1 Introduction

This technical note represents an attempt to document how the target polarization analysis was performed for Hall C experiment E93-026, also known as Gen01.

The analysis was performed at UVa by Paul McKee, with help and input from the rest of the Gen01 collaboration. All of the files and programs described in this document can be found on the computer `blur.phys.virginia.edu`, to which major collaboration members have access through the `target` account. Other details about the polarized target, including a link to this document, are located on the same machine at `http://blur.phys.virginia.edu/gen`.

1.1 Computer Programs Used

Most of the scripts used in this analysis are written in the Python programming language. It is similar to Perl in its capabilities, but is easier to learn and is significantly easier to read. Some of the smaller script tasks are handled by `awk`, a line-by-line text file processor that uses a C-style language. The `tcsh` shell is used for shell scripts. Finally, the program that extracts NMR signals from the event files and performs the fits and signal integration is written in C, mostly for speed reasons. This code calls routines in the GNU Scientific Library (GSL) for curve fitting and matrix algebra. GSL is similar in scope to the Fortran language SLATEC library.

Plots are made by one of two programs. The `graph` program from the GNU plotutils package takes data on standard input as x-y pairs, and produces publication-quality output in many formats. For plots generated by scripts, such as details of the signal fitting, `topdrawer` is used. Typically a program will store its output in a file, and then `topdrawer` is set to process that file into an X window or postscript file.

2 Technique of Polarization Determination

This section describes the programs used to perform the analysis and the details of their use. The base directory for all analysis code is `/exp/gen/target/anal/`.

The basic approach taken to accomplish this analysis is to create a separate directory for each of the physics quantities necessary for calculation of the target polarization. In each directory are the various scripts and programs necessary to calculate that quantity, as well as a master data file. This file has the same name as the directory in which it sits, with a ".dat" extension. The file is in text format, with two or more fields of fixed width. The first field is always the unix timestamp, a 32-bit integer representing the number of seconds since 1970. The following field or fields contains the value of the physics quantity at that time.

Many times a given physics variable depends on others, which means that several scripts may need access to each quantity. In order to reduce code duplication and avoid bugs, a library of Python routines, `resources/target.py`, was developed to provide access to these quantities. For example, to find the target position at a given time, a subroutine called `get_pos()` can be called, with the argument being the timestamp of interest. The return value indicates either the position of the target at that time, or an error. This avoids the situation in which multiple scripts would each have their own code to read and parse the `position.dat` file; should the format of that file change, only the `get_pos()` routine in the python library needs to be changed.

Many of the subdirectories have a file named `commands` that contains shell commands used to do various parts of the analysis. Many of these files are also heavily commented, describing the steps taken and the decisions made. It should be possible to execute each of the `commands` files with `source commands` in the tcsh shell.

2.1 Manually Entered Information

Although a great deal of information about the target was logged automatically by a suite of DAQ programs, some important events were only recorded manually, in the paper logbook and/or the electronic HCLog.

At the end of the experiment, a page-by-page search was made of the more than 500 pages of paper logbook entries, and any relevant information entered into a file, `logbook.txt`, in a format easily parsed by computer programs. A script, `log2web`, puts the information in this file into a web page, `target_history.html`, so that humans may also easily read it.

There was not a similar effort made to extract important information from the HCLog because of the enormous number of entries. However, any relevant information discovered in the HCLog was added to `logbook.txt` so that the discovery would not be lost.

The files discussed in this section are located in the `logbook/` directory. A copy of `target_history.html` is also maintained on the Gen01 Target Webpage, `http://blur.phys.virginia.edu/gen`.

2.2 EPICS Data

A computer in the counting house, `jeffylab`, ran an EPICS archiver program that captured a number of EPICS variables at 2 second intervals. The size of the archive files required that they be erased from one experiment to another, so relevant data was copied from `jeffylab` to the target analysis directory for further processing.

The perl script `pmm_dump.pl` extracts a variable from the archive into a text file containing two fields: the UNIX timestamp and the value of the variable at that time. It does this for three time periods:

- 29-Jul-2001 18:00:00 to 18-Sep-2001 00:00:00 ($Q^2 = 0.5$)
- 22-Oct-2001 00:00:00 to 21-Dec-2001 06:00:00 ($Q^2 = 1.0$)

- 19-Jan-2002 00:00:00 to 05-Mar-2002 00:00:00 (RSS)

The shell script `get_all_epics` calls `pmm_dump.pl` for each variable that might be of use in target analysis. The variables extracted are: `CFI6711C` (mass flow of liquid helium into Hall C), `EV9177` (warm return valve), `EV9178R` (cold return valve), `EV9180` (target JT valve), `ibcm1` (beam current monitor #1), `ibcm2` (beam current monitor #2), `LL91101` (buffer dewar level), `LL91110` (liquid nitrogen level), `LL91111` (magnet liquid helium level), `LL91112` (tail liquid helium level), `LL91113` (separator liquid helium level), `PI91104` (buffer dewar pressure), `PI91112` (magnet can pressure), `PI91125` (separator pressure), `PI91131` (isolation vacuum pressure, vacuum side of valve), `PI91132` (isolation vacuum pressure, pump side of valve), `PI91135` (warm return pressure), `PI91140` (refrigerator pump line pressure), `PI91145` (mechanical pumpset input pressure), `ptencoder` (target table encoder value), `ptpol` (reported online target polarization), `ptpos` (target position code), `ptrunnum` (current run number [zero for no run in progress]), `SV91110` (liquid nitrogen precool valve), `SV91115` (magnet boiloff vent to atmosphere valve), `SV91148` (main pumps vent to atmosphere valve), `TD9178` (buffer dewar return temperature), and `TD9180` (buffer dewar supply temperature).

A copy of the cryogenic control panel is included in Figure 1.

The files discussed in this section are in the `epics/` directory, but the scripts must be run on `jeffylab`, since that is where the EPICS archive is located.

2.3 Run Times

There are several means of determining the start and stop times of the runs taken during the experiment. The HCLog has entries for most runs, the EPICS archive logged the `ptrunnum` variable, various online lists of runs were maintained, etc. For target analysis, the most important issue is that the times used to average multiple polarization measurements be the same as those used in the analysis of the scattering data. For example, if a run lasted 30 minutes, but only the first 15 minutes of it were able to be analyzed due to a file corruption, then the target analysis should only average those 15 minutes of polarization measurements.

After discussion with the collaboration, two methods of determining the start and stop times emerged as the best choices.

