UNIVERSITY OF NEW MEXICO

HONORS THESIS

R&D Leading Toward the Deep Underground Neutrino Experiment

Author:
Brad Philipbar

Advisor:
Prof. Michael Gold

A thesis submitted in partial satisfaction of the requirements for the degree
Bachelors of Science in Physics

in the

UNM Department of Physics and Astronomy

May 4, 2016

"The day the TPC project was approved I felt a mixture of elation and dread. I think I must have felt the way a novice mountain climber does — rope coiled on his shoulder, pitons in his knapsack — walking towards the face of El Capitan and looking up."

- Berkeley Lab physicist David Nygren describing
his feelings about the construction of
the first Time Projection Chamber

UNIVERSITY OF NEW MEXICO

Abstract

Dr. Michael Gold
UNM Department of Physics and Astronomy

Bachelors of Science in Physics

R&D Leading Toward the Deep Underground Neutrino Experiment

by Brad Philipbar

The mini-CAPTAIN prototype liquid argon time projection chamber (LAr-TPC) will be used to eventually assist the large scale Deep Underground Neutrino Experiment (DUNE). The goal is to understand the contribution from neutrons to the energy reconstruction and to increase the ability to reject backgrounds such as neutron spallation in events that mimic electron neutrino interactions. The CAPTAIN collaboration aims to measure neutrino-argon interaction properties and spallation backgrounds below 50 MeV to determine the response of future large LAr-TPCs to supernova neutrinos, and measure neutrino-argon interaction properties above 2 GeV for neutrino oscillation physics. A crucial parameter for the experiment is the electron lifetime used to verify the purity of the argon in the mini-CAPTAIN prototype detector. The results of the measurements demonstrate that the current argon purity is insufficient for the full 32 cm drift length, 200 μs.

Acknowledgements

I would like to thank Prof. Michael Gold, Prof. Sudhakar Prasad for his patience in teaching, Guy Bennett for his many insights, and the Undergraduate Committee: Prof. Mousumi Roy, Prof. Jim Thomas, and Prof. David Dunlap. I would also like to thank my Grandparents Will & Lettie Carroll, without their motivation and support none of this would have been possible. Lastly, I would like to give special acknowledgment to Bradley Knockel for his guidance and coaching.

Contents

List of Figures

Chapter 1

Introduction

1.1 Review of neutrino physics

The Cryogenic Apparatus for Precision Tests of argon Interactions with Neutrinos (CAPTAIN) physics program is to develop a large scale LAr-TPC detector technology for DUNE neutrino physics. Neutrino-Ar scattering cross sections in the energies relevant to oscillation physics are not well measured. As a simple example, oscillations between two neutrino flavors depends on their mixing angle θ, L which is the travel length, and the mass difference $\Delta \mathrm{m}_{21}{}^2 = m_2{}^2 - m_1{}^2$ given by the following formula:

$$P\left(\nu_\mu \to \nu_e\right) = \sin^2(2\theta)\sin^2\left(\frac{\Delta \mathrm{m}^2_{21} L}{4E_\nu}\right)$$

E_ν is the neutrino energy which is essential to measure on an event by event basis. A clear goal is to measure mixings and mass differences for the three flavor neutrino case.

In the 1.5-5 GeV energy window, rich and complex neutrino-nuclei interactions will take place - more than half of neutrino interaction events will occur in the baryon resonance channel. The neutrons produced in neutrino interactions will complicate energy reconstruction of incoming neutrinos resulting in missing energy that will produce uncertainty in L/E. However, the CAPTAIN collaboration aims to study the uncertainties of cross-section measurements at energies relevant for supernova neutrinos (<50 MeV) and neutrino oscillation studies above 2 GeV, neutron tagging and reconstruction relevant for long-baseline neutrinos, and cosmic spallation backgrounds for supernova neutrinos.

The neutrino energy measurement in Ar is made difficult by the production of final state neutrons, and this complication is an important systematic for the energy reconstruction essential for oscillation physics.

High energy neutrino reconstruction is primarily done with calorimetry by looking at neutrino energy. In the case of charged-current interactions, neutrino energy is the sum of the charged lepton (muon or electron) energy and hadron energy, but you need to assume all hadron energy is visible (Figure 1.1). The kinematic reconstruction of the neutrino energy can be obtained by fully reconstructing the final state. The high-energy neutrino collides with a nucleus, which is assumed to be a free a neutron at rest ignoring the rest of the nucleus. If final state particles are missed, the neutrino energy is incorrectly reconstructed. CAPTAIN hopes to improve the understanding of the colorimetric reconstruction by improving neutrino cross section models and understanding hadronic models or hadron interactions, in particular, in argon.

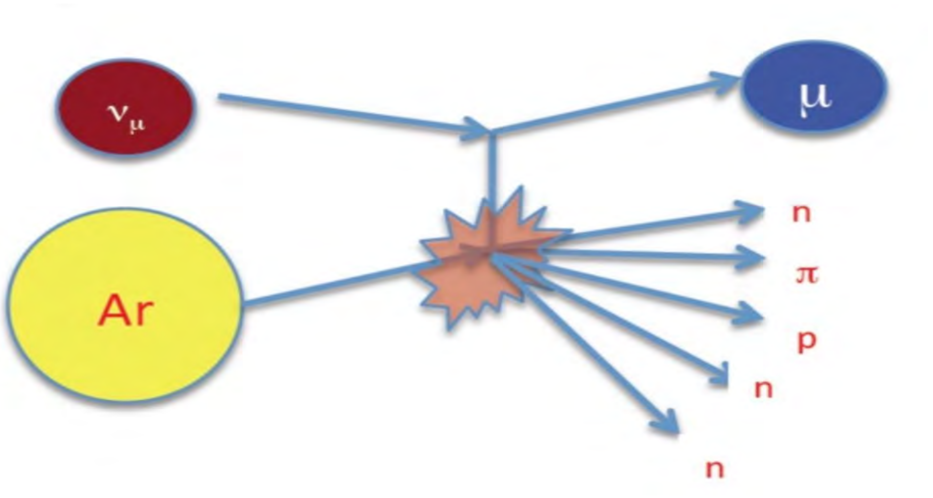


FIGURE 1.1: This is charged current, neutral current is same but no charged lepton is produced. Note the hadrons produced in the lower right under the muon.

The production and measurement of neutrons can be studied in CAPTAIN using a combination of neutron and neutrino beams. CAPTAIN began as part of a LANL Laboratory Directed Research and Development (LDRD) project and has evolved into a multi-institutional collaboration. [1] In addition to the measurement of TPC tracks, CAPTAIN will also detect the Ar scintillation light via photo-multiplier tubes (PMTs). Because the energy deposited by the charged particle gets shared between light and drifting electrons, detecting the light improves the energy resolution. Simulations show that by detecting several photons/MeV improves the projected energy resolution of the detector by (10-20)%.[2]

1.1.1 Review of Time Projection Chambers

The field of neutrino physics has an increasing demand for more advanced particle detectors. David Nygren invented the Time Projection Chamber (TPC) in 1974 at Lawrence Berkeley National Laboratory (LBNL) in California, USA.[3] In general liquid rare gases have been increasing in notoriety as detector media. There are a large number of challenges facing LAr-TPC detector technology and many can be thanked for their contributions until this point. One of which is Carlo Rubbia who adopted the use of LAr as a medium for TPCs.[4] A TPC consists of two parallel planes; the biased cathode and grounded anode, separated by the drift gap, that varies by detector, all of which is immersed in highly purified LAr ($\sim$ 10 ppb for mini-CAPTAIN).

1.1.2 LAr-TPC: working principle

As an ionizing particle traverses a LAr-TPC it creates pairs of positively charged ions Ar+ and quasi-free electrons along its path.[5] The electron and Ar+ pairs are separated by an applied E-field, and the lighter electron drifts at a constant speed (for mini-CAPTAIN 1.6 mm/μs is the drift velocity at an E-field of 500 V/cm). A fraction of the electrons later recombine with impurities. The remaining electrons drift towards electronics for readout (anode), and the argon cations tend toward the cathode. The necessary LAr purity helps to ensure only a very small number of drifting electrons are grabbed by electronegative impurities (the specific impurities addressed later). The electronics register the electrons together with their arrival times. Also, as an ionizing particle passes through the detector it creates an ionizing track producing UV scintillation light. The UV scintillation light is not emphasised in this paper.

1.1.3 CAPTAIN TPCs

With the increased interest in liquid argon time projection chambers (LAr-TPC), the CAPTAIN project was conceived as a study of the technological and systematic uncertainties associated with liquid argon detectors in event reconstruction. The CAPTAIN program consists of two-staged detectors: a primary 7,700-liter LAr-TPC, a prototype 1,700-liter LAr-TPC for configuration testing and a liquid argon scintillation-testing chamber (Figure 1.2 left). While the smaller LAr-TPC prototype detector (mini-CAPTAIN) will test various system designs, the primary (CAPTAIN) LAr-TPC will measure background neutron and muon yields and examine neutrino interactions with liquid argon.

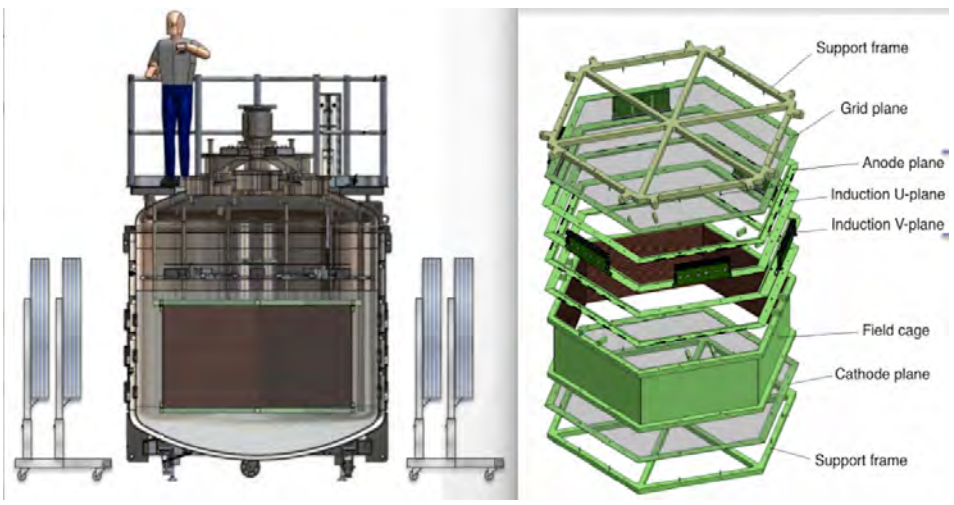


FIGURE 1.2: The full scale CAPTAIN detector (left) and exploded view of the TPC plane internal construction (right), image courtesy of Walter Sondheim (LANL, CAPTAIN)

1.2 mini-CAPTAIN overview

The design of the mini-CAPTAIN detector mimics that of the full-scale CAPTAIN detector. The difference between the two detectors is the cryostat size but the scaled geometries, and the applied electric field remain the same. Both CAPTAIN TPCs consist of a field cage of hexagonal shape with a cathode meshed plane on the lower hexagon extending upward to the U,V and X(anode) planes as seen above in (Figure 1.2 right).

1.2.1 mini-CAPTAIN: a unique wire mapping

In the volume spanned by the LAr-TPC of mini-CAPTAIN a unique wire mapping has been devised to define geometric wires connected to electronic channels. This consists of hexagonal geometry that can be described from the top-down view of the TPC planes. Three frames sit on top of the main volume. On frame 0 (U) wire numbers span from 1-332, frame 1 (V) from 333-664, and frame 2 (X) from 665 to 996. In order as defined by TPC layout each plane has 2 PC boards with 169 pads each, but 3 are obstructed. The sequential index of all wires (1-996) is ordered and defined to match the channel ordering of μBooNE boards.[6] The wire-ID or identifier for the wire used in each geometry definition (0-331 on each plane), is defined by the basis U, V and X vectors. The geometry and wiring mapping of the U, V and X planes has must be verified in the data.

