ESPACE

ESPACE

USER'S

GUIDE

Event Scanning Program for hall A Collaboration Experiments

Eddy Offermann
February 28, 1997

Contents

Chapter 1

General information

The program ESPACE can analyze experiments performed with one or two high-resolution spectrometers. In this manual we will discuss the implementation specific to the setup in hall A, CEBAF.

ESPACE can filter, histogram and/or calibrate variables of the experimental setup while applying conditions on the incoming data. The variables range from raw detector signals, like the value of a scintillator TDC channel to much more elaborate ones like the coordinates of the reaction point in the target.

The program interface is supplied by the KUIP CERN-library and, therefore, the program has a PAW-like look. Details about this interface can be found in the CERN documentation of [1] and [2].

1.1 Program concept

Let me try to explain the history and the concepts behind the analysis program ESPACE [1]. The program is build up in three layers where each layer on top of the lower one adds more information specific to the experiments.

The first layer performs operations on variables defined in higher layers for each event input. These operations can be histogramming, filtering, evaluating conditions and/or accumulating data for an optimization cycle. The routines are collected in the COOLHANDS [2]-library. The first version of this package was written by Joe Mandeville [3]. In COOLHANDS it is possible to define 'on the fly' new variables as a function of existing ones with a FORTRAN expression evaluator contained in the FCALC-library (also written by Joe).

In figure 1.1 a diagram shows how the data flows from a COOLHANDS perspective. After initialization in COOLHANDS and possibly in a routine supplied by the upper ESPACE-layer, commands supplied through the KUIP interpreter are executed. The most important ones are:

- definition of an ntuple or histogram through the `save` command which triggers a call to the MBOOK4D library, an interface to the CERN HBOOK-library.

[1] ESPACE: Experiment Scanning Program for hall A Collaboration Experiments

[2] COOLHANDS: Command Oriented Off-Line Histogramming AND Scanning program.

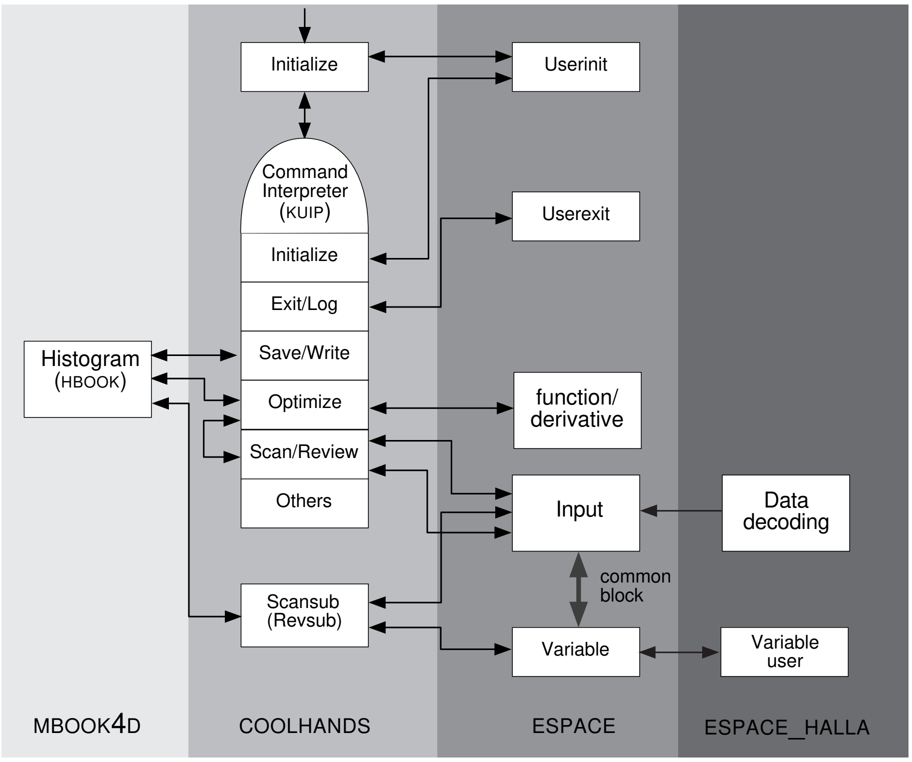

Figure 1.1: Diagram showing the flow of data around the COOLHANDS library.

- the `scan` command which calls in the ESPACE library the the INPUT routine. It opens all the necessary files and calls on its turn the raw data decoding routine in the top layer , which is experiment specific (Here at JEFFERSON LAB it is a set of CODA routines [4]). After receiving data, INPUT will call the routine SCANSUB, the central routine of COOLHANDS. This routine evaluates all the logical definitions and applies than the filters and stores the data than through MBOOK for histogramming and optimization. To analyze the logical definitions and store the appropriate values, SCANSUB has to call the ESPACE- routine VARIABLE where the different variables have been defined.

The story looks a bit different from an ESPACE-perspective. After the `scan` command has been issued, COOLHANDS calls the ESPACE routine INPUT which takes over til the data file has been analyzed. INPUT opens the data file and some other necessary files and starts a loop in which for each data event the routine PROCESS_DATA is called. This is the center piece of the two-spectrometer analysis in ESPACE. It gets from the experiment-specific decoding routine (ESPACE_HALLA all the detecor information.

PROCESS_DATA will call now for each spectrometer the routine PROCESS_SPECTROMETER which digests all the detector information and for instance, evaluates all the tracks through the wire chamber package. The track that made the trigger signal is determined and traced back through the magnet to the spectrometer entrance. Next, the reaction point in the target is determined with the vertex information of both spectrometers and the information of the beam position monitors for the incoming beam (REACTION_Z. Having the reaction point and the target geometry, corrections can be made for the energy losses of incoming and outgoing particles in the target. This is coordinated by the routine CALC_KINEMATICS, together with calculation of numerous kinematical

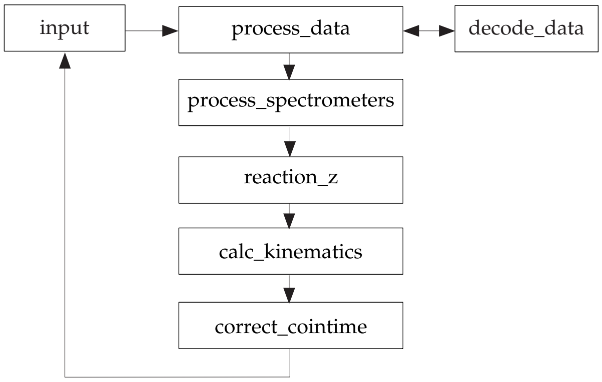

Figure 1.2: Diagram showing the flow of data in the ESPACE library.

variables (KINEMATICS-library). After correcting the coincidence time between the two spectrometers for flight-path differences wrt. the central ray, (CORRECT_COINTIME) control is given back to the routine INPUT which will call the COOLHANDS-routine SCANSUB to evaluate all the logicals, histograms, ·.

1.2 How to start ?

First make sure that the following lines are added to your login file (`.login` for the `csh`):

```
setenv HALLA_DIR /work/halla/software
setenv LOC_HALLA_BIN $HALLA_DIR/bin//$OSNAME
setenv PATH $LOC_HALLA_DIR/bin:$PATH
```

Execute the login file by giving the command `source .login` or logging out and in again. Now the necessary logicals have been defined and you are running the current version of ESPACE in the directory `$HALLA_DIR/bin`.

If you want to have your own version of the analysis software, follow the steps described in the document:
`http://www.cebaf.gov/halla/calibrations/quick_install.html`

To start a scanning session give the command `espace` or `espace -m` to get the X-window equivalent version. The prompt `espace >` has appeared now and commands can be typed in.

Before explaining the program in detail, an example is shown of how to analyze an ^{12}C(e,e′) run. Below is listed the contents of a macro file for the KUIP interface (known as KUMAC file). Instead of typing in each command, one can create a macro file containing all commands and then execute this file by giving in ESPACE the command `exec file_name`.

```
*
******************** define FILES **********************

set/file/output carbon_1434.hbook
set/file/database db_845_e
set/file/header hdr_1434
```

```
set/file/detmap detmap.config

******************** define CUTS ***********************

define/cut/1d spec_e.th_tg-1 -0.03 0.03
define/cut/1d spec_e.ph_tg-1 -0.06 0.6
define/cut/1d spec_e.p_kin-1 825.0 850.0
define/cut/1d spec_e.gas.adc-1 0.0 150.0
define/logical accept_e spec_e.th_tg-1&&spec_e.ph_tg-1

******************* spectrometer E ******************

set/hitbits (+spec_e.gas)
set/auto_window on
set/spectrum/bins/x1 1
spectra/save spec_e.s1.hp_r,spec_e.s2.hp_r
spectra/save spec_e.s1.adc_r,spec_e.s1.adc_l, _
             spec_e.s2.adc_r,spec_e.s2.adc_l accept_e
set/ntuple on
spectra/save spec_e.s1.adc_r/spec_e.s2.adc_r accept_e
spectra/save spec_e.s1.adc_l/spec_e.s2.adc_l accept_e
set/auto_window off
set/spectrum/bins/x1 300
set/spectrum/window/x1 -0.15 0.15
spectra/save spec_e.th_tg-1,spec_e.ph_tg-1 accept_e&&spec_e.p_kin-1
set/spectrum/bins/x1 100
set/spectrum/window/x1 -0.025 0.025
spectra/save spec_e.y_tg-1
set/spectrum/bins/x1 250
set/spectrum/window/x1 825.0 850.0
spectra/save spec_e.p_kin-1 accept_e&&spec_e.gas.adc

******************* scanning and output ****************

file/scan carbon_1434.dat FIRST=1 LAST=10000
spectra/write
quit
```

In the first few lines the names of the histogram, database and header file are given. The database contains constants of the spectrometer and detector package, like spectrometer tracing tensors and wire positions in the VDC's, see B. The header file contains information specific for the data run to be analyzed, like target material and spectrometer field settings, see A.

The next lines contain the definition of *cuts* on variables calculated with the detector package signals accumulated by the data acquisition code CODA [4], like the ADC value of the gas Cerenkov in the electron spectrometer (`spec_e.gas.ecer`) or variables derived from CODA information, like the reconstructed angles of the scattered electron, θ_{tg} and ϕ_{tg}. Two of these logical definitions are combined by *anding* (`&&`) them into a new one called `accept_e` defining the solid-angle acceptance of the electron spectrometer.

In the next part the histograms and ntuples are defined through `save` commands of variables. Some of these `save`'s are performed under the condition that a logical is true. (Compare `spectra/save spec_e.y_tg-1` and `spectra/save spec_e.p_kin-1 accept_e`). The variables are referred to through a tree-like structure. For example the `spec_e.s1.adc_l` string points to the ADC value of the left channel of the S1 scintillator of the electron spectrometer, `spec_e.y_tg` points the reconstructed y-coordinate at the target of the electron spectrometer. The next section shows which variables specific to a certain spectrometer detector or just spectrometer can be accessed.

Chapter 2

Variables

As was pointed out in section 1.1, the COOLHANDS subroutine SCANSUB evaluates logical expressions and increments histograms by obtaining the variable values through the routines VARIABLE in the ESPACE library or VARIABLE_USER in the experiment-specific ESPACE_HALLA part. The different locations where the variable can be defined is not important for the usage of the program, however, for maintenance/changes of the program care should be taken where one installs them. If the variable is of interest to may users, it should be installed in the ESPACE library. However, if it is only useful for testing purposes, installation should be done in ESPACE_HALLA.

In the first section an overview is given of the variables that have already been installed in the ESPACE library, followed by a section giving instructions for implementing new ones in the code. Finally, it is shown how new variables can be defined on-the-fly in a command file.

2.1 Units of variables

Before discussing which variables are installed in ESPACE and how to install new ones, a few words about the unit definition in ESPACE. We have tried to express variables that are going in and out of the program in one consistent set of units, as shown below:

QUANTITY	PURPOSE	UNITS
length	 target parameters	meters centimeters
angles	 vertex angles at target and focal plane wrt spectrometer central axis	degrees dimensionless
momenta		MeV/c
energy		MeV
field		kG
time		seconds

However, from the table it is clear that for length and angle variables, besides meters and degrees, two othe units are used. In case of length, target parameters like thickness or reaction point coordinates are expressed in cm. The angles of the particle vertex as measured at the target or in the detecor hut are actually tangents of angles, so dimensionless. Since the angles are fairly small, the tangents of an angle is close to the angle when expressed in radians.

2.2 Overview of pre-defined variables

In this section we want to give an overview of the variables that are defined in the program and, therefore, can be histogrammed and used in the definition of logicals. Most of the variable names are preceded by a string indicating to which spectrometer and/or detector they belong. Figure 2.1 gives an overview of the variables and a tree-like structure guiding the user how to compose the variable string.

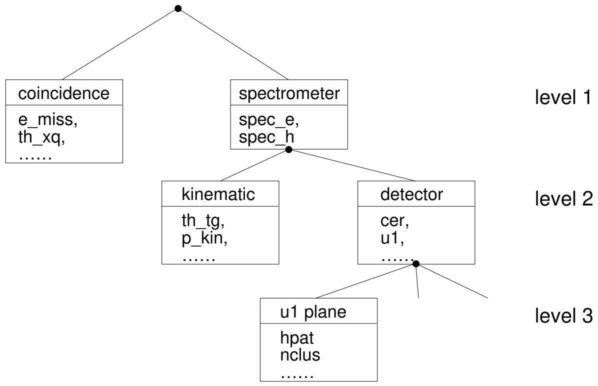

Figure 2.1: Structure guiding the user how to compose the variable string.

Starting at the top of the tree, the user has two options: choose a variable which is a function of the output of both spectrometers or select a variable linked to one of the spectrometers. In the first case there is no string, so the missing energy is simply referenced by `e_miss`. However, in the second case the string will start with the spectrometer name: `spec_e` or `spec_h`. The black dot at the bottom of the box indicates that the user should go at least one level deeper to have a valid variable name. The next choice is between a kinematic variable or a variable connected to one of the detectors, f.i. `spec_e.p` (the momentum of the particle detected in the electron spectrometer) or `spec_e.s1...` . In the last case the user has to go one level deeper, indicating the variable of that detector.

