

At each login, these three commands need to be run for the environment variables to be set and the needed modules to be available. They can be added to the user's bash script file for convenience. After this process, the macros for simulation and analysis described in the following two chapters are ready to be compiled and executed.

For non-NSCL users, installation on a personal computer will require the following dependent packages before installing ATTPCROOTv2:

1. FairSoft

2. FairRoot

3. FLANN, Eigen, PCL, and HDF5 libraries

Once these packages are installed, the ATTPCROOTv2 code can be cloned from https://github.com/ATTPC/ATTPCROOTv2. As of September 2018, the ATTPCROOTv2 code has been compiled with the following versions:

- FairSoft may18 [45],
- FairRoot 17.10 [46],
- boost 1.67.0,
- CMake 3.12.0,
- flann 1.9.1 [48],
- eigen 3.3.6 [49],
- pcl 1.8.0 [47],
- hdf5 1.10.2 [50]

Operating systems tested:

- Mac OS X 10.13 (High Sierra) & clang 10.0
- Ubuntu 16.04.2 &
- Debian GNU/Linux 8.11 & gcc 6.4.1

If the code needs to be installed on a server where the environment might change often, consider using CMake provided in FairSoft to make sure everything is compiled with the same versions.

FairSoft:

FairSoft is a software package that includes the external software to compile and run FairRoot. The package can be installed through git either in any Linux distribution (successfully tested in Fedora/Red Hat, Scientific Linux, Ubuntu/Debian and MacOs Yosemite and Mavericks. In order to avoid permission problems, it is better to install it in any folder below the home directory. Here is the link to the GitHub account for FairSoft: https://github.com/FairRootGroup/FairSoft and a link the official web-page and forums: https://fairroot.gsi.de.

There are several libraries needed to install FairSoft (detailed explanation in DEPENDENCIES text file on FairSoft repository). First, one should have installed CMake. Then other packages are needed that depend on the OS and are detailed below.

For installation of FairSoft on a Mac OS, see the following for the specific dependencies each version requires and some methods for install:

- Mac OS X 10.8 and earlier

  Xcode developer package (for make, g++, gcc, ld, libX11, etc.) X11User (for the X11 server) gfortran

- Mac OS X 10.9 (Mavericks) and 10.10 (Yosemite)

  Xcode from AppStore (after installing, run `sudo xcode-select --install` from a Terminal) XQuartz version 2.7.5 and later gfortran version 4.9 only. Note: It will be mandatory to install dpkg library through macports or brew (Install macports and

then `sudo port install dpkg`). The procedure using brew is explained in the next section.

- Mac OS X 10.11 (El Capitan) and newer

  In El Capitan other packages are needed. The easiest way is to install them with brew:

```
ruby -e "\$(curl -fsSL https://raw.githubusercontent.com/
    Homebrew/install/master/install)"
```

  Then:

```
brew install gcc cmake autoconf automake libtool openssl pkg-config
ln -s /usr/local/opt/openssl/include/openssl /usr/include
```

  El Capitan OS includes a System Integrity protection that prevents users to write in the /usr folder. If there are problems installing the packages because of this feature, disable it rebooting the computer pressing `CMD+R` and open a terminal to type `csrutil disable`. Then, reboot computer.

For installation of FairSoft on Linux, refer to this link depending on the user's OS https://gith- ub.com/FairRootGroup/FairSoft/blob/master/DEPENDENCIES. After installing these packages, one has to create a folder in the home directory and pull FairSoft from GitHub. It is recommended to check on their website which OS systems the new release has been tested with. It is hard to install FairSoft with an untested OS, so be sure to take the correct release for the OS being used (optional: check for new patches and tags with git tag). This can be done using the following commands:

```
cd ~

mkdir fair_install (for example)

cd fair_install

git clone https://github.com/FairRootGroup/FairSoft fairsoft.source

cd fairsoft.source

./configure.sh
```

Then, answer the questions depending on the OS:

1. Which compiler you want to use to compile the external packages?

   (i) **GCC (Linux, and older versions of Mac OSX) ←**

   (ii) Intel Compiler (Linux)

   (iii) CC (Solaris)

   (iv) Portland Compiler

   (v) **Clang (Mac OSX) ←**

   (vi) Quit

2. Do you want to compile the external packages with or without debug information or with optimization?

   (i) No Debug Info

   (ii) **Debug Info ←**

   (iii) Optimize

   (iv) Quit

3. Would you like to install FairMQ only? Choosing 'Yes' will skip building ROOT, GEANT, etc. The default option is 'No'.

   (i) Yes

   (ii) **No** ←

   (iii) FairMQ dependencies only

   (iv) Quit

4. Would you like to install Simulation engines and event generators?

   (i) **Yes** ←

   (ii) No

   (iii) Quit

5. Would you like to install the additionally available data files for the Geant4 package? To do so, you either need an internet connection (Internet) or you have to provide the files in the transport subdirectory (Directory).

   (i) Don't install

   (ii) **Internet** ←

   (iii) Directory

   (iv) Quit

6. Would you like to compile Geant4 in multithreaded mode? For that to work all detectors have to implement CloneModule.

   (i) **Yes** ←

   (ii) No

   (iii) Quit

7. Would you like to install the python bindings for ROOT and Geant4 (only if simulation engines are installed)?

   (i) **Yes** ←

   (ii) No

   (iii) Quit

   Please define a directory for the installation of the external packages. An installation in the source directory is not possible any longer. Please enter the full path of the installation directory

   path: ∼/fair_install/FairSoft_Inst (put the destination here)

8. Is /home/user/fair_install/FairSoft_Inst the correct path?

   (i) No

   (ii) **Yes** ←

   (iii) Quit

Then, the installation starts downloading and installing every package. Once this is done one has to export the SIMPATH variable, and the ROOTSYS variable (if there is another ROOT or GEANT4 version, all this can be put in a function in the user bash script to avoid for conflicts). This can be done using the following code:

```
export SIMPATH=~/fair_install/FairSoft_Inst

export ROOTSYS=~/fair_install/FairSoft_Inst

export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$ROOTSYS/lib

export PATH=$PATH:$ROOTSYS/bin
```

FairRoot:

FairRoot contains the base classes used by ATTPCROOTv2. The package is developed at GSI and allows the user to create simple detector geometries and analysis and simulation flows. First, one needs to clone the latest FairRoot repository:

```
cd ~/fair_install (Where FairSoft is installed)

git clone https://github.com/FairRootGroup/FairRoot.git

cd FairRoot

git tag     # (optional) check older versions with git reset --hard tagname

mkdir build

cd build

cmake -DCMAKE_INSTALL_PREFIX=~/fair_install/FairRoot_Inst

      -DCMAKE_CXX_COMPILER=(FairSoft compiler)

      -DCMAKE_C_COMPILER=(FairSoft compiler) ../

make -j(number of cores)

make install
```

This should install the libraries and binaries in the folder specified in the cmake command. After the installation is done, the path variable for FairRoot must be exported:

```
export FAIRROOTPATH=~/fair_install/FairRoot_Inst
```

Additional Libraries:

ATTPCROOTv2 introduces support for the Point Cloud Library [47]. Several additional packages will be needed in order to compile and run this new version. The ATTPCROOTv2 folder contains a script folder which can be used to download and build each package from the source.