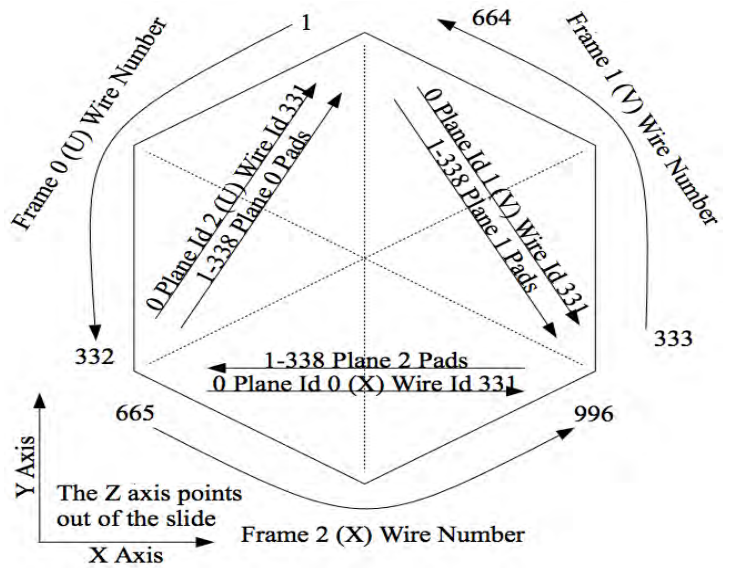


FIGURE 1.3: mini-CAPTAIN wire mapping looking down on the TPC planes. Z information is obtained from drift time

1.2.2 mini-CAPTAIN electronics

The electronic components for the TPC are based on the μBooNE experiment at Fermilab.[6] The front end motherboard (FEM) is designed with twelve custom CMOS Application Specific Integrated Circuits (ASIC). Each ASIC reads out 16 channels from the TPC. Each channel corresponds to a group of wires. There are three groups of wires, one for each plane, each going a different direction (see Figure 1.3 for wire layout). The motherboard is mounted directly on the TPC wire planes and is designed to be operated at liquid argon temperature.

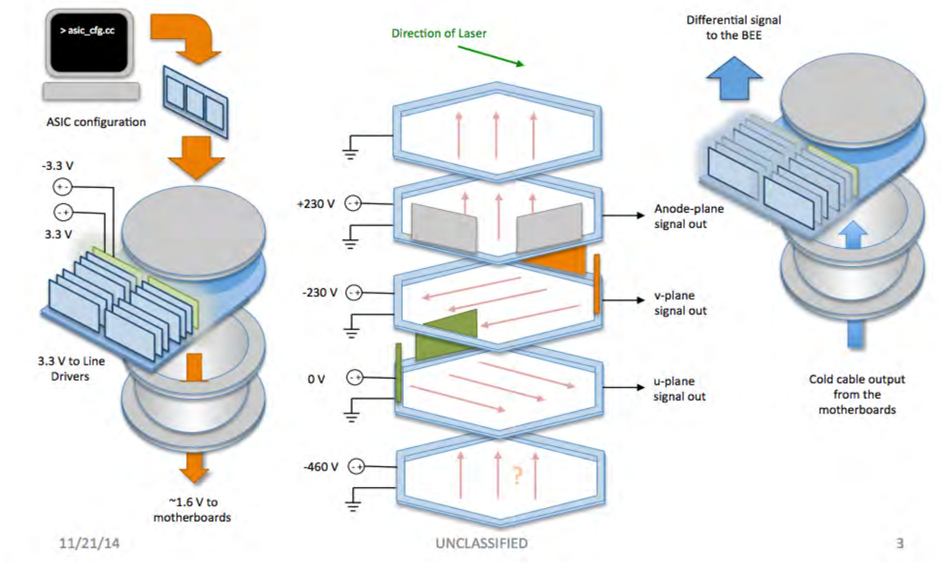


FIGURE 1.4: mini-CAPTAIN front end electronics, image courtesy of Charles Taylor (LANL, CAPTAIN).

The output signals from the motherboard are transmitted through the cold cables to the cryostat feed-thru to the intermediate amplifier board. The intermediate amplifier is designed to drive the differential signals through long cable lengths to the 64 channel receiver ADC board. The digital signal is then processed in an Field Programmable Gate Array (FPGA) on the FEM board. All signals are transmitted by fiber optic cable from a transmit module to the data acquisition computer. The Front end Electronics (FEE)(Figure 1.4) and Back End electronics (BEE)(Figure 1.5) function continuously after configuration. The data is continuously collected and dumped into the static access memory (SRAM). Without a triggering event the data is simply discarded.

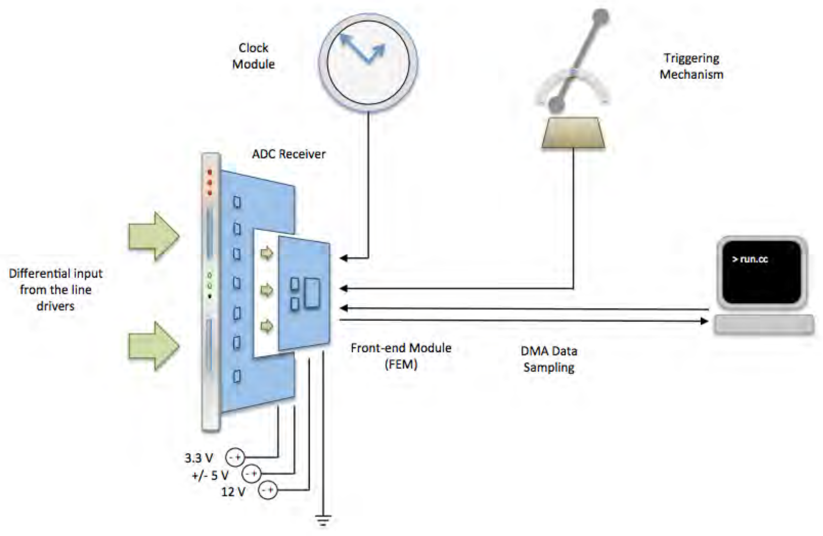
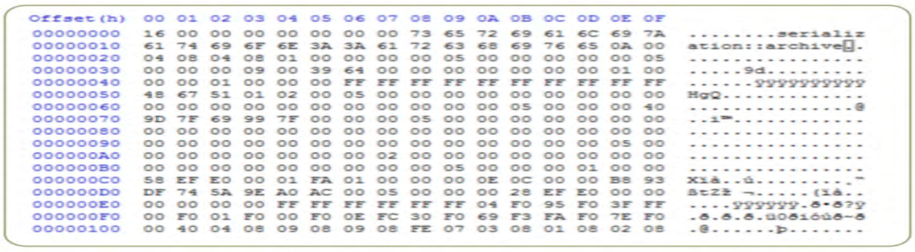


FIGURE 1.5: mini-CAPTAIN back end electronics, image courtesy of Charles Taylor (LANL, CAPTAIN).

1.2.3 Data structure

The data consists of "digits" for each of the detector elements, and represents either drift wire signals or photo-multiplier tubes (PMTs). These elements are digitized and stored by the data acquisition in a binary format. The digits consist of digitized voltages, timing and channel ID. The raw binary file is not readable except by custom internal code and requires separate channel mapping tables and calibration constants to reconstruct into data corresponding to the physical event, e.g. particle momenta and energy. A proprietary data structure is used and can be viewed in its elemental form (Figure 1.6) with a simple hex dump: hexdump –d –s <#_events - #_bytes_to_view> <filename>



```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00000000  16 00 00 00 00 00 00 00 73 65 72 69 61 6C 69 7A  ........serializ
00000010  61 74 69 6F 6E 3A 3A 61 72 63 68 69 76 65 0A 00  ation::archive▯.
00000020  04 08 04 08 01 00 00 00 00 05 00 00 00 00 00 05  ................
00000030  00 00 00 09 00 39 64 00 00 00 00 00 00 00 01 00  .....9d.........
00000040  00 00 01 00 00 00 FF FF FF FF FF FF FF FF FF FF  ......ÿÿÿÿÿÿÿÿÿÿ
00000050  48 67 51 01 02 00 05 00 00 00 00 00 00 00 00 00  HgQ.............
00000060  00 00 00 00 00 00 00 00 00 00 00 05 00 00 00 40  ...............@
00000070  9D 7F 69 99 7F 00 00 00 05 00 00 00 00 00 00 00  ..i™............
00000080  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000090  00 00 00 00 00 00 00 00 00 00 00 00 00 00 05 00  ................
000000A0  00 00 00 00 00 00 00 02 00 00 00 00 00 00 00 00  ................
000000B0  00 00 00 00 00 00 00 00 00 05 00 00 00 01 00 00  ................
000000C0  58 EF E0 00 01 FA 01 00 00 00 0E 0C 00 00 B8 93  Xïà..ú........¸"
000000D0  DF 74 5A 9E A0 AC 00 05 00 00 00 28 EF E0 00 00  ßtZž ¬.....(ïà..
000000E0  00 00 00 00 FF FF FF FF FF FF 04 F0 95 F0 3F FF  ....ÿÿÿÿÿÿ.ð•ð?ÿ
000000F0  00 F0 01 F0 00 F0 0E FC 30 F0 69 F3 FA F0 7E F0  .ð.ð.ð.ü0ðióúð~ð
00000100  00 40 04 08 09 08 09 08 FE 07 03 08 01 08 02 08  .@......þ.......
```

FIGURE 1.6: Sample portion of data viewed by using a hex dump to see the basic data structure that is not seen usually in analysis.

1.3 Event reconstruction with offline software

1.3.1 Review of event reconstruction

A general understanding of event reconstruction is needed to interpret the following sections involving the CAPTAIN software (captSoft). Event reconstruction is the process of interpreting electronic signals produced by a detector to determine the original particles that passed through the detector volume. The reconstruction produces measurements of momenta, directions, and the primary vertex(s) of physical events. To study the physical processes of neutrino interaction in liquid argon we need all of these systematics.

1.3.2 Hit finding

In brief, the hit finding is finding the electronic pulse of an expected form in wires above the background noise. The particles traverse the detector volume and their resulting ionization contacts the wires laced through the detector. This involves an understanding of signal processing applied to raw waveforms produced by the hits. The TPC signal is composed of noise from an electronic response with a pedestal offset. With an impulse response from the FEM and a mean power spectrum for the signal to noise ratio, one can take a measured signal and deconvolve it to find the signal of choice. In captSoft various noise filtering algorithms are implemented for this type of signal analysis. The process to optimize these algorithms was of great concern to this project.

1.3.3 Clusters and reconstruction

The hits are grouped into clusters that show ionization of an event of physical nature. These groups of hits are functions of time and space. The basic process of cluster finding across channels entails finding associated hits that belong to an ionization event. This event could be from a cascade of secondary particles interacting with denser matter, this is referred to as a shower. The incoming particles produce a shower that continues to cascade into multiple new particles of lesser energy. Clusters are then merged into tracks or showers depending on the timing of the event. The end goal of the intermediate clusters is to create a 3D reconstruction of the physical process.

Chapter 2

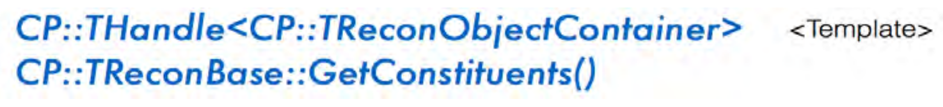
Offline Software Development

2.1 Overview of the CAPTAIN off-line software: captSoft

The CAPTAIN offline software (captSoft) is meant to do event reconstruction and offline analysis. The coding conventions are described in the software reference manual.[7]. Descriptions of captSoft packages are in Appendix A. The foundation of the software is held in a git repository which has packages managed by the Configuration Management Tool (CMT) and has been heavily used by the ATLAS, LHCb (and others at CERN), as well as several experiments around the world (e.g. Daya Bay and T2K). For captSoft, CMT handles a collection of packages and has knowledge about language, compiler usage, link editors and conditional code. When the work area is set up on a new machine CMT is automatically installed. Then each package has a manager which tells the tool what to do (a library or an application) and how to do it.

Many of CAPTAIN's collaborators do not have dedicated computing systems to analyze data, so captSoft was designed so it can be easily accessible to many collaborators using smaller remote computing systems. To assist users a shared directory was allocated at the Physics Detector Simulation Facility (PDSF) that is part of the National Energy Research Scientific Computing Cluster (NERSC). The physics computing cluster uses shares in a batch system proportional to the resources used, e.g. compute nodes or other infrastructure. PDSF has super computing power and caters to batch job processing for computationally intensive data requirements. In practical application it has proved cumbersome because of latency issues when using Secure Shell (SSH) for access. Because of this a local machine was often used. In our case the software was installed on a local machine named "Red" in the physics department at UNM.

The captSoft code uses sophisticated algorithms in packages developed for displaying simulations, signal processing, and particle tracking that hide many layers of object-oriented programming and templates reminiscent of the Generic programing paradigm. In this way, the data structure becomes an object that includes both data and functions accessed by the use of sophisticated pointers and fancy data containers(Figure 2.1).



```
CP::THandle<CP::TReconObjectContainer>    <Template>
CP::TReconBase::GetConstituents()
```

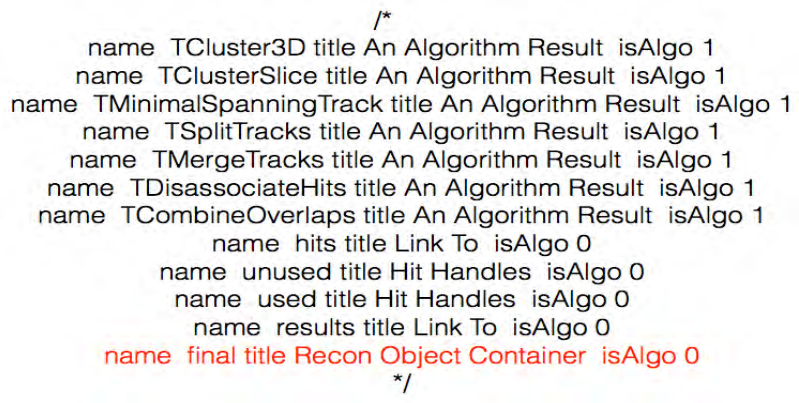
FIGURE 2.1: Pointer and container uses in captSoft.