Below we will discuss three tables in which the variables are grouped according to the number of levels in the variable string. We start with the variables giving information about the CODA properties of the event.

variable	class	type	sub	dim	description
`clock*`	i	s	no	sec	time since CODA start
`run_number*`	i	s	no		CODA run number
`event_type*`	i	s	no		CODA event type
`event_nr*`	i	s	no		CODA event number
`cur_cav*`	r	s	no		current (unscaled) in one of the beam cavities

Table 2.1: Level one variables, giving CODA information.

For each variable the parameters `type`, `size` and `sub` have been defined. The meaning of the parameters is the following:

`class` : The `class` of a variable is either `real` or `integer`. Its value will determine the result of the command `set/spectrum/bins`. In case of a `real` variable, it means the number of bins of the histogram, otherwise the number of channels that will be grouped together.

`type` : The `type` of a variable is either `scalar` or `array`. If a variable has a `type` value of `scalar`, a histogram of this variable can only be incremented once for each CODA event. However, an `array` variable can be incremented multiple times. An example of an `scalar` variable is `event_type`. `tdc` (TDC values of drift times in a VDC wire plane.) is an `array` variable and will contain the drift time of each wire exited by the tracks passing through this particular wire plane.

`sub` : Finally one can add to some variables a `sub-detector` number in order to be more specific when defining a cut or a histogram of a variable. for example, one could histogram the `tdc` times of wires 100 up to 200 of the `u1` plane in spectrometer `spec_e` with the command `spectra/save spec_e.u1.tdc[100;200]`. The command `spectra/save spec_e.u1.tdc` would have saved the `tdc` spectrum of the whole `u1` plane. Table 2.2 gives a summary of what the meaning of the sub detector number is for the different detectors.

detector	**sub-detector**
wire plane	wire number
scintillator	paddle number
cherenkov	mirror number
shower	block number

Table 2.2: Meaning of sub-detector numbers for the variables of different detectors.

In the next table (see 2.3) the variables are listed which are reconstructed with two spectrometers.

Table 2.3: Level one variables, involving both spectrometers.

variable	**class**	**type**	**sub**	**dim**	**description**
`e_miss`	r	s	no	MeV	missing energy
`m_miss`	r	s	no	MeV	missing mass
`p_miss`	r	s	no	MeV/c	missing momentum
`tc`	r	s	no	TDC chan	coincidence time
`tc_cor`	r	s	no	TDC chan	coincidence time corrected for path length difference wrt the central rays
`th_xq`, `ph_xq`	r	s	no	deg	spherical angles between momentum of particle x and the momentum transfer in the reaction A(e,e'x)B. z-axis is defined by the momentum transfer and the x-z plane by the electron scattering plane.
`th_bq`, `ph_bq`	r	s	no	deg	spherical angles between momentum of particle B and the momentum transfer in A(e,e'x)B
`react_x`	r	s	no	cm	x coordinate of reaction point in the target given in the laboratory frame

Table 2.3: (continued)

variable	class	type	sub	dim	description
react_y	r	s	no	cm	y coordinate
react_z	r	s	no	cm	z coordinate
mlu_in1* mlu_in2* mlu_in3* mlu_in4* mlu_in5*	r	s	no		coincidence event rate into the different channels of the memory lookup unit MLU; only channel 5 is meaningful for coincidences
mlu_out*	r	s	no		coincidence event rate coming out of the textscmlu
dead_time*	r	s	no		dead time in coincidence events

A '*' behind the variable name of the tables 2.3, 2.4, 2.5 indicates that the histogram of this particular variable is updated even when the necessary detectors did not fire. For instance, the number of hits in a scintillator would be updated even when this detector did not fire. For level 3-variables which detector had to fire is obvious from the variable string. Level-2 variables are a bit more complicated variables which are composed out of the response of several detectors. Here, the histogram is updated when the trigger detector of the spectrometer fired. For level-1 variables either the trigger detectors of both spectrometers should have fired in case of coincidence variables or no detector is necessary at all like for the variable `event_type`.

Table 2.4: Level two variables, involving one spectrometer.

variable	class	type	sub	dim	description
x_rot	r	s	no	m	dispersive position x of the *golden* track in the U1 plane in the *rotated* c.s. [1]
th_rot	r	s	no		tangent of angle θ of the particle track in the focal plane in the dispersive direction wrt the *local* central ray
y_rot	r	s	no	m	non-dispersive position y in the U1 plane in the *rotated* c.s.
ph_rot	r	s	no		tangent of angle ϕ of the particle track in the non-dispersive plane at the focal plane
x_rot_abr	r	s	no	m	dispersive position x_rot corrected for spectrometer aberations
x_rot_kin	r	s	no	m	dispersive position x_rot corrected for spectrometer aberations and for deviations of the nuclear recoil momentum [2]
x_tra th_tra y_tra ph_tra	r	s	no		Coordinates of *golden* track defined as above but now in the *transport* c.s.
p	r	s	no	MeV/c	momentum of the particle after exiting the target corrected for spectrometer aberations
p_kin	r	s	no	MeV/c	momentum of the particle after exiting the target corrected for spectrometer aberations and for deviations of the nuclear recoil momentum
dp	r	s	no		like p but relative to the central ray
dp_kin	r	s	no		like p_kin but relative to the central ray
dp_cor	r	s	no		momentum relative to the central ray corrected only for spectrometer aberations
th_tg	r	s	no		tangent of angle θ_{tg} of the particle track in the dispersive plane at the scattering point at the target

[1] The golden track is defined as the track having a `tzero_av` closest to zero, therefore, most likely being the particle track that made the trigger.

[2] This correction is different for electrons and protons. It makes only sense to use it for the hadron arm in case of H(e,e'p)

Table 2.4: (continued)

variable	class	type	sub	dim	description
y_tg	r	s	no	m	non-dispersive position y_{tg} at the scattering point at the target
ph_tg	r	s	no		tangent of angle ϕ_{tg} of the particle track in the non-dispersive plane at the scattering point at the target
path_diff	r	s	no	m	difference in path length between this track and the central ray through the spectrometer.
track*	i	s	no		number of reconstructed particle tracks.
tzero_av*	i	s	no	TDC chan	average time offset of TDC times in the clusters of the different planes that are in the track
mlu_in1* mlu_in2* mlu_in3* mlu_in4* mlu_in5*	r	s	no		single event rate into the different channels of the memory lookup unit MLU; only channels 1 and 2 are meaningful for spec_e and 3 and 4 for spec_h
mlu_out*	r	s	no		pre-scaled single event rate coming out of the textscmlu
dead_time*	r	s	no		dead time in pre-scaled single
det_mon*	r	s	no		detector package coincidence rate

Table 2.5: Level three variables, involving detectors in the spectrometers. spec_s can be either spec_e or spec_h

variable	class	type	sub	detector	dim	description
hit_type*	i	s	no	spec_s.u1 spec_s.v1 spec_s.u2 spec_s.v2		hitpattern of wire plane, see description in section **??**
clus	i	s	no	spec_s.u1 spec_s.v1 spec_s.u2 spec_s.v2		number of clusters in the wire plane
tdc	i	a	yes	spec_s.u1 spec_s.v1 spec_s.u2 spec_s.v2 spec_s.aero spec_s.gas	TDC chan	TDC values of drift times in wire plane or of photo multipliers in Cherenkov
tdc_c	i	a	yes	spec_s.u1 spec_s.v1 spec_s.u2 spec_s.v2	TDC chan	TDC values of drift times in wire plane corrected for COMMON STOP shifts due to path length differences to the time defining scintilator and corrected for drift to path length conversion.
tdc_cor	i	a	yes	spec_s.u1 spec_s.v1 spec_s.u2 spec_s.v2	TDC chan	correction to TDC values of drift times in wire plane
tdc_diff	i	a	yes	spec_s.u1 spec_s.v1 spec_s.u2 spec_s.v2	TDC chan	drift time difference determined by comparing measured drift time to the distance between wire plane and global fit divided through the drift velocity obtained from the database
wire	i	a	no	spec_s.u1 spec_s.v1 spec_s.u2 spec_s.v2		wire numbers that fired in wire plane

Table 2.5: (continued)

variable	class	type	sub	detector	dim	description
coars	i	a	no	spec_s.u1 spec_s.v1 spec_s.u2 spec_s.v2		wire number with the shortest drift time in each cluster
slope	r	a	no	spec_s.u1 spec_s.v1 spec_s.u2 spec_s.v2		slope of track through wire plane, determined for each cluster
pos	r	a	no	spec_s.u1 spec_s.v1 spec_s.u2 spec_s.v2	m	position where track passed through the wire plane given in the local wire plane coordinate system, determined for each cluster.
tzero	r	a	no	spec_s.u1 spec_s.v1 spec_s.u2 spec_s.v2	TDC chan	time offset of TDC times in cluster
chisq	r	a	no	spec_s.u1 spec_s.v1 spec_s.u2 spec_s.v2		reduced χ^2 of fit to the drift times in the wire plane for each cluster.
drift	r	a	no	spec_s.u1 spec_s.v1 spec_s.u2 spec_s.v2	m/s	reduced χ^2 of fit to the drift times in the wire plane for each cluster.
angle_diff	r	a	no	spec_s.u1 spec_s.v1 spec_s.u2 spec_s.v2		difference in slope as determined from the drift times in one plane and two planes
mult*	i	s	no	spec_s.u1 spec_s.v1 spec_s.u2 spec_s.v2 spec_s.s1 spec_s.s2 spec_s.aero spec_s.gas spec_e.preshow spec_e.show spec_h.s3		number of hits on detector
hp_l*	i	a	yes	spec_s.s1 spec_s.s2 spec_h.s3		paddle hit pattern for photo multipliers on the left side
hp_r*	i	a	yes	spec_s.s1 spec_s.s2 spec_h.s3		paddle hit pattern for photo multipliers on the right side
adc	i	a	yes	spec_s.aero spec_s.gas spec_e.preshow spec_e.show	ADC chan	ADC of Cherenkov mirrors or (pre)shower blocks
adc_c	i	a	yes	spec_s.aero spec_s.gas spec_e.preshow spec_e.show	ADC chan	ADC of Cherenkov mirrors or (pre)shower blocks corrected for pedestal and gain
adc_l	i	a	yes	spec_s.s1 spec_s.s2 spec_h.s3	ADC chan	ADC of scintillator paddles on the left side

Table 2.5: (continued)

variable	class	type	sub	detector	dim	description
adc_r	i	a	yes	spec_s.s1 spec_s.s2 spec_h.s3	ADC chan	ADC of scintillator paddles on the right side
adc_l_c	i	a	yes	spec_s.s1 spec_s.s2 spec_h.s3	ADC chan	ADC of scintillator paddles on the left side corrected for pedestal and gain
adc_r_c	i	a	yes	spec_s.s1 spec_s.s2 spec_h.s3	ADC chan	ADC of scintillator paddles on the right side corrected for pedestal and gain
tdc_l	i	a	yes	spec_s.s1 spec_s.s2 spec_h.s3	TDC chan	TDC of scintillator paddles on the left side
tdc_r	i	a	yes	spec_s.s1 spec_s.s2 spec_h.s3	TDC chan	TDC of scintillator paddles on the right side
tdc_l_c	i	a	yes	spec_s.s1 spec_s.s2 spec_h.s3	TDC chan	TDC of scintillator paddles on the left side corrected for timing offset
tdc_r_c	i	a	yes	spec_s.s1 spec_s.s2 spec_h.s3	TDC chan	TDC of scintillator paddles on the right side corrected for timing offset
x_det y_det	r	s	no	spec_s.u1 spec_s.v1 spec_s.u2 spec_s.v2 spec_s.s1 spec_s.s2 spec_s.aero spec_s.gas spec_e.preshow spec_e.show spec_h.s3	m	coordinates of crossing point of *golden* track with detector plane, given in the detector's local coordinate system
rate	r	a	yes	spec_s.aero spec_s.gas spec_e.preshow spec_e.show		count rate in mirrors and blocks, resp.
rate_l rate_r	r	a	yes	spec_s.s1 spec_s.s2 spec_h.s3	count rate in paddles on the left or right side, resp.	

A word of caution is necessary for the usage of variables that give the rates of the scalers in the experimental setup. Realize that the whole histogramming procedure in ESPACE is event-interrupt driven. So if you would save a one-dimensional histogram of, for instance, the coincidence rate in paddle 3 of scintillator s1 and its value is always 100 Hz and 1000 events are scanned, you will end up with a sharp spike at 100 containing 1000 entries. This is probably not very useful. If you are interested in the time evalution of this rate, its better to svae a 2-dim ntuple of rate versus time.

2.3 Coding new variables

In this section we discuss how to install in the code a new variable. It is assumed that the new variable is installed in the ESPACE library. Installation in the ESPACE_HALLA part would proceed along the same lines. The routines that should then be modified will have the same names as the ones shown below, however, the extension _USER has to be added.

In our example, we want to install the ADC value of the left side of the scintillator paddle in both spectrometer `spec_e` and `spec_h`. Realize that per interrupt the scintillator's left side could have more than one entry and that one would like to specify the paddle number of the scintillator. In other words, this variable will be an array and could be specified with a sub-detector number. The consequence is that the last two routines below (`index_range.f`, for array variables and `det_sub.f`, for variables with sub-detector numbers) need to be modified.

The following routines have to be changed in the ESPACE library:

BLOCK_DATA_VAR.F In this file the string array `var_txt` is initialized. It contains all the necessary parameter input for each variable. Each variable should have an entry of one line in the `data` statement:

```
data (var_txt(i),i=1,199)
............
> 'adc_l 5-6,21-22,49 none none 0.0 4095.0 i a s',!  49
............
```

The description of these entries (possible values btween square brackets) :

`name` [*alphabetic characters including '_' and '-'*] variable name.

`detector` [*numeric characters and `all` or `none`.*] String of numbers indicating which detectors have access to this variable. A `-` in the string indicates a 'from-to', so `5-7` means `5,6,7`. The correspondance between number and detector string is given by the array `det_txt`, also initialized in `block_data_var.f`. As you can see in table 2.5 with the `adc_l` entry, the string `5-6,21-22,49` corresponds to detectors `spec_e.s1`, `spec_e.s2`, `spec_h.s1`, `spec_h.s2` and `spec_h.s3`.

