CLAS-NOTE 1999-018

CLAS Drift Chamber Calibration: Software and Procedures

David Lawrence
University of Massachusetts

Mac Mestayer
Jefferson Lab

November 17, 1999

Abstract

This document describes the software used in calibrating the CLAS drift chambers. Throughout the course of this document, "calibration" refers to parameters used in the drift velocity function. Other calibration constants used by the drift chambers exist which do not vary significantly for long periods of time. These are not determined by the software packages described here and are thus, outside the scope of this document.

Contents

1 Introduction

This document describes software and procedures for calibrating the CLAS drift chambers. Some detail is paid to describing the features in the software even though some features will not be used by the average DC calibrator. Section 4 is intended as reference while section 5 is really the guide to DC calibration. If your goal is simply to calibrate a run or two, you can skip ahead to section 5 and refer back to the rest of the document as needed. Before jumping into the sofware descriptions though, an overview of some of the terms used to describe aspects of the drift chambers are given followed by a quick overview of the map parameters.

2 Drift Chamber Nomenclature

Here are some brief descriptions of terms used in discussing the CLAS drift chambers. This is intended mainly for someone with little or no experience with the CLAS drift chambers since these terms are used throughout this document.

The CLAS drift chambers can be separated in several ways when describing them. Figure 1 gives a basic idea of how the chambers are arranged.

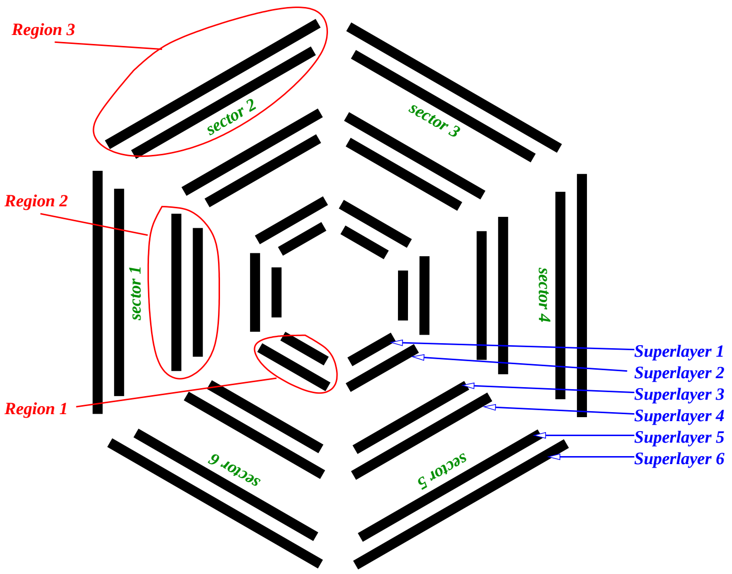

Figure 1: Schematic of CLAS Drift chambers showing how regions and superlayers are named

Each of the 6 sectors in CLAS has an identical set of drift chambers. Each set can be separated by region or by superlayer. Physically, each region is a separate physical volume containing 2 superlayers. Each superlayer contains 6 layers of sense wires (with the exception of superlayer 1 which has only 4 layers). Each superlayer of each sector is calibrated separately for a total of 36 sets of parameters (see sec. 3). There is a mechanism in place which allows the drift time to be scaled differently for each layer, but this is calculated automatically by the calibration software and not explicity determined by the fitting procedure.

When a charged particle goes through the drift chambers, each of the 34 layers is hit. [1] Each hit detected in the chamber is used to determine the particle's track via a least squares fit done inside the CLAS reconstruction program. Two terms are used to describe the distance a charged particle track is from a sense wire:

$DOCA$ - (Distance Of Closest Approach) The distance from the sense wire to the track as determined by time-based-tracking (tbt).

$DIST$ - The predicted distance from the sense wire to the track. This is calculated from the drift time as well as some other parameters.

Additionally, the *residual* is defined as:

$$RESI = abs(DOCA) - abs(DIST)$$

This is also known as the "time residual" because it's sign is determined by the sign of any systematic time shift. The residuals are the primary means of measuring the resolution of the drift chambers. Note also that $DIST$ is positive definite, while DOCA is assigned a sign determined by whether the track passed to the right or to the left of the wire.

3 Map Parameters

The CLAS drift chambers are calibrated by parameterizing the drift velocity function for every superlayer in every sector. The drift velocity function is the relation between the calculated distance of closest approach (DIST) of a particle track to the time it takes for ions created by the particle to drift to the sense wire (drift time). The function's parameters are determined by fits to DOCA vs. time plots produced from CLAS data. Figure 2 shows an example of a DOCA vs. time relationship. The DOCA values are obtained from fits to global tracks (i.e. fits which include all layers) and the drift times are calculated by from the wire's TDC values accounting for fixed cable delays and event_dependant delays such as flight time.

The parameters are stored in the map in the file DC_DOCA.map. The actual functional form used consists of a base function and several correction functions.

[1] Actually. we find an average of 30 hits per time based track. This is mostly due to inefficiencies or holes in the chamber's fiducial volume