- Pre-requisites for installation: boost, cmake, (ccmake for pcl recommended).
- The minimal install consists of FLANN, Eigen, PCL, and HDF5 libraries.
- The full installation (with visualization) requires also Qt4/5 and VTK (these are explained at the bottom of this page and are NOT necessary for most users).

It is recommended to follow the minimal install. Follow the instructions below for flann (don't forget to delete static libraries as explained in PCL section) and eigen.

For PLC, it is recommended to install ccmake (cmake curses interface) to disable the optional libraries. To install ccmake use `sudo apt-get install cmake-curses-gui`. Once FLANN and Eigen are installed following the instructions below, just proceed manually (as explained below) for PCL but instead of typing the cmake command from the build folder, just type `ccmake ../` (from build). A menu will appear (with several pages that can be navigated with scroll down) wherethe cmake options can be enabled or disabled.

To build PCL without QT and VTK just disable (OFF): BUILD_surface (page 2), WITH_QT and WITH_VTK (page 5). Alternatively, just use the command line proce-

dure with -D (for example -DWITH_QT=OFF) for all of them as explained below, if ccmake does not need to be installed.

Minimal Install:

FLANN - The Fast Library for Approximate Nearest Neighbors (FLANN) is a library for performing fast approximate nearest neighbor searches in high dimensional spaces. Download the tar from http://www.cs.ubc.ca/research/flann/. Unpack and compile with:

```
cd ~/path_to_flann/flann-X.X.X
mkdir build
cd build
cmake ../
make install
```

Eigen3 - Eigen is a C++ template library for linear algebra: matrices, vectors, numerical solvers, and related algorithms. Download at http://eigen.tuxfamily.org/index.php?title=Main_Page. Same procedure as Flann (except make is not needed as Eigen3 is a header library, sudo make install is needed though). IMPORTANT: PCL MUST be linked with the dynamic version of FLANN (libflann_cpp.so). The easiest way is to delete the static libraries if they are not needed (libflann_cpp_s.a). This is similar to the OpenCV implementation of FLANN.

In addition, boost libraries are needed to compile PCL. With a previous installation of ATTCROOT, just run the config.sh script on the build folder. For a fresh installation, one can run the same script on the FairRoot installation folder or install boost manually (with yum, apt-get, dnf, brew...). For the latter, just try to choose the latest version and be cautious which boost library will the ATTPCROOTv2 cmake will find (it is always better to use the same version to compile PCL and ATTCROOTv2).

PCL - The Point Cloud Library (PCL) is a standalone, large scale, open project for 2D/3D image and point cloud processing. Download source code from https://github.com/PointCloudLibrary/pcl/releases. Note: Support for visualization must be enabled here though not mandatory. Unpack and compile with:

```
tar xvfj pcl-pcl-1.X.X.tar.gz
cd pcl-pcl-1.X.X
mkdir build
cd build
cmake -DCMAKE_BUILD_TYPE=Release ../ (or ccmake ../)
make -j(number of cores)
sudo make -j(number of cores) install
```

HDF5 Library - The Hierarchical Data Format (HDF) is a set of file formats designed to store and organize large amounts of data. Download the source code from https://www.hdfgroup.org/downloads/hdf5/source-code/. Unpack and compile with:

```
cd ~/path_to_hdf5/hdf5-X.X.X
mkdir build
cd build
cmake ../
make -j(number of cores)
make install
```

Optional:

For the full install, just follow the instructions below to install VTK (http://www.vtk.org-/Wiki/VTK/Configure_and_Build) with QT (version 4.8.6 tested):

```
cd ~

mkdir qt-X.X.X-build

cd qt-X.X.X-build

wget http://download.qt-project.org/official_releases/qt/X.X/

    X.X.X/qt-everywhere-opensource-src-X.X.X.tar.gz

tar xzf qt-everywhere-opensource-src-X.X.X.tar.gz

cd qt-everywhere-opensource-src-X.X.X

./configure # go through the dialogue

make -j(number of cores)

sudo make install

cd ~/

git clone git://vtk.org/VTK.git

cd VTK

mkdir build

cmake -DQT_QMAKE_EXECUTABLE:PATH=/path/to/qt-X.X.X-build/qt-

    everywhere-opensource-src-X.X.X/bin/qmake \

      -DVTK_Group_Qt:BOOL=ON \

      -DBUILD_SHARED_LIBRARIES:BOOL=ON \

      /path/to/VTK

make -j(number of cores)

sudo make install
```

ATTPCROOTv2:

The installation of the ATTPCROOTv2 framework is rather straightforward. Currently, ATTPCROOTv2 only supports ROOT6, which should be taken care of in the installation of FairSoft:

```
cd ~/fair_install

git clone -b develop https://github.com/ATTPC/ATTPCROOTv2.git

cd ATTPCROOTv2

mkdir build

cd build

cmake ../

make -j(number of cores)
```

This should install the libraries and the headers. Then, load the environmental variables for ATTPCROOTv2:

```
source config.sh (In the build folder)
```

At each login, this command needs to be run for the environment variables to be set. It can be added to the user bash script file for convenience. After this process, the macros for simulation and analysis described in the following two chapters are ready to be compiled and executed.

A.2 ATTPCROOTv2 Simulation

A.2.1 FairRoot Simulation

ATTPCROOTv2 utilizes FairRoot for simulation of events within user defined environments. The runs are completely defined in ROOT macros and can be executed using the following code:

```
root -l macroName.C
```

where macroName ought to be the same name as the main function within the macro for convenience of execution. A basic FairRoot macro can be constructed using the C++ code outlined below.

1. Create Simulation Run

   The experimental set up is defined by calling the member functions of the FairRunSim() class. Here the user sets up the Monte Carlo engine (e.g. "TGeant4") and the final output file for the data (e.g. "./data/attpcsim_reactionName.root").