`optimize` [*numeric characters and `all` or `none`.*]

`fire string` [*numeric characters and `all` or `none`.*] String of numbers pointing at detectors that should have fired in addition to the one whose variable is asked for. This is a useful feature if a variable is referenced that has a detector in both spectrometers, like all level-one variables, see table 2.3.

`range` [*numeric characters*] Default range of the variable which is used when the command `set/auto-window on` was given.

`class` [*i,r*] `integer` or `real` variable

`type` [*`s,a`*] `scalar` or `array` variable.

`sub` [*`y,n`*] sub-detector number allowed, `y(es)` or `n(o)`.

VARIABLE.F The function VARIABLE.F should return the value of the variable pointed at by `ispec`and `det_txt`. `ispec` corresponds to the entry number in the array `det_txt`, see routine `block_data_var.f`.

indx is the hit number on the detector and its value is given by the routine DET_SUB.F, see below.

```
............
else if (ivar.eq.49) then
if (ip_sc.ne.0) then
ip_pad = spdetector.sp(ispectro).trigger.
> sc(ip_sc).padnr(indx).adc.left
ip_hit = spdetector.sp(ispectro).trigger.
> sc(ip_sc).hitnr(indx).adc.left
dvar = dble(spdetector.sp(ispectro).trigger.
> sc(ip_sc).pad(ip_pad).
> val(ip_hit).adc.left)
endif
else
............
```

In the example given above, first the pointers to the correct scintillator paddle and position in the array containing the different hits (ip_pad and ip_hit) have to be determined. Then the variable dvar is filled.

INDEX_RANGE.F Changing this routine is only necessary for array variables. This subroutine should return through imax the number of entries for the variable pointed at by ivar.

```
............
else if (ivar.eq.49) then
if (ip_sc.ne.0) then
imax = spdetector.sp(ispectro).trigger.sc(ip_sc).
> nhits.adc.left
endif
imax = max(1,imax)
else
............
```

The example above shows how imax is set if ivar equals 49.

DET_SUB.F This routine has to be changed only for variables with subdetector indices. It should return the sub-detector number pointed at by ivar and indx.

```
............
else if (ivar.eq.49) then
if (ip_sc.ne.0) then
det_sub = spdetector.sp(ispectro).trigger.sc(ip_sc).
> padnr(indx).adc.left
endif
else
............
```

2.4 Defining new variables

Instead of coding a variable into ESPACE, a new one can also be defined *on the fly* through the command interface. For example, suppose the missing mass has to be defined for the reaction H(e,e'p). This would be done through the following steps:

```
espace > m_proton=938
espace > e0 = 845
espace > omega:e0-spec_e.p
espace > tx:sqrt(spec_e.p**2+m_proton**2)-m_proton
espace > m_miss:sqrt((omega-tx)**2-p_miss**2)
```

Notice that constants are assigned through an equal sign: `m_proton=938`. Variables are assigned through a colon: `omega:e0-spec_e.p`. After this assignment the new variable can be used, like the coded ones, in logical statements, histograms and ntuples.

Before summarizing the syntax for these assigments, notice the following restriction:

In the definition of a new variable, only scaler variables can be used

New variables are defined by combining existing variables through the following operations:

```
( ) + - * / ^ **
```

and intrinsic functions

```
abs acos acosd asin asind atan atand cos
cosd cosh deg exp log log10 rad sin
sind sinh sqrt tan tand tanh
```

The definition string should look like, `NEW : operation(OLD1,OLD2,...)`. Notice the assignment with ':'.

Constants are defined by a string like, PI=3.142 . Notice the assignment with '='.

Chapter 3

Defining histograms, logicals and filters

In the previous section was described how to access a particular variable. What still has to be explained is how to histogram that variable and how to use it in the defining a condition.

3.1 Histograms

A histogram is defined with the command:

`spectra/save variable_string condition_string`

The `variable_string` can be any variable or combination of variables (for multi-dimensional histos and ntuples) as defined in tables 2.1,2.3,2.4 and 2.5. The `condition_string` string is discussed in the next section.

A multi-dimensional histogram or ntuple (give command `set/ntuple on`) is created by making a variable string like `var2/var1`. For a histogram this has to be limited to two variables, an ntuple can contain upto 12 variables. The first variable mentioned in the `variable_string` is the last one in the histogram, so `var2/var1` will result in a 2-dim histogram with `x=var1` and `y=var2`.

A group of variables can be saved several times with different logical conditions. In this case the user has to supply at the end of the `variable_string` either a counting index or leave it up to the program ESPACE to increment it internally. For instance,

```
spectra/save spec_e.dp condition1
spectra/save spec_e.dp condition2
```

or

```
spectra/save spec_e.dp-1 condition1
spectra/save spec_e.dp-2 condition2
```

will have the same result, however in the second case the user has explicitly given the counter numbers and has, therefore, its values under control.

As shown in the previous chapter, a variable has several parameters which can influence the `save` command. Before histogramming a variable one should check whether its `class` is `real` or `integer`. In the example below the command `set/spectrum/bins/x1 10` makes a histogram of the `real` variable `spec_e.th_tg` containing 10 bins between 100. and 200. while for the `integer` variable `spec_e.u1.tdc` it means that a histogram running from 100 to 200 should be made with a bin size of 10.

```
set/spectrum/window/x1 100 200
set/spectrum/bins/x1 10
spectra/save spec_e.th_tg
```

or

```
set/spectrum/window/x1 100 200
set/spectrum/bins/x1 10
spectra/save spec_e.u1.tdc
```

The `sub-detector` parameter for the variable allows to specify a subset of the detector for which the variable should be stored (see section 2.2 for the meaning of sub-detector for the different detectors). The syntax for the `sub-detector` range is,

```
spectra/save var[start;stop] condition
```

This can be used to create an ADC spectrum for the read-out of every paddle on the right side of scintillator `s1` (notice the usage of the DO loop as implemented in the KUIP command interpreter):

```
do i=1,6
spectra/save spec_e.s1.adc_r[[i];[i]] condition
enddo
```

3.2 Logicals

The definition of logicals can be done in two different ways:

- Define logicals by making cuts on the variables, either one- or two- dimensional.
- Define new logicals by combining the previously defined logicals through logical operators.

The syntax of the `cut` command in one-dimension is

```
define/cut/1d variable_string start_value stop_value
```

The syntax of the `variable_string` is exactly along the lines of defining a `variable_string` in the `spectra/save` command. The logical value of the cut is TRUE if `start_value` $\geq$ `variable_string` $>$ `stop_value`.

In two dimensions a cut can be defined in two ways:

- Define a curve with a 2$^{\text{nd}}$-order polynomial, and demand that a point is `below` or `above` it. The syntax is

  ```
  define/cut/2d variable_string direction p0,p1,p2
  ```

  Here the `variable_string` has the same syntax as for a 2-dim histogram, so `var2/var1`. The parameters p_0, p_1 and p_2 are polynomial coefficients describing the line, $\text{var2} = p_0 + p_1 * \text{var1} + p_2 * \text{var1}^2$. The parameter `direction` should have the value `above` ($\text{var2} \leq p_0 + p_1 * \text{var1} + p_2 * \text{var1}^2$) or `below` ($\text{var2} \geq p_0 + p_1 * \text{var1} + p_2 * \text{var1}^2$).

- Define a region with a polygon and demanding that the data point is inside the polygon. Syntax for this command is:

  ```
  define/cut/polygon variable_string file_name
  ```

  `file_name` is the name of a file containing the coordinates of the polygon corners. The numbers of points should be limited to 20 !

A way to make such a file is by plotting the appropriate 2-d histogram or ntuple in PAW and then execute the following lines in PAW:

```
vector/delete x,y
vlocate x y -LPS
write x,y file_name '(g13.7,2x,g13.7)' 'OC'
```

This command sequence will generate a file called `file_name` in which the `x` and `y` coordinates are written of each left-button mouse-click. It stops after a right-button mouse click.

In both 1-dim and 2-dim `cut` commands one is allowed to use variables with `type=array`. This can make cuts more specific, e.g. `define/cut/1d spec_e.s1.adc_r[2;2] 100 200` is only .true. when paddle 2 of s1 has a value between 100 and 200. A statement like `define/cut/1d spec_e.s1.adc_r 100 200` is .true. if any of the paddles had a value between 100 and 200.

New logicals can be defined by combining existing ones through LOGICAL operands. A list of the implemented operands is given in table 3.1.

symbol	operation
!	negate
&&	logical AND
||	logical OR

Table 3.1: LOGICAL operands allowed in ESPACE

The order of operations can be guided by brackets. For the moment a logical definition containing an 'OR' should be surrounded by quotes because the KUIP command interpreter will otherwise stop reading at the appearance of a | character.

It is important to realize that logical definitions are expanded only upon execution, so during scanning an event file. For example,

```
define/logical new 'old1 && old2'
spectra/save spec_e.th_tg-1 new
```

followed by

```
define/logical new 'old1 || old2'
spectra/save spec_e.th_tg-2 new
```

will result in two `th_tg` spectra stored under identical conditions, namely the logical *or* of ttfsold1 and `old2`.

3.3 Filters

A filter is defined with the command:

```
file/filter file_name condition_string ,
```

where `file_name` is the file into which the data will be stored if it fulfills `condition_string`. Currently the filtered data is stored again in CODA format so that it can be replayed with the same input routines as the raw data. However, the user can change that; all it takes is a change of the routines in the file FILTER.F in the `espace_halla` package. The current version is shown below:

```
subroutine filter_event(ihandle)
c
c-----------------------------------------------------------------------
c
implicit none

include 'coda.h'

integer ihandle
character filename*(*)

integer evopen

integer istatus

c Write CODA event
call evwrite(ihandle,evbuffer)

return


entry filter_open(ihandle,filename)

c Open CODA file

istatus = evopen(filename,'w',ihandle)
if (istatus.ne.0) then
call derror('FILTER_OPEN: error opening CODA output file')
endif

return


entry filter_close(ihandle)

c Close CODA file

call evclose(ihandle)

return
end
```

The file should contain the routines

FILTER_OPEN This routine opens the filtered data file.

FILTER_EVENT This routine is called for each event that fulfills the filter condition. In it is defined what data is stored and in which format.

FILTER_CLOSE The closing of the data file is performed here.

If the user prefers a format different from CODA and wants to be able to replay it later again with ESPACE, the input routines, controlled by FILTER_DATA_TIL_EOF.F (ESPACE_HALLA should be changed too. When scanning the file, one should not forget to give the 'f' option with the `scan` command so that FILTER_DATA_TIL_EOF.F controls the input instead of RAW_DATA_TIL_EOF.F.

3.4 Global conditions

In the previous sections was discussed how to provide selection criteria when saving histograms or defining filters through the `command_string`. It is also possible to define more global conditions, meaning that it is applied to all following commands. This global condition is defined with the command:

```
set/hitbits (+detector1,-detector2,...)  ,
```

where one can fill in the list any `detector` as mentioned in table 2.5. Each `detector` should be preceded by a '+' or '−'. A '+' means that the condition is only .TRUE. if the `detector` has fired while a '−' means just the opposite, so .TRUE. if the `detector` did not fire. The total condition is the obtained by AND'ing the result of each `detector` in the list.

3.5 When is what stored ?

For calculating absolute numbers, it is important to understand what is stored in a histogram or the filter file when or a condition is not fulfilled or a detector does not fire.

If the condition for a particular operation in ESPACE is not fulfilled, then it will not be executed for that event. So if the operation was histogramming a variable, saving data in a filter file, accumulating data for an optimization, etc., it will not be performed for this event. After a global condition all further operations are skipped up to the point where a new global condition was defined.

If a detector did not fire either of the following can happen:

- The variable of this detector is well defined even when the detector did not fire. For example the multiplicity of the detector event, being zero. These variables have an '*' in tables 2.1, 2.3, 2.4, 2.5 behind their name.
- The variable of the detector is undefined, ESPACE puts it on a very large number (10^9).

In either case, the number of entries in the histogram/ntuple should reflect the number of times the condition was fulfilled.

For variables not directly linked to one particular detector (so level-1 and level-2) it is not immediately clear which detectors should be checked. For the moment the following convention has been adapted. Level-2 variables are linked to the trigger detectors ((s1) and (s2)) in the appropriate spectrometer (see table 2.4 while part of the level-1 variables are linked to the trigger packages in both spectrometers, see table 2.3, and the other half is completely detector independent, see table 2.1.

Chapter 4

Coordinate Systems

4.1 Definition of Coordinate Systems

In this section five coordinate systems are discussed. One coordinate system is connected to the hall while a second one is a coordinate system in the hall that is connected to each of the magnetic spectrometers. Coordinate systems three, four and five are linked to the detector packages in the focal planes of the spectrometers.

4.1.1 Hall A Laboratory Coordinate System

The origin of the hall A Laboratory Coordinate System is defined by the intersection (or at least the point of closest approach) of the non-rastered electron beam and the axis of rotation of the solid-target system. The non-rastered electron beam direction is defined by the last two beam-position monitors. The incident unscattered beam leaves the target along the positive z direction. The y-axis is pointing upward as defined by gravity.

Angles are defined with respect to this origin and a ray pointing along the positive z-axis. Convenient is the use of *geographical* angles. Here, the angle θ is 0 along z and covers a 0 to 180 $\deg$ range in the $x - z$ plane. ϕ is the out-of-plane angle formed between the vector under consideration and its projection on the $x - z$ plane. In contrast to the spherical angle system, geographical angles can be added and subtracted, see section 4.2.3. In appendix C an overview is given of available algorithms that translate between the different systems.