2.2 The captSoft analysis packages

The captSoft package is not optimized for doing analysis with ROOT ".C" macros. It uses ROOT I/O, but the classes are not fully root compatible. Specific captSoft packages provide the low-level framework for handling I/O and interfacing with other databases. Another package (clusterCalib) provides the data processing and calibration of data. When run, clusterCalib applies the calibration parameters that are stored in its database to the offline data stream. Fine tuning the parameters fed to algorithms used for reconstruction is the purpose of measuring the electron lifetime, but a minimum electron lifetime is needed to do anything useful.This is because the electron lifetime limits track lengths, i.e. τv_d = track length. the e If the purity of the argon is known to be a certain value, adjustments can be made to increase sensitivity of peak finding algorithms used to identify hits. These systematics increase the spatial resolution of the detector and the accuracy of finding rare event interactions.

2.3 captAna

In lieu of the learning curve associated with captSoft, captAna was created. All of its code was developed by Dr. Michael Gold at the University of New Mexico. The use of captAna will get new users analyzing data quickly. The primary working function of captAna is to fill the containers of captRoot. The captSoft code is an analysis macro with the following hierarchy: captSoft->captAna (linked with captRoot, which I developed to give captSoft full access to ROOT classes). The captAna code allows simple access to the captSoft data structure and algorithms without needing to understand the complicated pointers of captSoft. When captAna is executed on an output file from the clusterCalib command line executable it generates a ".root" (ROOT file format) file containing branches of data for quick analysis. It should be noted captAna uses CMT packages as well. The objects are put in $\sim$/fits/TCaptainRecon, which are shown in Figure 2.2. The objects in TCaptainRecon are the reconstruction algorithms inherited by captAna from captSoft.



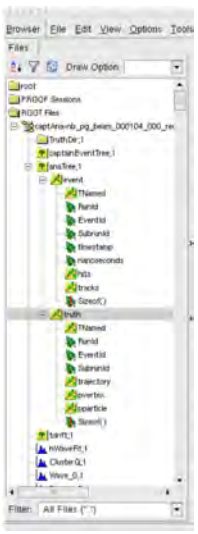
```
/*
name  TCluster3D title An Algorithm Result  isAlgo 1
name  TClusterSlice title An Algorithm Result  isAlgo 1
name  TMinimalSpanningTrack title An Algorithm Result  isAlgo 1
name  TSplitTracks title An Algorithm Result  isAlgo 1
name  TMergeTracks title An Algorithm Result  isAlgo 1
name  TDisassociateHits title An Algorithm Result  isAlgo 1
name  TCombineOverlaps title An Algorithm Result  isAlgo 1
name  hits title Link To  isAlgo 0
name  unused title Hit Handles  isAlgo 0
name  used title Hit Handles  isAlgo 0
name  results title Link To  isAlgo 0
name  final title Recon Object Container  isAlgo 0
*/
```

FIGURE 2.2: TCaptainRecon: Using name="final" title = "Recon Object Container" produces a list of objects.

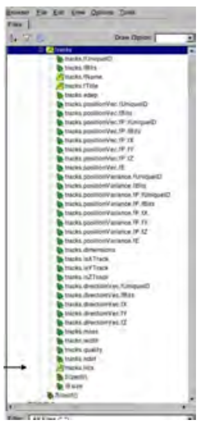
Truth information corresponds to the generated events vs. simulated data. The

2.4 captRoot

Thus far objects for experimental and generated data have been put into separate library, captRoot. The captRoot portion functions like a macro, I have used this macro extensively (and others in captSoft) to analyze tracks that I have found using captSOFT's event-display package. There is a ROOT file structure with the tracks' hits, each track has its own full list of hits accessible from a ROOT Tree (Figure 2.3a & 2.3b). Similar ROOT Trees have the signal waveforms per wire, clusters and artificial PMT information. All reconstructed data is stored in ROOT classes and a ROOT loadable class library (Figure 2.4).

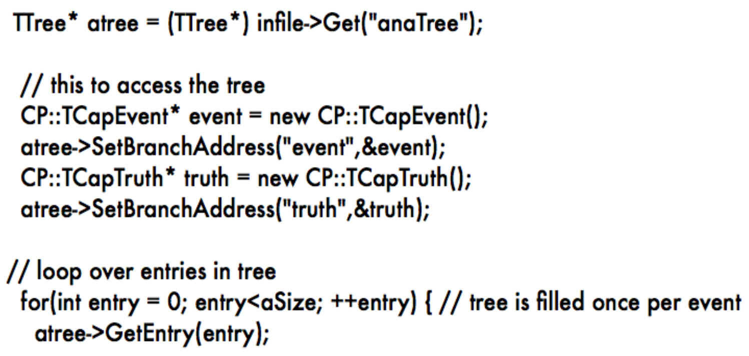


(A) ROOT trees from captAna.


Browser File Edit View Options Tools
Files
Draw Option
tracks
tracks.fUniqueID
tracks.fBits
tracks.fName
tracks.fTitle
tracks.edep
tracks.positionVec.fUniqueID
tracks.positionVec.fBits
tracks.positionVec.fP.fUniqueID
tracks.positionVec.fP.fBits
tracks.positionVec.fP.fX
tracks.positionVec.fP.fY
tracks.positionVec.fP.fZ
tracks.positionVec.fE
tracks.positionVariance.fUniqueID
tracks.positionVariance.fBits
tracks.positionVariance.fP.fUniqueID
tracks.positionVariance.fP.fBits
tracks.positionVariance.fP.fX
tracks.positionVariance.fP.fY
tracks.positionVariance.fP.fZ
tracks.positionVariance.fE
tracks.dimensions
tracks.isXTrack
tracks.isYTrack
tracks.isZTrack
tracks.directionVec.fUniqueID
tracks.directionVec.fBits
tracks.directionVec.fX
tracks.directionVec.fY
tracks.directionVec.fZ
tracks.mass
tracks.width
tracks.quality
tracks.ndof
tracks.hits

(B) ROOT tree structure of tracks.

FIGURE 2.3: ROOT file structure from captAna using captRoot



```
TTree* atree = (TTree*) infile->Get("anaTree");

 // this to access the tree
 CP::TCapEvent* event = new CP::TCapEvent();
 atree->SetBranchAddress("event",&event);
 CP::TCapTruth* truth = new CP::TCapTruth();
 atree->SetBranchAddress("truth",&truth);

// loop over entries in tree
 for(int entry = 0; entry<aSize; ++entry) { // tree is filled once per event
   atree->GetEntry(entry);
```

FIGURE 2.4: Example to read captAna tree data.

Chapter 3

Liquid Argon Purity Measurements

3.1 UV Laser

A Quantel quadrupled Nd:YAG laser with a wavelength of 266 nm (4.66 eV) was pointed into the mini-CAPTAIN cryostat. There are 10 pulses every second, each lasting 5 ns led by a periscope from the top of mini-CAPTAIN into the inner cryostat by angled mirrors that were aligned before closing the detector (Figure 3.1). [8] The laser has an energy per pulse of 0.1 joule. [8] The laser entry ports are on the side of the inner mini-CAPTAIN TPC (Figure 3.2). where D(x) is axis diameter after passing a distance of x, D_0 is the initial axis diameter, and θ is the laser divergence. The axis diameter as a function of passing distance (x) can be found by:

$$D(x) = D_o + 2x \tan\theta \tag{3.1}$$

using a divergence of 0.169 mrad. Knowing the divergence of the laser path it was approximated a 5 m path length was needed to traverse the inside and outside (exit and entry) of the cryostat corresponding to an axis diameter increase of approximately 1 cm.

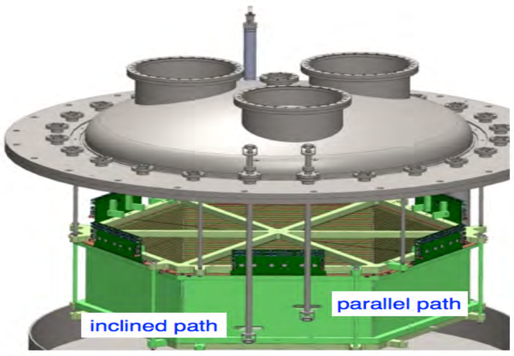


FIGURE 3.1: mini-CAPTAIN angled mirror assembly.

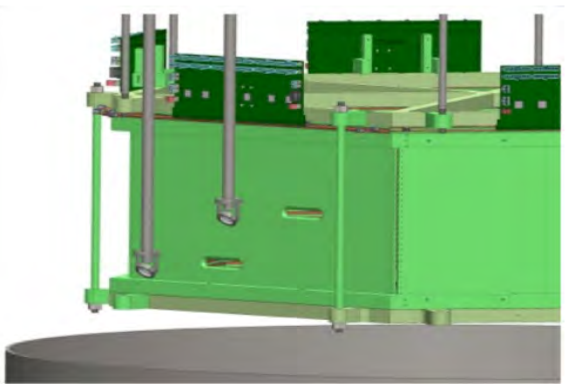


FIGURE 3.2: mini-CAPTAIN TPC laser entry locations.

From this data a preliminary calculation showed that the slits should be 18 cm long with 5 cm between them. Once the final design parameters were established the horizontal distance from the periscope to the panel was confirmed to be 70 cm from the mirror to anode plane. The first entry path is parallel to the plane to measure the energy loss as the laser propagates. The second angled entry path was designed such that the path would be inclined in the upward direction for time separation, so the drift time from the two laser paths will be different. From the artificially produced UV laser ionization tracks, an electron lifetime can be determined. The electron lifetime is critically dependent on the O_2 and H_2O concentrations in the liquid argon. The electron lifetime of a minimum ionizing particle (MIP) yields 18,000 electrons (29fC) every 3 mm, which is the wire spacing in mini-CAPTAIN. So the base pre-amplifier setting of 12 fC corresponds to 100 ADC counts implying the electron drift speed is 1.6 mm/μs so signal smearing would be more than 1 μs or 2 DAQ bins of 500 ns/bin. From this analysis it is well understood that a high electron survival rate is critical to distinguish signal from noise and should be at least 3 ADC counts per bin. The electron lifetime is used to determine the drift distance of electrons, for mini-CAPTAIN the required drift distance is 32 cm corresponding to 1.5 ppb O_2 and 0.3 ppb H_2O. A 32 cm drift distance corresponds to a 200 μs electron lifetime. As can be seen H_2O is not as much a concern as the O_2.

3.1.1 Laser Ionization Track Observation

The approach applied here is based on principles commonly used for LAr purity monitors and the reference table from the Liquid Argon Purity Demonstrator (LAPD) at Fermilab to associate a relationship between electron lifetime and oxygen equivalent contamination [9].

By August of 2015 the cryostat electronics were fully immersed in highly purified argon and the UV laser testing began. During this time period the overall purity of the liquid argon read by the purity analyzers was shown to to 5 ppb (Figure 3.3). From the table from LAPD in Appendix B the purity level can be related to an electron lifetime. A laser track would be the first confirmation of the purity of argon in the detector and that the detector is in its functional range.[9] During this time, the first UV laser track was confirmed by LANL using alternative software, my measurement here was in agreement. The difference between the two measurements is that my measurement used the captSoft software to find the track and captAna to measure the electron lifetime.

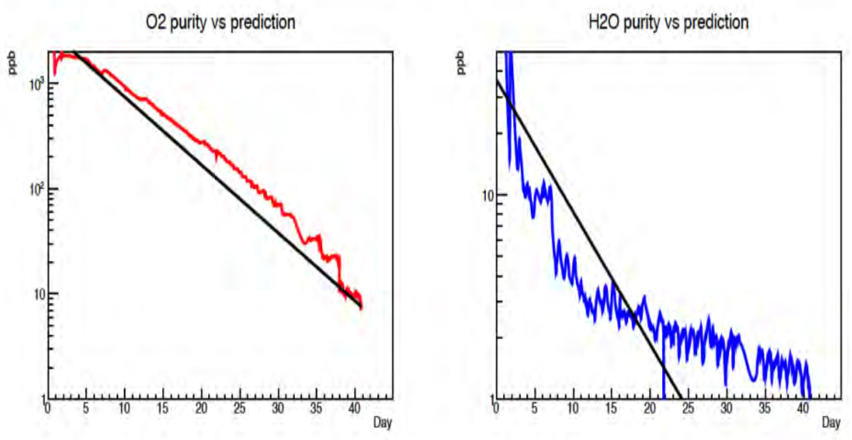


FIGURE 3.3: Purity vs. time O_2 and H_2O predictions from the purity monitors at LANL, for mini-CAPTAIN. These values correspond to less than 2 ppb of O_2. As filtering and distilling of the argon continues the impurities go down.