   ```
   FairRunSim* run = new FairRunSim();
   run->SetName(mcEngine);
   run->SetOutputFile(outFile);
   ```

2. Create Run-time Database

   The run-time database works as a parameter manager which handles initialization and saving to an output file (e.g. "./data/attpcpar_reactionName.root") and which contains a complete list of runs.

```
FairRuntimeDb* rtdb = run->GetRuntimeDb();
Bool_t kParameterMerged = kTRUE;
FairParRootFileIo* parOut = new FairParRootFileIo(kParameterMerged);
parOut->open(parFile.Data());
rtdb->setOutput(parOut);
rtdb->saveOutput();
rtdb->print();
```

3. Create Media

   Load the definitions for the materials involved in the simulation environment.

```
run->SetMaterials("media.geo");
```

4. Create Geometry

   In order to create the simulation environment, a "world" volume to contain all other object must first be created with the AtCave() FairModule. Then the user can add detectors created using the geometry files described in Section A.2.2.

```
FairModule* cave= new AtCave("CAVE");
cave->SetGeometryFileName("cave.geo");
run->AddModule(cave);

FairDetector* ATTPC = new AtTpc("ATTPC", kTRUE);
ATTPC->SetGeometryFileName("ATTPC_versionName.root");
run->AddModule(ATTPC);
```

5. Create PrimaryGenerator

After creating the PrimaryGenerator, generators for each event must be added to it. The beam event must be added first. It is generated by the ATTPCIonGenerator(). The reaction event is added second. This generator will be specific to the reaction involved, like a two-body reaction in the example below. The initial particle parameters, like charge, momentum, and energy, used as inputs for the generators must be defined before creating the new generator. Information on the vertex location and the momentum and energy of the beam particle at this location is carried between the generators by a global variable called the ATVertexPropagator.

```
FairPrimaryGenerator* primGen = new FairPrimaryGenerator();
ATVertexPropagator* vertex_prop = new ATVertexPropagator ();

ATTPCIonGenerator* ionGen = new
    ATTPCIonGenerator("Ion",z,a,q,m,px,py,pz,BExcEner,Bmass, NomEnergy);
ionGen->SetSpotRadius(0,-100,0);
primGen->AddGenerator(ionGen);      // Add beam event

ATTPC2Body* TwoBody = new
    ATTPC2Body("TwoBody",Zp,Ap,Qp,mult,Pxp,Pyp,Pzp,Mass,ExE,
    ResEner,ThetaMinCMS,ThetaMaxCMS);
primGen->AddGenerator(TwoBody);      // Add reaction event

run->SetGenerator(primGen);
```

6. Initialize and Start Simulation Run

   After the following code, the simulation will begin and run for $\frac{\texttt{nEvents}}{2}$ runs (since each run generates two events). The events will be stored in the ROOT file designated earlier by the outFile.

```
run->Init();
run->Run(nEvents);
```

A.2.2 Geometry

Some of the inputs to the FairRunSim needed for propagation of the particles through space by Monte Carlo engine are the geometry position of detectors and the media involved in the reaction environment. In the ATTPCROOTv2/geometry/ directory, there are macros for creating these geometries that utilize TGeoManager object and accompanying classes from the ROOT geometry package.

To construct the detector, the geometry file contains a list of volumes characterized by a number of parameters. The geometry will consist of a top volume which contains all of the other modules that make up specific detector components. To describe a module, the following requirements need to be met:

- Construct all necessary materials
- Define shapes/solids required to describe the geometry
- Construct and place volumes of the detector geometry
- Define sensitive detectors and identify detector volumes with which to associate them
- Associate magnetic field to detector regions (if needed)
- Define visualization attributes for the detector elements

The media.geo file acts a dictionary for materials that are commonly used in simulations since there are no predefined materials in FairRoot. Each medium is defined by several numerical parameters as seen below and described on the FairRoot HowTo page [46]. The following parameters are required:

- **int ncomp** - number of components in the material (ncomp= 1 for a basic material and < 1 or > 1 for a mixture. If ncomp > 0 the array wm contains the proportion by weight of each material in the mixture. If ncomp < 0 the array wm contains the proportion by number of atoms of each kind.)
- **float aw[ncomp]** - atomic weights A for the components
- **float an[ncomp]** - atomic numbers Z for the components
- **float dens** - density DENS in g/cm^{-3}
- **float wm[ncomp]** - weights WMAT of each component in a mixture (only for a mixture)
- **int sensflag** - sensitivity flag ISVOL
- **int fldflag** - fieldflag IFIELD
- **float fld** - maximum field value FIELDM in kilogauss
- **float epsil** - boundary crossing precision EPSIL
- **int npckov** - number of values used to define the optical properties of the medium. This variable is 0 for all media except some special media used for the Rich where the tracking of the Cherenkov photons is necessary. These media have additional parameters
  - **float ppckov[npckov]** - photon momentum in eV
  - **float absco[npckov]** - absorption length in case of dielectric and of absorption probabilities in case of a metal
  - **float effic[npckov]** - detection efficiency
  - **float rindex[npckov]** - refraction index for a dielectric, rindex[0]=0 for a metal

  Remark: In the present program version a mixture may contain a maximum of 5 components. If this is not sufficient one has to change MAXCOMP in hgeomedium.h.

The following parameters are normally not read. The default values are -1 and the real values are automatically calculated by Geant. If the user wants to set these values manually,

they must type the keyword AUTONULL in the media file. After this keyword all media must contain these additional 4 parameters before the Cherenkov (int npckov).

- **float madfld** - maximum angular deviation TMAXFD due to field
- **float maxstep** - maximum step permitted STEMAX
- **float maxde** - maximum fractional energy loss DEEMAX
- **float minstep** - minimum value for step STMIN

Many materials have already been defined but new definitions can be added to the file as needed, examples of which are shown in Figure 3.4.

A.2.3 Generators

Another input to the FairRunSim is provided using particle generators classes with inherited members from the FairGenerator class. These generators create single particles or multiple particles involved in a reaction, calculating the kinematics involved, given the initial conditions of the particles. Each contains the ReadEvent() function that adds tracks to the the stack in the PrimaryGenerator.

The FairPrimaryGenerator class handles the MC input of the event generation, allowing the user to set various aspects of the beam, target, and number of events. This object is only instantiated once but several generators can be registered to it depending on the reaction being simulated.

Reactions in the AT-TPC must be simulated using two events. In simulations involving solid targets, the beam prior to the reaction does not need to be considered. However in the case of a gas target like in the AT-TPC, the gas has an effect on the energy and trajectory of the beam in the moments leading up to the reaction. This means that an event must be generated for both the beam and the reaction itself so that the physics engine can propagate

the particles before and after the reaction.

A.2.4 FairRoot Analysis

Data analysis in FairRoot is managed using the FairRunAna class. The input to this class is a root file with the "cbmsim" tree containing the data arrays to be analyzed. The output from simulation will be such a root file. Files containing real data from experiment will need to be unpacked prior to analysis to create this input. The output of data analysis using this class will be a new root file containing the "cbmsim" tree with arrays of analyzed data.

The data analysis is processed using a list of user-defined tasks derived from the FairTask class. Each task processes a specific type of data that could be a branch in the input file or output from another task. Every task produces a data array that becomes a branch in the "cbmsim" tree of the final output file. To check the available data arrays, load the root file into Root by entering `root -l myRootFile.root` into the terminal and open a TBrowser using `new TBrowser` in the Root terminal, as seen in Figure A.1.