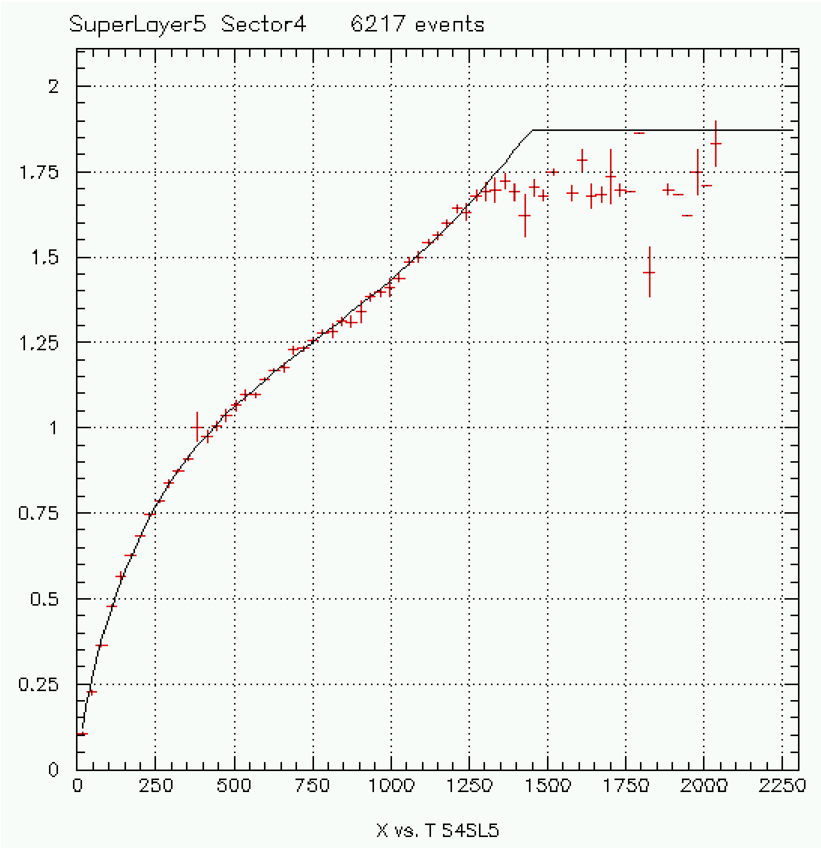

Figure 2: Drift velocity function for superlayer 5 sector 4

The correction functions have been parameterized once by Limin Qin and we currently work under the assumption that these do not change significantly. The DC_DOCA.map file has the following format.

```
Subsystem: Sigma,       nitems: 3
   Item: Region1,  length: 50,     type: float,    narray:3
   Item: Region2,  length: 50,     type: float,    narray:3
   Item: Region3,  length: 50,     type: float,    narray:3
Subsystem: Timewalk,    nitems: 5
   Item: Region1,  length: 10,     type: float,    narray:3
   Item: Region2,  length: 10,     type: float,    narray:3
   Item: Region3,  length: 10,     type: float,    narray:3
   Item: factor,   length: 3,      type: float,    narray:3
   Item: tau,      length: 3,      type: float,    narray:3
Subsystem: t_max,       nitems: 7
```

```
   Item: Sector1,  length: 36,     type: float,    narray:48
   Item: Sector2,  length: 36,     type: float,    narray:48
   Item: Sector3,  length: 36,     type: float,    narray:48
   Item: Sector4,  length: 36,     type: float,    narray:48
   Item: Sector5,  length: 36,     type: float,    narray:48
   Item: Sector6,  length: 36,     type: float,    narray:48
   Item: comment,  length: 1024,   type: char,     narray:46
Subsystem: xvst_par_Sect2,      nitems: 7
   Item: SL1,      length: 24,     type: float,    narray:48
   Item: SL2,      length: 24,     type: float,    narray:48
   Item: SL3,      length: 24,     type: float,    narray:48
   Item: SL4,      length: 24,     type: float,    narray:48
   Item: SL5,      length: 24,     type: float,    narray:48
   Item: SL6,      length: 24,     type: float,    narray:48
   Item: comment,  length: 1024,   type: char,     narray:46
                                  .
                                  .
                                  .
```

There is a separate xvst_par_Sect* Subsystem for every sector. For historical reasons, sector 1 does not follow the naming scheme of the other sectors and instead, has values stored in Subsystem "xvst_params". The same items exist for xvst_params as for the other sectors though.

3.1 Subsystem Sigma

The Sigma subsystem contains error arrays for the DOCA values in units of microns. Each 50 element array contains errors as a function of the DOCA normalized to the cell size.

3.2 Subsystem Timewalk

The Timewalk subsystem contains parameters used to correct for timewalk due mostly to clustering effects in the drift chambers (e.g. minimum ionizing particles can leave a very sparse ion trail which leads to smaller signals and a smearing to larger times). The timewalk parameters are not fit in the normal calibration procedure.

3.3 Subsystem t_max

The t_max subsystem contains values for the maximum drift times for every sector and every layer. In principle, this is the amount of time it takes for an ion created at the edge of a drift chamber cell to drift to the sense wire at the center. The values for Tmax are determined for a selected layer near the center of each superlayer and exptrapolated from there to the rest of the layers in that

1	average local angle (the real local angle has a - sign.)
2	Tzero
3	Tmax_SL (tmax for selected layer (parm 10) in this superlayer)
4	average magnetic field strength
5	effective cell size (unit: 1)
6	para 1
7	para 2
8	para 3
9	para 4
10	layer number for the parameters (i.e. "excluded layer")
11	para 1 for angular dep (tau)
12	para 2 for angular dep
13	para 3 for angular dep
14	spare
15	correction to maximum drift time per layer in %
16	para 1 for R2 B dep
17	para 2 for R2 B dep
18	para 3 for R2 B dep
19	para 4 for R2 B dep
20	spare
21	para 1 for R2 tmax B dep
22	para 2 for R2 tmax B dep
23	para 3 for R2 tmax B dep
24	spare

Table 1: Definition of elements in DC_DOCA.map xvst_par arrays. A separate 24-element array is stored in the map for each superlayer/sector

superlayer. The extrapolation parameter for each sector/superlayer is stored in parameter 15 in the corresponding xvst_par subsystem (see table 1).

3.4 Subsystem xvst_par_Sect...

The xvst_par_Sect* subsystems contain the parameters which determine the shape of the drift velocity function. A list of what each element in the array represents can be seen in table 1.

The four base parameters are kept in elements 6-9. Two different functional forms are used for the base drift velocity function. One for regions I and II and the other for region III:

Region I and II:

$$d = (p1 * \hat{t}) + (p2 * \hat{t}^2) + (p3 * \hat{t}^3) + (p4 * \hat{t}^4) + ((Dmax - p1 - p2 - p3) * \hat{t}^5)$$

Region III:

$$d = (p1 * t) + \left(\frac{(Dmax - (p1 * t_{max}))}{1 - p2}\right)\hat{t}^{p3} - p2 * \hat{t}^{p4}$$

where $p1, p2, p3, p4$ are the parameters from elements 6,7,8,9 of Table 1, D_{max} is the cell size[2], and $\hat{t}$ is the time normalized to t_{max}. Both equations are constrained to hit D_{max} at $t = t_{max}$ (i.e. $\hat{t} = 1$).

Elements 6,7,8,9 of Table 1, therefore, have different meanings depending on which region the parameters are being used for. The values normally determined by the DC calibration program are the four base parameters, the average local angle, the average magnetic field strength, T0, and Tmax. The average local angle and average magnetic field strength are those of the data sample used to produce the calibration. These values are extrapolated from in the reconstruction software.

4 Software

This section gives descriptions of the programs used to calibrate the drift chambers. All of the programs described here have source code under CVS control on the JLab CUE [3]. All can be compiled with a simple *make* command, except where indicated below.

At the time of this writing, there are two tagged versions of DC calibration software in the CLAS CVS repository. These are the old *dc_calib_check*-1-0 and the current *dc_calib_check*-2-0. In this document, I specify the *dc_calib_check*-2-0 tag so persons unfamiliar with CVS will know exactly what to type. This same tag is placed on the dc package of recsis so a working set of executables can always be reproduced. You can always get the latest version of the code by not specifying the tag (i.e. omit the $-r$ *dc_calib_check*-2-0 from the cvs command line). This, however is not recommended since the latest version is often in a state of development and not neccessarily working.

4.1 dc_calib_check

The source for this program can be obtained via CVS on the JLab CUE:

cvs co -r dc_calib_check-2-0 packages/reccal/dc_calib_check

This program requires CERNLIB and Tcl/Tk. These packages exist on the Unix cluster and access to them is usually achieved by a line such as

setup cernlib tcl

in your .cshrc file. *dc_calib_check* takes as input an hbook file which has the appropriate Ntuples in it [4] . This program will provide a graphical interface for fitting the drift velocity function and writing the results to the DC_DOCA.map file pointed to by your $CLAS_PARMS environment variable.

[2] corrected for local angle and scaled by parameter 5 of Table 1.

[3] Common User Environment or Common Unix Environment. Specifically, the jlabs, ifarms, or ifarml machines running either Solaris or Linux

[4] See section 4.2 on trk_mon for instructions on generating an hbook file of the proper format.