The first method is to look for the time indicated on the `INFO:GO` line in the `4????.log` files (where question marks are replaced by each run number). This time is in UTC (previously known as GMT), so a conversion is made to Eastern Standard Time (EST), and then a correction made, if necessary, to Daylight Savings Time (EDT) [1]. This time is then the start of the run as analyzed by the `syncfilter` program. The length of the run is determined by dividing the number of helicity buckets listed in the `syncfilter.4????` files by 30, and adding that number of seconds to the start time. This method is implemented by a python script called `run_times_sync`, which must be run on `ifarml1` in

[1] This correction is hard coded only for fall 2001 and spring 2002. Use of these scripts in other analyses would require an update of this correction

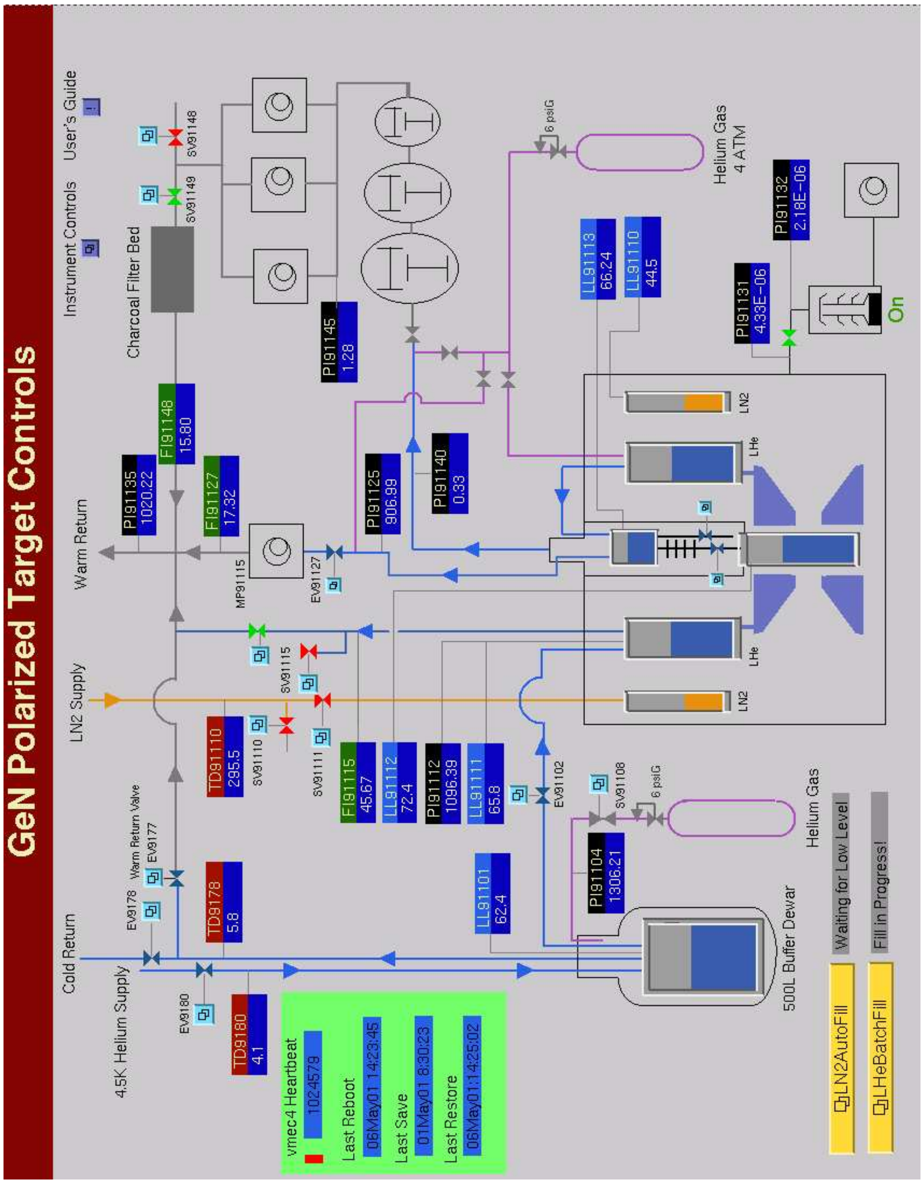

Figure 1: Copy of the cryogenic control program's main screen.

order to access the appropriate directories. Output is in three fields – the run number and the unix timestamps of the run start and stop – and is saved in the file `run_times_sync.dat`.

The second method to determine the run start and stop times is to examine the second line of the `epics4????.txt` files, which contain the timestamp of the first EPICS event of the run. These occur every 6 seconds under normal conditions; the start of the run is then taken as 3 seconds before this time.

The `gen4????.txt` files contain the `syncfilter` reports. In the Summary section, a line that begins `Time of run =` gives the run length in seconds. This is added to the run start to get the run stop time.

This method is implemented by a python script `run_times_epics`, which also must be run on `ifarml1`. Output is saved, in the format described above, in `run_times_epics.dat`.

Each time a new pass is made through the scattering data, the name of the new directory holding the results must be added to the lists at the beginning of `run_times_sync` and `run_times_epics`.

The `run_times_sync.dat` and `run_times_epics.dat` files are then copied to UVa, and combined as detailed in the `commands` file, with the results stored in `run_times.dat`. If both methods return a result for a given run, the value from `run_times_sync` is given precedence.

A histogram of the lengths of all runs is presented in Figure 2.

The files discussed in this section are copied in the `run/` directory.

2.4 Liquid Helium Level

Knowledge of the liquid helium (LHe) level in the tail is important because the presence or absence of LHe affects the dilution factor significantly. In addition, if the LHe level dropped below beam hight during a polarized target run, the polarization would drop precipitously and possible damage to the material would occur.

The EPICS archive is the primary source of information on the LHe level in the tail. However, the archive does have a few holes in it. To get around these, the tail level stored in the target event files was used.

These two sources of data are combined, bad readings filtered out, and the results sorted chronologically by a shell script called `determine_level`.

During run 42742 the LHe level dropped below beam height. By examining the LHe readings in the archive and comparing them to polarization measurements taken during that run, it was determined that at least part of the raster pattern extends above the surface of the liquid for levels below 48%, see Figure 3. A level of 50% or more is therefore required for runs with rastered beam. This limit is defined as the variable `TAIL_DRY` in the python library `resources/target.py`.

The files discussed in this section are located in the `lhe/` directory.