Many of the tasks will need to access parameter files in order to process the data. The parameter files are managed by the FairRuntimeDb class. Depending on the format of the parameter file, they can be opened in the FairParRootFileIo or FairParAsciiFileIo classes and added to the run-time database.

Below is the framework for an analysis macro showing the methods for setting input, output, and parameter files. Additional tasks can be added to the FairRunAna object in the order in which the data arrays need to be processed. Some tasks may have additional methods to define parameters of the analysis. Descriptions of the tasks already in the ATTPCROOTv2 framework are in the following sections.

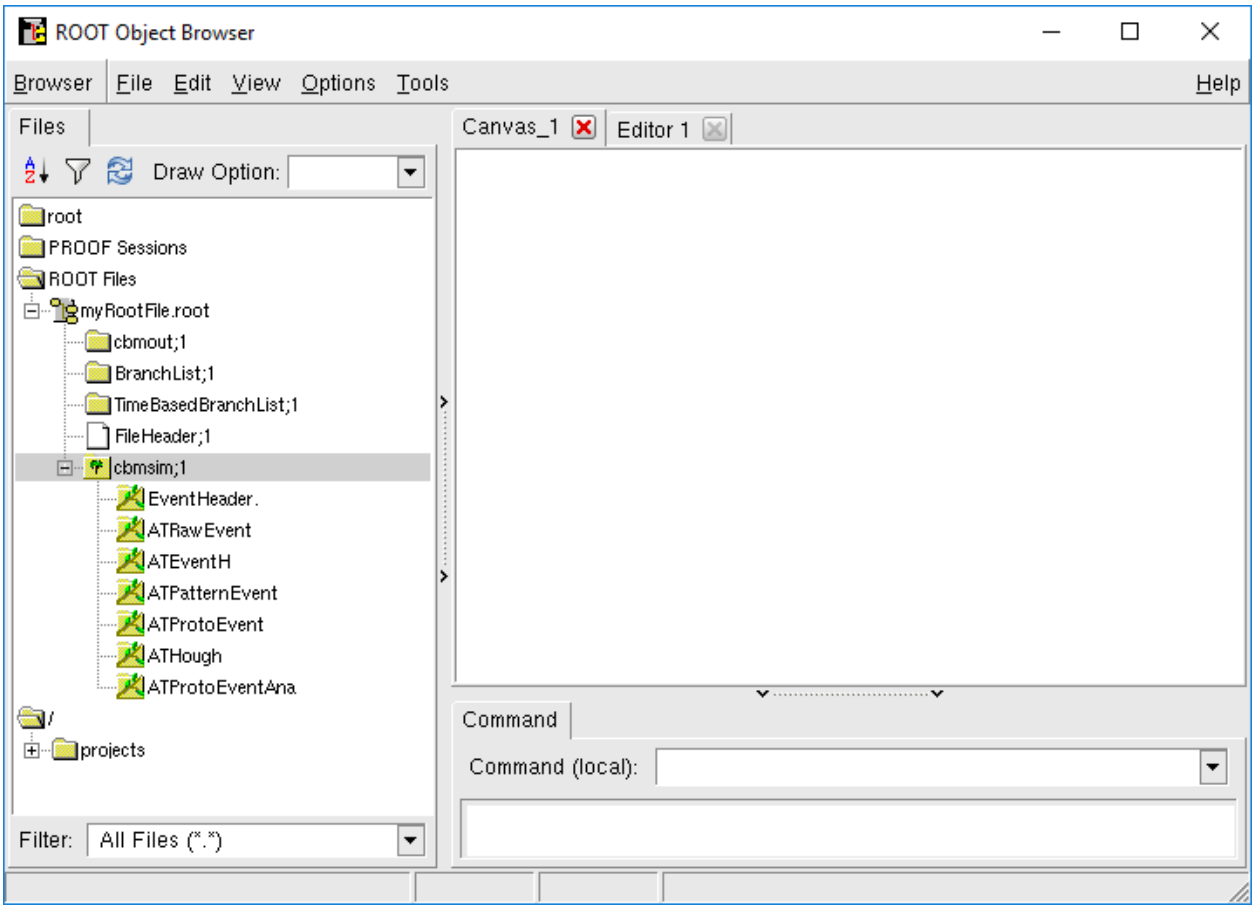


Figure A.1: View of root file loaded into TBrowser, showing "cbmsim" tree and some examples of the data arrays that may be contained within the tree.

```
// -----  Files  -----------------------------------------------

TString inputFile = "myInputFile.root",

TString outputFile = "myOutputFile.root",

TString parFile1 = "myParFile1.root",

TString parFile2 = "myParFile2.par"

// -----  Timer  -----------------------------------------------

TStopwatch timer;

timer.Start();

// ----- Run Manager -------------------------------------------

FairRunAna* fRun = new FairRunAna();

fRun -> SetInputFile(inputFile);

fRun -> SetOutputFile(outputFile);

// ----- Parameters -------------------------------------------

FairRuntimeDb* rtdb = fRun->GetRuntimeDb();

FairParRootFileIo* parIo1 = new FairParRootFileIo();

parIo1 -> open(parFile1.Data(), "in");

rtdb -> setFirstInput(parIo1);
```

```
FairParAsciiFileIo* parIo2 = new FairParAsciiFileIo();

parIo2 -> open(parFile2.Data(), "in");

rtdb -> setSecondInput(parIo2);


// -----  Tasks  -------------------------------------------------


ATMyFirstTask* myFirstTask = new ATMyFirstTask();

myFirstTask -> SetPersistence(kTRUE);


ATMySecondTask* mySecondTask = new ATMySecondTask();

mySecondTask -> SetPersistence(kTRUE);


fRun -> AddTask(myFirstTask);

fRun -> AddTask(mySecondTask);


// ----- Init & Run ----------------------------------------------


fRun -> Init();

fRun -> Run();


std::cout << std::endl << std::endl;

std::cout << "Macro finished succesfully." << std::endl << std::endl;


// -----  Finish  -------------------------------------------------
```

```
timer.Stop();
Double_t rtime = timer.RealTime();
Double_t ctime = timer.CpuTime();
cout << endl;
cout << "Real time " << rtime << " s, CPU time " << ctime << " s" << endl;
cout << endl;
```

A.2.5 Digitization

The resulting root file from simulation has an array of AtTpcPoint objects. These objects contain the information describing a particle in an event, including the position and momentum in three dimensions, the volume occupied, the energy and angle, and the mass and charge. In a real experiment, this information is not explicitly available and must be extracted by analysis. In order to reformat the data from the FairRoot simulation to the same data type as from experiment, FairRoot Analysis must be performed. This is the last step in ATTPCROOTv2 simulation and all the ATTPCROOTv2 analysis described in Section A.3 can be applied to these results in the same manner as experimental data.