The UV laser system functions as a controllable tool to measure and calibrate the electron lifetime. By shining a UV laser pulse into the TPC argon volume, an amount of charge Q_o is produced by laser induced ionization. The charge produced is in the form of an electron cloud that is drifted upward to the wire planes. During the drift upward electrons in the cloud attach to electronegative impurities. The charge amount changes as the drifting electrons attach to these impurities. This process of attachment can be described by a characteristic time constant τ (eq. 3.2). This constant is referred to as the electron lifetime in TPC experiments. The electron lifetime varies as the inverse product of the electron attachment rate constant k_s and n_s the molar solute impurity concentration in LAr) (eq. 3.3).

$$Q(t) = Q_0 \cdot e^{-t/\tau}. \tag{3.2}$$

$$\tau = (k_s \cdot n_s)^{-1}. \tag{3.3}$$

The attachment rates of O_2, H_2, and N_2 add as in (eq. 3.4), but O_2 is our concern as the other impurities contribute negligibly according to [9].

$$\frac{1}{\tau_e} = k_e^{(O_2)}\,[O_2] + k_e^{(N_2)}\,[N_2] + k_e^{(H_2O)}\,[H_2O] \tag{3.4}$$

The attachment rate constant depends on electric field, for mini-CAPTAIN the electric field is known to be 500 V/cm, and the attachment rate constant is referenced from an LBNL LAr-TPC conceptual design report. [9]

In September of 2015 a laser track was observed in run 4547 event 250 using the captSoft software. At this time a laser track had already been seen but a comparison track was not yet confirmed. The laser track was found using the captSoft event-display executable (Figure 3.4). The application of captSoft to find a track had not yet been done, this was a first for the CAPTAIN collaboration and a mile stone in captSoft development.

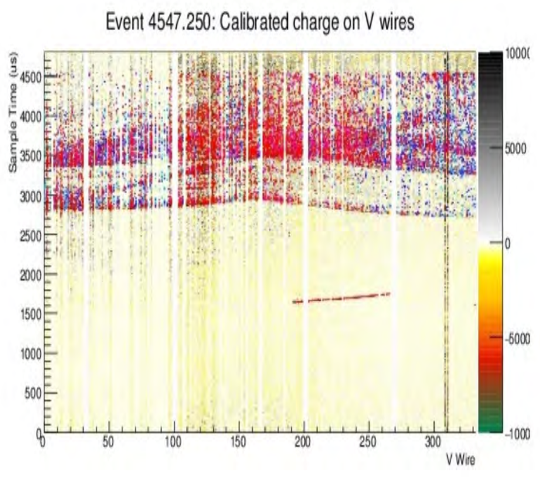
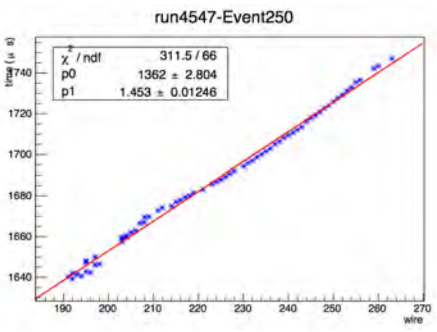
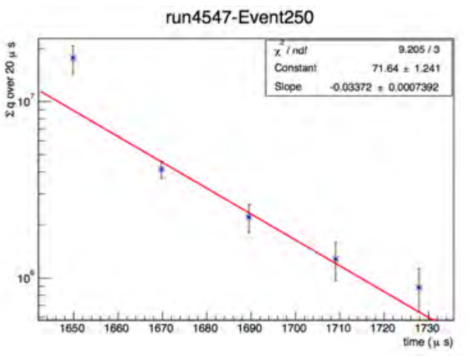


FIGURE 3.4: Laser track in run 4547 event 250, the upper half region is believed to contain acoustic noise from the laser.

The laser track from run 4547 event 250 was analyzed using an analysis macro, charge1.C that inherits objects and classes from captSoft, captAna and captRoot (see appendix B for the charge1.C code). The electron lifetime for run 4547 event 250 was found to be τ = 29.7 +/- 0.6 μs (Figure 3.5a & 3.5b). During this time period the statistics needed to calculate the charge error were not fully accounted for in the code. The charge error for this measurement was set to the square root of the number of charge hits, which is a reasonable approximation. Later the charge error was corrected, it was set as the square root of the charge plus discrete components, see appendix B for how the statistics of all wires were taken into account. The charge errors in the code used to make these plots was written by Prof. Gold. I suspect they are not correct as the errors have not been confirmed by fellow collaborators looking at the same data. The code calculates a time interval from a start and stop time, finds the number of bins, does summing over the bins, then finally calculates the RMS for the summed charge. Lastly, systematic errors for laser measurements are neglected, but are expected to be small.



(A) fit of laser track run 4547 event 250.



(B) electron lifetime measurement of run 4547 event 250 with charge error.

FIGURE 3.5: electron lifetime of run 4547 event 250: τ = 29.7 +/- 0.6 μs

The electron lifetime can be related to the O_2 concentration in ppb from [10] by the relationship given in (eq. 3.5)

$$\tau = \frac{500 \cdot \mu s}{O_2 \cdot ppb}. \tag{3.5}$$

Accordingly, the 30 μs electron lifetime measured from run 4547 event 250 corresponds to approximately 16.6 ppb O_2.

3.2 Motivation for Electron Lifetime Measurements

My goal for this thesis was to reconstruct neutron events. The laser track measurement previously described was needed during preparation for neutron beam runs in February of 2016. During this time the detector was emptied and moved into the path of the neutron beam line at the Los Alamos Neutron Science Center (LANSCE). After the move, and during the argon re-filling of the detector, purity problems were detected by the O_2 monitor. The neutron beam runs were carried out in late February with the O_2 level not well known. Using another known background of cosmic ray events I wanted to calibrate our, potentially, noisy data so as to extract the signal from the neutron beam data.

3.3 Electron Lifetime Measurements

Through-going cosmic-muon tracks are used to determine electron lifetime in liquid argon against electro-negative impurities. Specifically the ionization of cosmic ray muons as a function of drift distance is a key parameter needed to analyze detector performance. A cosmic ray muon track was found in run 6388 event 45 (Figure 5.1) on April 16, 2016. The track was fitted to find the respective electron lifetime using the charge1.C code in appendix B. The charge1.C code slices the hit area proportional to the charge as a function of drift time for a selected cosmic-muon track. The selected track is isolated with a bounding time and wire intervals.

The calculation of the charge error was a problem while making the electron lifetime measurement. The fundamental uncertainty is in the number of electrons that obey a Poisson distribution. Next in the code, the uncertainty for the sample to sample RMS is added, which are fluctuations introduced by Gaussian fluctuations of electrons (including short noise, a type of electronic noise). It follows that the channels that have baseline subtraction, will have a positive baseline sigma (this only affects the induction planes). In this case we add a correlated uncertainty for the "wandering of the baseline". This is analogous to the deviation of the track from a "best-fit" straight line. The RMS is correlated between all the channels, but is uncorrelated with the other uncertainties (number of electrons, and electronics noise). The uncertainty for the "baseline wander" is approximated as the sample sigma. This should be calculated separately, but since the wander is from the same physics as the sample sigma, it should be a good approximation. Finally we take the square root of the variance to get the uncertainty.

Also there is smearing from electronic noise, track length and variations from the cosmic ray energy spectrum that contribute to systematic errors for each wire. A statistical error from the peak determination of the summed charge per hit will contribute to the average charge from each time drift bin.[11] The systematic uncertainty from these contributing parameters has been neglected in our measurements of electron lifetime. They are expected to be approximately 0.3% (each wire affected by ~0.8% from systematic error) as seen in previous experimental results.[11]

Multiple measurements have been performed of a large number of cosmic-muon tracks using the charge1.C code (appendix B) to find electron lifetime. Only those with a reasonable chai-square statistic are used for analysis. A comparison of this data for electrons lifetimes from an earlier laser track measurement to after the detector was refilled is presented in the concluding section of this paper. Figure 3.7a shows a single measurement corresponding to an effective electron lifetime for run 6388 event 45: τ = 42.84 +/- 2.96 μs. Using equation 3.5 the argon O_2 concentration is estimated to be approximately 10 ppb. The additional plots used to obtain the electron lifetimes for the time evolved analysis are located in appendix D.

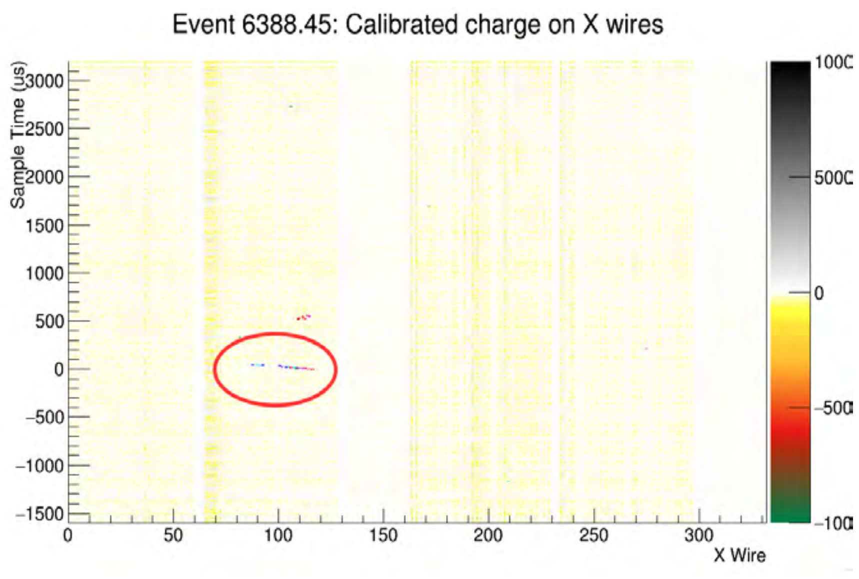
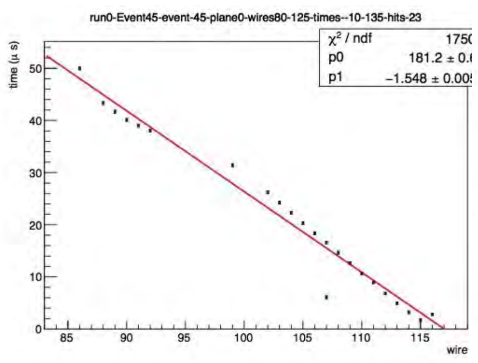
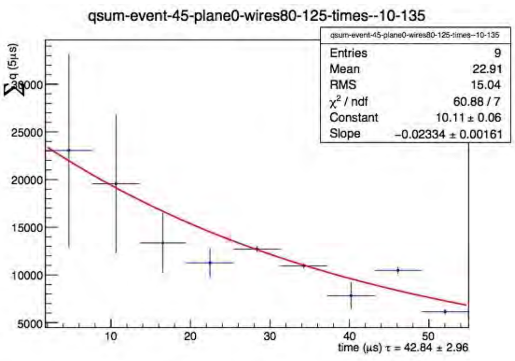


FIGURE 3.6: run 6388 event 45.



(A) charge fit of run 6388 event 45.



(B) electron lifetime measurement of run 6388 event 45, sum charge vs. time

FIGURE 3.7: 6388 event 45: τ = 42.84 +/- 2.96 μs.

Chapter 4

Longitudinal Electron Diffusion in Liquid Argon

4.1 A Comment on the Affect of Longitudinal Electron Diffusion in Liquid Argon

I wish to discuss solutions of the Fick's equation. Fick's equation describes the density profile of a cloud of electrons diffusing through a region in which there is an electric field as Figure 4.1 depicts.