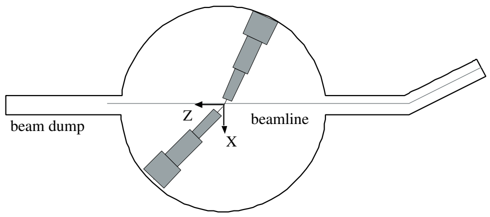

Figure 4.1: Hall A laboratory coordinate system

4.1.2 Spectrometer Reconstructed Coordinate System

The origin of the spectrometer reconstructed coordinate system is defined as a point at a distance of 1.25 m from center of the central sieve-slit hole. The line going from the origin to the center of this hole is perpendicular to the sieve-slit surface and defines the z-axis. The x-axis is the line going through the center sieve-slit holes and points downward. Under optimal circumstances this origin should coincide with the origin of the hall A laboratory coordinate system and the center of rotation of the spectrometer. The $x - z$ plane and the y axis of the hall system should be parallel.

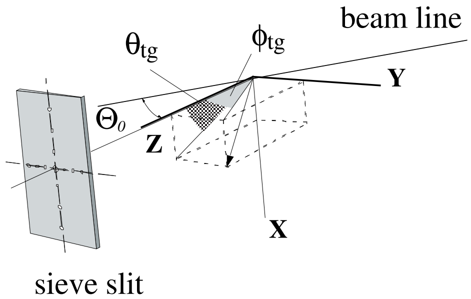

Figure 4.2: Spectrometer reconstructed coordinate system

4.1.3 Spectrometer Detector Coordinate System

The origin of this coordinate system is defined by the intersection of wire 184 in the first wire plane and the projection on the first wire plane of wire 184 in the second plane (assuming 368 wires in each plane). The z axis is perpendicular to the wire planes and its direction is fixed by demanding that its product with the central spectrometer ray (see section 4.1.4) is larger than zero. The x axis is defined as the projection on the first wire plane of the vector difference between the the spectrometer central ray and a ray for which the momentum has been increased by an infinitesimal amount. Its direction is fixed by requiring an increase in momentum. It would be optimal if the $x - z$ plane would coincide with the spectrometer symmetry plane. In section 4.2 we will discuss how these coordinates can be calculated with the wire chamber information and how corrections can be performed for misalignments of the detector package.

4.1.4 Spectrometer Focal-Plane Coordinate System

The spectrometer focal-plane coordinate system shares its origin with the detector system and the $x - z$ planes coincide. However, its z-axis (and therefore also its x-axis) has a different orientation. The z axis is is defined as the projection of the local central ray on the $x - z$ plane. The consequence is that x and z axis are a function of the fractional particle momentum $\frac{\Delta p}{p}$.

4.2 Calculation of Coordinates

In this section is discussed how the coordinates are calculated in the different coordinate systems. We will start in the detector system and will finish with the particle vertex in the hall A system. Along the way offsets are introduced to correct for misalignments. Of course, there are many ways

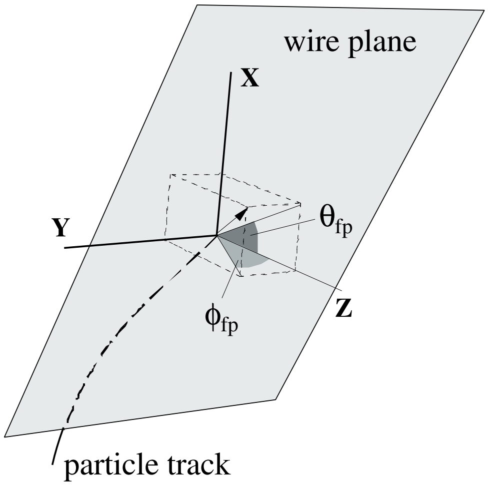

Figure 4.3: Spectrometer focal plane coordinate system

in which parameters can be introduced to make these corrections, however, it is our goal to choose that set that fulfills the following two important requirements:

1. The parameter set that corrects for the misalignments should be as small as possible.
2. There should be a clear procedure how to determine each of them in a unique way.

After this section the experimental procedures will be discussed with which the parameters will be calibrated.

4.2.1 Detector Vertex

A particle track, p, in the detector package of a magnetic spectrometer can be written in the detector coordinate system as

$$\begin{pmatrix} p_x \\ p_y \\ p_z \end{pmatrix} = \begin{pmatrix} x_{ds} \\ y_{ds} \\ 0 \end{pmatrix} + \lambda \begin{pmatrix} \sin\theta_{ds}\cos\phi_{ds} \\ \sin\phi_{ds} \\ \cos\theta_{ds}\cos\phi_{ds} \end{pmatrix} \quad \text{,where} \tag{4.1}$$

θ_{ds} and ϕ_{ds} are geographic angles, see 4.1.1. The four unknowns (x_{ds}, y_{ds}, θ_{ds} and ϕ_{ds}) are determined with four measurements in the drift-chamber package.

In figure 4.4 the wire direction in the first plane is shown making an angle of $-\xi$ with the y-axis of the detector coordinate system. Indicated also is a particle track through the plane. The drift chamber can measure the projection of this track on a plane perpendicular to its wire plane. The projection of an arbitrary vector $v = (v_x,v_y,v_z)$ is

$$\begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix} \stackrel{\text{projection}}{\longrightarrow} \begin{pmatrix} v_x\cos^2\xi - v_y\cos\xi\sin\xi \\ v_y\sin^2\xi - v_x\cos\xi\sin\xi \\ v_z \end{pmatrix} \tag{4.2}$$

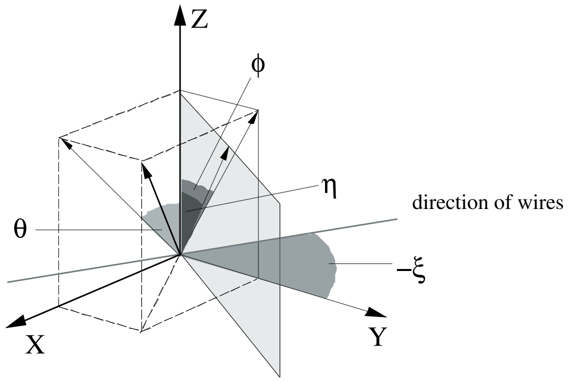

Figure 4.4: Spectrometer detector coordinate system. Two track angles θ_{ds} and ϕ_{ds} are indicated together with the angle η that is measured with one or two wire plane(s).

The projection is characterized by two quantities: the angle η (see fig. 4.4 and the position p_{vdc} where the projected track crosses the wire plane relative to the projection of the origin of the detector coordinate system on this wire plane. Using equation 4.2, the expression for η and p_{vdc} in terms of the unknowns of equation 4.1 are:

$$\tan \eta_n = \tan \phi_{ds} \sin \xi_n + \tan \phi_{ds} \cos \xi_n \tag{4.3}$$

$$p_{vdc;n} = p_x \cos \xi_n - p_y \sin \xi_n \quad , \tag{4.4}$$

where the index n refers to the wire-plane number.

Equation 4.4 shows that two independent measurements of η will give ϕ_{ds} and θ_{ds}. This can be done with two drift chamber planes with different wire angles. However, in this configuration the η-resolution is limited to typically 8 mrad (FWHM). This can be improved by adding two more drift chamber planes, resulting in a configuration with two wire lanes with wire angle ξ_1 and two with wire angle ξ_2. In figure 4.5 the spatial configuration of the wire planes is sketched.

Figure 4.5: Configuration of the wire chamber package.

Using only the position information $p_{vdc,n}$, $n = 1 \cdots 4$, we can calculate the vertex parameters in the SCDS as follows:

$$\tan\eta_1 = \frac{p_{vdc;3} - p_{vdc;1}}{d_2}\ , \ \tan\eta_2 = \frac{p_{vdc;4} - p_{vdc;2}}{d_2}$$

$$\begin{aligned}
\tan\theta_{ds} &= \frac{\tan\eta_1\sin\xi_2 + \tan\eta_2\sin\xi_1}{\sin(\xi_2-\xi_1)} \\
\tan\phi_{ds} &= \frac{\tan\eta_2\cos\xi_1 + \tan\eta_1\cos\xi_2}{\sin(\xi_2-\xi_1)} \\
x_{ds} &= \frac{p_{vdc;1}\sin\xi_2 - (p_{vdc;2} - d_1\tan\eta_2)\sin\xi_1}{\sin(\xi_2-\xi_1)} \\
y_{ds} &= \frac{p_{vdc;1}\cos\xi_2 - (p_{vdc;2} - d_1\tan\eta_2)\cos\xi_1}{\sin(\xi_2-\xi_1)}
\end{aligned}$$

Note that the coordinates of the point where the particle crosses the first plane are calculated with information mainly coming from the first two planes. This minimizes the effects on x_{ds} of multiple scattering in the drift chambers. A correction has been applied to the position $p_{vdc;2}$ for the fact that the first two planes are separated by a distance d_1.

Some assumptions have been made in this derivation:

- Wire angles are known and constant.
- Wires are positioned in planes
- Planes are parallel and at known distances.
- Location of the center of the wire planes wrt. the SCDS origin is known.

Deviations from the nominal values will lead to offsets in the SCDS coordinates. In order to correct for these deviations at least four offsets have to be introduced. In principle one could introduce offsets to correct for each of the items mentioned above, however, this would result in a set of highly correlated parameters. Therefore, we prefer to have one offset for each of the coordinates, x_{ds}, θ_{ds}, y_{ds} and ϕ_{ds}. The offsets will be introduced in this two sections describing the transformation of the vertex to the spectrometer focal plane coordinate system and the spectrometer reconstructed coordinate system. Some of the offsets can be combined with parameters describing these transformations.

4.2.2 Focal-Plane Vertex

Before reconstructing the particle vertex at the spectrometer entrance the particle vertex in the detector system is transformed to the spectrometer focal-plane system (a name slightly misleading as one will see further on). This coordinate system is obtained by rotating SDCS along its y-axis over an angle ρ, where ρ is the angle between the projection of the *local* central ray ($\theta_{tg} = \phi_{tg} = 0$, see section 4.1.4 on the $x - z$ plane and the z-axis of the SDCS [1] So the new z-axis rotates as a function of the relative particle momentum $\frac{\Delta p}{p}$. The main advantage of expressing the vertex in

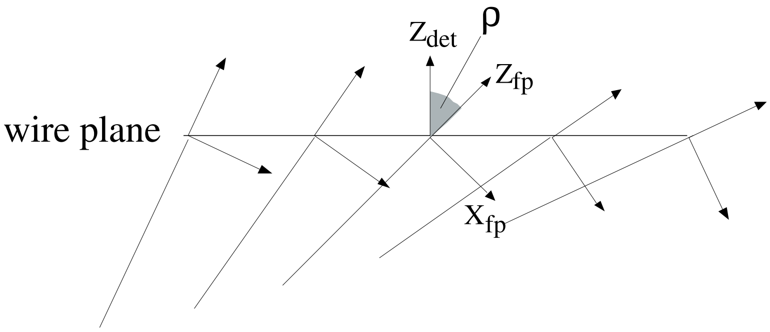

Figure 4.6: Spectrometer focal plane coordinate system which is rotated over an angle ρ along the y-axis wrt the SDCS. ρ is a function of the relative momentum.

this system is the fact that the dispersive angle θ will be small making the expressions for the reconstructed vertex converge faster.

Writing down the expressions for the new coordinates, the following points should be noted. 1) Offset correction and rotation can be combined in one parameter. 2) Since the particle momentum is the variable with the highest resolution (and therefore most sensitive) we do not want to apply a rotation to this coordinate that is not constant. Then, a new calibration of the momentum dependence of ρ would immediately have consequences for the dispersion constants, see equation 6.4. Here we apply only the

The offset for x_{ds} will be introduced when calculating the particle momentum and the θ_{ds} offset will be absorbed in the transformation to the spectrometer focal-pane coordinate system. So up to here we have:

$$\boxed{\begin{aligned}
x_{fp} &= \cos\rho_0(x_{ds} - x^{offset}) \\
y_{fp} &= y_{ds} - y^{offset} \\
\tan\theta_{fp} &= \frac{\tan\theta_{sd} - \tan\rho}{1 - \tan\theta_{sd}\tan\rho} \\
\tan\phi_{fp} &= \frac{\tan\phi'_{ds}}{\cos\rho - \sin\rho\tan\theta_{ds}} \\
\tan\phi'_{ds} &= \frac{\tan\phi_{ds} - \tan\phi^{offset}}{1 + \tan\phi_{ds}\tan\phi^{offset}}
\end{aligned}}$$

4.2.3 Hall A Laboratory Vertex

Suppose the normal to the sieve slit surface that goes through the central hole is given by the geographical angles Θ_0 and Φ_0 in the hall A laboratory coordinate system. [2] The geographical

[1]under ideal conditions, such as spectrometer mid-plane symmetry, the local central ray is in the $x - z$ plane.

[2]I guess some of you will argue that to define the orientation of the spectrometer uniquely in space also the rotation around this normal should be specified. I guess you are right but usually spectrometer can not (or better formulated *should not*) perform this motion.

angles of the particle vertex in the HLCS is given by:

$$\boxed{\begin{aligned} \Theta_{hlcs} &= \Theta_0 \pm \phi_{tg} \\ \Phi_{hlcs} &= \Phi_0 + \theta_{tg} \end{aligned}}$$

The question whether to add or subtract ϕ_{tg} depends on the position of the spectrometer with respect of the beam line. In order to calculate the reaction point (x, y, z), additional information is needed from another spectrometer, detecting a particle in coincidence, and/or the beam position from the rastering system.

4.3 Calibration of Coordinates

Up to now we have defined the different coordinate systems and discussed how the particle vertex as determined in the spectrometer detector package leads to the vertex of the particle emerging from the reaction point in the target. However, the usefulness of all this stands or falls with the existence of procedures to calibrate these coordinates.

In the following discussion it is assumed that the orientation of the sieve slit with respect to the hall a coordinate system is aligned carefully through optical procedures. At this point we assume also that the hall A coordinate system itself can be checked through a straightforward procedure. So starting with a well calibrated system of sieve slit and magnet positions wrt the HLCS, elastic electron scattering measurements should be performed so that a sharp peak in momentum is detected. This measurement should be repeated after changing the magnetic field, thereby after several changes covering the whole momentum acceptance. The central hole of the sieve slit provides us with data to determine the offsets introduced in the spectrometer detector coordinate system for θ, y and ϕ. The offset in x is absorbed in the dispersion coeficient d_0 of equation 6.4. Determination of this coefficient needs different type of measurements as discussed in [6]

Chapter 5

Tracking

In this chapter we will discuss the track reconstruction with the spectrometer VDC package. This is the most important part in the data analysis with a high-resolution spectrometer, justifying a whole chapter about this subject. First, the program flow for tracking will be discussed, followed by an explanation of the *minimum spanning tree* algorithm, used in disentangling multiple tracks. We finish with showing the tracking efficiency of the algorithm by analyzing a simulated ^{16}O(e,e'p) data.