In order to get experiment-like results from the simulation data, the data must undergo digitization through a few analysis tasks. After running the simulation, two files were created: the simulation file ("attpcsim_*.root") and the parameter file ("attpcpar_*.root"). The simulation file will be the input file for the FairRunAna object and the parameter file will be the first input for the run-time database.

Clusterization: The ATClusterizeTask class found in the ATTPCROOTv2/digi/ directory is a FairTask that accepts a branch of AtTpcPoint objects as input. This task

loops through every point in each event and calculates the number of electrons (with some Gaussian fluctuation dependent on the Fano factor) ionized by a particle passing through the detector's gas using the energy loss since the previous point. The distance between the current point and the previous point is divided into equal spaces such that one of the ionized electrons can be placed at a point in each one. Each electron's position is calculated with some Gaussian fluctuation using the longitudinal and transverse coefficients of diffusion found in the parameter file to account for dispersion. The drift time is determined by dividing the Z-component of each electron's position by the drift velocity found in the parameter file. The task produces a branch of ATSimulatedPoints each of which contain the X- and Y-components of an electron as well as the drift time.

This task requires adding the AT.digi.par parameter file in the ATTPCROOTv2/parameters/ directory as the second input to the run-time database. The parameter file provides the ionization energy, the drift velocity, and the coefficients of diffusion. Instantiation of this task in a FairRoot analysis macro should be done using the following code:

```
ATClusterizeTask* clusterizer = new ATClusterizeTask();
clusterizer -> SetPersistence(kFALSE);
fRun -> AddTask(clusterizer);
```

Pulse Creation: The ATPulseTask class found in the ATTPCROOTv2/digi/ directory is a FairTask that accepts a branch of ATSimulatedPoint objects as input. The task produces a branch of ATRawEvent objects each of which contain the collection of ATPad objects. An AtTpcMap object is created which can be used to map the X- and Y-components of each ATSimulatedPoint to one of 10240 pads in the pad plane. This task loops over each event, creating an ATPad for every new X- and Y-coordinate and adding hits to existing ATPads

when multiple electrons are mapped to the same pad. Each hit is registered with a time stamp corresponding to the electron's drift time. Once all ATSimulatedPoints in the event have been registered to a pad, the task loops over the pads and creates signals according to a pad response function. Multiple hits result in a superimposition of pulses. The binning parameters and gain in ATTPC.d2He.par are needed for this task.

This task requires adding the AT.digi.par parameter file in the ATTPCROOTv2/parameters/ directory as the second input to the run-time database. The parameter file provides the gain in the pads of the detector. Instantiation of this task in a FairRoot analysis macro should be done using the following code:

```
ATPulseTask* pulse = new ATPulseTask();
pulse -> SetPersistence(kTRUE);
fRun -> AddTask(pulse);
```

A.3 ATTPCROOTv2 Analysis

The data up to this point came from FairRoot simulation of reactions in the AT-TPC and digitization through FairRoot Analysis. The ATTPCROOTv2 analysis described in the following discussions can be applied to both data created in simulation and data recorded in experiment. After digitizing simulated data and unpacking experimental data, each should have the same data type: a collection of pulses in the AT-TPC pad plane.

A.3.1 Pulse Shape Analysis

The ATPSATask class found in the ATTPCROOTv2/reco/ directory is a FairTask that accepts a branch of ATRawEvent objects as input. The task produces a branch of ATEvent

objects each of which contain the collection of ATHit objects. Each ATHit object contains the X-, Y-, and Z-coordinates of a particle. An ATHit is created for each ATPad in the ATRawEvent where the X- and Y-coordinates correspond to the position of the pad in the plane. The Z-coordinate is determined by analyzing the pulse in the ATPad in one of two ways: (1) finding the maximum value of the pulse or (2) fitting the pulse and and finding peaks. This provides the time of the hit which can be used with the drift velocity to find the Z-coordinate.

This task requires adding the AT.digi.par parameter file in the ATTPCROOTv2/parameters/ directory as the second input to the run-time database. The parameter file provides the gain in the pads of the detector. Instantiating this task in a FairRoot analysis macro requires setting a few of its parameters as seen in the following code:

```
ATPSATask* psaTask = new ATPSATask();
psaTask -> SetPersistence(kTRUE);
psaTask -> SetThreshold(10);
psaTask -> SetPSAMode(1); //NB: 1 is ATTPC - 2 is pATTPC - 4 is FULL
// Use either peak finder or maximum finder but not both at the same time
psaTask -> SetMaxFinder(); // psaTask -> SetPeakFinder();
psaTask -> SetBaseCorrection(kFALSE); //Directly apply the base line
    correction to the pulse amplitude to correct for the mesh induction. If
    false the correction is just saved
psaTask -> SetTimeCorrection(kFALSE); //Interpolation around the maximum of
    the signal peak
fRun -> AddTask(psaTask);
```

A.3.2 RANSAC Tracking

In order to fit the tracks made by particles passing through the detector, a method for modeling in the presence of outliers (background, multiple tracks, etc.) is needed. The Random Sampling Consensus (RANSAC) algorithm satisfies this need, as it can extract features of a data set that fit a specified model. This model depends on the particle trajectories expected, e.g. linear (no magnetic field) or circular (with magnetic field).

The ATRansacTask class found in the ATTPCROOTv2/reco/ directory is a FairTask that accepts a branch of ATEvent objects as input. The task produces a branch of ATRansac objects each of which contain an array of ATTracks found by the algorithm and the vertex between each track (point closest to each pair). Each ATTrack object contains an array of ATHits, the parameters for their fit, and can calculate the angles of the track.

This task does not require a parameter file as input. Instantiating this task in a FairRoot analysis macro requires setting a few of its parameters as seen in the following code:

```
ATRansacTask* RansacTask = new ATRansacTask();
RansacTask->SetPersistence(kTRUE);
RansacTask->SetDistanceThreshold(10);
RansacTask->SetFullMode();
fRun -> AddTask(RansacTask);
```

A.3.3 User Analysis

In order to perform further analysis using the results of the tasks, a macro can be created which opens a branch of the final output file from FairRoot analysis. This file contains a tree

as seen in Figure A.1 with branches corresponding to the results from each task. Loading this file in a ROOT macro and opening the desired branch will allow the user to perform their own analysis using the parameters stored in whichever ATTPCROOTv2 object that branch contains.