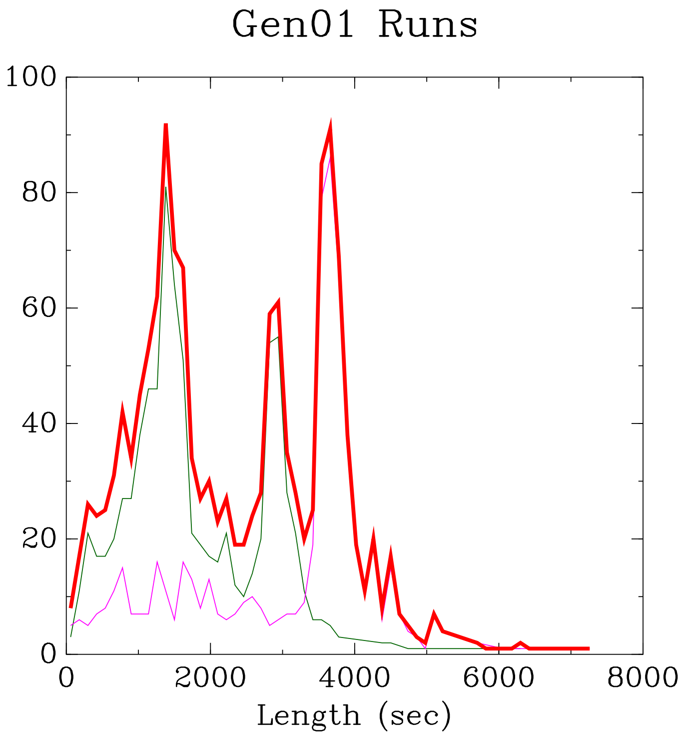

Figure 2: Lengths (in seconds) of Gen01 runs. Red is all runs, green is only $Q^2 = 0.5$ runs, magenta is only $Q^2 = 1.0$ runs.

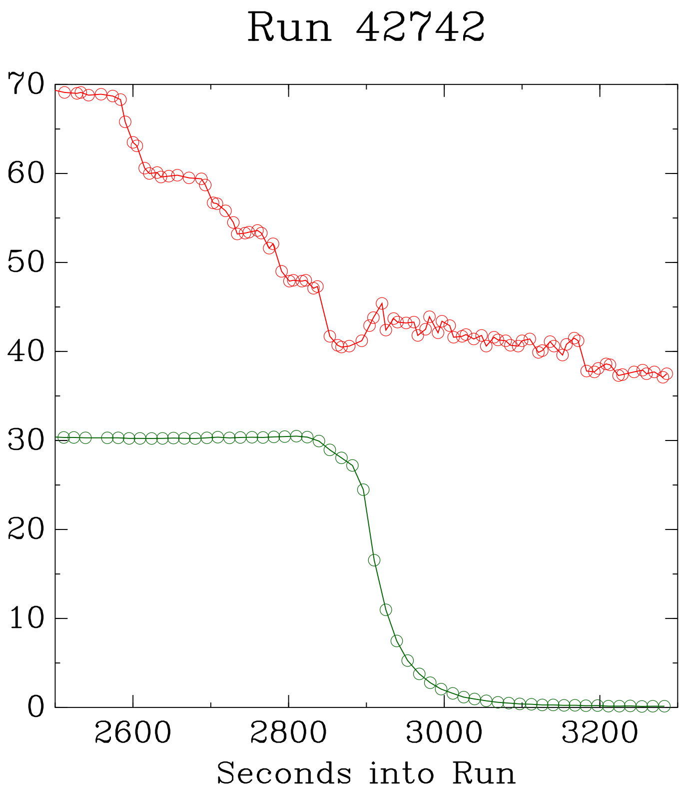

Figure 3: Liquid Helium level (red) and target polarization (green) for run 42742.

Date	Source	Unit	Top	Hole	Bot	Carb	MT
29Jul01	HCLog 37009	count	81600	2000	39700	70700	1728
29Jul01	Log I-36	count	81600	2000	39700	70700	1728
1Aug01	Log I-48	count	81660	2000	39700	70700	1728
12Aug01	EPICS	count	81660	2000	39700	70700	1728
18Aug01	EPICS	inch	3.220	0.032	1.532	2.766	0.000
10Sep01	TMC	inch	3.220	0.032	1.532	2.766	0.000
15Oct01	TMC	inch	3.220	0.032	1.532	2.766	0.000
5Nov01	TMC	inch	3.220	0.032	1.532	2.766	0.000
7Nov01	TMC	inch	3.220	0.032	1.532	2.766	0.000
19Nov01	TMC	inch	3.220	0.032	1.532	2.766	0.000
29Nov01	TMC	inch	3.220	0.032	1.532	2.766	0.000
1Dec01	Log II-291	inch	3.120	3.913	1.432	2.666	3.878
4Dec01	HCLog 44175	inch	3.160	3.953	1.472	2.706	3.918
6Dec01	TMC	inch	3.220	0.032	1.532	2.766	0.000
10Dec01	TMC	inch	3.160	3.953	1.472	2.706	3.918
7Jan02	TMC	inch	3.160	3.953	1.472	2.706	3.918
15Jan02	TMC	inch	3.160	3.953	1.472	2.706	3.918
20Jan02	TMC	inch	3.160	3.953	1.472	2.706	3.918
28Jan02	TMC	inch	3.160	3.953	1.472	2.706	3.918
7Feb02	TMC	inch	3.160	3.953	1.472	2.706	3.918
15Feb02	HCLog 47704	inch	3.200	0.013	1.512	2.746	3.958
19Feb02	TMC	inch	3.200	0.013	1.512	2.746	3.958
28Feb02	TMC	inch	3.200	0.013	1.512	2.746	3.958

Table 1: History of encoder values for each target position. "Log Y-xx" indicates that the values were obtained from Target Logbook Y, page xx. "TMC" indicates that the values were obtained from an archived copy of the Target Motion Control program.

2.5 Target Position

The position of the target insert was controlled by a compressed air motor that moved a table on which the insert and the microwave components were mounted. An optical shaft encoder was attached to this motor to allow a computer to read the position of the table, and thus determine which of the several targets were in the beam. In the beginning of the experiment, this encoder read in raw counts, from 0 to 100,000. At around 10:00 on 16 Aug 2001, the readout unit of the encoder was given a scale factor so that the encoder value represented inches of travel of the target table. The values then ranged from 0" to 3.978". The table, however, was capable of moving more than 4", so the encoder would wrap from 3.978" back to 0" at some point along its motion. Table 1 lists the history of which encoder values represented which insert positions.

There are several sources of information on what target position was in the beam at a given time. The target encoder value was stored as the EPICS variable `ptencoder`, the position was stored as the EPICS variable `ptpos`, the HCLog lists target position in the run start entries, the target operators were instructed to write changes to the position in the paper logbook, the messages from the target control computer included position information that was logged to disk, and the NMR channel used to measure the polarization can be used to indicate some of the target positions (top or bottom).