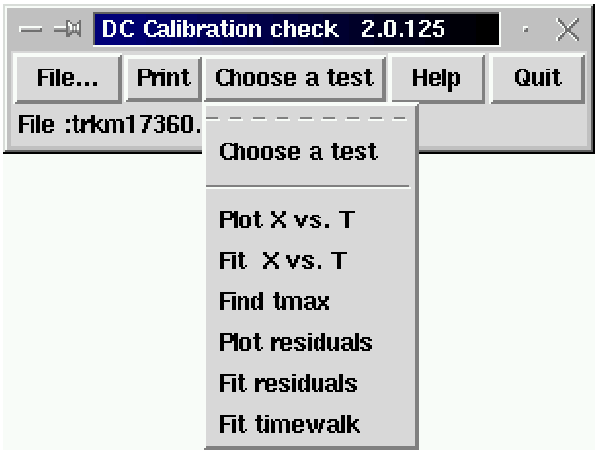

Figure 3: Main window for dc_calib_check

To build this program, simply enter *make* from the dc_calib_check directory. It is important to remember that the *dc_calib_check* executable will still need the tcl scripts which reside in the dc_calib_check/tcl directory. The program will first look in the directory pointed to by the DC_CALIB_PATH environment variable. If the environment variable is not set, it will look in the dc_calib_check/tcl source directory which is compiled into the executable. (So if you delete the source code, it could cause *dc_calib_check* to stop working.)

Start the program by typing *dc_calib_check file.hbook* (where *file.hbook* is the name of an existing hbook file). Figure 3 shows the main control window you should see. The "Choose a test" menu lists the sections available in the program.

4.1.1 Plot X vs. T

In this mode, *dc_calib_check* will simply display data from the Ntuple for the specified sector/superlayer and overlay the function parameterized by the specified function source. All of the functionality of this mode is contained in the "Fit X vs. T" mode described below.

4.1.2 Fit X vs. T

This mode allows you to adjust the calibration parameters through a least squares minimization via MINUIT. Figure 4 shows the main control window for the xvst fitter.

Here are descriptions of some of the features of the xvst fitter:

Function Source *dc_calib_check* keeps a couple of sets of parameters in memory by which to draw the function on the plot. The set that will be written to the map will always be shown if "last fit" is selected as the function

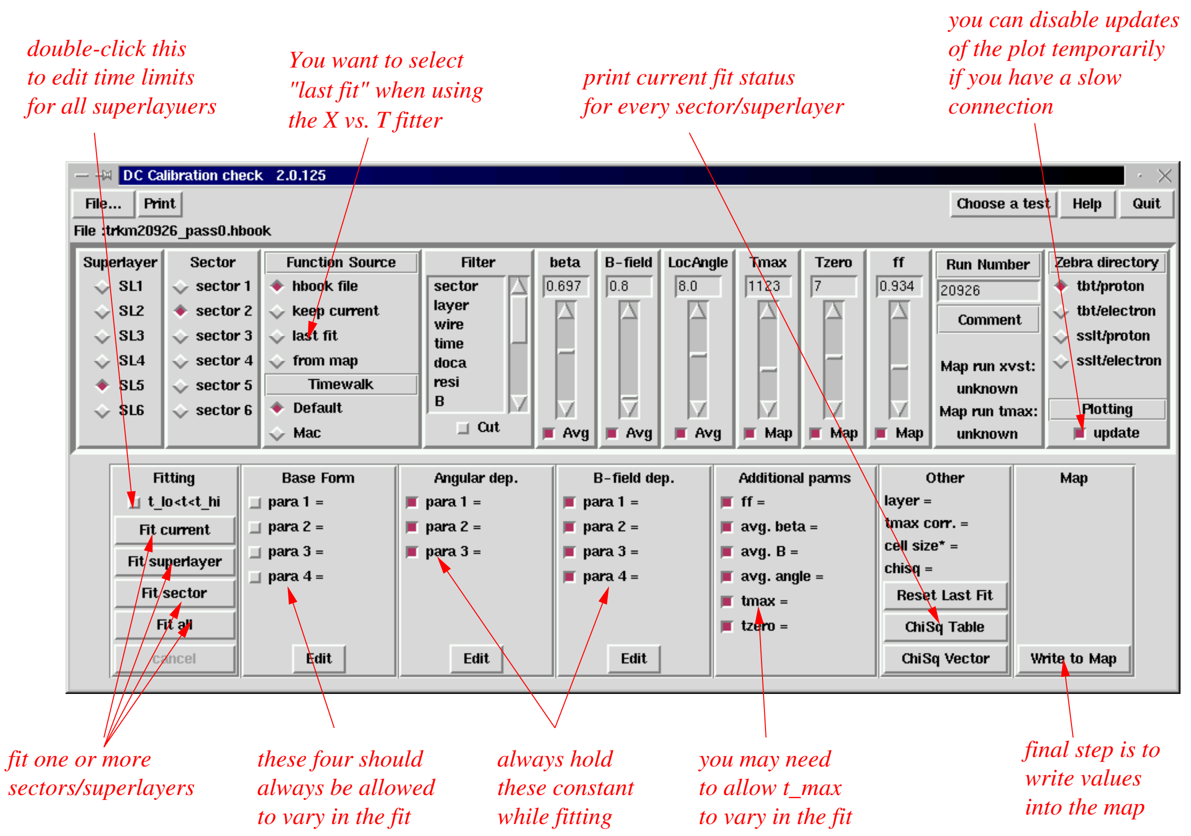

Figure 4: Control window for fitting DOCA vs. drift time

source [5] . If parameters exist in the hbook file, those are copied into "last fit" when the program starts. Otherwise "last fit" will be initialized using the current Map values. Recent versions of *trk_mon* will store map parameters in the hbook file automatically. The same set of"last fit" parameters are used by all modes of *dc_calib_check*.

The four options for function source are:

hbook file - parameters are read from the hbook file

keep current - don't change parameters when a new sector/superlayer is chosen

last fit - the values obtained from the last fit (if any)

from map - paramters are read from the map pointed to by $CLAS_PARMS

$t\_lo < t < t\_hi$ **checkbutton** Due to certain biases and inefficiencies in the data samples and the tracking algorithim, the DOCA vs. drift time spectra are not accurate towards the edge of the cell. Also, due to the nature of how the plot is generated [6] the very low end in time must be excluded from the fit. This can be done by selecting the $t\_lo < t < t\_hi$ checkbutton. Double clicking this button will open a window by which you can edit these limits by superlayer (see figure 5). These values tend to change run period to run period so you will need to look at the DOCA vs. drift time plots in order to set these properly. The upper limit should be to the left of the "kink" (at 1400 ns in 2). The $autofind$ button can be used to have the upper limit in time determined automatically by the fraction of the dirft time spectrum specified in the field next to the $autofind$ button. This works in the same way as "Find Tmax" only it just sets limits on the drift times used for the fit (it does not change t_{max}). The lower limit on drift times used when fitting is always $zero$ when the $autofind$ option is enabled.

By having the $t\_lo < t < t\_hi$ button selected, only those points which will be included in the fit will be displayed upon selecting a new sector/superlayer [7].