Below is an sample of the code that can be used to analyze the results from FairRoot simulation which is done before digitization (figures in Section 3.2.4 of thesis):

```
AtTpcPoint *point = new AtTpcPoint ();
TClonesArray *pointArray = 0;
TFile *file = new TFile (mcFileName.Data (), "READ");
TTree *tree = (TTree *) file->Get ("cbmsim");
tree = (TTree *) file->Get ("cbmsim");
tree->SetBranchAddress ("AtTpcPoint", &pointArray);
Int_t nEvents = tree->GetEntriesFast ();
for (Int_t iEvent = 0; iEvent < nEvents; iEvent++) {
    tree->GetEvent (iEvent);
    Int_t Npoints = pointArray->GetEntries ();
    for (Int_t i = 0; i < Npoints; i++) {
        point = (AtTpcPoint *) pointArray->At (i);
        ...
    }
}
```

This code opens the "AtTpcPoint" branch of the "cbmsim" tree, which provides the parameters detailing the trajectory of every particle in the reaction stored in AtTpcPoint objects. This data can be used to create the "real" results of analysis that the results of analysis from

digitized data can be compared against.

To open the data produced using FairRoot analysis (i.e. digitization, pulse shape analysis, and RANSAC tracking), the following code can be used:

```
FairRunAna* fRun = new FairRunAna(); // Force dummy run
TFile* file = new TFile(anaFileName.Data(),"READ");
TTree* tree = (TTree*) file -> Get("cbmsim");
TTreeReader Reader1("cbmsim", file);
TTreeReaderValue<TClonesArray> eventArray(Reader1, "ATEventH");
Int_t nEvents = tree -> GetEntries();
for(Int_t iEvent=0;iEvent<nEvents;iEvent++){
   Reader1.Next();
   ATEvent* event = (ATEvent*) eventArray->At(0);
   if(event!=NULL){
      Int_t nHits = event->GetNumHits();
      for(Int_t iHit=0; iHit<nHits; iHit++){
            ATHit* hit = event->GetHit(iHit);
            ...
         }
     }
}
```

This code opens the "ATEventH" branch of the "cbmsim" tree, which provides the results from pulse shape analysis stored in ATEvent objects, each of which contain ATHit objects. This is a general method for opening branches of the analysis output files. Users can then access the member functions of the ATTPCROOTv2 objects to extract the parameters. For example, the X-, Y-, Z-components defined by a hit in the pad plane is given by a TVector3 object returned from the GetPosition() function of the ATHit object class.

These ATTPCROOTv2 classes are defined in files that can be found using the index of file directories in Appendix B.

Appendix B

Index of File Directories

Macros

ATTPC_d2He.C ..... geometry/

d_2He_sim_14O.C macro/Simulation/d2He/

d2He_ana_14O_07atm.C ..... macro/Simulation/d2He/Analysis_d2He/

rundigi_sim.C ..... macro/Simulation/d2He/Analysis_d2He/

d2He_ana_ransac.C ..... macro/Simulation/d2He/Analysis_d2He/

Inputs

all2_14O.dat macro/Simulation/d2He/

ATTPC.d2he.par ..... parameters/

Lookup20150611.xml ..... scripts/

energyRangeCoefs.txt ..... macro/Simulation/d2He/Analysis_d2He/

energyRangeKnots.txt ..... macro/Simulation/d2He/Analysis_d2He/

FairGenerators

ATTPCIonGenerator ..... ATGenerators/

ATTPC2Body ..... ATGenerators/

ATTPC_d2He ..... ATGenerators/

ATVertexPropagator ..... ATGenerators/

FairTasks

ATClusterizeTask ..... digi/

ATPulseTask ..... digi/

ATPSATask ..... reco/

ATRansacTask ..... reco/

ATTPCROOTv2 Data Types

AtTpcPoint ..... tpc/

ATSimulatedPoint ..... digi/

ATRawEvent ..... reco/ATDecoder/

ATEvent ..... reco/ATDecoder/

ATRansac ..... reco/ATRansac/

ATPad ..... reco/ATDecoder/

AtTpcMap ..... AtMap/

ATHit ..... reco/ATDecoder/

ATTrack ..... reco/ATDecoder/

ATd2HeAnalysis ..... reco/ATAnalysis/

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] H. A. Bethe. Supernova mechanisms. *Rev. Mod. Phys.*, 62:801, 1990.

[2] C. Iliadis. *Nuclear Physics of Stars*. Wiley-VCH, Weinheim, Germany, 2015.

[3] S. Chandrasekhar. The Maximum Mass of Ideal White Dwarfs. *Astrophys. J.*, 74:81, 1931.

[4] A. Heger, C. L. Fryer, S. E. Woosley, N. Langer, and D. H. Hartmann. How Massive Single Stars End Their Life. *Astrophys. J.*, 591(1):288, 2003.

[5] Chris L Fryer and Kimberly C. B. New. Gravitational Waves from Gravitational Collapse. *Living Rev. Relativity*, 6(2), 2003.

[6] C. Sullivan, E. O'Connor, R. G. T. Zegers, T. Grubb, and S. M. Austin. The Sensitivity of Core-collapse Supernovae to Nuclear Electron Capture. *Astrophys. J.*, 816(1):44, 2016.

[7] S. E. Woosley and T. A. Weaver. The Evolution and Explosion of Massive Stars. II. Explosive Hydrodynamics and Nucleosynthesis. *Astrophys. J., Supplement*, 101:181, 1995.

[8] C. D. Ott, E. P. O'Connor, S. Gossan, E. Abdikamalov, U. C. T. Gamma, and S. Drasco. Core-Collapse Supernovae, Neutrinos, and Gravitational Waves. *Nucl. Phys B Proceedings Supplements*, 235:381, 2013.

[9] H. Dimmelmeier, C. D. Ott, A. Marek, and H.-T. Janka. Gravitational wave burst signal from core collapse of rotating stars. *Phys. Rev. D*, 78:064056, 2008.

[10] S. Richers, C. D. Ott, E. Abdikamalov, E. O'Connor, and C. Sullivan. Equation of state effects on gravitational waves from rotating core collapse. *Phys. Rev. D*, 95:063019, 2017.

[11] R. J. Hall. "What is a Supernova?", 2006. Fact sheet created by undergraduates in the Department of Physics, University of Warwick.

[12] G. M. Fuller, W. A. Fowler, and M. J. Newman. *Astrophys. J.*, 42:447, 1980.

[13] G. M. Fuller, W. A. Fowler, and M. J. Newman. *Astrophys. J.*, 252:715, 1982.

[14] G. M. Fuller, W. A. Fowler, and M. J. Newman. *Astrophys. J.*, 48:279, 1982.

[15] G. M. Fuller, W. A. Fowler, and M. J. Newman. *Astrophys. J.*, 293:1, 1985.

[16] Georgios Perdikakis, R. Zegers, Sam Austin, D. Bazin, C. Caesar, Jenna Deaven, A Gade, D. Galaviz, Gwen Grinyer, Carlos Guess, C. Herlitzius, George Hitt, M. Howard, R. Meharchand, S Noji, Hideyuki Sakai, Yoshihiro Shimbara, E. Smith, and C. Tur. Gamow-Teller Unit Cross Sections for $(t, {}^3\mathrm{He})$ and $({}^3\mathrm{He}, t)$ Reactions. *Phys. Rev. C*, 83, 2011.