5.1 Program flow for tracking

5.2 Minimum spanning tree algorithm

5.3 Definition of wire chamber hit type HPCH

Each wire chamber hit is assigned a hit type. At the moment the following seven different conditions are being distinguished:

HPCH value		Description
dec	bin	
2	00000010	a nohit or all clusters are of ERR1 type.
4	00000100	number of wires hit > 64
8	00001000	some drift times resulting in drift distances larger than 1.25 times distance between wire plane and cathode
16	00010000	some drift times are correcting for the stop offset negative

Table 5.1: Hit types for a wire chamber hit

For an error condition HPCH has a value ≥ 2. In this case the hit type can be a combination of several error conditions.

5.4 Tracking efficiency

Chapter 6

Calibration

The resolution and position of several variables can be optimized. To start an optimization give either the command

```
optimize/calibrate variable_string condition_string
```

The `condition_string` can be any valid logical statement. The variable combinations that can be optimized are:

variable	comments
`spec_e.th_tg/spec_e.ph_tg`	Reconstructed theta and phi target coordinates of the electron spectrometer. These angles can only be optimized for a measurement with a *sieve* collimator. Both of them are optimized at the same time, see [5]
`spec_e.y_tg`	Reconstructed y target coordinate of the the electron spectrometer. The non-dispersive position can be optimized after a series of measurements have been performed in which the beam spot has been put at different positions.
`tc_cor`	Coincidence time between e and h spectrometer after correction for path length differences wrt the central rays and time offsets between the scintillators.
`e_miss`	Missing energy for (e,e'x)
`e_miss/p_miss`	Missing energy and momentum for H(e,e'p)
`spec_e.s1.adc_l_c`	scintillator ADC values corrected for pedestals and attenuation
`spec_e.beta`	particle β as calculated by the time-of-flight between `s1` and `s2`.

In the table above, one can replace `spec_e` by `spec_h`, `s1` by `s2` and `adc_l_c` by `adc_r_c`.

Except for the optimization of `tc_cor`, an additional input file has to be supplied. Since the selection of events used in the optimization procedure is usually too complicated to be accomplished through a series of cuts defined in ESPACE, this file contains information for the event selection. The file can be assigned through the command `set/file/fit_input`, see reference section.

For the different optimization options this file has a different format. First we discuss the case that the vertex reconstruction of a spectrometer is optimized, so `dp_kin`, `theta_tg/phi_tg` and `y_tg`. Then this file should contain in the first line the number of positions along the focal plane where one wants to optimize, followed by a line for each position containing the center of this region, its new position (only useful when `dp_kin` is being optimized) and the half width of this region.

If one wants to optimize `theta_tg/phi_tg` or `y_tg` additional data has to be supplied. In case of `theta_tg/phi_tg` this should be the number of holes in the *sieve* collimator. It has to be followed by a line for each hole giving the center of the hole in θ_{tg} before optimization, θ_{tg} hole position after optimization and half the width of the ellipse in the θ direction which will be put around the initial coordinates of the hole center. The same should be given for `phi_tg`. Only those events are used which have their initial values inside one of the ellipses. The beginning of a FIT-FILE for optimizing `theta_tg/phi_tg` with a 77-hole *sieve*-collimator would look like

```
 77 (theta_ini, theta_final, dtheta, phi_ini, phi_final, dphi)
-0.04500 -0.07024 0.0075 -0.08600 -0.09757 0.0050
```

The input for optimizing `y_tg` would consist out of first a line with the number of beam-spot positions which is being used in the optimization. It is followed by a line for each beam-spot position with the initial position, final position and half the width of the region around the initial position containing the events that are accepted for the optimization.

An important parameter in the optimization procedure is the variable that controls the number of events per peak per file that is being used (command `set/fit/npts`). With this variable one can prevent that the data used in the fit is coming from just the first few files being scanned.

If `e_miss` is optimized the file should contain in the first line the number of peaks in the missing energy spectrum that one wants to optimize. This is followed by a line for each peak containing the center of this region, its new position and the half width of this region. All in units of MeV.

6.1 ADC gains and pedestals

6.2 Scintillator timing

6.3 VDC time offets

6.4 VDC time to distance

6.5 Vertex reconstruction

Well, one could devote a whole chapter to this topic, actually a whole book. But let's not get carried away and just explain the basics to get you prepared to do your spectrometer calibration.

The basis for magnetic design and discussion is a six dimensional vector and the corresponding transfer tensor calculated by such programs as TRANSPORT [7]. The components of this vector include position and tangent of angle in the dispersive plane (usually designated x and θ), the non-dispersive coordinates (y and ϕ), and the fractional deviation of the momentum from the central design momentum (δ). A sixth dimension describing ray path length variations is included, but it is ignored in our discussion for simplicity. Note that all tensor elements in this document with subscript '5' correspond to the TRANSPORT element with subscript '6'.

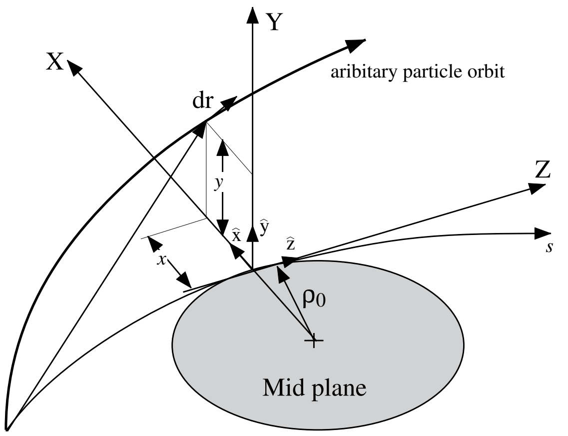

Figure 6.1: Spectrometer transport coordinate system.

The positions x and y are defined in a right-handed rectangular coordinate system with $\hat{z}$ along the particle direction and $\hat{x}$ directed away from the center of curvature in the bending plane. Thus, for an upward bending spectrometer like HRS, at the target $\hat{x}$ points downward. The angles θ and ϕ are measured from the z axis to the projection of the particle vector onto the $x - z$ and $y - z$ planes, respectively. At any position through the spectrometer, these coordinates are measured with respect to the *central ray* (i.e., $\hat{z}$ points along the central ray), where the central ray is defined to be that ray with all initial coordinates equal to zero at the target.

The form of the first-order forward transfer tensor is shown in equation 6.1. **tg** designates a vector at the target, and **fp** designates a vector measured in the focal plane region.

$$\begin{bmatrix} x \\ \tan\theta \\ y \\ \tan\phi \\ \delta \end{bmatrix}_{fp} = \begin{bmatrix} \langle x|x\rangle & \langle x|\theta\rangle & 0 & 0 & \langle x|\delta\rangle \\ \langle\theta|x\rangle & \langle\theta|\theta\rangle & 0 & 0 & \langle\theta|\delta\rangle \\ 0 & 0 & \langle y|y\rangle & \langle y|\phi\rangle & 0 \\ 0 & 0\langle\phi|y\rangle & \langle\phi|\phi\rangle & \langle\phi|\delta\rangle & \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ \tan\theta \\ y \\ \tan\phi \\ \delta \end{bmatrix}_{tg} \quad (6.1)$$

Notice that symmetry in the $x - z$ plane is enforced by making several tensor elements zero. We can easily find the invserse matrix which is shown in equation 6.2. Again, those elements which are explicitly zero result from midplane symmetry. In the notation below, $\langle a|b\rangle^{-1}$ means $(\langle a|b\rangle^{-1})_{ij}$ and not $(\langle a|b\rangle_{ij})^{-1}$.

$$\begin{bmatrix} x \\ \tan\theta \\ y \\ \tan\phi \\ \delta \end{bmatrix}_{tg} = \begin{bmatrix} \langle x|x\rangle^{-1} & \langle x|\theta\rangle^{-1} & 0 & 0 & \langle x|\delta\rangle^{-1} \\ \langle\theta|x\rangle^{-1} & \langle\theta|\theta\rangle^{-1} & 0 & 0 & \langle\theta|\delta\rangle^{-1} \\ 0 & 0 & \langle y|y\rangle^{-1} & \langle y|\phi\rangle^{-1} & 0 \\ 0 & 0 & \langle\phi|y\rangle^{-1} & \langle\phi|\phi\rangle^{-1} & \langle\phi|\delta\rangle^{-1} \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ \tan\theta \\ y \\ \tan\phi \\ \delta \end{bmatrix}_{fp} \quad (6.2)$$

The representation above is not adequate, however, to describe the reconstruction of target variables from measured ones. The full five dimensional vector is not detected by the focal plane instrumenation, which measures only positions and angles; the momentum information δ, is absent. Thus, only two of the three dispersive coordinates (δ and θ_{tg}) can be reconstructed. The third dispersive target coordinate must in principle be fixed to allow a solution for the problem. The logical choice is to constrain, of course, x_{tg}.

A matrix representation appropriate to the reconstruction procedure is trivially related to the above matrix by using that

$$\delta = \frac{x_{tg} - \langle x|x\rangle^{-1} x_{fp} - \langle x|\theta\rangle^{-1}}{\langle x|\delta\rangle^{-1}}$$

which results in the relation

$$\begin{bmatrix} \delta \\ \tan\theta \\ y \\ \tan\phi \\ x \end{bmatrix}_{tg} = \begin{bmatrix} \langle\delta|x\rangle^\star & \langle\delta|\theta\rangle^\star & 0 & 0 & \langle\delta|x_{tg}\rangle^\star \\ \langle\theta|x\rangle^\star & \langle\theta|\theta\rangle^\star & 0 & 0 & \langle\theta|x_{tg}\rangle^\star \\ 0 & 0 & \langle y|y\rangle^\star & \langle y|\phi\rangle^\star & 0 \\ 0 & 0 & \langle\phi|y\rangle^\star & \langle\phi|\phi\rangle^\star & \langle\phi|\delta\rangle^\star \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{fp} \\ \tan\theta_{fp} \\ y_{fp} \\ \tan\phi_{fp} \\ x_{tg} \end{bmatrix}_{fp} \tag{6.3}$$

This will hereafter be referred to as the *reverse* tensor (as opposed to the *inverse* tensor above). The dependence of δ and θ_{tg} on x_{tg} is explicit.

In practice (ESPACE) one proceeds a bit different. Only the the upper 4x4 part can be calibrated for a known x_{tg} position (hopefully zero). In the analysis the x_{tg} position is determined with the information of the beam position on target and the reconstructed θ, y and ϕ position in the target of one or two spectrometers. This value of x_{tg} is used together with equation 6.3 to correct the focal plane coordinates to values corresponding to $x_{tg} = 0$. [1] Realize that one has to obtain part of the tensor coefficients in equation 6.3 from a program like RAYTRACE [8] or COSY [9]. The new focal-plane vertex is used in the calculation of the reconstructed target vertex.

The transformation from the focal plane to the spectrometer entrance is described with a set of tensors, D_{ijkl}, T_{ijkl}, Y_{ijkl} and P_{ijkl}:

$$\boxed{\begin{aligned} \frac{\Delta p}{p} &= \sum_{ijkl} D_{ijkl} x^i_{fp} \tan\theta^j_{fp} y^k_{fp} \tan\phi^l_{fp} \\ \tan\theta_{tg} &= \sum_{ijkl} T_{ijkl} x^i_{fp} \tan\theta^j_{fp} y^k_{fp} \tan\phi^l_{fp} \\ y_{tg} &= \sum_{ijkl} Y_{ijkl} x^i_{fp} \tan\theta^j_{fp} y^k_{fp} \tan\phi^l_{fp} \\ \tan\phi_{tg} &= \sum_{ijkl} P_{ijkl} x^i_{fp} \tan\theta^j_{fp} y^k_{fp} \tan\phi^l_{fp} \end{aligned}}$$

If the spectrometer has a symmetry plane that coincides with the $x - z$ plane of SRCS, the following restrictions should be imposed on the tensor coefficients: Only combinations of powers

[1] So, this x_{tg} correction is only correct upto first order.

of y_{fp} and $\tan\phi_{fp}$ with even values for the power sum $k+l$ occur in the calculation of $\frac{\Delta p}{p}$, $\tan\theta_{tg}$ while only odd values for $k+l$ are in the calculation of y_{tg}, $\tan\phi_{tg}$.

It is important to notice that $\frac{\Delta p}{p}$ for particles entering the spectrometer with $\theta_{tg} = \phi_{tg} = 0$ only depends on x_{fp}. The expression for the absolute momentum of these particles is given by:

$$p_f = p_{\text{centr}}(1 + \sum_{j=1}^{n} d_j {x_{fp}}^j) \qquad \text{where,} \tag{6.4}$$

p_{centr} central momentum setting of the spectrometer. Under ideal circumstances its dependence on the spectrometer magnetic field is linear. However, saturation effects will introduce non-linear terms:

$$p_{\text{centr}} = \sum_{j=0}^{m} \Gamma_j B^j$$

d_j spectrometer dispersion coefficients

This simple relation is due to the choice of our focal-plane coordinate system. In the next section where the calibration of the different coordinates is discussed we will see that as a result of this choice a special calibration procedure will give information about the important spectrometer dispersion coefficients.

6.5.1 Non-dispersive position y_{tg}

6.5.2 Angles

6.5.3 Momentum aberrations

To give you some idea how a scanning session with optimization will look like, an example is discussed in which the transfer tensor for calculation of momentum aberations is being optimized. In figure 6.5.3 the dispersive projection of the focal plane is drawn. Imagine that the spectrometer momentum setting is such that we observe the elastic peak of ^{12}C(e,e′). The ray with momentum p and $\theta_{tg} = 0$ is indicated by a dashed line. The position where it crosses the first wire plane is indicated by x_0. Rays with the same momentum and an infinitesimal small θ_{tg} cross this wire plane at the same spot and a relation can be established between momentum and x-position in that wire plane. However, rays entering the spectrometer collimator with the same momentum but a larger θ_{tg} will typically cross the wire plane at a different spot. In other words, the particle momentum can not be deduced solely from the dispersive position x, unfortunately all the focal plane vertex parameters (θ_{fp}, y_{fp}, ϕ_{fp}) are involved. It is now our task to describe this deviation Δx as a function of the focal plane vertex, the so-called transfer tensor for the momentum aberrations. Most spectrometer designs are such that the dependence on θ_{fp} is the most important one.