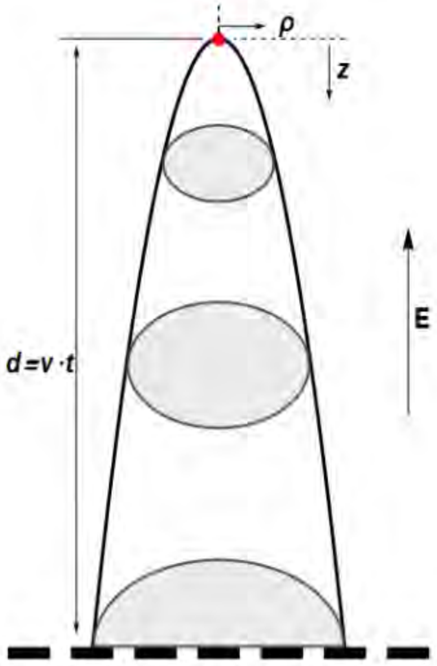


FIGURE 4.1: Diffusion process starting from a point source to the detection. Longitudinal drift expected to be smaller than that of the transverse direction. [11]

Fick's equation accounts for two effects: the growth in volume of the cloud and the drift velocity of the centroid of the cloud.

$$\frac{\partial n}{\partial t} = D_L \frac{\partial^2 n}{\partial z^2} + D_T \left(\frac{\partial^2 n}{\partial x^2} + \frac{\partial^2 n}{\partial y^2} \right) - v \frac{\partial n}{\partial z} - \lambda \text{vn} \tag{4.1}$$

I suspect, as this is a diffusive process, that with suitable manipulations the equation should take the form of a standard diffusion equation. This strategy is appealing as we already know the solution to the standard diffusion equation. To that end we redefine n in the following way:

$$n = \bar{n}e^{-\lambda \mathrm{vt}} \tag{4.2}$$

By taking a time derivative of n we now see that we have removed the constant term and have an equation for $\bar{n}$. The other obstructive term is the term linear in the velocity of the centroid. Examining the first derivative terms, we see that if we recast our solution into the following form:

$$\bar{n}(x, y, z - \mathrm{vt}, t) \tag{4.3}$$

we now have a standard diffusion equation.

$$\frac{\partial \bar{n}}{\partial t} = D_L \frac{\partial^2 \bar{n}}{\partial (z - \mathrm{vt})^2} + D_T \left(\frac{\partial^2 \bar{n}}{\partial x^2} + \frac{\partial^2 \bar{n}}{\partial y^2} \right) \tag{4.4}$$

We can rescale each of these coordinates to make the equation homogeneous:

$$\overline{x},\ \ \overline{y},\ \ \overline{z - vt} \ = \ \frac{x}{\sqrt{D_T}},\ \frac{y}{\sqrt{D_T}},\ \frac{z - vt}{\sqrt{D_L}} \tag{4.5}$$

And write the explicit Green's function solution of the diffusion equation in 3D. [12]

$$\bar{n} = \frac{1}{\left(\sqrt{4\pi t}\right)^3} e^{-\left[\frac{\left(\overline{z - \mathrm{vt}}\right)^2}{4t} + \frac{\overline{x}^2}{4t} + \frac{\overline{y}^2}{4t}\right]} \tag{4.6}$$

All that remains is now to rewrite this solution in terms of x, y, z, and t. This will give us the solution to Fick's equation.

$$n = \frac{1}{\left(\sqrt{4\pi t}\right)^3} e^{-\left[\frac{(z - \mathrm{vt})^2}{4tD_L} + \frac{x^2}{4tD_T} + \frac{y^2}{4tD_T}\right] - \lambda \mathrm{vt}} \tag{4.7}$$

I now want to analyze how this affects mini-CAPTAIN. Assuming the argon purity is capable of producing an electron lifetime of 200 μs, I can calculate the effects of longitudinal diffusion.

The terms in Fick's equation correspond to the following:

$$\frac{\partial n}{\partial t} = (\text{diffusion}) + (\text{drift}) + (\text{absorption}) \tag{4.8}$$

Considering only the drift term, we have:

$$\frac{\partial n}{\partial t} = -v \frac{\partial n}{\partial z} \Rightarrow \frac{\partial z}{\partial t} = -v \tag{4.9}$$

Where v is the centroid velocity. That is, the velocity of the electron cloud if no diffusion existed. From [9], v = 1.6 mm/μs. The time of arrival of the centroid is $t_c = \frac{d}{v}$ shown in Figure 4.2 as a function of distance of wire grid detector, d.

Just looking at the absorption term:

$$\frac{\partial n}{\partial t} = -\lambda \mathrm{vn} \Rightarrow n = e^{-\lambda \mathrm{vt}} = e^{-t/\tau} \Rightarrow \lambda = \frac{1}{\tau \mathrm{v}} \tag{4.10}$$

I have found λ = 0.0031 1/mm using what we hope to target 200 μs electron lifetime, τ.

With this λ, we can calculate the time the maximum charge reaches the wire grid, which will be called the peak time, t_p. This time is shorter than t_c because the back of the cloud is being absorbed for a longer time and arrives weaker (eq. E.11, from reference: [11]).

$$t_p = \frac{-D_L + \sqrt{{D_L}^2 + d^2 v\,(v + 4D_L\lambda)}}{v\,(v + 4D_L\lambda)} \tag{4.11}$$

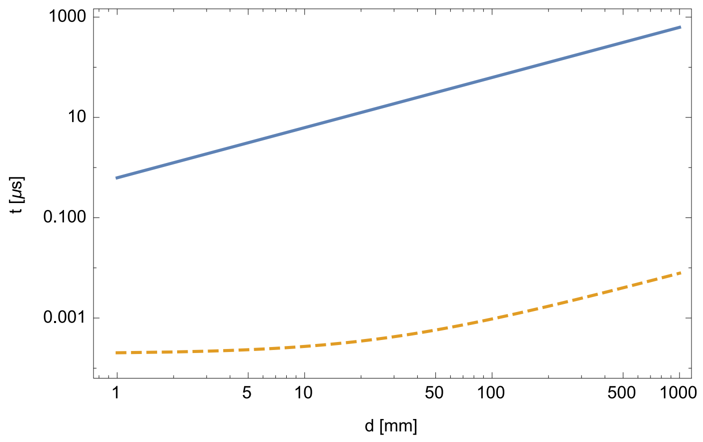


FIGURE 4.2: Solid line is drift time. The dashed line is the difference between the centroid and the peak time.

Using $D_L = 5 \times 10^{-4}$ mm$^2/\mu$s from [9] I get Figure 4.2 which shows $t_c - t_p$. If precision information is required, the longitudinal electron diffusion coefficient is non-negligible (see 4.2). Note constants are referenced from the document from LBNE.

Analysis of the effects of longitudinal electron diffusion have been considered for mini-CAPTAIN, the affects are found to be negligible. However, the longitudinal electron diffusion in the DUNE experiments 3.6 m drift will degrade measurements according to reference [11].

Chapter 5

Results and Conclusions

The software developed has been applied to determine the deposited charge by through-going muons to obtain an estimate of the electron lifetime. When these results are compared to the UV laser results, a trend has been observed corresponding to an increase in the argon purity needed for the mini-CAPTAIN apparatus to eventually reach the full 32 cm drift length equivalent to 200 μs (5.1).

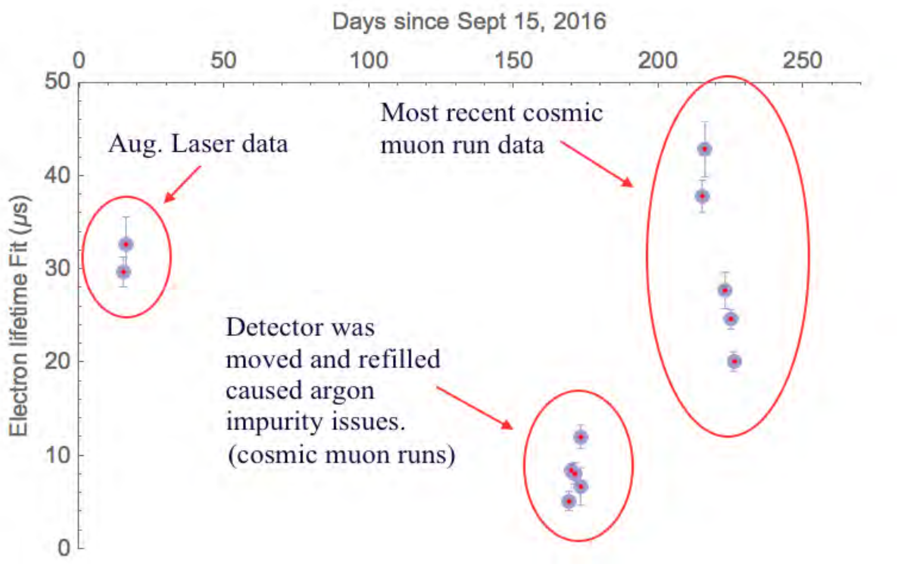


FIGURE 5.1: Electron lifetime vs. date.

Finally, a comment on the analysis of the effects of longitudinal electron diffusion has been made for mini-CAPTAIN, the affects are found to be negligible. However, the longitudinal electron diffusion in the DUNE experiments 3.6 m drift will degrade measurements according to reference [11].

Appendix A

captSoft Packages

captEvent – The main I/O library for CAPTAIN. This defines the event format and is required by any package that expects to read or write events. This library is intended to remain small and stable.

captControl – The main job control and configuration library for CAPTAIN. This provides a set of bash functions to make running CAPTAIN jobs a cinch. It is useful for everyday CAPTAIN work, but is aimed at batch processing. Functions are provided for standard configuration options, and most operations can be done by calling a single function.

captTrans – This library provides access to the raw data structures written out by the DAQ. It is used to translate the DAQ data into the offline format.

testBase – A package of tools for testing software. This contains the tut (Testing Using Templates) tools, and the nd280-validate script used to run the package validation scripts found in the validate.d directory.

The Event Display – The event display is closely tied to the event format, and is intended to display all objects that can be stored in the captEvent data format.

captSummary – A package to summarize the captEvent file into a smaller DST (data summary tree) for physics analysis. This package generates files that are (best) analyzed using the captDST package.

The simulation packages provide the simulation of the detector. These are the programs that will probably be of most interest to the first-time user. The detector simulation. The particle transport is done using geant 4 and it has a (relatively) flexible geometry that can be changed at run time. This only simulates the energy as it is deposited in the detector. It does not simulate the electronics response which is done in elecSim.

elecSim – This handles the response, digitization and electronics simulation. It contains several classes that simulate different types of electronics. This takes care turning raw MC "hits" into digitized electronics output which represent uncalibrated* detector hits.

The database packages provide various interface to databases.

captDBI – Defines the data base interface routines. This derived from the MINOS DBI with a few additions made while it was used at T2K (many of the authors of the original minosDBI library are also T2K authors). This is used to access the calibration constant database.

Handle channel and geometry information. – Provides methods to translate between electronics channels and detector geometry objects.

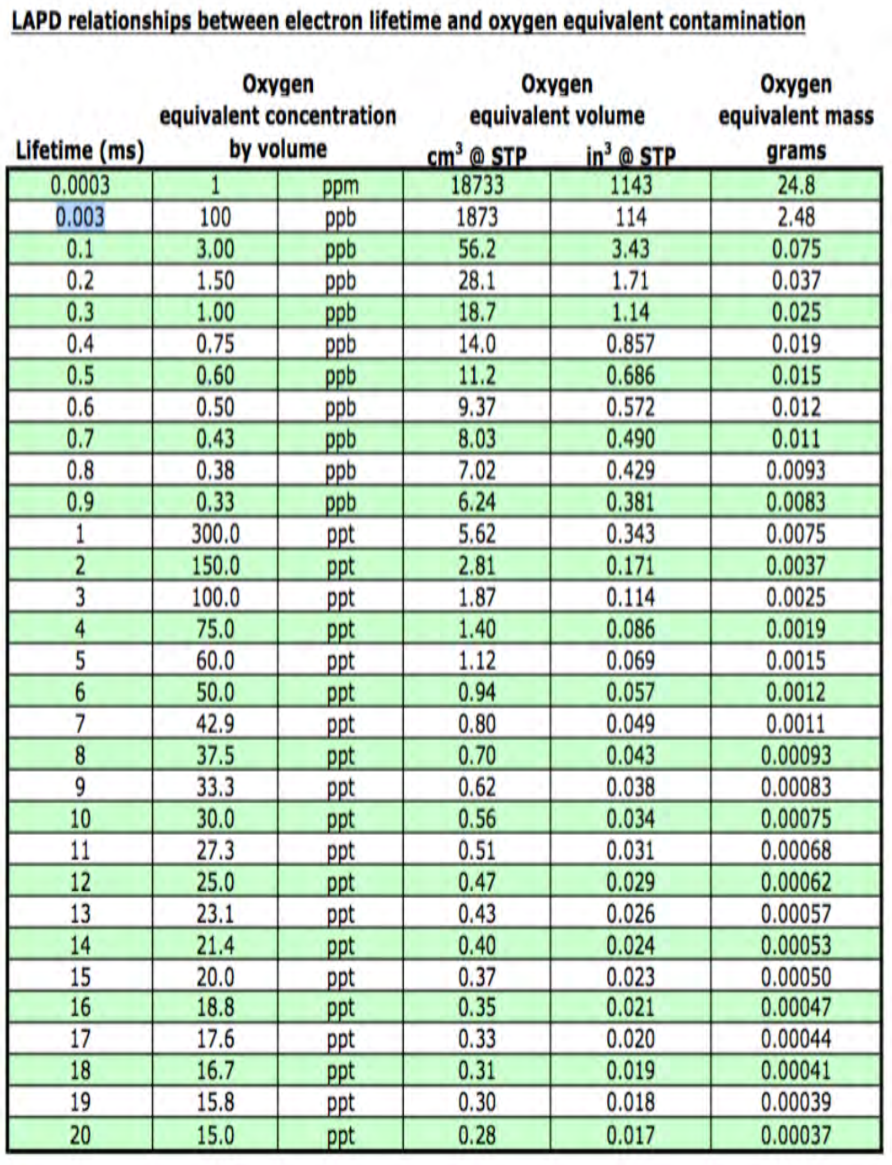
The Calibration Database Table Definitions. – Defines the database tables which can be accessed using captDBI.