[17] T. N. Taddeucci, C. A. Goulding, T. A. Carey, R. C. Byrd, C. D. Goodman, C. Gaarde, J. Larsen, D. Horen, J. Rapaport, and E. Sugarbaker. The (p, n) reaction as a probe of beta decay strength. *Nucl. Phys. A*, 469(1):125, 1987.

[18] F. Osterfeld. Nuclear spin and isospin excitations. *Rev. Mod. Phys.*, 64:491, 1992.

[19] R. G. T. Zegers, H. Akimune, Sam M. Austin, D. Bazin, A. M.van den Berg, G. P. A. Berg, B. A. Brown, J. Brown, A. L. Cole, I. Daito, Y. Fujita, M. Fujiwara, S. Galès, M. N. Harakeh, H. Hashimoto, R. Hayami, G. W. Hitt, M. E. Howard, M. Itoh, J. Jänecke, T. Kawabata, K. Kawase, M. Kinoshita, T. Nakamura, K. Nakanishi, S. Nakayama, S. Okumura, W. A. Richter, D. A. Roberts, B. M. Sherrill, Y. Shimbara, M. Steiner, M. Uchida, H. Ueno, T. Yamagata, and M. Yosoi. The (t,^{3}He) and (^{3}He, t) reactions as probes of Gamow-Teller strength. *Phys. Rev. C*, 74:024309, 2006.

[20] R. G. T. Zegers, T. Adachi, H. Akimune, Sam M. Austin, A. M. van den Berg, B. A. Brown, Y. Fujita, M. Fujiwara, S. Galès, C. J. Guess, M. N. Harakeh, H. Hashimoto, K. Hatanaka, R. Hayami, G. W. Hitt, M. E. Howard, M. Itoh, T. Kawabata, K. Kawase, M. Kinoshita, M. Matsubara, K. Nakanishi, S. Nakayama, S. Okumura, T. Ohta, Y. Sakemi, Y. Shimbara, Y. Shimizu, C. Scholl, C. Simenel, Y. Tameshige, A. Tamii, M. Uchida, T. Yamagata, and M. Yosoi. Extraction of Weak Transition Strengths via the (^{3}He, t) Reaction at 420 MeV. *Phys. Rev. Lett.*, 99:202501, 2007.

[21] R. Madey, B. S. Flanders, B. D. Anderson, A. R. Baldwin, C. Lebo, J. W. Watson, Sam M. Austin, A. Galonsky, B. H. Wildenthal, and C. C. Foster. Gamow-Teller strength in the $^{26}\mathrm{Mg}(p, n)\,^{26}\mathrm{Al}$ reaction at 135 MeV and its fractionation into T = 0, 1, and 2 isospin channels. *Phys. Rev. C*, 35:2011, 1987.

[22] M. E. Howard, R. G. T. Zegers, Sam M. Austin, D. Bazin, B. A. Brown, A. L. Cole, B. Davids, M. Famiano, Y. Fujita, A. Gade, D. Galaviz, G. W. Hitt, M. Matos, S. D. Reitzner, C. Samanta, L. J. Schradin, Y. Shimbara, E. E. Smith, and C. Simenel. Gamow-Teller strengths in ^{24}Na using the $^{24}\mathrm{Mg}(t, {}^3\mathrm{He})$ reaction at 115A MeV. *Phys. Rev. C*, 78:047302, 2008.

[23] S. El-Kateb, K. P. Jackson, W. P. Alford, R. Abegg, R. E. Azuma, B. A. Brown, A. Celler, D. Frekers, O. Häusser, R. Helmer, R. S. Henderson, K. H. Hicks, R. Jeppesen, J. D. King, G. G. Shute, B. M. Spicer, A. Trudel, K. Raywood, M. Vetterli, and

S. Yen. Spin-isospin strength distributions for fp shell nuclei: Results for the ^{55}Mn(n,p), ^{56}Fe(n,p), and ^{58}Ni(n,p) reactions at 198 MeV. *Phys. Rev. C*, 49:3128, 1994.

[24] H. Ohnuma, K. Hatanaka, S. I. Hayakawa, M. Hosaka, T. Ichihara, S. Ishida, S. Kato, T. Niizeki, M. Ohura, H. Okamura, H. Orihara, H. Sakai, H. Shimizu, Y. Tajima, H. Toyokawa, H. Y. Yoshida, and M. Yosoi. $(d,^2\text{He})$ reactions at E_d=260 MeV as a possible probe to nuclear spin-isospin excitation. *Phys. Rev. C*, 47:648, 1993.

[25] H. M. Xu, G. K. Ajupova, A. C. Betker, C. A. Gagliardi, B. Kokenge, Y. W. Lui, and A. F. Zaruba. $(d,^2\text{He})$ reactions at E_d=125.2 MeV. *Phys. Rev. C*, 52:R1161, 1995.

[26] C. Bäumer, A. M. van den Berg, B. Davids, D. Frekers, D. De Frenne, E.-W. Grewe, P. Haefner, M. N. Harakeh, F. Hofmann, M. Hunyadi, E. Jacobs, B. C. Junk, A. Korff, K. Langanke, G. Martínez-Pinedo, A. Negret, P. von Neumann-Cosel, L. Popescu, S. Rakers, A. Richter, and H. J. Wörtche. High-resolution study of the Gamow-Teller strength distribution in ^{51}Ti measured through ^{51}V$(d,^2\text{He})^{51}$Ti. *Phys. Rev. C*, 68:031303, 2003.

[27] D. Frekers and M. Alanssari. Charge-exchange reactions and the quest for resolution. *The European Physical Journal A*, 54(10):177, 2018.

[28] M. Scott, Y. Shimbara, Sam M. Austin, D. Bazin, B. A. Brown, J. M. Deaven, Y. Fujita, C. J. Guess, S. Gupta, G. W. Hitt, D. Koeppe, R. Meharchand, M. Nagashima, G. Perdikakis, A. Prinke, M. Sasano, C. Sullivan, L. Valdez, and R. G. T. Zegers. Gamow-Teller transition strengths from ^{56}Fe extracted from the ^{56}Fe$(t,^3\text{He})$ reaction. *Phys. Rev. C*, 90:025801, 2014.

[29] M. Sasano, G. Perdikakis, R. G. T. Zegers, Sam M. Austin, D. Bazin, B. A. Brown, C. Caesar, A. L. Cole, J. M. Deaven, N. Ferrante, C. J. Guess, G. W. Hitt, R. Meharchand, F. Montes, J. Palardy, A. Prinke, L. A. Riley, H. Sakai, M. Scott, A. Stolz, L. Valdez, and K. Yako. Gamow-Teller Transition Strengths from ^{56}Ni. *Phys. Rev. Lett.*, 107:202501, 2011.