Fit Buttons The *Fit Current* button will fit only the currently selected sector/superlayer. The $Fit\ superlayer$, $Fit\ sector$,and *Fit All* buttons will

[5] When using the residual fitter, what is displayed is always the difference between "last fit" and whatever is selected as the function source. You should therefore select either "hbook file" or "from map" when fitting to residuals

[6] The DOCA vs. drift time plots are generated by making a profile histogram of the absolute value of DOCA for each time bin . Once the scatterplot extends below zero, the absolute value makes the average found by the profile histogram higher than it should be making the function curve up at the end rather than extend all the way down to zero as it should.

[7] If you want to display the entire spectrum again, you will need to uncheck the $t\_lo < t < t\_hi$ checkbutton, select "time" in the Filter box and uncheck the *Cut* checkbutton there.

Figure 5: Edit limits on drift times used in fitting X vs. T

loop through all sectors or all superlayers, or all sectors and superlayers respectively. A summary of the χ^2 for every sector/superlayer is printed when either multiple fits are done or the *Chisq table* button is pressed. The χ^2 values for a good fit are typically around 0.5 or less.

Write to Map Once you've fit all of the sector/superlayer combinations, you can write them to the Map via a dialog window(see figure 6) which is opened by pressing the *Write to Map* button. Each of the checkbuttons for which a reasonable fit was achieved will be checked. Two additional checkbuttons exist: *replace current map contents* and *copy sl2 values into sl1*. The *replace current map contents* checkbutton can be unchecked if you want to ensure you do not overwrite a previous entry for this run[8]. The *copy sl2 values into sl1* checkbutton will copy the values you obtained from the fits to superlayer two into those of superlayer one. You may get better results by doing this if you are using the X vs. T fitter[9] . There is an area by which you add additional comments to the Map along with the calibration constants. Remember that the date, your username, the dc_calib_check version number and several other pieces of info are automatically written to the comment. Any additional observations or comments can be entered here and will be stored with the calibration constants in the map..

[8] only one entry may exist in the map for a given run. If you are calibrating an uncalibrated data set, then there should be no map entries for this run number so the state of this box will have no effect.

[9] You generally do NOT want this option while using the residual fitter. The residual fitter does a good enough job fitting to superlayer 1 that you can include the results of those fits in the calibration.

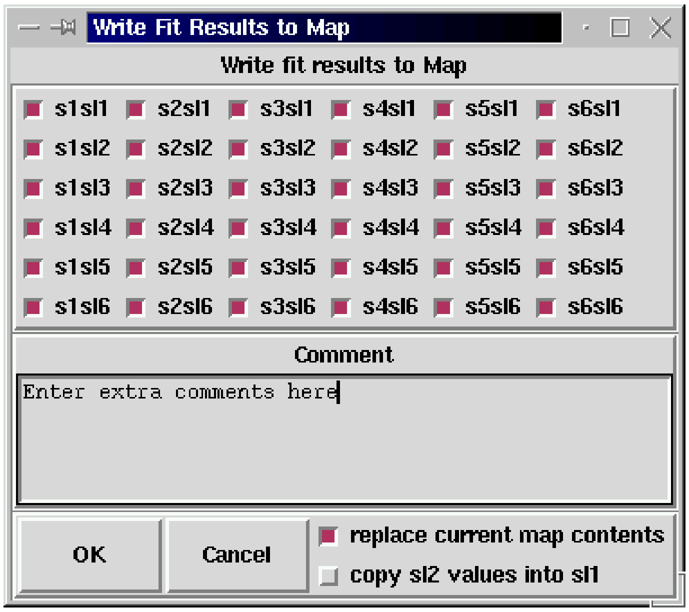

Figure 6: Dialog window for writing fit results to map

4.1.3 Find Tmax

The TMax finder uses drift time distributions to determine the values for tmax[10] as seen in figure 7. The method just looks for the point where the fraction of the total integral of the drift time distribution is equal to some value (typically 97% or 98%). The method is limited by the fact that the last 2% or 3% determines this point and can vary dramatically with low statistics. Any values found using the TMax Finder will be written to the "last fit" area.

Figure 8 shows the tmax finder control window. The "find all" button will find the point of the indicted integral fraction for every sector/superlayer. Unchecking the "update" box will keep the plot in the HIGZ window from updating. This is useful for slow network connections.

4.1.4 Plot Residuals

This mode allows you to plot residuals vs anything else contained in the Ntuple. It is useful for viewing calibration quality and for looking for correlations. Figure 9 shows the residual plotter control window. This mode is mostly useful as a diagnostic tool for drift chamber experts.

[10]tmax is the amount of time it would take ions to drift from the outer edge of a cell to the sense wire in the center of the cell.

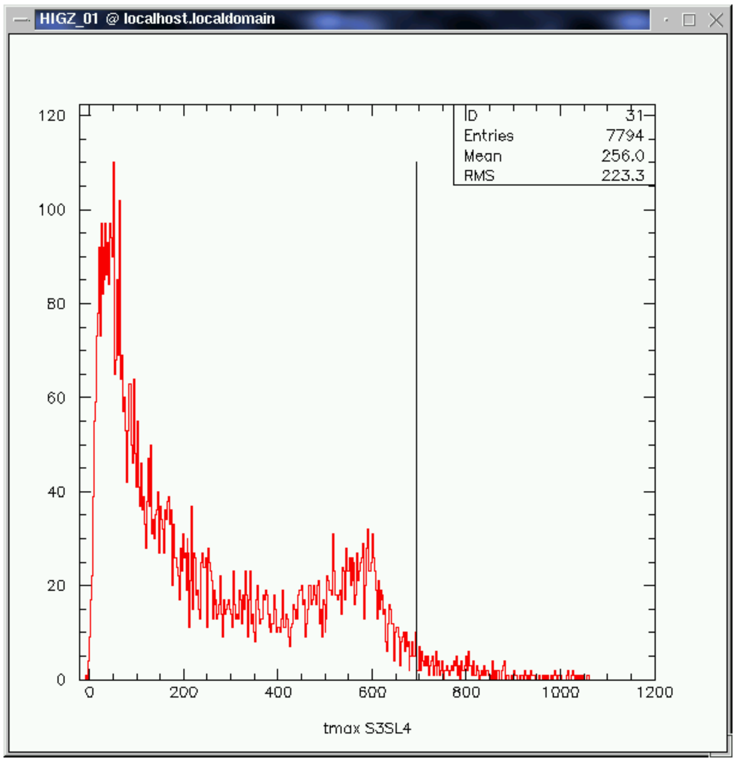

Figure 7: Example of a drift time distribution. For region 1 the bump on the right for the most part merges with the peak on the left.