Below an example of a *kumac*-file is given in which the transfer tensor for the momentum aberations is being optimized. Seven measurements of an elastic electron scattering peak have been made at different values of $\Delta p/p$. After the assignment of the different input files and definitions

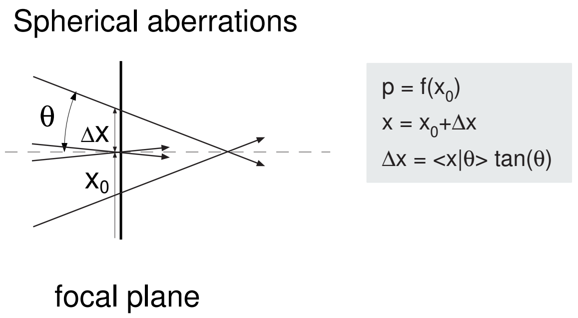

Figure 6.2: Explanation of spherical momentum aberations in the focal plane of a magnetic spectrometer. The dashed line is made by a ray with momentum p and $\theta_{tg} = 0$.

of *cut*'s, histograms and scatter plots are defined. Especially the θ_{fp}/dp_{kin} plots around the seven elastic peak positions are of interest since they will reveal the momentum aberration as a function of θ_{fp}. This is followed by a command indicating that at the end of the event-by-event scan an optimization of the momentum transfer tensor has to be performed.

```
define/batch on
set/file/output c12_full_conv1.hbook
set/file/database db_hrs_7x7
set/file/opt_database db_hrs_7x7_opt
set/file/detmap detmap.config
set/file/fit fit_full_conv1.inp
******************** Define cuts ***********************
define/cut/1d spec_e.y_tra-1 -0.144 0.144
define/cut/1d spece.x_tra-1 -0.749 0.749
define/cut/1d spece.th_tg-1 -0.07 0.07
define/cut/1d spece.ph_tg-1 -0.030 0.030
define/logical vdchole spece.y_tra-1&&spece.x_tra-1
define/logical accept_e spece.th_tg-1&&spece.ph_tg-1
******************** spectrometer HRS ******************
set/hitbits (+spece.s1)
set/auto_window off
set/spectrum/bins/x1 560
set/spectrum/window/x1 -0.07 0.07
spectra/save spec_e.th_tg
set/spectrum/bins/x1 320
set/spectrum/window/x1 -0.04 0.04
spectra/save spec_e.ph_tg
set/spectrum/bins/x1 500
set/spectrum/window/x1 -0.065 0.065
spectra/save spec_e.dp-1
spectra/save spec_e.dp_kin-1
set/spectrum/window/x1 -0.7 0.7
spectra/save spec_e.x_tra-1
set/ntuple 5000

**** dp-aberration plots
```

```
set/spectrum/window/x2 -0.5 0.5
set/spectrum/window/x1 -0.06 -0.04
spectra/save spec_e.th_tra/spec_e.dp_kin-1
set/spectrum/window/x1 -0.035 -0.0275
spectra/save spec_e.th_tra/spec_e.dp_kin-2 vdc_hole
set/spectrum/window/x1 -0.025 -0.0175
spectra/save spec_e.th_tra/spec_e.dp_kin-3 vdc_hole
set/spectrum/window/x1 -0.015 0.015
spectra/save spec_e.th_tra/spec_e.dp_kin-4 vdc_hole
set/spectrum/window/x1 0.015 0.0225
spectra/save spec_e.th_tra/spec_e.dp_kin-5 vdc_hole
set/spectrum/window/x1 0.0225 0.030
spectra/save spec_e.th_tra/spec_e.dp_kin-6 vdc_hole
set/spectrum/window/x1 0.040 0.060
spectra/save spec_e.th_tra/spec_e.dp_kin-7 vdc_hole

***** optimize momentum resolution in electron spectrometer

calibrate/optimize spec_e.dp_kin

******************** scanning and output *****************
set/file/header hdr_c12_dpm5.prm
file/scan ../monte_car/data_full_conv1_dpm5 LAST=1000 FIRST=1
set/file/header hdr_c12_dpm4.prm
file/scan ../monte_car/data_full_conv1_dpm4 LAST=1000 FIRST=1
set/file/header hdr_c12_dpm3.prm
file/scan ../monte_car/data_full_conv1_dpm3 LAST=1000 FIRST=1
set/file/header hdr_c12.prm
file/scan ../monte_car/data_full_conv1 LAST=1000 FIRST=1
set/file/header hdr_c12_dpp3.prm
file/scan ../monte_car/data_full_conv1_dpp3 LAST=1000 FIRST=1
set/file/header hdr_c12_dpp4.prm
file/scan ../monte_car/data_full_conv1_dpp4 LAST=1000 FIRST=1
set/file/header hdr_c12_dpp5.prm
file/scan ../monte_car/data_full_conv1_dpp5 LAST=1000 FIRST=1 CHOPT=-o

spectra/write
quit
```

Which parts of the tensor should be optimized is indicated in the database, see appendix B. The first number in each line of the tensor elements indicates whether it is kept fixed or is being fitted. A summary of the fitting is listed in a file which name is deduced from the file name `fit_input`, f.i. `momentum_aber.inp` produces `momentum_aber.res`.

In figure 6.3 the result of such an optimization is shown. On the left side the scatter plot $\theta_{fp}/dp/p$ is shown for elastic ^{12}C(e,e′) scattering without any aberration corrections. On the right is shown the same data but now after correction for momentum aberations.

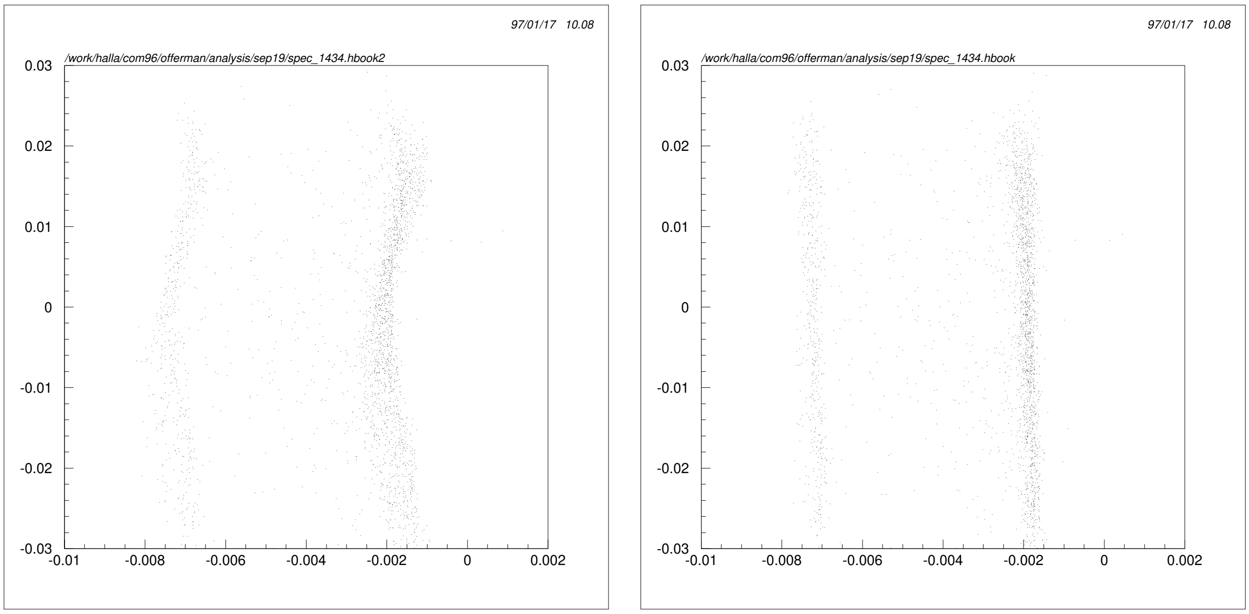

Figure 6.3: Two scatter plots showing θ_{rot} versus dp/p for elastic ^{12}C(e,e′) scattering. In the one on the left side, no aberration corrections have been performed on the calculated momentum. The plot on the right is the result of an optimization like the example shown in this section.

6.6 Coincidence time

6.7 Missing energy

6.8 Adding a new optimization

Chapter 7

Espace Reference Manual

Event Scanning Program for hall A Collaboration Experiments

7.1 Espace/Debug

This menu contains several switches controling output for debugging purposes. At the moment the CODA event stream and the VDC tracks can be monitored.

vdc [vdc]

`vdc` C "vdc tracking" `D='off'`
Possible `vdc` values are:
`off`

`on`

Switch the VDC debug mode. In the VDC debug mode, a graphics window will pop up and for each CODA event a track will be shown for each cluster in the U1,V1,U2 and V2 plane.

coda [coda]

`coda` C "coda" `D='off'`
Possible `coda` values are:
`off`

`on`

Switch the CODA debug mode. In the CODA debug mode, part of the raw CODA data is displayed.

sync [sync]

`sync` I "synchronization" `D=0 R=0:3`
Switch the synchronization debug mode. In the SYNC debug mode, CODA information is displayed together when synchronization problems appear.

hbook [hbook]

`hbook` C "hbook" `D='off'`
Possible `hbook` values are:
`off`

`on`

Switch the HBOOK debug mode. In the HBOOK debug mode, HBOOK memory headers are displayed.

7.2 Espace/Set

Menu to assign parameters to ESPACE variables.

test [test]

`test` C "test mode" `D='off'`
Possible `test` values are:
`on`

`off`

Switch the test mode. If the test mode is on, the command syntax and the existance of data files is checked. No scanning or optimizing is performed.

auto_window [awindc]

`awindc` C "auto-windowing" `D='on'`
Possible `awindc` values are:
`on`

`off`

Switch the automatic histogram windowing. If the auto-window mode is on, the LOW_X, HIGH_X and NUM_X parameters are put on the default values for the particular variable to be histogrammed.

hitbits hitbits

`hitbits` C "hitbits" `D='␣'`
Choose the condition, which detector should have fired or not, that is imposed on all histograms defined after this point.

ntuple [ntuple]

`ntuple` C "NTUPLE format" `D='off'`
Possible `ntuple` values are:
`on`

`off`

If the NTUPLE switch is on, the data are store in NTUPLE format, otherwise it will be a multi-dimensional histogram.

trig_cor [trig_cor]

`trig_cor` C "vdc com. stop corr." `D='on'`
Possible `trig_cor` values are:
`on`

`off`

If TRIG_COR is switched on, the VDC TDC drift times are corrected for time of flight to thetiming scintillator and signal propagation through this scintillator.

cosmic_rays [cosmic_rays]

`cosmic_rays` C "cosmic corr." `D='off'`
Possible `cosmic_rays` values are:
`on`

`off`

If COSMIC_RAYS is switched on, the VDC TDC drift times are corrected with an opposite sign since the cosmic rays drift from timing scintillator to VDC.

iterate_vdc iterate_vdc

`iterate_vdc` I "iterate VDC" `D=1 R=1:4`
Choose the number of iterations in the determination of a track through a VDC plane. If ITERATE_VDC ⟩ 1 the global angle coming from two wire planes is used in the calculation of several corrections.

7.3 Espace/Set/file

Assign file names.

header hdr_file

`hdr_file` C "header file" `D='␣'`
Set the name of the header file.

database db_file

`db_file` C "database file" `D='␣'`
Set the name of the database file.

opt_database optdb_file

`optdb_file` C "opt database file" `D='datab_opt'`
Set the name of the optimized database file in which results are stored of an optimization triggered with the FIT command.

detmap detmap_file

`detmap_file` C "detmap_config file" `D='tstand_detmap.config'`
Set the name of the detector configuration file. This file describes how the raw data of the data acquisition code CODA should be interpretted.

fit_input fit_file

`fit_file` C "fit input file" `D='fit.inp'`
Set the name of the input file containing parameters for optimization. This file contains starting parameters for fitting th_tg, ph_tg, y_tg, t_c, e_miss and e_miss/p_miss.

output output_file

`output_file` C "output file" `D='espace.hbook'`
Set the name of the HBOOK output file containing histograms and Ntuples.

7.4 Espace/Set/timing

Assign trigger definition parameters.

e plane [side]

`plane` C "scintillator plane" `D='s1'`
`side` C "paddle side" `D='left'`
Possible `plane` values are:
`s1`

`s2`

Possible `side` values are:
`left`

`right`

Choose the number scintillator plane and the paddle side that will define the trigger timing for e-spectrometer.

h plane [side]

`plane` C "scintillator plane" `D='s1'`
`side` C "paddle side" `D='left'`
Possible `plane` values are:
`s1`

`s2`

Possible `side` values are:
`left`

`right`

Choose the number scintillator plane and the paddle side that will define the trigger timing for h-spectrometer.

7.5 Espace/Set/type

Assign CODA event types

e type

`type` I "e-spec type" `D=1 R=1:16`
Choose the CODA event type for single e-spectrometer event.

h type

`type` I "h-spec type" `D=2 R=1:16`
Choose the CODA event type for single h-spectrometer event.

coin type

`type` I "coin type" `D=4 R=1:16`
Choose the CODA event type for coincidence event.

scal type

`type` I "scaler type" `D=15 R=1:16`
Choose the CODA event type for scaler event.

7.6 Espace/Set/fit

Assign fit parameters.

npts npts_file

`npts_file` I "max points/file" `D=2000 R=0:2000`
Set the maximum number of data points accumulated in one file for optimization.