The calibration packages provide the low level data processing and calibration algorithms for data analysis. These packages apply the calibration constants stored in the databasePackages to the offline data stream.

clusterCalib – Translates digits (low level hit information) into calibrated 2D hits. This is done in two steps. The first is to apply low level calibrations from ADC samples to charges vs time. The second is to find peaks and to determine the peak time and total charge. The hit time, charge, and object (i.e. the CP::TGeometryId value) are assigned to a THit. The resulting hits are ready for use in the reconstruction.

Appendix B

LAPD tank lifetime O_2 equivalent



LAPD relationships between electron lifetime and oxygen equivalent contamination

Lifetime (ms)	Oxygen equivalent concentration by volume		Oxygen equivalent volume		Oxygen equivalent mass
			cm^3 @ STP	in^3 @ STP	grams
0.0003	1	ppm	18733	1143	24.8
0.003	100	ppb	1873	114	2.48
0.1	3.00	ppb	56.2	3.43	0.075
0.2	1.50	ppb	28.1	1.71	0.037
0.3	1.00	ppb	18.7	1.14	0.025
0.4	0.75	ppb	14.0	0.857	0.019
0.5	0.60	ppb	11.2	0.686	0.015
0.6	0.50	ppb	9.37	0.572	0.012
0.7	0.43	ppb	8.03	0.490	0.011
0.8	0.38	ppb	7.02	0.429	0.0093
0.9	0.33	ppb	6.24	0.381	0.0083
1	300.0	ppt	5.62	0.343	0.0075
2	150.0	ppt	2.81	0.171	0.0037
3	100.0	ppt	1.87	0.114	0.0025
4	75.0	ppt	1.40	0.086	0.0019
5	60.0	ppt	1.12	0.069	0.0015
6	50.0	ppt	0.94	0.057	0.0012
7	42.9	ppt	0.80	0.049	0.0011
8	37.5	ppt	0.70	0.043	0.00093
9	33.3	ppt	0.62	0.038	0.00083
10	30.0	ppt	0.56	0.034	0.00075
11	27.3	ppt	0.51	0.031	0.00068
12	25.0	ppt	0.47	0.029	0.00062
13	23.1	ppt	0.43	0.026	0.00057
14	21.4	ppt	0.40	0.024	0.00053
15	20.0	ppt	0.37	0.023	0.00050
16	18.8	ppt	0.35	0.021	0.00047
17	17.6	ppt	0.33	0.020	0.00044
18	16.7	ppt	0.31	0.019	0.00041
19	15.8	ppt	0.30	0.018	0.00039
20	15.0	ppt	0.28	0.017	0.00037

FIGURE B.1: LAPD-Tank Lifetime O2 Equivalents Table.

Appendix C

charge1.C macro code

```
/*
** vector sort must run compiled laser.C++
*/
#include <fstream>
#include <vector>
#include<algorithm>
#include<TStyle.h>
#include<TTree.h>
#include<TH1F.h>
#include<TF1.h>
#include<TLeaf.h>
#include<TFile.h>
#include<TBranch.h>
#include<TCanvas.h>
#include<TGraph.h>
#include<TGraphErrors.h>
#include <TLorentzVector.h>
#include <TVector3.h>


int runId;
TTree* atree;
CP::TCapEvent* event;
TF1 *flin;
TF1 *fexp;
TFile *fout;
using namespace std;
enum {XPL,VPL,UPL,NPLANES};
bool isMC;

void charge1(int iev=13) {

  //TString tag("cluster-6389-12");
  //TString tag("cluster-6388-50");
  TString tag("mc_pg_muon-20-n100");
  isMC = false;
  if(tag.Contains("pg")) isMC = true;

  char foutName[250];
  sprintf(foutName,"charge-%s.root",tag.Data());
  printf(" out file name %s \n",foutName);
  fout = new TFile(foutName,"RECREATE");
  //TString directory("PDSF_DATA/");
  TString directory("$BRIDGE/WNRoffline/");
  // now get captana data file
  //TString tag("cluster-6300-n500-Cosmics");
  TString inputFileName = directory + TString("captAna-")+tag+
    TString(".root");
  cout << " opening " << inputFileName << endl;
  TFile *infile = new TFile(inputFileName,"READONLY");
  if(!infile) { printf(" input file %s not found \n",inputFileName
    .Data()); return ; }

  atree=NULL;
  int aSize = 0;
  atree = (TTree*) infile->Get("anaTree");
  if(atree) aSize=atree->GetEntriesFast();
  printf(" anaTree with %i entries \n",aSize);

  if(aSize==0) return;
  // this access the tree
  event = new CP::TCapEvent();
  atree->SetBranchAddress("event",&event);

 /*for(int entry=0; entry < atree->GetEntriesFast(); ++entry) {
    atree->GetEntry(entry);
    printf(" .... entry %i eventId %i \n ",entry,event->EventId);
  }*/

    if(iev==45) laser1(tag,45,0,80,125,-10,135,5);
    if(iev==47) laser1(tag,47,0,10,40,-500,-300,5);
    if(iev==41) laser1(tag,41,0,0,38,900,1150,5);
    if(iev==23) laser1(tag,23,0,60,100,-110,100,5);
    if(iev==15)  laser1(tag,15,0,240,275,1100,1500,5);
    if(iev==14) laser1(tag,14,0,200,275,100,200,5);
    if(iev==50) laser1(tag,50,0,200,245,-50,100,5);
    if(iev==8) laser1(tag,8,0,30,45,1800,1900,5);
    if(iev==6) laser1(tag,6,0,80,130,-200,300,5);
    if(iev==12) laser1(tag,12,0,0,60,-20,60,5);
    if(iev ==13) laser1(tag,13,0,320,400,200,500,10);

  fout->ls();
  fout->Write();
}


float laser1(TString atag, int eventId, int thePlane,  int
    wire_low, int wire_high, float startTime , float stopTime ,
    float delta )
{
  char chtag[120];
  sprintf(chtag,"event-%i-plane%i-wires%i-%i-times-%.0f-%.0f",
    eventId, thePlane, wire_low, wire_high,startTime,stopTime);
  TString tag(chtag);
  printf(" calling laser1: event %i plane %i  wires (%i,%i) times
    (%.0f,%.0f) delta %.1f tag=%s\n",
      eventId, thePlane, wire_low, wire_high,startTime,stopTime,
    delta,tag.Data());

  //return false;


  bool ifound=false;
  for(int entry=0; entry < atree->GetEntriesFast(); ++entry) {
    atree->GetEntry(entry);
    if(event->EventId == eventId ) {
      ifound=true;
      break;
    }
  }
  if(!ifound) {
    printf(" event not found %i \n",eventId);
    return -1.;
  }

   event->print(entry);
   vector<float> tvec(0);
   vector<float> tsort(0);
   vector<float> dtvec(0);
   vector<float> xvec(0);
   vector<float> dxvec(0);
   vector<float> twec(0);
   vector<float> xwec(0);
   vector<float> qvec(0);
   vector<float> dqvec(0);

  std::vector<CP::TCapHit> fhits= event->hits;
  printf(" number of calib hits in %i event is %i \n",eventId,(int
    ) fhits.size());


  if(fhits.size()==0) return 0;


  // x plane
   vector<float> tvec0(0);
   vector<float> xvec0(0);
  // u plane
   vector<float> tvec1(0);
   vector<float> xvec1(0);
  // v plane
   vector<float> tvec2(0);
   vector<float> xvec2(0);



   // Loop over all entries of the TTree or TChain.
   for(int ihit =0 ; ihit<fhits.size() ; ++ihit) {
     CP::TCapHit ahit = fhits[ihit];
     //ahit.print(entry);
     float hitTime =ahit.t*1E-3; // convert to mircroseconds
     //if(ahit.plane!=thePlane) continue;
     //if(ahit.wire<wire_low||ahit.wire>wire_high) continue;
     //if(hitTime<startTime || hitTime>stopTime ) continue;


     if(ahit.plane==0) {
       tvec0.push_back(hitTime);
       xvec0.push_back(ahit.wire);
     }
     if(ahit.plane==1) {
       tvec1.push_back(hitTime);
       xvec1.push_back(ahit.wire);
     }
     if(ahit.plane==2) {
       tvec2.push_back(hitTime);
       xvec2.push_back(ahit.wire);
     }

   }
   // Loop over all entries of the TTree or TChain.
   for(int ihit =0 ; ihit<fhits.size() ; ++ihit) {
     CP::TCapHit ahit = fhits[ihit];
     float hitTime =ahit.t*1E-3; // convert to mircroseconds
```

```
      printf(" hit in track  %i q %f dq %f (%f) wire %i time = %f \
    n",ihit,ahit.q,ahit.dq,ahit.dq/ahit.q,ahit.wire,hitTime);
      if(ahit.plane!=thePlane) continue;
      float hitDt =ahit.dt*1E-3; // convert to mircroseconds
      twec.push_back(hitTime);
      int wire = ahit.wire;
      if(wire<150&&!isMC) {
        if(wire%2==0) wire = wire-1;
        else wire=wire+1;
      } else if(!isMC){
        if(wire%2==0) wire = wire+1;
        else wire=wire-1;
      }
      xwec.push_back(wire);
      if(ahit.q<1) continue;
      if(ahit.wire<wire_low||ahit.wire>wire_high) continue;
      if(hitTime<startTime || hitTime>stopTime ) continue;
      tsort.push_back(hitTime);
      tvec.push_back(hitTime);
      dtvec.push_back(hitDt);
      xvec.push_back(wire);
      dxvec.push_back(0);
      qvec.push_back(ahit.q);
      // dq has changed in clusterCalib
      //dqvec.push_back(qrt(ahit.q));
      float wnoise = pow(ahit.noise,2.0)*float(ahit.nsamples);
      float wq = ahit.q;
      float qerr = sqrt(wnoise+wq);
      dqvec.push_back(qerr);
      //ahit.print();
    }


     printf(" \n\n plotting all hits for plane %i \n",thePlane);

     char gatitle[120];
     char catitle[120];
     if(thePlane==0) {
     sprintf(gatitle,"run%i-Event-%i-x-plane-%s",runId,eventId,tag.
    Data());
     sprintf(catitle,"all-time-wire-run%i-Event-%i-x-plane-%s",
    runId,eventId,tag.Data());
     TCanvas *ca = new TCanvas(catitle,catitle);
     TGraph *gaxt = new TGraph(tvec0.size(),&(xvec0[0]),&(tvec0[0])
    );
     gaxt->SetTitle(gatitle);
     gaxt->GetHistogram()->GetXaxis()->SetTitle("wire x plane ");
     gaxt->GetHistogram()->GetYaxis()->SetTitle("time ({#mu}s)");
     gaxt->SetMarkerStyle(21);
     gaxt->SetMarkerSize(.4);
     gaxt->SetMarkerColor(kBlue);
     gaxt->Draw("ap");
     ca->Print(".pdf");
     fout->Append(ca);


     } else if(thePlane==2) {

     sprintf(gatitle,"run%i-Event-%i-u-plane-%s",runId,eventId,tag.
    Data());
     sprintf(catitle,"all-time-wire-run%i-Event-%i-u-plane-%s",
    runId,eventId,tag.Data());
     TCanvas *cau = new TCanvas(catitle,catitle);
    TGraph *gaut = new TGraph(tvec2.size(),&(xvec2[0]),&(tvec2[0])
    );
    gaut->SetTitle(gatitle);
    gaut->GetHistogram()->GetXaxis()->SetTitle("wire u plane ");
    gaut->GetHistogram()->GetYaxis()->SetTitle("time ({#mu}s)");
    gaut->SetMarkerStyle(21);
    gaut->SetMarkerSize(.4);
    gaut->SetMarkerColor(kBlue);
    gaut->Draw("ap");
    cau->Print(".pdf");
    fout->Append(cau);

    } else if(thePlane==1) {

    sprintf(gatitle,"run%i-Event-%i-v-plane-%s",runId,eventId,tag.
    Data());
    sprintf(catitle,"all-time-wire-run%i-Event-%i-v-plane-%s",
    runId,eventId,tag.Data());
    TCanvas *cav = new TCanvas(catitle,catitle);
    TGraph *gavt = new TGraph(tvec1.size(),&(xvec1[0]),&(tvec1[0])
    );
    gavt->SetTitle(gatitle);
    gavt->GetHistogram()->GetXaxis()->SetTitle("wire v plane ");
    gavt->GetHistogram()->GetYaxis()->SetTitle("time ({#mu}s)");
    gavt->SetMarkerStyle(21);
    gavt->SetMarkerSize(.4);
    gavt->SetMarkerColor(kBlue);
    gavt->Draw("ap");
    cav->Print(".pdf");
    fout->Append(cav);
    }



   if(tsort.size()==0) { printf(" !!!!!!!!!!!!!!!! no hits passed
     cuts ! \n"); return 0; }
   printf("  hits passed cuts = %i \n", tsort.size() );
   if(tsort.size()<7) return;



   sort(tsort.begin(),tsort.end());

   printf("tsort : ");
   for(unsigned iw =0; iw<tsort.size(); ++iw) printf(" %i) %f; ",
    iw,tsort[iw]);
   printf("\n");

  fout->cd();
  // summed charge
  char htitle[120];
  float nbins = (tsort[tsort.size()-1]-tsort[0])/delta;
  sprintf(htitle,"qsumUn-%s",tag.Data());
  TH1F *hsumUn = new TH1F(htitle,htitle,nbins,tsort[0],tsort[tsort
    .size()-1]+delta);
  hsumUn->Sumw2();
  hsumUn->Print();

  sprintf(htitle,"qsumsq-%s",tag.Data());
  TH1F *hsumsq = new TH1F(htitle,htitle,nbins,tsort[0],tsort[tsort
    .size()-1]+delta);