Unfortunately, flaws were discovered with all of these methods:

- In the beginning of the experiment, the position control program did not write the encoder value and target position to EPICS variables.
- The EPICS archive frequently missed recording updates to `ptencoder` and `ptpos`.
- The HCLog entries depended on the EPICS variables to determine position information, and even then were often not updated by the shift workers before beginning new runs.
- The target operators occasionally forgot to write a position change in the paper logbook.
- The control computer messages were not logged to disk until well into the experiment (07 Nov 2001).
- The NMR channel only indicated on which cell measurements were being made - occasionally the bottom target would be monitored while running beam on the carbon target, or other similar situations would occur.

Despite this rather unfortunate list of problems, by combining the sources and doing a great deal of manual checking, accurate information can be obtained.

For the period from the beginning of the experiment until 07 Nov 2001 08:16, target encoder values from the paper logbook were typed into a file `log_info.dat`, which is then converted into `log_pos.dat` by a short awk script in `commands`. After the above date, the control computer logs were parsed for messages indicating target movement by an awk script in `commands` and saved as `pdp_encoder.dat`. These encoder values are then interpreted into target positions by the python script `decode_encoder` and saved to `pdp_pos.dat`. Corrections to `pdp_pos.dat` are then applied by an awk script in `commands`, the data combined with `log_pos.dat`, and the results stored in `position.dat`. The corrections to `pdp_pos.dat` were determined from comparisons with NMR channel information, by looking at anneal times, and from inspections of the paper logbook and EPICS archive.

Since this process is rather involved and prone to possible errors, a cross-check was developed. The target event files were scanned by giving the file `qmeter.control` to the `control` awk script, which produces an output file (`qmeter.dat`) that lists time and Q Meter number for each event. This file

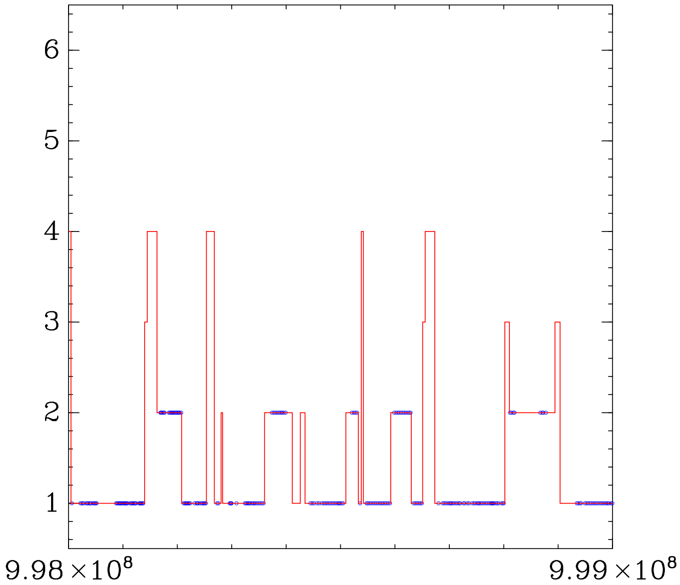

Figure 4: Corrected position information (red lines) plotted against position determined from runs using one Q Meter almost exclusively (blue circles). The x-axis indicates the UNIX timestamp, with each tick mark representing just over one day. The y-axis values correspond to Top, Bottom, Carbon, Empty, N/A, and 10mm Hole, respectively, starting at $y = 1$.

is then used by the python script `count_qmeter`, which groups the data together by run. If one Q Meter (either the top or bottom) was used over 4 times more than the other, then the target was probably in that position during the run, and the result is saved to the file `qmeter_pos.dat`. By plotting the contents of `position.dat` and `qmeter_pos.dat`, any disagreements between the two files should become apparent. Figure 4 is an example of such a plot.

The files discussed in this section are in the `position/` directory, with the exception of the `control` script, which is in the `bin/` directory.

2.6 Insert and Material Changes

Before the experiment, four identical target inserts were prepared, labeled "Insert A" through "Insert D." Also, several bottles of target material were prepared, each with a unique name that usually indicated details of the material's creation as well as the person that created it.

A "target load" refers to one or more bottles poured into a single cell of a target insert. The term "stick" refers to a specific insert filled with zero to two target loads. Gen used four insert/target load combinations, denoted Stick 1 through Stick 4.

The logbook summary file, `logbook/logbook.txt` is the authoritative source of information on insert changes. This information is used in the code that tracks accumulated charge, below. It is also used in making various anneal plots in the `anneal/` directory.

2.7 Target Anneals

The cumulative effect of radiation damage in the target material is a gradual reduction in the maximum achievable polarization. The cause of this reduction is the production of radicals in the material that allow the nuclear spins to relax back to an unpolarized state. During a target anneal, the temperature of the material is raised to allow these radicals to recombine back into nonmagnetic molecules.

Knowledge of when target anneals occurred is necessary for various studies of target material performance. An example would be a plot of polarization vs. accumulated charge since last anneal. Such a plot can indicate fatigue of the material that could be solved by installing a new batch. Understanding of material performance can also highlight which preparation techniques produce the best material.

The `logbook/logbook.txt` file is the authoritative source of information on when anneals occurred. Changes of the target insert also count as anneals, since long term storage in liquid nitrogen has effects similar to an anneal.

Figure 5 shows all anneals performed during the experiment.

2.8 Beam Current

In order to make a proper average of the polarization measurements for a given run, the values must be weighted by the charge accumulated since the previous measurement. In order to calculate the deposited charge, the beam current must first be determined.

BCM1 died about 50 days into the experiment. A comparison of the two BCM readings for the first 50 days was made (see `bcm1_vs_bcm2.commands`) which resulted in the decision to use BCM2 for the current readings for the entire experiment. A second study looked at the non-zero values returned by the BCMs when there was no beam. By making a histogram of all current readings for the entire experiment, a large spike is seen below 18 nA. Because of this, any BCM2 reading below 20 nA is taken as a no-beam measurement, while any value above 20 nA is taken as an accurate current measurement.

The EPICS archive supplies current measurements from the BCMs every two seconds, which is more than adequate for use in performing charge-weighted polarization averages. There were a few times during the experiment when the archiver did not seem to be working, and thus there are holes in the data for those times. There are three such holes that are longer than 2 hours. For them, the BCM reading was extracted from the target event file, which provides measurements about every 13 seconds (see `commands` for how this was done).