7.7 Espace/Set/fit/time

Assign control parameters for the spectrometer path length fit.

paddle_e [fit_paddle_e]

`fit_paddle_e` C "fit time-offset paddles spec e" `D='off'`
Possible `fit_paddle_e` values are:
`on`

`off`

Fit the time offsets of the scintillator used in the trigger of the electron spectrometer when t_c is being optimized.

paddle_h [fit_paddle_h]

`fit_paddle_h` C "fit time-offset paddles spec h" `D='off'`
Possible `fit_paddle_h` values are:
`on`

`off`

Fit the time offsets of the scintillator used in the trigger of the hadron spectrometer when t_c is being optimized.

time_walk [fit_timewalk order]

`fit_timewalk` C "fit time walk coefficients" `D='off'`
`order` I "order of polynome" `D=1 R=1,2,3`
Possible `fit_timewalk` values are:
`on`

`off`

Fit the time walk of the timing scintillator signals. It is assumed that it is the same for all scintillators of the electron and hadron spectrometer when t_c is being optimized.

7.8 Espace/Set/fit/momentum

Assign control parameters for the spectrometer momentum fit.

gamma_e [fit_gamma_e]

`fit_gamma_e` C "fit gamma spec E" `D='off'`
Possible `fit_gamma_e` values are:
`on`

`off`

Fit the electron spectrometer constant defining the central momentum.

gamma_h [fit_gamma_h]

`fit_gamma_h` C "fit gamma spec H" `D='off'`
Possible `fit_gamma_h` values are:
`on`

`off`

Fit the hadron spectrometer constant defining the central momentum.

7.9 Espace/Set/spectrum

Set histogram parameters.

7.10 Espace/Set/spectrum/window

Set range.

x1 low high

`low`	R	"x1 lower limit"
`high`	R	"x1 upper limit"

LOW and HIGH value of the x1 variable

x2 low high

`low`	R	"x2 lower limit"
`high`	R	"x2 upper limit"

LOW and HIGH value of the x2 variable

x3 low high

`low`	R	"x3 lower limit"
`high`	R	"x3 upper limit"

LOW and HIGH value of the x3 variable

x4 low high

`low`	R	"x4 lower limit"
`high`	R	"x4 upper limit"

LOW and HIGH value of the x4 variable

7.11 Espace/Set/spectrum/bins

Set number of bins.

x1 num

`num` I "number of bins x1" `D=1`
Number of X1 bins for REAL variable, binsize for INTEGER variable.

x2 num

`num` I "number of bins x2" `D=1`
Number of X2 bins for REAL variable, binsize for INTEGER variable.

x3 num

`num` I "number of bins x3" `D=1`
Number of X3 bins for REAL variable, binsize for INTEGER variable.

x4 num

`num` I "number of bins x4" `D=1`
Number of X4 bins for REAL variable, binsize for INTEGER variable.

7.12 Espace/Define

Menu to define logicals and variables.

init

Set all parameters back on their default values.

batch [batch]

`batch` C "batch mode" `D='off'`
Possible `batch` values are:
`on`

`off`

Stop execution if an error condition has occured.

reset hist

`hist` C "hist list" `D='all'`
Empty contents of histograms in list. Correct syntax for the list is f.i. all: empty all histograms, 1-3,10: histograms 1,2,3 and 10. The numbers refer to the histogram HBOOK identifier.

logical name str

`name` C "logical name"
`str` C "logical expr"
Define a new logical by combining existing logical through the following operations:

```
!   negate
&&  logical AND
||  logical OR
```

The order of operations can be guided by brackets.

variable str

`str` C "expression"
With this command, constants and new variables can be defined.
New variables are defined by combining existing variables through the following operations:

```
( ) + - * / ^ **
```

and intrinsic functions

```
abs     acos    acosd   asin    asind   atan    atand   cos
cosd    cosh    deg     exp     log     log10   rad     sin
sind    sinh    sqrt    tan     tand    tanh
```

The definition string should look like, NEW : operation(OLD1,OLD2). Notice the assignment with ':'.
Constants are defined by a string like, PI=3.142 . Notice the assignment with '='.

7.13 Espace/Define/cut

Define cuts n varaibles calculated on a event-by-event base.

1d var_1d xmin xmax

`var_1d`	C	“variable” `D='␣'`
`xmin`	R	“low range” `D=-100.`
`xmax`	R	“high range” `D=100.`

Define a window on a variable. This definition results in a logical which is TRUE if the variable value is inside the window and FALSE otherwise. The variable can be a scalar or an array. If it is an array the logical is TRUE if the contents of the whole array fulfill the condition.
Examples:

```
scalar cut:  /Define/cut/1d spec_e.ph_tg -0.03 0.03
array  cut:  /Define/cut/1d spec_e.u1.tdc  2000 2500
             /Define/cut/1d spec_e.u1.tdc[100;200] 2000 2500
The last cut is performed on the TDC values of wires 100 upto 200.
```

2d var_2d direction par

`var_2d`	C	“variable” `D='␣'`
`direction`	C	“cut direction” `D='above'`
`par`	C	“polynome parameters” `D='1,0,0'`

Possible `direction` values are:
`above`

`below`

Define a region in 2 dimensions. This definition results in a logical which is TRUE if the combination of the values of the 2 variables is on the right side of the given polynomial, which is either ABOVE or BELOW. Again variables can be scalars or arrays.
Example:

```
/Define/cut/2d spec_e.ph_tg/spec_e.th_tg above 0.,1.,0.
```

7.14 Espace/Spectra

Histograms commands.

save spec [condition]

`spec` C "spectrum list " `D='␣'`
`condition` C "logical expr" `D='␣'`
Define a histogram of a particular variable. In this command several histograms can be defined by giving a list of variables separated by commas. After the the variable list a logical expression can be given which has to be fulfilled for incrementing the histograms.
Examples:

```
 1-dim histogram: /Spectra/save spec_e.p_kin  spec_e.ph_tg&&spec_e.th_tg
 2-dim histogram: /Spectra/save spec_e.th_fp/spec_e.p_kin
spec_e.ph_tg&&spec_e.th_tg
 a '/Set/ntuple on ' before the last save statement would result in a
 ntuple instead of a histogram.
 1-dim histogram of an array variable:
                  /Spectra/save spec_e.u1.tdc[100;200]
 This results in 1-dim histogram containing the TDC values of wires 100-200.
 DO I = 1,368
    ISTA = [I]
    ISTO = [I]
    spectra/save spec_e.u1.tdc[[ISTA];[ISTO]]
 ENDDO
 This will give you a TDC histogram of each wire in the U1 VDC plane of the
 electron spectrometer. Notice that the KUIP macro language has been used.
```

rotate list rot

`list` C "list of spectra" `D='1-100'`
`rot` I "rotation number" `D=1`
Do not use this command.

write [list]

`list` C "list of spectra" `D='1-4999'`
Save the defined spectra to a file. Correct syntax for the spectra list is f.i. all: save all histograms, 1-3,10: histograms 1,2,3 and 10. The numbers refer to the histogram HBOOK identifier.

7.15 Espace/File

Menu containing commands related to the data file.

scan file [first last chopt rvf]

`file`	C	"file name"	`D=' '`
`first`	I	"first event"	`D=1`
`last`	I	"last event"	`D=-1`
`chopt`	C	"options"	`D=' '`
`rvf`	C	"review file"	`D='review.dat'`

Possible `chopt` values are:

`' '`

- `o` fit : Start fitting after scanning file
- `r` review : review file
- `w` write review : write review report after this file
- `f` filter : filter data input
- `l` loop : Do not open the file again but continue scan

Scan a file. Start a scanning session in which the previously defined histograms and logicals are updated event-by-event with the specified file. In the scan command parameters like the first interrupt, # of interrupts and several options can be specified.
Among the options are f.i. the possibility to incorporate the data from this file in an optimization (specified before through a /calibrate/fit statement) or to filter data from this file (specified through a /file/filter statement).
Example:

```
/File/scan file.dat 1 1000 -f
```

In this example the first 1000 interrupts of the file 'file.dat' are analyzed and used in te filter process.

review file [first max chopt rvf]

`file`	C	"file name"	`D=' '`
`first`	I	"first event"	`D=1`
`max`	I	"max event"	`D=-1`
`chopt`	C	"options"	`D=' '`
`rvf`	C	"review file"	`D='review.dat'`

Possible `chopt` values are:

`' '`

- `w` write review : write review report after this file
- `f` filter : filter data input

Review the detector coincidences. Issuing this command results in a review file in which a summary is given of the hitpattern of the detectors: how many times did a detector fire and how many coincidences and between which detectors are there.

option name

`name`	C	"option name"	`D=' '`

Execute a USER defined command. The command /File/option NAME executes the part in the user subroutine OPTION_DO linked to the string NAME as defined in the user subroutine OPTION_DEFINE.

filter file condition

`file`	C	"file name"	`D=' '`
`condition`	C	"logical expr"	`D=' '`

The name of the file in which the filtered data are stored is defined here and the LOGICAL condition for the filtering.

7.16 Espace/Calibrate

This menu contains commands defining the optimization to be performed after scanning a CODA data file.

optimize spec [condition]

`spec`	C	"spectrum name"	`D='␣'`
`condition`	C	"logical expr"	`D='␣'`

Start an optimization. Possible variables that can be optimized:

```
--------------------------+---------------------------------------------
spec_e.dp_kin             | Relative momentum after correction for
                          | aberations and kinematic broadening.
spec_e.th_tg/spec_e.ph_tg | Reconstructed theta and phi target coordinate
                          | of the electron spectrometer.
spec_h.th_tg/spec_h.ph_tg | Reconstructed theta and phi target coordinate
                          | of the hadron spectrometer.
spec_e.y_tg               | Reconstructed y target coordinate of the
                          | the electron spectrometer.
spec_h.y_tg               | Reconstructed y target coordinate of the
                          | the hadron spectrometer.
tc_cor                    | Coincidence time between e and h spectrometer
                          | after correction for time-of-flight and paddle
                          ! offsets.
e_miss                    | Missing energy.
e_miss/p_miss             | Missing energy and momentum for H(e,e'p).
spec_e.s1.adc_l_c         |
spec_e.s1.adc_r_c         |
spec_e.s2.adc_l_c         |
spec_e.s2.adc_r_c         |
spec_h.s1.adc_l_c         |
spec_h.s1.adc_r_c         |
spec_h.s2.adc_l_c         |
spec_h.s2.adc_r_c         |
spec_h.s3.adc_l_c         |
spec_h.s3.adc_r_c         |
spec_e.s1.tdc_l_c         |
spec_e.s1.tdc_r_c         |
spec_e.s2.tdc_l_c         |
spec_e.s2.tdc_r_c         |
spec_h.s1.tdc_l_c         |
spec_h.s1.tdc_r_c         |
spec_h.s2.tdc_l_c         |
spec_h.s2.tdc_r_c         |
spec_h.s3.tdc_l_c         |
spec_h.s3.tdc_r_c         |
--------------------------+---------------------------------------------
```

A LOGICAL condition can be specified for selection of the data to be used in the optimization.

Appendix A

Header file description

The header file contains information which is needed during the scan to construct some of the variables. At the end of the scan this information is saved with each histogram. This way it is available to the next programs in the analysis chain. An example of a header file for analyzing `C(e,e'` at 845 MeV is given below:

```
reaction c (e,e)
e0 r 1 845.0
particle_e c lepton
spece r 5 3.4730 +16.00 0.0 0.030 0.060
target r 5 0.05580 2.20 2 1 1
tar_angle r 1 +8.00
isotope1_name c 12C
isotope1 r 3 0.989 12.000 6
isotope2_name c 13C
isotope2 r 3 0.011 13.003 6
```

The order of the lines is not important. They are built up in the following way. The first string indicates the name of the item. It is followed by a character which should be either `c` which stands for character string or `r` which indicates that REAL's will follow. The next number gives the number of fields if REAL's follow. Finally, the values of the item should be given. Table A.1 gives a description of each item and its entries. Note that items followed by an '*' are mandatory, you will have to give it in the header file, otherwise an error message will appear.

Table A.1: Description of header file items

item	type	field	units	Description
react*	c			reaction string
e0*	r		MeV	accelerator electron energy
spece*	r			parameters of the electron spectrometer
		field	kG	magnetic induction of the dipole
		angle	deg	angular position of spectrometer in-plane
			deg	angular position of spectrometer out-of-plane
		solid		spectrometer collimator acceptances
			rad	half horizontal acceptance
			rad	half vertical acceptance
spece_offset	r			position of the electron spectrometer optical point in the hall coordinate system
		x	m	x component

Table A.1: (continued)

item	type	field	units	Description
		y	m	
		z	m	
spech*	r			parameters of the hadron spectrometer
		field	kG	magnetic induction of the dipole
		angle	deg	angular position of spectrometer in-plane
			deg	angular position of spectrometer out-of-plane
		solid		spectrometer collimator acceptances
			rad	half horizontal acceptance
			rad	half vertical acceptance
spech_offset	r			position of the hadron spectrometer optical point in the hall coordinate system
		x	m	x component
		y	m	
		z	m	
target*	r			target parameters
		d	g/cm^2 ; cm	total target thickness (model=1); target radius (model=3)
		rho	g/cm^3	target density
		niso		number of isotopes
		isop		isotope being investigated
		model		target model (1:foil; 2:circular; 3:cylindrical)
foil	r			target foil parameters
		d	g/cm^2	target thickness
		width	cm	foil width
		pos_x		
		pos_y		
		pos_z	cm	posdistance of foil from target center
tar_angle*	r			target angle parameters; only for model=1
		tangle	deg	angle between target c.s. z-axis and hall c.s. z-axis;
		fangle	deg	angle between normal to target foil and target c.s. z-axis
tar_offset	r			position of target center in the hall coordinate system; it is assumed that this is also the center of rotation for the target.
		x	m	x component
		y	m	
		z	m	
isotope1_name *	c			name of isotope 1
isotope1*	r			
		f		abundance ≤ 1.0
		A	amu	atomic mass
		Z		isotope number
isotope2_name*	c			name of isotope 2
isotope2*	r			
		f		abundance ≤ 1.0
		A	amu	atomic mass
		Z		isotope number
reconstruct	r			parameter controling extended target corrections
		0		It is assumed that the reaction happened in the center of the target
		1		The position information of the beam and the vertex reconstructed with the spectrometers are used to calculate the reaction point in the target.
		2		In addition to option 1 the vertices of all spectrometers are corrected for the fact that the particle did not emerge from the optical point.