```

```
 sprintf(htitle,"qerr-%s",tag.Data());
TH1F *hqerr = new TH1F(htitle,htitle,nbins,tsort[0],tsort[tsort.
  size()-1]+delta);


 sprintf(htitle,"qnorm-%s",tag.Data());
 // make the sum for each bin, store in histograms
TH1F *hnorm = new TH1F(htitle,htitle,nbins,tsort[0],tsort[tsort.
  size()-1]+delta);
 for(unsigned iq=0; iq<qvec.size() ; ++iq) {
   int ibin = hsumUn->FindBin(tvec[iq]);
  hsumUn->SetBinContent(ibin,   hsumUn->GetBinContent(ibin)+qvec
  [iq]);
  hsumsq->SetBinContent(ibin, hsumsq->GetBinContent(ibin)+pow(
  qvec[iq],2.));
   hqerr->SetBinContent(ibin,  hqerr->GetBinContent(ibin)+pow((
  dqvec[iq]/qvec[iq]),2.0));
  hnorm->SetBinContent(ibin,  hnorm->GetBinContent(ibin)+1.0);
 }
 sprintf(htitle,"qsum-%s",tag.Data());
TH1F *hsum = new TH1F(htitle,htitle,nbins,tsort[0],tsort[tsort.
  size()-1]+delta);
hsum->Sumw2();
hsum->Print();

 // calculate qrms for each bin
 //for(int ibin = 1; ibin<=hsum->GetNbinsX(); ++ibin)  hsum->
  SetBinError(ibin, hsum->GetBinContent(ibin)*qrt(hqerr->
  GetBinContent(ibin))/hnorm->GetBinContent(ibin));
 for(int ibin = 1; ibin<=hsum->GetNbinsX(); ++ibin)  {
   float qnorm = hnorm->GetBinContent(ibin);
   if(qnorm<1) continue;
   float qsum2 = hsumsq->GetBinContent(ibin)/qnorm;
   float qsum =  hsumUn->GetBinContent(ibin)/qnorm;
   float qerr =  qsum*sqrt(hqerr->GetBinContent(ibin));
   float qrms=0;
   if(qnorm>1) qrms = sqrt(qsum2-qsum*qsum);
   else qrms=qerr;
   float low = hsumUn->GetBinLowEdge(ibin);
   float up = low + hsumUn->GetBinWidth(ibin);
   printf(" %i qnorm %0.f (%0.1f,%0.1f) qsum2 %E qsum2^2 %E qerr
  %E qrms = %E \n",ibin, qnorm,low,up, qsum2 , qsum*qsum, qerr,
  qrms );
  hsum->SetBinContent(ibin, qsum);
  hsum->SetBinError(ibin, qrms);
 }

 //hsum->Print("all");

 /*
 printf(" summed plot number of bins %i \n",hsum->GetNbinsX());
 for(int ibin = 1; ibin<=hsum->GetNbinsX(); ++ibin) if(hsum->
  GetBinContent(ibin)>0) printf(" ... %i %f q = %f +/- %f (%f)  \
  n",
     ibin,hsum->GetBinCenter(ibin),hsum->GetBinContent(ibin),hsum
  ->GetBinError(ibin),hsum->GetBinError(ibin)/hsum->GetBinContent
  (ibin));
 */


 gStyle->SetOptFit(1);

 char gtitle[120];
```

```
  char ctitle[120];
  TGraphErrors *gxt = new TGraphErrors(xvec.size(),&(xvec[0]),&(
    tvec[0]),&(dxvec[0]),&(dtvec[0]));
  sprintf(gtitle,"run%i-Event%i-%s-hits-%i",runId,eventId,tag.Data
    (),gxt->GetN());
  sprintf(ctitle,"time-vs-wire-run%i-Event%i-%s-hits-%i",runId,
    eventId,tag.Data(),gxt->GetN());
  TCanvas *c1 = new TCanvas(ctitle,ctitle);
  gxt->SetTitle(gtitle);
  gxt->GetHistogram()->GetXaxis()->SetTitle("wire");
  gxt->GetHistogram()->GetYaxis()->SetTitle("time (#mu s)");
  gxt->SetMarkerStyle(21);
  gxt->SetMarkerSize(.4);
  gxt->SetMarkerColor(kBlue);
  printf(" >>>>>>>>>>>>>>> fitting %s %i <<<<<<<<<<<<<<<<<<< \n",
    gtitle,gxt->GetN());
  flin = new TF1("flin","[0]+x*[1]",xvec.front(),xvec.back());
  gxt->Fit("flin");
  flin->Print();
  gxt->Draw("ap");
  c1->Print(".pdf");
  fout->Append(c1);

  // fit function
  TF1* fexp = new TF1("fexp","expo",tvec.front(),tvec.back());


  char c2title[120];
  sprintf(c2title,"average-charge-time-run%i-Event%i-%s",runId,
    eventId,tag.Data());
  TCanvas *c2 = new TCanvas(c2title,c2title);
  //c2->SetLogy();
  hsum->Fit("fexp");
  double tau = -1./fexp->GetParameter(1);
  double tauError = fexp->GetParError(1)/fexp->GetParameter(1)/
    fexp->GetParameter(1);
  char sumTitle[80];
  char yTitle[80];
  sprintf(sumTitle,"#sum q (%.0f#mus) ",delta);
  sprintf(yTitle,"time (#mus) #tau = %0.2f #pm %0.2f",tau,tauError
    );
  hsum->SetMarkerStyle(21);
  hsum->SetMarkerSize(.4);
  hsum->SetMarkerColor(kBlue);
  hsum->GetYaxis()->SetTitle(sumTitle);
  hsum->GetXaxis()->SetTitle(yTitle);
  hsum->Draw("");
  c2->Print(".pdf");
  fout->Append(hsum);
  fout->Append(c1);
  fout->Append(c2);


  char gqtitle[120];
  char cqtitle[120];
  TGraphErrors *gqt = new TGraphErrors(tvec.size(),&(tvec[0]),&(
    qvec[0]),&(dtvec[0]),&(dqvec[0]));
  sprintf(gqtitle,"charge-vs-time-run%i-Event%i%s-hits-%i",runId,
    eventId,tag.Data(),gqt->GetN());
  sprintf(cqtitle,"charge-time-run%i-Event%i%s-hits-%i",runId,
    eventId,tag.Data(),gqt->GetN());
  gqt->SetTitle(gqtitle);
  gqt->SetMarkerStyle(21);
  gqt->SetMarkerSize(.4);
  gqt->SetMarkerColor(kBlue);
  gqt->GetHistogram()->GetYaxis()->SetTitle("charge");
  gqt->GetHistogram()->GetXaxis()->SetTitle("time (#mu s)");
  /*TCanvas *cq1 = new TCanvas(cqtitle,cqtitle);
  gqt->Draw("ap");
  cq1->Print(".pdf");
  */


  TGraph* grfit=gqt;

  if(grfit->GetN()==0) return -1;
  TF1* ffexp = new TF1("ffexp","expo",tvec.front(),tvec.back());


  float timeZero = tvec[0];
  char c3title[120];
  sprintf(c3title,"charge-time-fit-run%i-Event%i-%s-hits-%i",runId
    ,eventId,tag.Data(),grfit->GetN());
  TCanvas *c3 = new TCanvas(c3title,c3title);
  //c2->SetLogy();
  double tau = -1./fexp->GetParameter(1);
  double tauError = fexp->GetParError(1)/fexp->GetParameter(1)/
    fexp->GetParameter(1);
  char sumTitle[80];
  char yTitle[80];
  sprintf(sumTitle,"charge ");
  sprintf(yTitle,"time (#mus) #tau = %0.2f #pm %0.2f",tau,tauError
    );
  grfit->Fit("ffexp");
  grfit->Print();
  grfit->SetMarkerStyle(21);
  grfit->SetMarkerSize(.4);
  grfit->SetMarkerColor(kBlue);
  grfit->GetHistogram()->GetYaxis()->SetTitle(sumTitle);
  grfit->GetHistogram()->GetXaxis()->SetTitle(yTitle);
  grfit->Draw("ap");
  c2->Print(".pdf");
  fout->Append(c3);

  return timeZero;

}

```

LISTING C.1: charge1.C macro code

Appendix D

electron lifetime plots

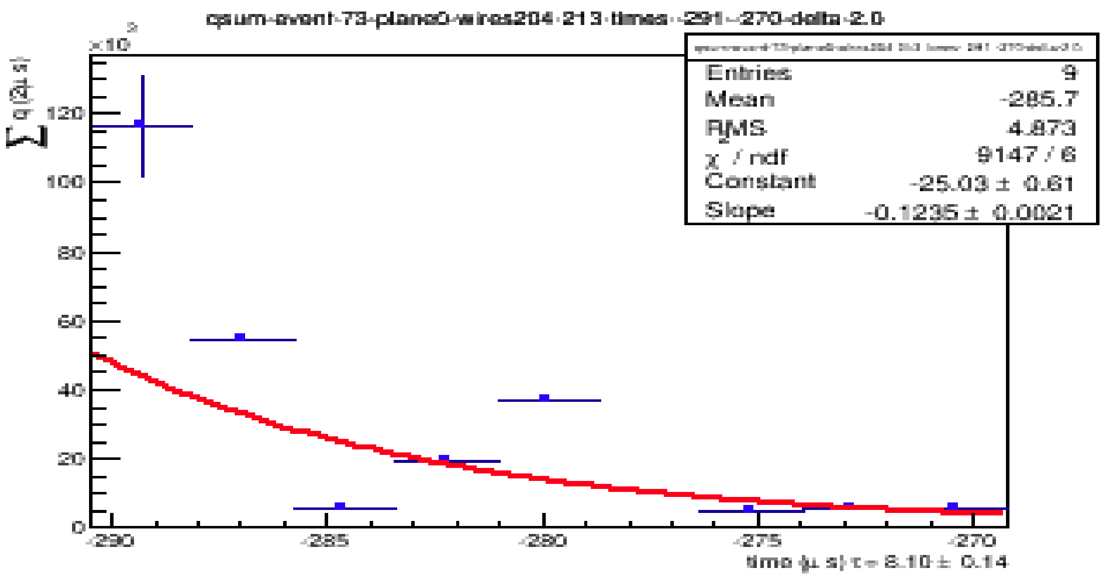


FIGURE D.1: 6300-E73.

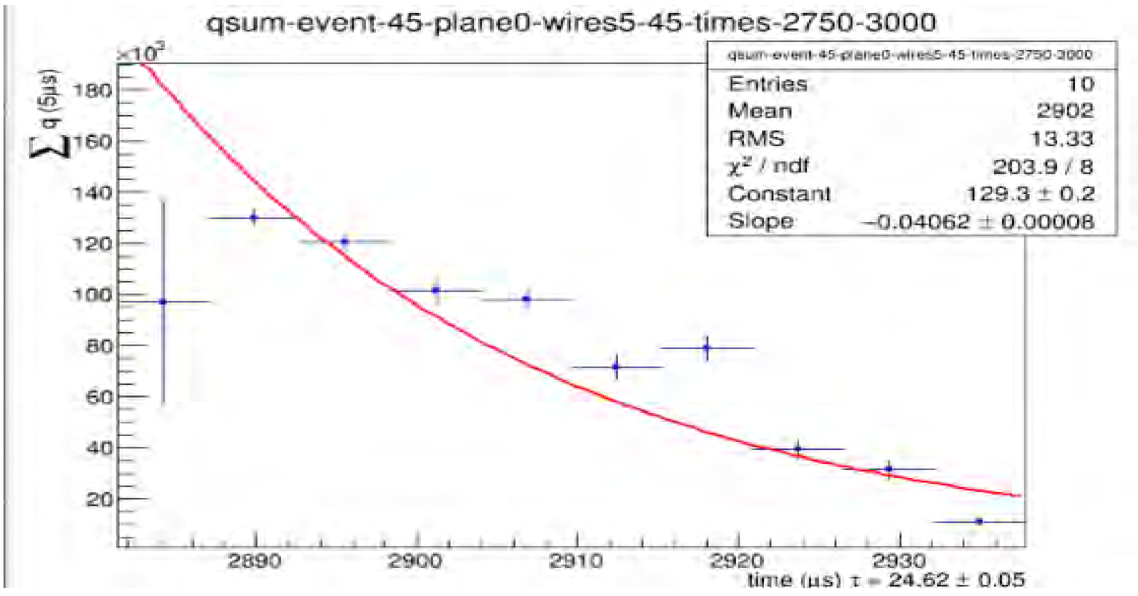


FIGURE D.2: 6395-E45.