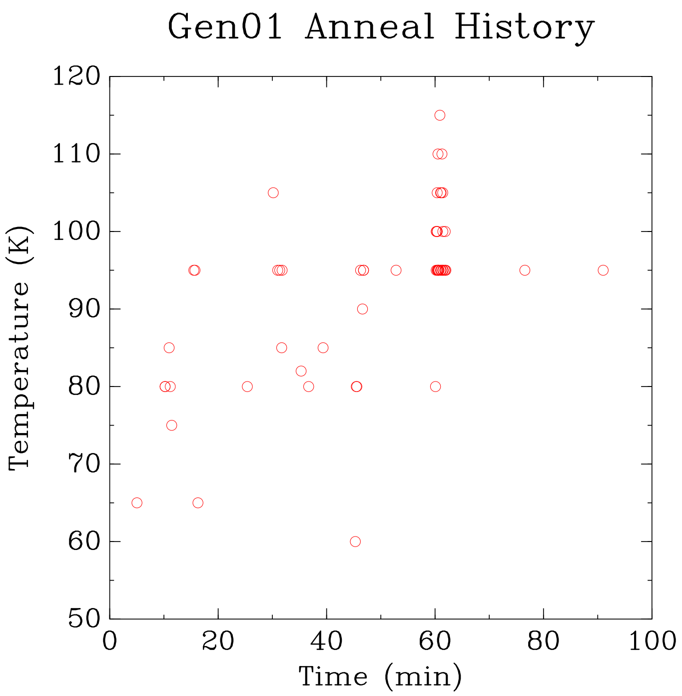

Figure 5: History of all anneals performed during the experiment. Values on the x-axis have a small random value added to help indicate the number of anneals at a given time/temperature combination.

The EPICS archive and event file data were combined, and the 20 nA cutoff applied, by the `determine_current` awk script to create the `current.dat` file.

The files discussed in this section are located in the `current/` directory.

2.9 Beam Charge

The python script `accumulate_charge` calculates total charge since the beginning of the experiment for each of the five target positions. In addition, for the top and bottom polarizable targets, it calculates the charge since the last anneal or insert change. This requires information on beam current (from `current/current.dat`), target position (`position/position.dat`), and anneal and insert changes (`logbook/logbook.txt`). The python library described in Section 2 provides this access.

The accumulated charge is written to a file, `charge.dat`, for each of the timestamps of the NMR measurements. Having the charge known at exactly the same time as a polarization measurement makes the process of performing weighted polarization averages much more easy.

2.10 Magnetic Field

The NMR and microwave system are designed to drive the polarization through the DNP process only for a narrow range of magnetic fields. Thus the magnet was set to a standard field value, 5.003 T, any time the target was being polarized. The frequency at which the deuteron NMR signal occurred provides the most accurate determination of magnetic field, but this technique is obviously only useful while polarizing.

After 14-Aug-2001 18:54:18, the daq system recorded the current in the magnet coils. It should therefore be straightforward to apply a current-to-field ratio to determine the field strength. Unfortunately, small drifts in the calibration of the magnet power supply meant that the magnet current required to place the deuteron NMR signal at a given frequency varied over the course of the experiment. A typical current-to-field ratio is 15.408 A/T. The amount of the drift was less than 0.100 A around the typical setting of 77.085 A.

To create the `field.dat` file, a three categories are used:

1. For times before 14-Aug-2001 18:54:18 there is no magnet current information. If the polarization is above 5% in magnitude, the field is declared to be 5.003 T, if not, 0.000 T.

2. After this time, for currents above 77.000 A, the field is declared to be 5.003 T.

3. If the current is below 77.000 A, the field is the current times 15.408 A/T.

This policy is implemented in the `commands` file, which uses `field.control` as input to the `control` program to extract information from the event files for the second and third case, above.

The files discussed in this section are located in the `field/` directory.

2.11 NMR Signal Analysis

Polarization of a spin species in the target is determined up to a scale factor by calculating the area under that species' NMR resonance signal. The scale factor, or calibration constant, is determined by putting the target into a known state, called thermal equilibrium, in which the polarization can be calculated from the temperature of the material and the applied magnetic field.

The C program `polcalc` is used to measure the area of an NMR signal. By default, it uses information found in the target event file, but an "override file" can also be specified, which allows the user to supply values that should replace the data in the event file for a specified range of timestamps. Typical uses of an override file include the use of a different baseline than the one used online, insertion of an offline calibration constant in place of the online one, and flagging known bad signals so that they may be cut.

The output of `polcalc` is very flexible, and can include the area, polarization, coefficients of the fit, the signal at several stages of the analysis, and even a `TopDrawer`-compatible summary of all analysis steps.

2.12 TE Calibrations

Analysis of the TE calibrations is a more manual process than many other aspects of the target analysis. The summary file of the paper logbook was used to identify when TE calibrations were made, resulting in a total of 110 measurements for entire experiment.

Inherent in the TE process is the acquisition of very small NMR signals, on the order of 0.07% polarization – a factor of several hundred smaller than the enhanced signals measured during production running. Because of this, the NMR signal analysis is very sensitive to stray noise included in the measurement, and the fit obtained from each signal must be examined by a human to classify it as good or containing an unacceptable amount of noise. This was a lengthy process and resulted in a total of 2095 good signals, with from 12 to 36 signals in a given TE measurement.

In addition to inspection of the signals, all baseline signals taken up to four hours before or after the TE were compared with each signal, and the baseline giving the best results chosen.

For each TE, the results of the baseline selection, the elimination of bad signals and any other relevant data not contained in the event file are put into an override file so that `polcalc` can use them when determining the signal areas.

After this step, the TE measurements were put into relevant groups so that they could be averaged. There is a group for each target position of each stick used in the experiment, as well as groups to hold TE measurements that were deemed bad. Some of the reasons to discard TE measurements include:

- Noisy Signals (Stick 1, Top & Bottom)
- Under-irradiated material (Early Stick 3 Bottom)
- Material fell out of cups (Late Stick 3 Top & Bottom)

Groups are defined in a text file in the following format. The first line lists the beginning and ending timestamps for which the group is relevant. The second line is either `cal_const_top` or `cal_const_bottom`, indicating the target position of the group. The third line is a title to use in plots. Each line after that specifies a TE to include in the group. There are 5 required fields, separated by spaces: 1) the name of the override file to use (the TE name), the 2) starting and 3) ending timestamps of the TE, 4) the event filename, 5) the baseline file name. A blank line terminates the definition of the group, after which the file may end, or another group may be defined.

The python script `group_te` takes one or more group definition files as input, and then performs many steps:

- `polcalc` is called to calculate signal areas for each signal of each TE, using the override file created during inspection of the TE, described above.
- The signal areas are piped into `te_calc`, an awk script that calculates a calibration constant from the area, field, and temperature information.