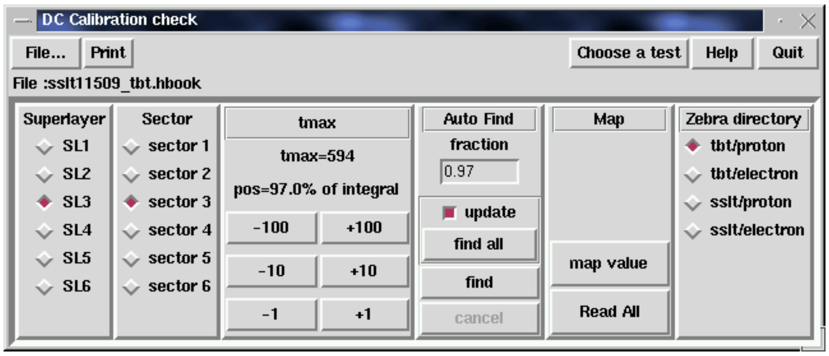

Figure 8: TMax finder control window

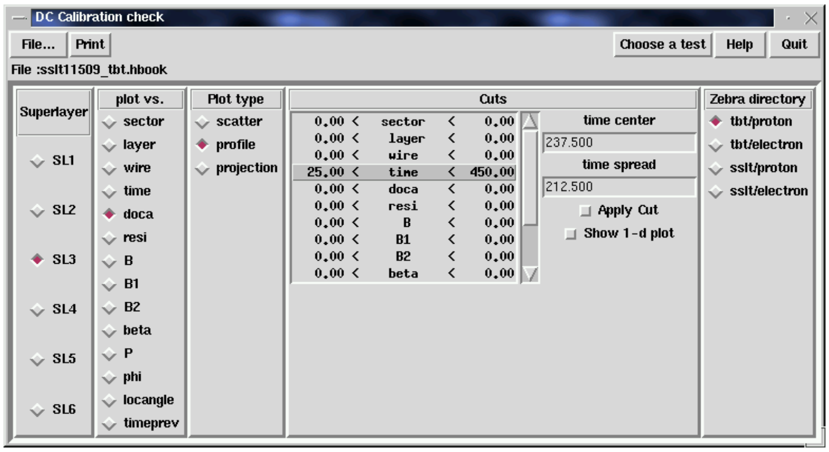

Figure 9: Residual plotter

4.1.5 Fit Residuals

The residual fitter is the currently reccomended method for calibrating the drift chambers. The residual fitter is similar to the xvst fitter except it uses information contained in the residuals to produce a more accurate fit. Figure 10 shows the residual fitter control window. Many features of the residual fitter are shared with those of the X vs. T fitter. The reader is referred to sections 4.1.2 and 5.3.2.

The residual fitter uses the difference between drift velocity functions evaluated at two sets of calibration parameters. This means it is vital for the residual fitter to know the calibration parameters used by time-based-tracking at the time the data sample being calibrated was cooked. It was for this reason that the *trk_mon* program was modified to archive the calibration parameters from the map into the hbook file containing the calibration Ntuples.

Note: The residual fitter tries to do a pretty thorough job of fitting. This means it takes a lot of computer power to run it. Typically, *dc_calib_check* cannot get through all 36 fits [11] using the residual fitter before the process is "niced" on a CUE machine. If you have access to a fast PC on which you can run dc_calib_check, it is recommended that you do so.

The residual fitter has an advantage over the older X vs. T fitter for the following:

The χ^2 value calculated using the X vs. T fitter is defined as:

$$\chi^2 = \sum_{i=1}^{N} \left(\frac{\overline{DOCA(t)} - DIST(t, \vec{p}_f, \vec{\alpha}_{avg})}{ERR(t)} \right)^2$$

[11] 6 sectors times 6 superlayers

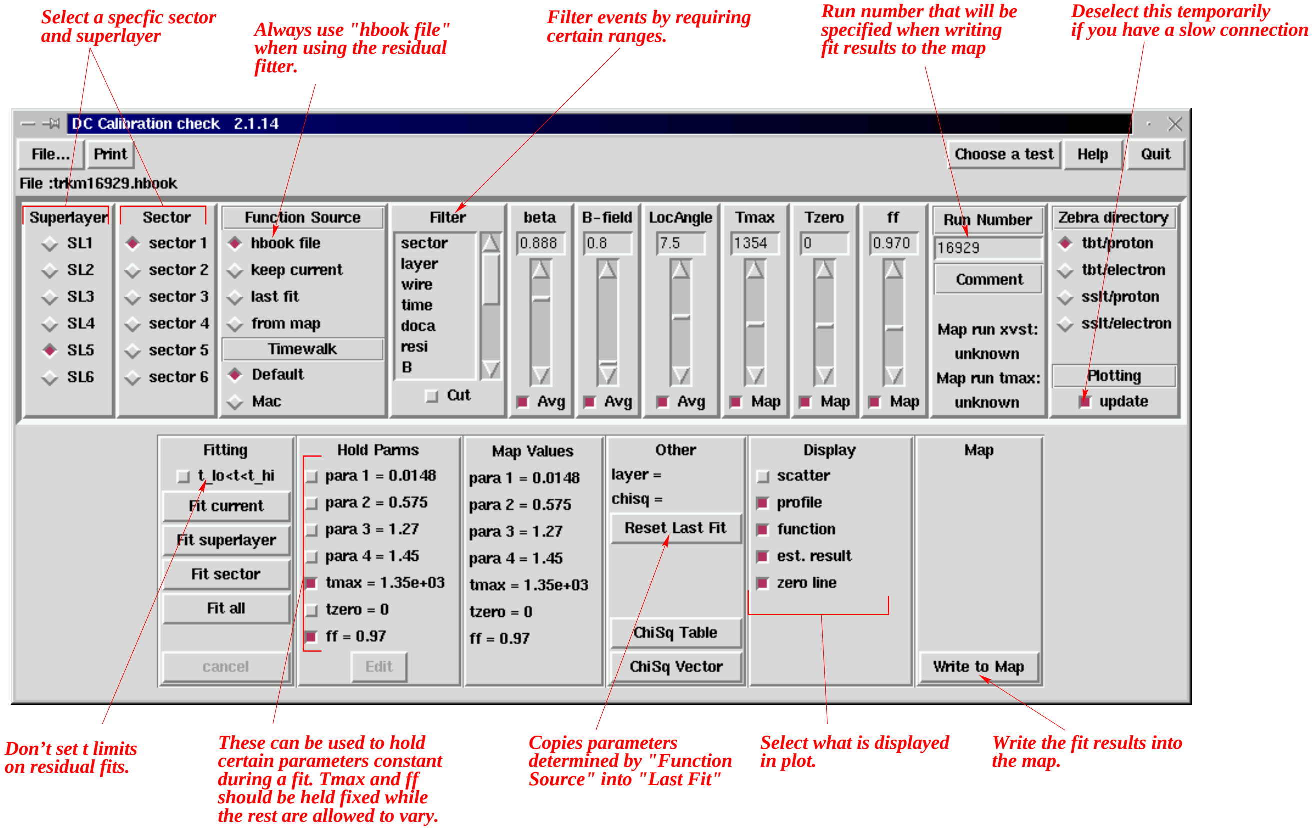

Figure 10: Residual fitter

where:

$\vec{p}_f$ =parmeters (varied in fit)
$\vec{p}_m$ =parmeters (from map)
$\vec{\alpha} = \beta, \vec{B}, locangle$
$\vec{\alpha}_{avg} = \beta_{avg}, \vec{B}_{avg}, locangle_{avg}$

The χ^2 value calculated using the residual fitter is defined as:

$$\chi^2 = \sum_{i=1}^{N} \left(\frac{\left[\overline{DOCA(t) - DIST(t, \vec{p}_m, \vec{\alpha})}\right] - \left[DIST(t, \vec{p}_f, \vec{\alpha}_{avg}) - DIST(t, \vec{p}_m, \vec{\alpha}_{avg})\right]}{ERR(t)} \right)^2$$

or, slightly rearranged

$$\chi^2 = \sum_{i=1}^{N} \left(\frac{\overline{DOCA(t)} - \left[DIST(t, \vec{p}_f, \vec{\alpha}_{avg}) + \left(\overline{DIST(t, \vec{p}_m, \vec{\alpha})} - DIST(t, \vec{p}_m, \vec{\alpha}_{avg})\right)\right]}{ERR(t)} \right)^2$$

The first two terms in the residual fitter χ^2 are identical to those in the X vs. T χ^2. The two DIST terms in the round brackets of the above equation should be interpreted as a correction to the first DIST term. This accounts for the difference between evaluating a function at the averages of its input distributions and averaging over the distribution of a function. Generally, $\overline{DIST(t, \vec{p}_m, \vec{\alpha})} \neq DIST(t, \vec{p}_m, \vec{\alpha}_{avg})$. This seems to be particularly true for the smaller times.