[30] M. Sasano, G. Perdikakis, R. G. T. Zegers, Sam M. Austin, D. Bazin, B. A. Brown, C. Caesar, A. L. Cole, J. M. Deaven, N. Ferrante, C. J. Guess, G. W. Hitt, M. Honma, R. Meharchand, F. Montes, J. Palardy, A. Prinke, L. A. Riley, H. Sakai, M. Scott, A. Stolz, T. Suzuki, L. Valdez, and K. Yako. Extraction of Gamow-Teller strength distributions from ^{56}Ni and ^{55}Co via the (p,n) reaction in inverse kinematics. *Phys. Rev. C*, 86:034324, 2012.

[31] Samuel Lipschutz and NSCL E10003 Collaboration. (p,n) Charge-Exchange Reactions in Inverse Kinematics. In *APS Division of Nuclear Physics Meeting Abstracts*, volume 2014, page FF.002, 2014.

[32] J. Yasuda, M. Sasano, R. G. T. Zegers, H. Baba, D. Bazin, W. Chao, M. Dozono, N. Fukuda, N. Inabe, T. Isobe, G. Jhang, D. Kameda, M. Kaneko, K. Kisamori, M. Kobayashi, N. Kobayashi, T. Kobayashi, S. Koyama, Y. Kondo, A. J. Krasznahorkay, T. Kubo, Y. Kubota, M. Kurata-Nishimura, C. S. Lee, J. W. Lee, Y. Matsuda, E. Milman, S. Michimasa, T. Motobayashi, D. Muecher, T. Murakami, T. Nakamura, N. Nakatsuka, S. Ota, H. Otsu, V. Panin, W. Powell, S. Reichert, S. Sakaguchi, H. Sakai, M. Sako, H. Sato, Y. Shimizu, M. Shikata, S. Shimoura, L. Stuhl, T. Sumikama, H. Suzuki, S. Tangwancharoen, M. Takaki, H. Takeda, T. Tako, Y. Togano, H. Tokieda, J. Tsubota, T. Uesaka, T. Wakasa, K. Yako, K. Yoneda, and J. Zenihiro. Extraction of the Landau-Migdal Parameter from the Gamow-Teller Giant Resonance in ^{132}Sn. *Phys. Rev. Lett.*, 121:132501, 2018.

[33] R. G. T. Zegers, R. Meharchand, Y. Shimbara, Sam M. Austin, D. Bazin, B. A. Brown, C. Aa. Diget, A. Gade, C. J. Guess, M. Hausmann, G. W. Hitt, M. E. Howard, M. King, D. Miller, S. Noji, A. Signoracci, K. Starosta, C. Tur, C. Vaman, P. Voss, D. Weisshaar, and J. Yurkon. ^{34}P(^{7}Li,^{7}Be$+\gamma$) Reaction at 100A MeV in Inverse Kinematics. *Phys. Rev. Lett.*, 104:212504, 2010.

[34] R. Meharchand, R. G. T. Zegers, B. A. Brown, Sam M. Austin, T. Baugher, D. Bazin, J. Deaven, A. Gade, G. F. Grinyer, C. J. Guess, M. E. Howard, H. Iwasaki, S. McDaniel, K. Meierbachtol, G. Perdikakis, J. Pereira, A. M. Prinke, A. Ratkiewicz, A. Signoracci, S. Stroberg, L. Valdez, P. Voss, K. A. Walsh, D. Weisshaar, and R. Winkler. Probing Configuration Mixing in ^{12}Be with Gamow-Teller Transition Strengths. *Phys. Rev. Lett.*, 108:122501, 2012.

[35] D. Bazin, J.A. Caggiano, B.M. Sherrill, J. Yurkon, and A. Zeller. The S800 Spectrograph. *Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms*, 204:629, 2003. 14th International Conference on Electromagnetic Isotope Separators and Techniques Related to their Applications.

[36] D. P. Stahel, R. Jahn, G. J. Wozniak, and Joseph Cerny. Charge-exchange reaction (d,^{2}He). *Phys. Rev. C*, 20:1680, 1979.

[37] H. Okamura. Three-body treatment of the (d,^{2}He) reaction on the basis of the adiabatic approximation. *Phys. Rev. C*, 60:064602, 1999.

[38] Carl Gaarde. Spin-isospin excitations. *Nucl. Phys A*, 507(1):79, 1990.

[39] M. Ericson, A. Figureau, and C. Thvenet. Pionic field and renormalization of the axial coupling constant in nuclei. *Physics Letters B*, 45(1):19, 1973.

[40] H. Hyuga, A. Arima, and K. Shimizu. Exchange magnetic moments. *Nucl. Phys A*, 336(3):363, 1980.

[41] A. Arima. History of giant resonances and quenching. *Nucl. Phys A*, 649(1):260, 1999. Giant Resonances.

[42] J. Bradt, D. Bazin, F. Abu-Nimeh, T. Ahn, Y. Ayyad, S. Beceiro Novo, L. Carpenter, M. Cortesi, M.P. Kuchera, W.G. Lynch, W. Mittig, S. Rost, N. Watwood, and J. Yurkon. Commissioning of the Active-Target Time Projection Chamber. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 875:65, 2017.

[43] D. Suzuki, M. Ford, D. Bazin, W. Mittig, W.G. Lynch, T. Ahn, S. Aune, E. Galyaev, A. Fritsch, J. Gilbert, F. Montes, A. Shore, J. Yurkon, J.J. Kolata, J. Browne, A. Howard, A.L. Roberts, and X.D. Tang. Prototype AT-TPC: Toward a new generation active target time projection chamber for radioactive beam experiments. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 691:39, 2012.

[44] ROOT Version 6.12/06, 2018. http://root.cern.ch/.

[45] FairSoft may18 Release, 2018. https://fairroot.gsi.de/.

[46] FairRoot Version 18.00, 2018. https://fairroot.gsi.de/.

[47] PCL 1.8.1, 2018. http://www.pointclouds.org/.

[48] FLANN 1.9.1, 2018. https://www.cs.ubc.ca/research/flann/.

[49] Eigen 3.3.6, 2018. eigen.tuxfamily.org/.

[50] HDF5 1.10.2, 2018. https://www.hdfgroup.org/solutions/hdf5/.

[51] Geant4 10.4.1, 2018. https://geant4.web.cern.ch/.

[52] Florian Uhlig. Fairroot tutorial, 2015. R3BRoot Development Workshop.

[53] J. L. Pack and A. V. Phelps. Drift Velocities of Slow Electrons in Helium, Neon, Argon, Hydrogen, and Nitrogen. *Phys. Rev.*, 121:798, 1961.

[54] Martin A. Fischler and Robert C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Commun. ACM*, 24(6):381, 1981.

[55] The Stopping and Range of Ions in Matter (SRIM), 2008. http://www.srim.org/.