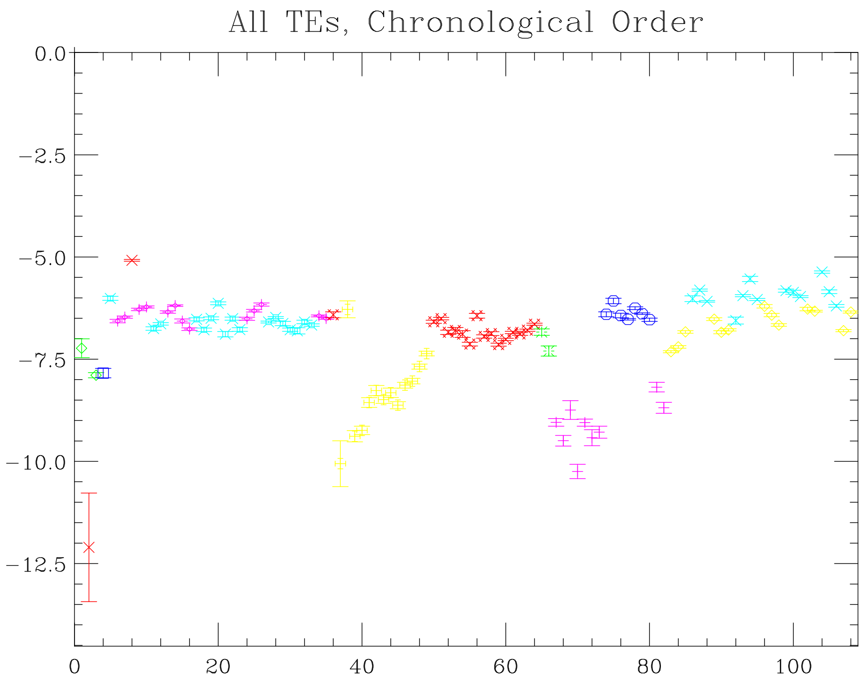

Figure 6: Offline calibration constants calculated from all TE measurements. Constants not used (and the reasons why) are: red X (noisy signals), yellow crosshairs (material not thermalized), magenta pluses (material fell out of cup), blue circles (material fell out of cup).

- Optionally, a plot is made of each signal in the TE.
- The calibration constants from each TE in the group are averaged.
- A summary file, `group_avg.dat`, is created indicating the average and error of each group.
- A `TopDrawer` file, `all_te.top`, is made showing the members of each group plotted along with the average of that group.
- An override file, `offline_cc.override`, is made for use in calculating polarizations of enhanced signals.

Figure 6 shows the calculated calibration constants from all TE measurements. The caption indicates those measurements deemed bad.

The files discussed in this section can be found in the `te/` directory.

2.13 Polarization Measurements

Calculation of the polarization from the NMR signals acquired during the experiment is relatively straightforward. The `polcalc` program is run on every event file, using the `offline_cc.override` file discussed above, and the output saved to a file. In order to facilitate large scans through all the data, an awk script called `control` is used. This script takes as input a text file that specifies the event file, the baseline file, the file in which to place the output, and finally additional commands to pass to `polcalc`, usually specifying the data to be output and the override file to use.

The `pol.control` file specifies a pass through each event file/baseline file combination (some event files require use of more than one baseline file depending on when the target operators decided to create new files). The output is stored in a file called `pol.all`.

One of the event files has a portion that contains all zeros due to a problem with the target daq computer during data taking. The problem started at 26aug01 22:06:41 and continued until 27aug01 13:47:55. To get around this problem, online polarization values are used, and then scaled by the ratio of offline to online calibration constant.

The recovered data and the data from the `pol.all` file are then combined, signals with errors eliminated, the surviving signals sorted chronologically, and the results stored in a file `pol.dat`. Details of this step and the recovery of the online data can be found in the `commands` file. The result is 376,445 polarization measurements.

The files discussed in this section are located in the `pol/` directory.

2.14 Error on Polarization

There are several ways to determine the accuracy of the polarization measurements performed in the experiment. Because the size of the enhanced signals is so large, there is very little error made in measuring and integrating enhanced signals. TE calibrations present a much greater challenge because TE signals are hundreds of times smaller, because errors in temperature measurement affect the predicted TE polarization, and because an error in calculating the calibration constant has a systematic impact on the determination of all enhanced signals using that constant.

A useful tool to investigate possible errors in the temperature and area measurements is the Curie Plot, in which the area of the TE signals is plotted against the inverse of the temperature[2]. If there are no gross errors in the temperature or area measurement, a linear fit to the points should intersect the origin.

For sticks 1, 3 and 4, almost all TE measurements were made in a very narrow range of temperatures around 1.5 K. With stick 2, however, a study was made 31aug01 to 02sep01 in which TE measurements were made over a range

[2]This is a small-angle approximation to the tanh function, since T appears in the denominator of the argument of the tanh function in the TE polarization equations. The actual equation is dependent on the spin of the species being measured.

of temperatures from 1.4 K to 1.9 K. This gives enough lever arm to perform a meaningful fit to the data. As can be seen in Figures 7 and 8, the fit comes remarkably close to intersecting the origin in both top and bottom targets.

As can be seen in the Curie plots, there is scatter in the values of the calibration constants, and the degree of this scatter represents a statistical error in determination of the constants. By normalizing each group and taking the error on its average, an error on the polarization measurement is obtained. Figure 9 shows each group, plotted with its error and Table 2 lists the final error by stick and target cup. The details of this calculation are located in the `commands` file.

Group Name	Error
Stick 2 Top	2.74 %
Stick 2 Bottom	3.49 %
Stick 3 Top	3.30 %
Stick 3 Bottom	4.61 %
Stick 4 Top	4.90 %
Stick 4 Bottom	5.24 %

Table 2: Final absolute errors for each target cell used in production running.

The files discussed in this section are located in the `error/` directory.