4.1.6 Fit timewalk

At the time of this writing, the timewalk fitter is in the early stages of development and should not be used.

4.2 trk_mon

The source for this program can be obtained via CVS on the JLab CUE:

cvs co utilities/trk_mon

The *trk_mon* program was written by John McNabb, Rob Feuerbach, David Lawrence, and Joe Manak. This program can create the hbook files required by *dc_calib_check*. It takes as input a file containing BOS banks[12]. The file must have been "cooked" by a reconstruction program such as recsis or a1.

[12] The file must contain the following BOS banks: DC0, DC1, TRKS, TBER, and TBTR.

To compile this program, enter *make trk_mon* from the trk_mon source code directory.

This program must be run with the -N option in order for it to create the calibration ntuples. Additionally, it should usually be run with the -K option to filter the events used for calibration. These are the cuts currently defined:

$$\beta < 1.001$$
SL1 localangle in range $-29 : -22$
SL2 localangle in range $-27 : -20$
SL3 localangle in range $-12 : -2$
SL4 localangle in range $-12 : -2$
SL5 localangle in range $+2 : +20$
SL6 localangle in range $+2 : +20$

One of the most attractive features of *trk_mon* is the TBLA bank rebuilder. The TBLA banks hold most of the information from which the calibration ntuples are built. Unfortunately, these banks take a very large amount of disk space to store. Therefore, they are normally not saved when cooking CLAS data. The TBLA bank rebuilder in *trk_mon* can reconstruct the TBLA banks from other tracking information. This means DC calibration ntuples can be made from normal cooked files with no TBLA banks thus, saving you the time and trouble of re-cooking the data a second time while saving the TBLA banks. To enable the TBLA bank rebuilder, the -R flag must be set when running *trk_mon*.

4.3 dc_tool

The source for this program can be obtained via CVS on the JLab CUE:

cvs co packages/reccal/dc_tool

The *dc_tool* program is a kind of Swiss army knife for DC_DOCA.map manipulation. It is a collection of small executables written at one time or another to perform various small tasks. They are combined into this single program for convienience. This program can remove calibrations or copy calibrations from run to run. A list of all the features can be seen by running *dc_tool* with no arguments.

This program is compiled with just a *make* command.

One of the features of dc_tool most commonly used during calibration is to copy parameters from one mapfile to another. If you have a DC_DOCA.map file with a calibration for a specific run which you'd like to copy to another DC_DOCA.map file, use the *copymap* feature of *dc_tool*. Just run *dc_tool copymap* with no other arguments for information on how to do this.

5 Calibration procedure

This section describes the recommended procedure for calibrating the CLAS drift chambers. Though not essential, familiarity with CLAS reconstruction software[13] would be useful.

Before getting started, you need to make sure your $CLAS\_PARMS$ environment variable is set to something appropriate. The nominal place for storing CLAS calibration constants is in files in the */group/clas/parms* directory on the JLab computers. These files have write access for anyone belonging to the *clas* Unix group. This makes things a bit dangerous (which has been proven several times in the past). To avoid some potential problems, each run group should have it's own parms directory (such as /work/clas/production/e1/pass2.x/PARMS) which will be used when calibrating and eventually cooking $CLAS$ data. You should NOT write calibrations directly to */group/clas/parms*. Eventually, when you become satisfied with a particular set of calibrations, you should notify a DC expert[14] to copy the parameters into the general map.

Once a parms area private to your run group is set up, set your $CLAS\_PARMS$ environment variable to point to it. You can always check what your $CLAS\_PARMS$ environment variable is set to with:

printenv $CLAS\_PARMS$

Make sure this is set correctly before continuing.

5.1 Calibration frequency

The first question asked about CLAS DC calibration is usually how often is re-calibration required. Though this has been discussed at length, there does not exist a checklist of quantitative items which will tell you precisely when re-calibration is neccessary. The nature of the map which holds the DC calibrations only allows one calibration per run number. This is not a limitation though since even calibrating every run would be an overwhelming amount of work and largely unbeneficial. As a rule, however, the chambers should be re-calibrated whenever there is any configuration change in the experimental setup.

Additionally, you can check the WWW-based online time histories accessible via the Hall-B web page[15] for constancy of any DC values tracked while the data is being taken. Any sudden shifts certainly warrant a calibration point. Less obvious are gradual changes occuring over a long period of time. A general guideline is to re-calibrate whenever TMax changes by 10,20, or 40 ns for regions 1,2, or 3 respectively. Figure 11 shows an example of the type of such a plot with some possible recalibration points indicated.

5.2 Generating hbook files

The hbook file needed for calibration is made by running *trk_mon* on a cooked data file. You will need approximately 55MB of disk space for each hbook file produced by *trk_mon*. Cooked data files are produced by CLAS reconstruction programs. Currently, there are two such programs, recsis and a1c. The cooked

[13] Recsis or a1c.
[14] davidl@jlab.org or mestayer@jlab.org
[15] http://claspc10.cebaf.gov/TIMELINE/

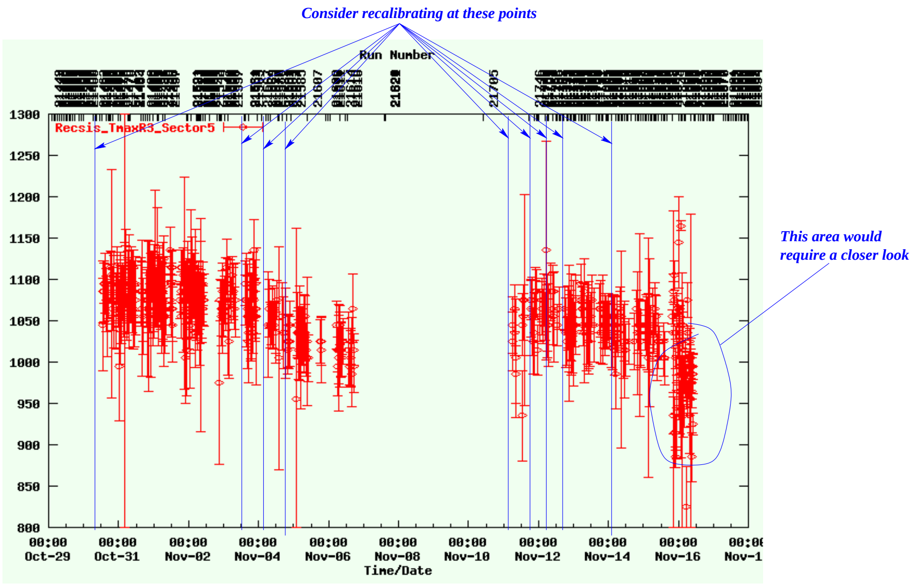

Figure 11: An example of a plot from the online time histories of tmax which can be found at $http://claspc10.cebaf.gov/TIMELINE/$