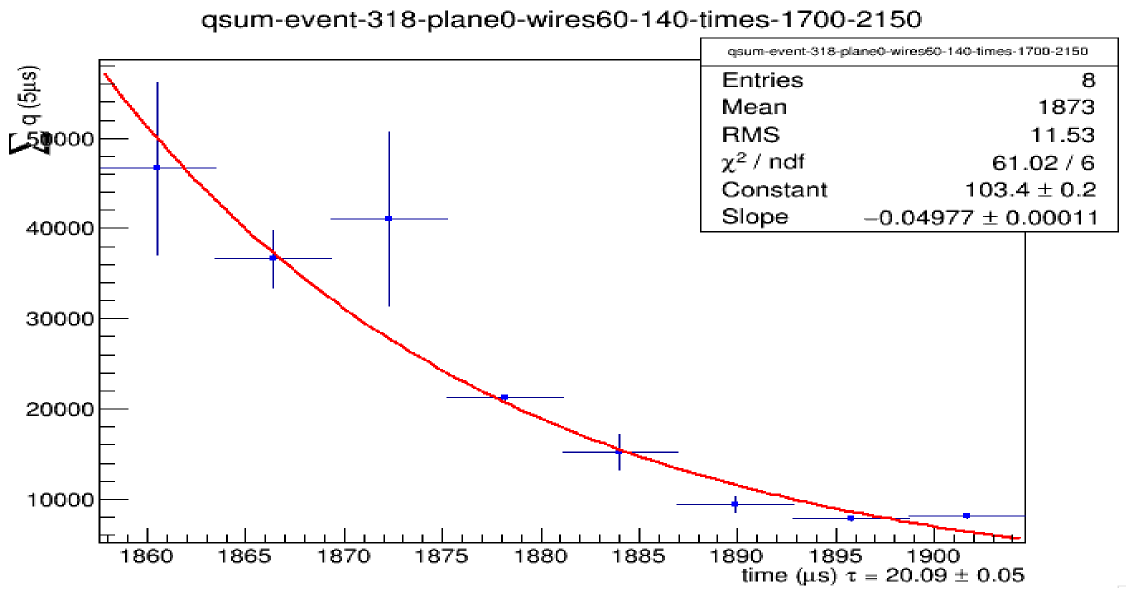


FIGURE D.3: 6395-E318.

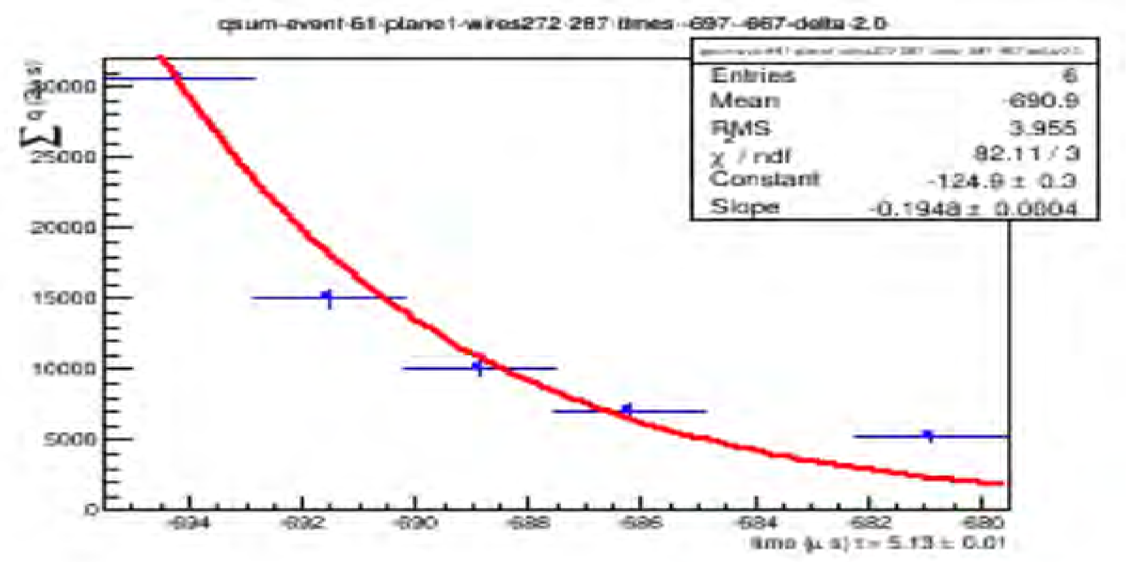


FIGURE D.4: 6395-E61.

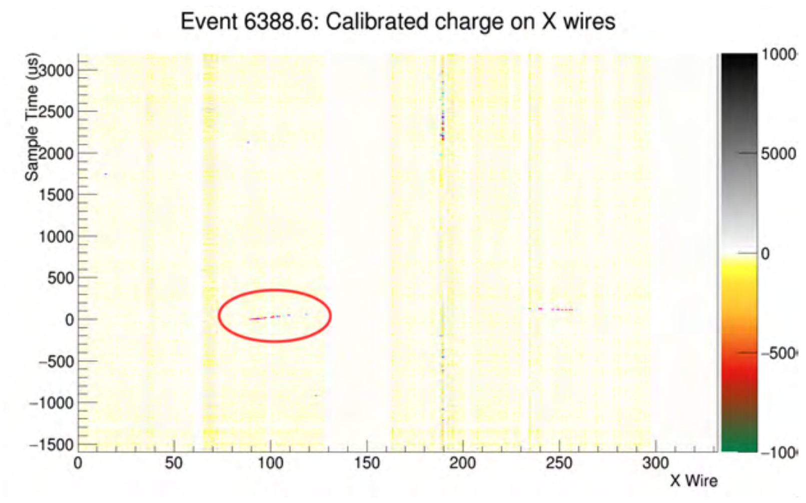
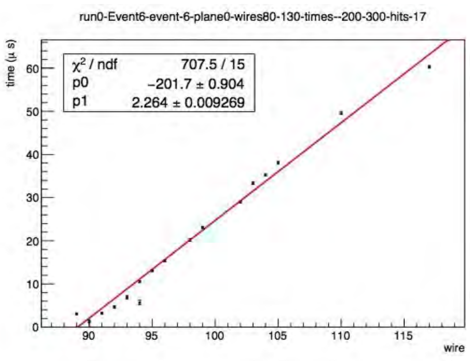
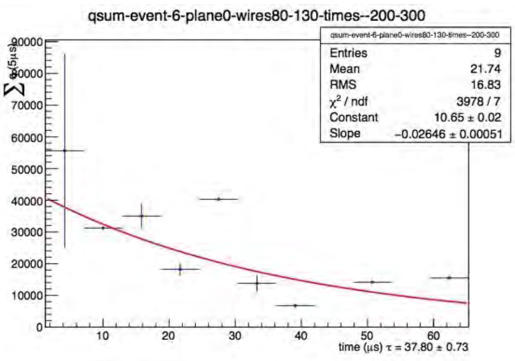


FIGURE D.5: run 6389 event 6.





(A) charge fit of run 6388 event 6.

(B) electron lifetime run 6388 event 6

FIGURE D.6: Cosmic run 6388 event 6, τ = 37.80 +/- 0.73 μs.

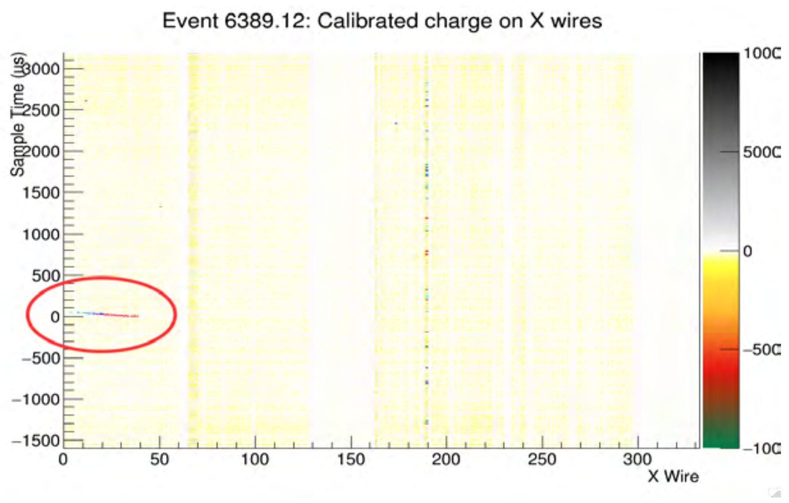
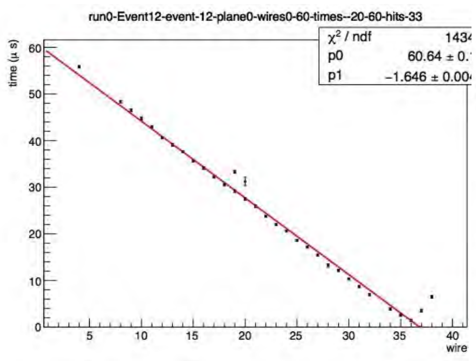
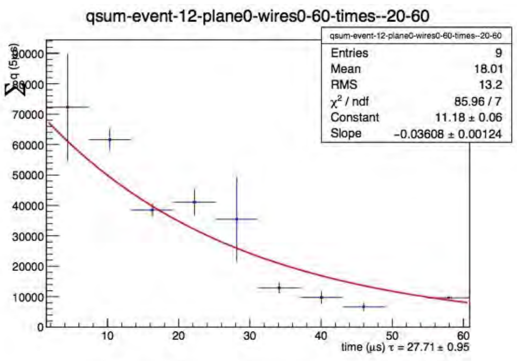


FIGURE D.7: run 6389 event 12.



(A) charge fit of run 6389 event 12.



(B) electron lifetime run 6389 event 12

FIGURE D.8: Cosmic run 6389 event 12, $\tau = \tau = 29.7$ +/- 2.96 μs.

References

[1] H. Berns et. al. "[The CAPTAIN Collaboration]". In: *arXiv:1309.1740 [physics.ins-det].* (2015.).

[2] P5. "Particle Physics Project Prioritization Panel (P5) report (2014)" Panel (P5) report". In: (2015). URL: http:%20//%20science.%20energy.%20gov/%20~%20/%20media/%20hep/%20hepap/%20pdf/%20May%2020%7B%5C/%7D2014/%20FINAL_%20DRAFT2%7B%5C_%7D%20P5Report%7B%5C_%7D%20WEB%7B%5C_%7D%2005.2114.%20pdf (visited on 08/14/2015).

[3] J.N. Marx and D.R. Nygren. "The Time Projection Chamber." In: *Physics Today, 31(10):46-53, 1978.* ().

[4] C. Rubbia. "The liquid-argon time projection chamber: A new concept for neutrino detectors." In: *CERN-EP, (77-8)* (May 1977.).

[5] M. Schenk. "Studies with a Liquid Argon Time Projection Chamber: Addressing Technological Challenges of Large-Scale Detectors." In: Springer Spektrum, 1.1 Working Principle ((6-7), May 2015.).

[6] FNAL. "MicroBoone CD3b review". In: (2016). URL: http://www-microboone.fnal.gov/ (visited on 03/16/2016).

[7] CERN. "CERN documentation". In: (2016). URL: https://root.cern.ch/ (visited on 16/16/2016).

[8] John Ramsay Qiuguang Liu Gus Sinnis. "CAPTAIN The CAPTAIN Laser System". In: *Docdb, (25-v1)* (June 2015.).

[9] LBNE. "Volume 5: A Liquid Argon Detector for LBNE". In: (2016). URL: http://lbne.fnal.gov/reviews/farsite-dec2011-LAr.shtml (visited on 04/16/2016).

[10] LBNE. "LAr properties LAr Properties LBNE". In: (2014). URL: http://lbne2-docdb.fnal.gov/cgi-bin/RetrieveFile?docid=4482&filename=Properties%20of%20LAr%20v9a.pdf&version=1 (visited on 02/14/2016).

[11] Yichen Lia et. al. "Measurement of Longitudinal Electron Diffusion in Liquid Argon". In: *arXiv* 1 (2016), pp. 1–22. DOI: 00.1. arXiv: 0012 [physics].

[12] CRC/Taylor and Francis et al. "wikipedia Green's Function Library". In: (2011). URL: https://en.wikipedia.org/wiki/Heat_equation#Some_Green.27s_function_solutions_in_1D (visited on 05/05/2016).