2.15 Polarizations by Run

The final step of target analysis is to use the offline polarization measurements to produce average polarizations for each run. This task is performed by the python script `average_runs`. It draws on most of the quantities discussed so far: the target position, the start and stop times of the run, the magnetic field value, the liquid helium level measurements taken during the run, the accumulated charge, and the polarization. It outputs a file, `target_pol.dat`, that contains a summary of all information known about each run. The fields of this file, and their meanings, are:

1. Run number
2. Timestamp of the start of the run
3. Date of the start of the run
4. Time of the start of the run
5. Timestamp of the end of the run
6. Date of the end of the run
7. Time of the end of the run

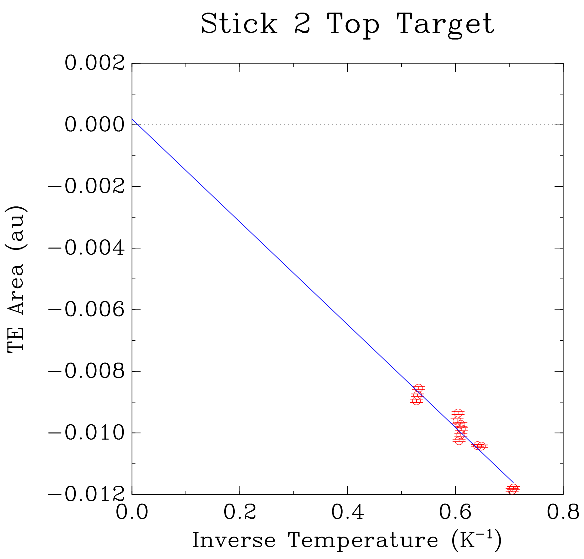

Figure 7: Curie plot for Stick 2, Top Target

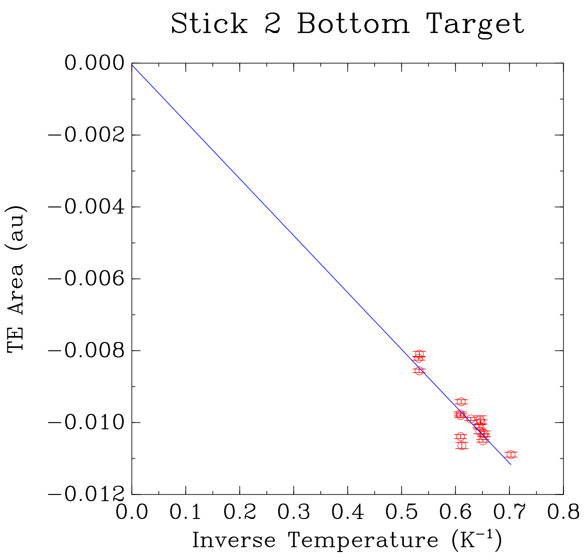

Figure 8: Curie plot for Stick 2, Bottom Target

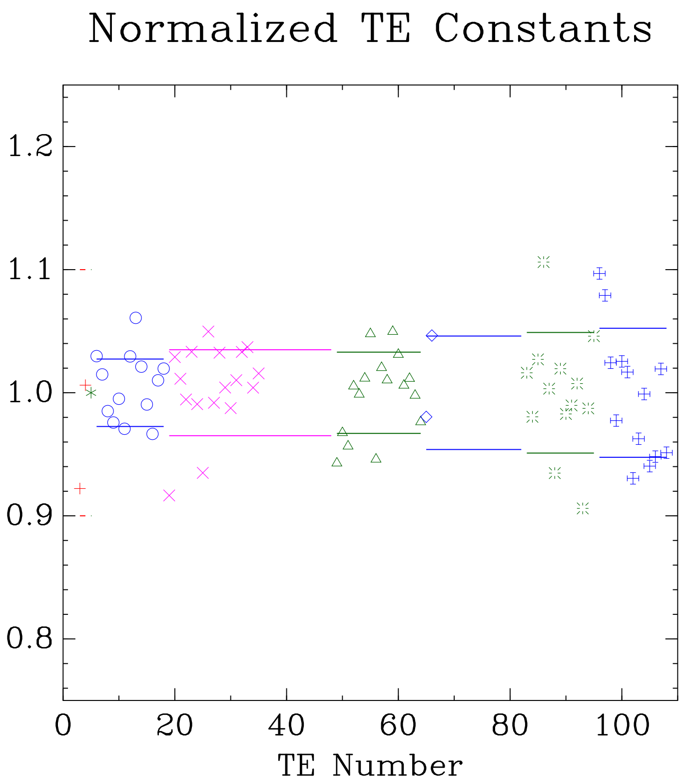

Figure 9: All TE measurements, normalized to the average of their groups. Solid lines indicate error on each group. Plus=Stick 1 Top, Asterisk=Stick 1 Bottom, Circle=Stick 2 Top, X=Stick 2 Bottom, Triangle=Stick 3 Top, Diamond=Stick 3 Bottom, Puff=Stick 4 Top, Crosshair=Stick 4 Bottom.

8. A code indicating the liquid helium level trend during the run. Possible values are:

   - W: the average LHe level was above 50% and fewer than 4% of the measurements were below 50%. This indicates a trustworthy wet run.
   - D: the average LHe level was below 50% and fewer than 4% of the measurements were above 50%. This indicates a trustworthy dry run.
   - w: the average LHe level was above 50%, but more than 4% of the measurements were below 50%. This could be a run during which the tail inadvertently ran dry.
   - d: the average LHe level was below 50%, but more than 4% of the measurements were above 50%. This could be a dry run during which the tail was filled.

9. Target field in tesla
10. Target position:
    - 1: Top polarizable cell
    - 2: Bottom polarizable cell
    - 3: Carbon target
    - 4: Empty target
    - 6: 10mm hole target
11. Charge (in C) on this target position at the beginning of the run
12. Charge (in C) added to this target position during this run
13. Charge-weighted average polarization, in %
14. Absolute error on average polarization

As a crosscheck, a plot was made for each run with the online, offline and average polarizations plotted. By scanning through each of the runs, possible errors in the offline calculations or the averaging could be spotted. The python script `check_runs` performed this task, producing the `TopDrawer` file `check_runs.top`.

Finally, runs were averaged for each "segment" of the experiment, where a segment could be all the data for a particular stick, for a given Q^2 point, or for the entire experiment. The segments are defined in `resources/segments.dat`, and are used by the python script `pol_averages`. The output is the averages (and the total charge comprising them) for positive polarizations, negative polarizations and all polarizations in each segment, stored in the file `pol_averages.dat`. A second file, `pol_averages.graph`, used in conjunction with the `graph` commands in the `commands` file, produces plots of each run's polarization and the average polarization of each segment. The plots for each Q^2 point are included as Figures 10 and 11.

All files discussed in this section are located in the `runpol/` directory.

3 Target Models

While running on Stick 3, a precipitous drop in scattering rate was observed starting at around run 42430. The effect was much more pronounced for the bottom cell, with the rate falling by almost 25% (see HCLog 43620 for further information).

Upon removing the insert, both top and bottom cells were obviously radiation damaged, in a pattern consistent with an anomalous rotation of the insert in a counter-clockwise direction about a vertical axis (HCLog 43671 contains a picture of the damaged insert).