data files should typically have 200K events or more to be useful for calibrating the drift chambers. You will need to run trk_mon using the $-K$, $-N$,and $-R$ options:

$$trk\_mon - R - N - K - v \quad run020926\_cooked.A00.00$$

The $-v$ option just tells trk_mon to periodically print a line of status info to the screen. The $-R$ option turns on the TBLA bank rebuilder (see section 4.2. The $-N$ option tells it to make the DC calibration ntuple, and the $-K$ tells it to implement the standard calibration filter on events written to the calibration ntuples. When the $-N$ option is used, the trk_mon program will automatically quit shortly after the file has been completely filled. It is normal to see some error messages come up after making calibration hbook files since you generally want ot overfill the file a little to make sure as much data as possible makes it into the calibration Ntuples.

If you are planning to calibrate several runs, you can save a little time by using a small script to automate the procedure. First, create a file containing the names of the cooked files you wish to calibrate [16] like the one shown in figure 12. Next, create a script similar to the one shown in figure 13 [17]. This script can be run on the batch farm using $jsub$ or just run on an interactive farm by

[16] You will only need one from each run. A typical run can have several raw data files (.A00, .A01, .A02 ,...) Each of these can get split at cooking time (.A00.00, .A00.01 ,...).

[17] This script assumes you're using the cache disk

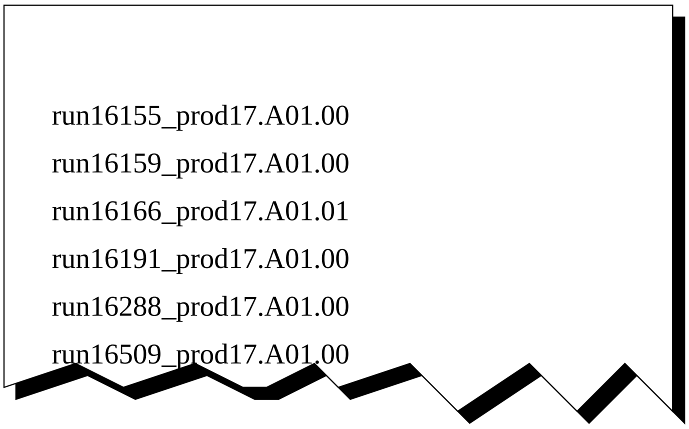

Figure 12: a Sample file containing names of cooked data files

hand [18]. It may take the better part of a day to process a list of 8 files or more depending on system loads.

5.3 Running dc_calib_check

If the previous steps have been completed successfully, then you should have a number of hbook files which can be used to calibrate the drift chambers. Choose one and start the *dc_calib_check* program passing it the name of an hbook file on the command line. The main window will open and some information will be printed in the terminal from which you started the program.

5.3.1 Running the Tmax Finder

Physically, the value of Tmax represents the maximum time it would take ions created at the edge of a cell in the drift chambers to drift to the sense wire in the center of the cell. The value of Tmax is determined using the Tmax finder (see section 4.1.3). Choose *Find Tmax* using the *Choose a test* menu and hit *find all*. The program will cycle through each sector/superlayer and determine a Tmax value for each. This step should only take a minute or so.

5.3.2 Running the Residual Fitter

The recommended method for doing CLAS drift chamber calibration at this time is to use the residual fitter (see section 4.1.5). Select *Fit residuals* from the *Choose a test* menu to bring up the residual fitter control window. Figure 14 shows an example of a residual fit.

[18] You're really not supposed to do run long jobs on the interactive farms, but sometimes you may find yourself in a pinch

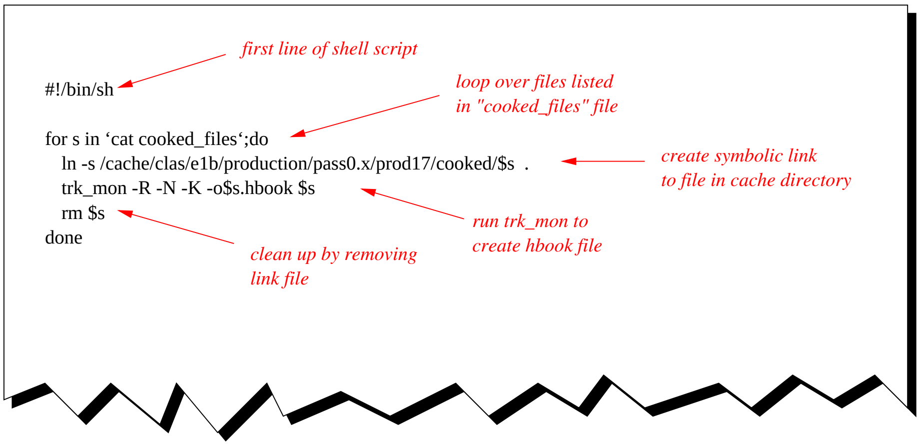

Figure 13: a Sample script for producing hbook files

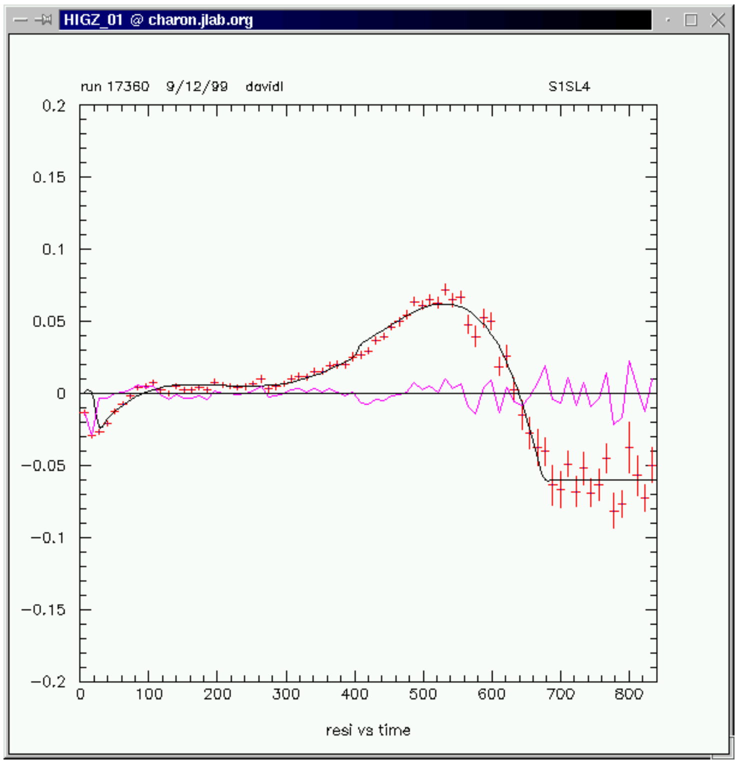

Figure 14: An example of a residual fit. The y-axis is the residuals in cm and the x-axis is time in ns. The black line is fit to the red crosses (a profile histogram of the residuals). The pink line is the difference between the black line and red crosses and serves as a best guess for where the residual will be after cooking.