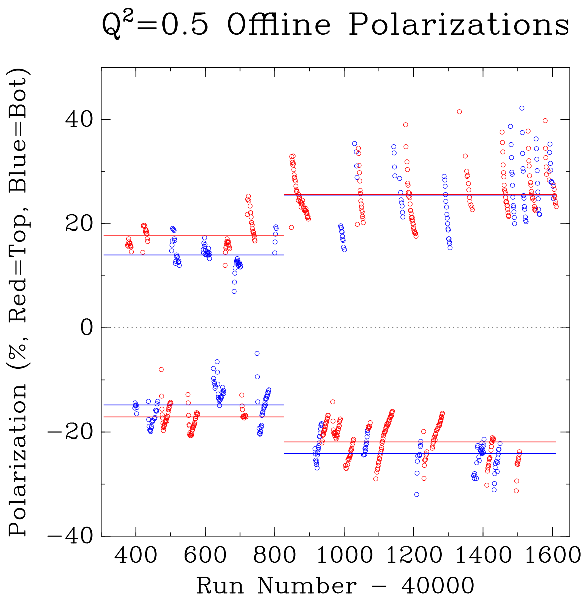

Figure 10: Average polarizations for top (red) and bottom (blue) targets for $Q^2 = 0.5$ runs.

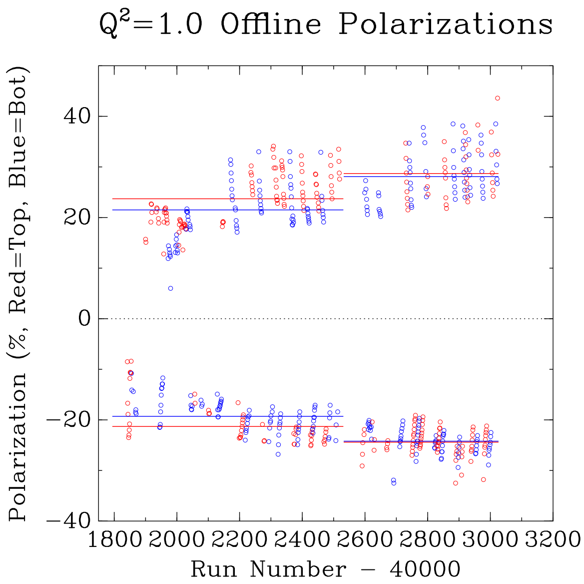

Figure 11: Average polarizations for top (red) and bottom (blue) targets for $Q^2 = 1.0$ runs.

The size of the damage pattern on the cup wall indicated a large rotation angle, large enough to significantly affect the dilution factor. In order to determine the rotation angle and address the impact on the dilution factor, two models of the target were made.

3.1 PovRay Visual Model

The first model was made in a 3D rendering program called PovRay. This program uses a text file of simple commands that describe fundamental geometrical shapes such as cones and rectangles to define a scene. A camera may be placed anywhere inside the world created by the file, as may any number of light sources.

A file `target.pov` was used to define the insert ladder, target cups, microwave horns, 4K shield, beam, and raster volume in the PovRay language. From this file a number of scenes can be rendered, by uncommenting various lines at the top of the file. A makefile takes care of invoking the `povray` program with the proper arguments, and converting the output to JPG format. This is all done by typing `make target.jpg`.

By adjusting the positions and angles of objects in the `target.jpg` file, the scene created was made to match a photograph of the damaged target cup. A rotation of 15 degrees produced the best agreement between the 3D model and the photograph. This provides only crude indication of the correct angle, however. To get a more precise value, it is necessary to consider how the rotated insert appeared to the beam.

When Stick 3 was first installed, it appeared that the target had been translated 4mm to beam right. To correct this, the target was moved 4mm after a few days of Stick 3 running. In actuality the target was rotated, which moved the upstream face of the target to beam right and the downstream face to beam left. The upstream face sits inside a 0.125 inch thick aluminum plate, so it is the hole in this plate that shows up in scattering rate vs. raster position plots. From HCLog 41278 we see that the 10mm Hole target is off by 4mm. Using precise values from the CAD drawing of the insert ladder, this yields a rotation angle of 15.82 degrees, in excellent agreement with the 3D model. If there is a $\pm$1mm error in the raster ADC calibration, this translates to a 3.8 degree error in the angle.

The files used to make the 3D model are located in the `visuals/` directory.

3.2 Geant 4 Materials Model

With an accurate calculation of the rotation angle, verified by a more crude agreement between the 3D model and photographs of the insert, the impact on the dilution factor may be examined.

To calculate the dilution factor, the average length of each type of material traversed by the beam must be determined. To perform this task, a C++ program was written using the Geant 4 libraries. This program allows the user to specify the position and rotation of the target, the raster radius to use and the number of samples to generate. It populates the raster radius with the requested number of sample points in a uniform way (using the "golden section," φ), and then passes an imaginary beam through the target for each of the raster positions. As the beam traverses the target, its path is broken down according to which materials are intersected. Counters are maintained for each material, allowing average path lengths to be computed once all raster positions have been processed.

Because this program allows the position of all elements of the target to be specified, it is not limited to the case of a rotated target. This allows average path lengths to be calculated for all four sticks used in the experiment. Tables 3 through 5 summarize the results for all materials within the HMS acceptance.

The files discussed in this section are located on `jlabl1`. The base directory is `/home/pmm3w/geant4/`. The source code to the program is located in `pmm/`. To compile the program, simply type `gmake`. The executable is `bin/Linux-g++/exampleN01`.

Material	Length
4K Shield	1.179
Drift Space	36.677
Tail Window	0.208
LHe	11.535
Cup Window	0.051
Cup Contents	29.201

Table 3: Average lengths for all materials in Stick 1 and Stick 2. The entry for Cup Contents refers to the combined length of ammonia and LHe within the target cup. The length of ammonia is (Cup Contents)$\times pf$, where pf is the packing fraction. The length of LHe *within the cup* is (Cup Contents)$\times(1-pf)$.

Material	11.94°	15.82°	19.70°
4K Shield	1.874	1.874	1.874
Drift Space	52.860	52.860	52.860
Tail Window	0.213	0.213	0.213
LHe	10.931	11.010	11.045
Cup Window	0.048	0.047	0.046
Cup Wall	0.573	0.637	0.687
Cup Contents	28.384	28.243	28.159

Table 4: Average lengths for all Stick 3 materials for the minimum, central and maximum rotation angles.

Material	Length
4K Shield	1.822
Drift Space	49.927
Tail Window	0.208
LHe	11.535
Cup Window	0.051
Cup Contents	29.201

Table 5: Average lengths for all materials in Stick 4.