Make sure that tmax and ff will be held constant during the fits. The checkbuttons next to $tmax = ...$ and $ff = ...$ on the lower half of the screen should be in and reddish colored. Do not select the $t\_lo < t < t\_hi$ button.

You should now just need to click *Fit All* and the program will begin fitting each sector/superlayer in sequence. Be sure not to hit the sector or superlayer buttons while *dc_calib_check* is looping over them since this could confuse it.

Even on a fast machine, this could take up to half an hour so take a break for a little while. When the program has completed all 36 fits, it will print two summary tables of the results. The second one is the important one to look at. It lists the χ^2 values obtained for each fit. Any that are above a certain threshold[19] will be red to draw your attention to them. It is normal for the average χ^2 to vary from superlayer to superlayer. Look at sectors with the largest χ^2 values for each superlayer by selecting them in the control window. Make sure the fit looks reasonable by eye (see fig 14). Remember that there are limitations in the functional form which prohibit it from ever fitting certain shapes very well.

5.3.3 Writing to the Map

Once you are convinced that all the fits are reasonable, click on the *Write to Map* button. This will bring up a dialog described in section 4.1.2. Enter any additional comments and click on the *OK* button.

If you are not working on a JLab CUE machine or you set your $CLAS_PARMS environment variable to use a different map file than the global map[20], then you'll need to copy your new calibration parameters to the global map. The *dc_tool* program can do this quite easily. See section 4.3 for information on using *dc_tool*.

If everything went well and you wrote new calibration parameters to the map, then you can send an e-mail to me at *davidl@jlab.org*. I will re-run the script which produces the web page summarizing the current calibration status of the drift chambers. This page can be seen at:

http://www.jlab.org/~davidl/dc_calib/dc_calib_html/dc_calib_status.html

5.4 Checking Calibration Quality

The quality of the calibration can only truly be checked by recooking the data using the new calibration parameters. Once the data is re-cooked, you'll need to produce new hbook files using *trk_mon*. The *trk_mon* program produces some useful output besides the calibration ntuples. By examining the hits per track and the tbt/hbt ratios reported by *trk_mon* before and after calibration, you can get an idea of how much things have improved.

An attempt at making a standard for comparing different DC calibrations from different run periods is manifest in a small script called *dc_summary.kumac*.

[19] at the time of this writing, the threshold is set at $\chi^2 > 3$.
[20] /group/clas/parms/Maps/DC_DOCA.map

Despite the name, this is really a shell script. It will create a temporary kumac file and run paw in batch mode on it in order to produce a table summary in a postscript file. The script is now stored with the *dc_calib_check* source code so you should have gotten it when you checked that out. Just give the script the name of a hbook file on the command line and it will produce a postscript file by the same name. Figure 15 shows an example of a summary sheet.

For reference, your calibration should result in tracking which averages more than 29 hits per TBT, has average tracking χ^2s of 3.0 or less, and should have a tbt/hbt ratio of around 0.70 or more.

Once you are satisfied with a set of DC calibrations, send email to the DC expert in charge of maintaining the general map[21]. Make sure you include the location of the Map file where your calibrations are and the run numbers so the calibrations can be copied into the map in */group/clas/parms*.

If you run into any difficulties or have any questions not answered by this document, then please contact David Lawrence (*davidl@jlab.org*) or Mac Mestayer (*mestayer@jlab.org*). Happy Calibrating!

Special Thanks to the numerous persons who contributed to the work on which this document is based. These include Rustam Niyazov, John McNabb, Rob Feuerbach, Franz Klein, and Liming Qin to name just a few.

[21] *davidl@jlab.org* or *mestayer@jlab.org*

Summary for trkm20926_pass1.hbook

Time residual sigmas, narrow (in microns)

	SL1	SL2	SL3	SL4	SL5	SL6	avg.
Sec1	291.54	315.567	300.359	281.663	267.365	272.144	288.107
Sec2	294.663	298.159	297.393	287.751	272.203	290.579	290.125
Sec3	300.292	285.985	310.126	290.286	293.637	294.778	295.852
Sec4	306.068	294.515	288.437	288.692	260.448	271.051	284.868
Sec5	287.372	302.01	287.343	298.346	265.438	276.813	286.22
Sec6	296.438	315.219	294.875	320.412	288.711	284.382	300.005
avg.	296.062	301.91	296.423	294.525	274.633	281.623	290.863

Time residual sigmas, wide (in microns)

	SL1	SL2	SL3	SL4	SL5	SL6	avg.
Sec1	803.242	762.153	765.084	728.148	754.136	746.424	759.865
Sec2	841.462	741.131	773.699	730.213	773.254	790.628	775.063
Sec3	854.62	681.136	825.244	753.355	816.613	810.604	790.262
Sec4	838.776	717.064	751.111	736.541	747.483	738.632	754.933
Sec5	831.866	810.935	758.969	774.516	751.896	776.516	784.118
Sec6	817.249	746.547	738.569	772.675	792.306	770.821	773.03
avg.	831.203	743.162	768.778	749.243	772.615	772.27	772.878

Time residual means (in microns)

	SL1	SL2	SL3	SL4	SL5	SL6	avg.
Sec1	-36.1288	-85.6705	-21.0494	-64.0706	-106.787	-153.607	-77.8855
Sec2	-97.1576	-65.9982	-52.2337	-76.7915	-106.024	-141.322	-89.9213
Sec3	-77.0752	-71.0241	-53.1618	-85.1225	-106.37	-110.264	-83.8363
Sec4	-108.905	-80.4843	-38.1067	-67.3345	-110.103	-131.435	-89.3948
Sec5	-105.432	-77.5214	-51.9719	-84.8368	-120.222	-128.842	-94.8043
Sec6	-99.0706	-72.3749	-42.4861	-75.0821	-95.1332	-135.437	-86.5973
avg.	-87.2948	-75.5122	-43.1683	-75.5398	-107.44	-133.484	-87.0732

Hits per TBT

	SL1	SL2	SL3	SL4	SL5	SL6	avg.
Sec1	3.38147	4.64863	5.48171	5.2779	5.41117	5.11402	29.3149
Sec2	3.42389	5.41682	5.41211	5.31827	4.85946	4.98054	29.4111
Sec3	3.71927	5.68747	5.48696	5.50858	5.26102	5.15674	30.82
Sec4	3.82114	5.63762	5.61788	5.56261	5.54833	5.34082	31.5283
Sec5	3.82792	5.55614	5.64368	5.53704	5.44165	5.18082	31.1871
Sec6	3.8507	5.68369	5.70544	5.39455	5.10156	4.83073	30.5667
avg.	3.67072	5.4384	5.55797	5.43317	5.27052	5.1006	30.4713

Avg. Chisq per DOF

Sec1	Sec2	Sec3	Sec4	Sec5	Sec6	avg.
2.61709	2.77624	2.7718	2.46141	2.61541	2.89732	2.68987

Figure 15: Example of a summary sheet produced by *dc_summary.kumac*