All values are initialized on zero. Although `spece` and `spech` are mandatory items, entry of one will suffice, the same is true for the isotope information.

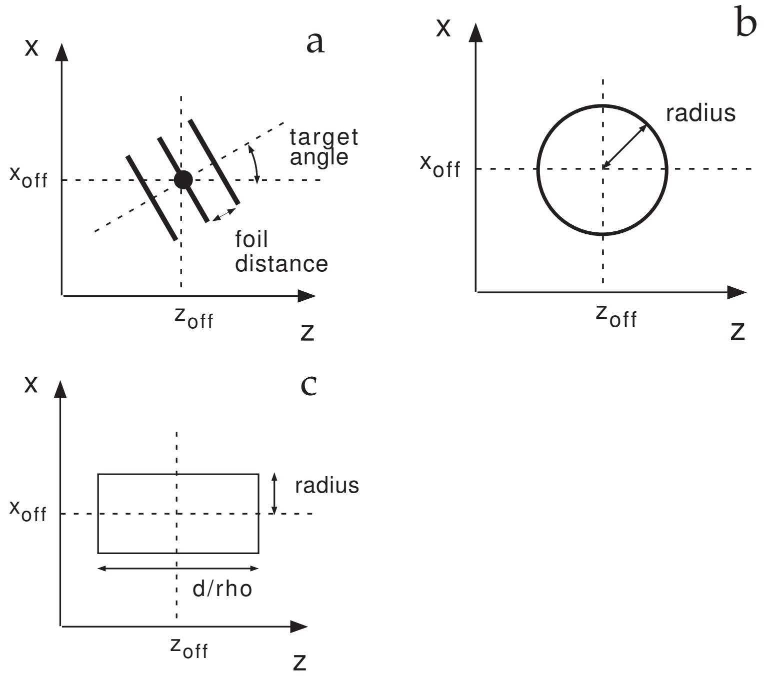

Figure A.1: .

In figure A.1 the configuration of the three target models is shown. Indicated are the different parameters describing the target. Realize that if target `model`=1 is chosen and a `foil` card is not given, the number of foils is assumed to be 1 with a target thickness as given in the `target` card. When the `foil` card is given, the target thickness in the `target` card is ignored.

Appendix B

Database file description

The database file contains constants of the spectrometers and their detectors. In the future this database can be accessed with a package developed around HEPDB. For the time being it is a plain ASCII file. We describe here how the part for spectrometer E looks like, the part for the other spectreometer has the same structure.

The first seven lines of the database file contain drift chamber information and should look as follows:

```
10.0 10.0 10.0 10.0
5.7e+04 5.7e+04 5.7e+04 5.7e+04
4096.  4096.  4096.  4096.
5.0e-10 5.0e-10 5.0e-10 5.0e-10
2.0 0.0 0.0 0.0 -4.479051e-5 1.588897e-4 3.861646e-4 -6.223269e-5
```

The meaning of these lines is:

line	Description
1	Estimate of drift time resolution (chan).
2	Electron drift velocity (m/s).
3	Range of the VDC-TDC
4	VDC-TDC resolution (s/chan)
5	coefficients of a polynomial for drift length correction in the VDC (m)

Next lines contain for each wire plane and each wire, the wire number and the TDC offsets. Each line contains information for five wires, f.i.:

```
1 503.0 2 503.0 3 503.0 4 503.0 5 503.0
```

If the wire number is negative then this wire will not be used in the analysis. This should be followed by lines containing wire number and position:

```
1 1.42057 2 1.41557 3 1.41057 4 1.40557 5 1.40057
```

Now information about the trigger detectors comes (first S1, followed by the same information for S2)

```
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
0.0 1.0 0.0 1.0 0.0 1.0 0.0 1.0 0.0 1.0 0.0 1.0
0.0 1.0 0.0 1.0 0.0 1.0 0.0 1.0 0.0 1.0 0.0 1.0
0.700
1.9e+8
1.0e-10
0.00 42.46 37.18 36.65 39.97 36.02
30.57 16.25 37.34 18.28 74.78 66.88
0.0
0.0 1.0 0.0 1.0 0.0 1.0 0.0 1.0 0.0 1.0 0.0 1.0
0.0 1.0 0.0 1.0 0.0 1.0 0.0 1.0 0.0 1.0 0.0 1.0
```

The meaning of these lines is:

line	Description
1-2	Parameters for correction of amplification of the ADC signal for each Cherenkov mirror. For each mirror first the offset is given and then an amplification factor.
3	inverse of attenuation length of light in the scintillator (m^{-1})
4	speed of light in scintillator (m/s).
5	TDC resolution in scintillator plane (s/chan)
6-7	time offset of each scintillator paddle on both sides (chan)
8	Parameter for correction timewalk effects in the scintillator timing with the ADC signal.
9-10	Parameters for correction of amplification of the ADC signal for paddle on the right side, followed by the left side. For each paddle an offset and an amplification factor is given.

At the end of the database the transport information of the spectrometer is given. The first number is the radius of the central ray through the dipoles. Then the coefficients for polynome of the spectrometer constant Γ (MeV/kG) are given and it is followed by the dispersion constants d_i which together link dispersive position x_{fp} and momentum p through the relation:

$$p = \left(\sum_{i=0}^{2} \Gamma_i B^i\right)\left(\sum_{i=1}^{6} d_i x^i_{fp}\right)$$

The number following the dispersion constants indicates the definition of the transport matrix:

Number	Matrix definition
1	Transport elements using focal plane coordinates defined wrt local central ray.
2	Transport elements using focal plane coordinates defined wrt the detector system

```
1.400
41.972 0.0 0.0
1.2063E-01 1.1105E-02 -1.0599E-03 -1.1160E-04 -1.2408E-04 0.0
2
```

It is completed with a list of the transfer elements. Each line contains the following information. The first INTEGER tells whether the transfer element will be fitted when an OPTIMIZE command has been given. A `0` means keep fixed, a `1` means fit it and `-1` indicates that when reconstructing the vertex at the target the first time use the values in the database but keep them fixed on zero during the fit. This is followed by the x_{fp} dependence of the particular transfer coefficient and its name:

$$M(x_{fp}) = \sum_{i=1}^{6} M_i x_{fp}^{i-1}$$

For example, the line for $\langle\theta|\theta\rangle$ looks like

```
 1 1 0 0 -5.748E-1 1.254E-3 -1.028E-5 6.655E-8 0.000E+0 -1.21E-12 'T100'
```

Appendix C

Algorithms to transform vectors between different coordinate systems

In this section we will use the following abbreviations for the different coordinate systems:

SDCS	Spectrometer Detector Coordinate System
SFPCS	Spectrometer Focal Plane Coordinate System
SRCS	Spectrometer Reconstructed Coordinate System
HLCS	Hall A Laboratory Coordinate System

Below a set of FORTRAN routines is described that transforms a vector between the different coordinate systems. These routines are part of the `kinematics`-library. Sources are located in the directory `$HALLA_DIR/src/kinematics` and the library is in the directory `$HALLA_DIR/lib/$OSNAME`.

ROUTINE	DESCRIPTION
`beam_to_transport`	This routine transforms a LCS-vector in spherical coordinates to the SRCS in transport coordinates. This transformation is defined by a call to the entry `define_beam_to_transport`.
`define_beam_to_transport`	Initialization routine for the transformation used in `beam_to_transport`.
`transport_to_beam`	This routine converts a SRCS-vector in transport coordinates to the LCS in cartesian coordinates. This transformation is defined by a call to the entry `define_trans`.
`define_transport_to_beam`	Initialization routine for the transformation used in `transport_to_beam`.
`ctrans`	Routine transforming geometrical angles to spherical angles and vice versa.
`spherang`	Routine transforming geometrical angles to spherical angles and vice versa. This routine gives also the option to switch LCS and a variant of the LCS with the z-axis along the momentum transfer q.
`newang`	

Appendix D

COOLHANDS Programming Guide

Joe Mandeville / Eddy Offermann

D.1 Concept

The COOLHANDS system is designed to attack the problem of scanning and histogramming event-mode data in a flexible and *general* way. The program assumes only that the data contain events in which one can define

1. the number of words in the event,
2. the detector firing sequence (or hitbit pattern),
3. a set of variables which the user may wish to histogram and make cuts upon.

If the data can be represented in these general terms, then it can be scanned with COOLHANDS by attaching a few simple subroutines. The program is designed so that a user-programmer needs to know very little about the bulk of the program to interface a particular system to it.

D.2 FORTRAN components

The executable scanning program is formed by linking several object files together. Some of the components have been made into libraries. The following shows the *makefile* which is used to obtain the executable of a program called *analyzer*.

```
LOC_HALLA_LIB = /work/halla/com96/offerman/halla/lib/SunOS
LDFLAGS        = -L$(LOC_HALLA_LIB)
LIBS          = $(LOC_HALLA_LIB)/coolhands.o
                -lcool_default -lfcalc -lmbook4d -lutil -lvms
FC            = f77
LINKER        = f77
MAKEFILE      = Makefile
OBJS          = analyzer.f
PROGRAM       = analyzer
SRCS          = analyzer.f
all:          $(PROGRAM)
$(PROGRAM):   $(OBJS)
```

```
$(LINKER) $(LDFLAGS) $(OBJS) $(LIBS) -o $(PROGRAM)
```

The source file `analyzer.f` should contain the main program and a group of subroutines describing the processing of an event. Further on in this document is described the contents of the `analyzer.f` file. If the user is not interested in defining some of the routines one has to include the library `cool_default.f` at the end of the library list. It contains a default set of user subroutines and by including it in the linking process, routines not found in the user files, will be picked up from here.

The object file `coolhands.o` contains the important routines for scanning, cutting and histogramming the event stream. The user-supplied file `analyzer.f` contains all routines which are specific to the data acquisition of the event stream. Included here are an input subroutine to read data and interpret the event format and a routine to define and return variables for scanning and histogramming. These routines are described later in this document.

The library of default routines cool_default exists so that options may be added which the user need not know about. Many of the subroutines which may be included in the user set are not essential and the default routines may suffice. Since object libraries such as this only look for routines which have not been found in the linking process, the user may supercede any or all of these routines in the user set.

The histogram manipulation library `mbook4d` provides an interface to the CERN `hbook` package for initializing, defining, filling, and storing histograms. The *util* library contains very general routines which are used in many programs. Most of the routines in this library are all simple 'independent' routines which do not require routines from other libraries.

D.3 FORTRAN unit numbers and common blocks

The FORTRAN components of COOLHANDS, excluding the user-written subroutines (eg, analyzer.f), use the following FORTRAN unit numbers:

- 5: Standard program input (STDIN)
- 6: Standard program output (STDOUT)
- 29: graphical cut info are being used.

The following named common blocks are used in the `coolhands` system and should not be redefined by the user.

common	file
/cool_i/ /cool_r/ /cool_c/ /cool_l/ /cool_rec/	coolhands/coolhands.h
/defvars/	coolhands/defspec.h
/mbook_i/ /mbook_l/ /mbook_s/ /mbook_comm/	mbook4d/mbook.h
/menu/	util/util.h
/fcalc1/ /fcalcpass/	fcalc/fcalc.h

The problem of duplicating subroutine names should be taken care of by the compiler.

D.4 Use Supplied Subroutines

The `coolhands` user-programmer must supply a set of routines. All but the first two of these listed below may be included by linking to the default set cool_default.f.

- `input`
- `variable`
- `det_sub.f`
- `filter_close`
- `filter_event`
- `filter_open`
- `index_range.f`
- `insert_header`
- `optimize_do`
- `optimize_event`
- `optimize_ini`
- `option_do`
- `option_setup`
- `user_exit`
- `user_init`

The `input` routine reads the event format for a file name which is passed. It is called after a `scan` command. This routine should read a block of data and, for each event, call one or both of `scansub` and `revsub`. `scansub` makes cuts and increment histograms, whereas `revsub` acquires statistics on the data. The routines `scansub` and `revsub`, which are found in the main set of `coolhands` routines, call the second required user routine, `variable`, to get information for a given spectrum from each event. This routine should communicate with the `input` routine via user-defined common blocks. The routine `user_init` is called upon program entry and after the command `initialize`. Its function is to initialize the histogram package or to begin a `cpu` clock or define names needed for the `option` command to work. The routine `user_exit` is called only after the `exit` command and may be used by the user for miscellaneous purposes. For instance, one might stop the `cpu` clock and print the elapsed time. After the return from this routine to the main COOLHANDS program, a FORTRAN stop is encountered. The `filter_event` and `filter_open` routines allow the user to accomplish filtering.

What follows now are examples of the routines the user has to provide to COOLHANDS for some fictitious data stream.

SUBROUTINE INPUT(SCANFILE,TEST,SCAN,REVIEW,LOOP,FIRST,LAST)

```
        CHARACTER SCANFILE*(*)
        LOGICAL TEST,SCAN,REVIEW,LOOP
        INTEGER FIRST,LAST
```

The function of this routine is to read the data, interpret the event format, and call the proper COOLHANDS routines to perform the action desired. `scanfile` is name of the data file to be scanned. `test` is a logical telling the routine that it is to simply open the file to test for its existence and output an error message if necessary before closing the file and returning. `scan` tells whether or not to call a COOLHANDS routine for each event to scan the file, make cuts, and accumulate histograms. `review` tells whether to call the proper COOLHANDS routine for reviewing each file to accumulate statistics. See the reference chapter for explanations of these functions.

The routine should be organized to first open the file, then to return if `test` is `.true.` or otherwise to read in a block of data. This data should then enter a loop in which each event is interpreted and placed into a common block (e.g., `common /event/`) which is shared with the user-supplied function `variable` described below. This common block should contain the number of words in the event, the hitbit pattern, and a list of variables required by the function `variable`. The hitbit pattern is a 32 bit integer in which each bit represents a detector and is set if that detector fired. An example of such a common is shown in the example routine below.

After an event has been decoded into the common block within this loop, the user should call the appropriate COOLHANDS routines:

```
        IF (SCAN) CALL SCANSUB
        IF (REVIEW) CALL REVSUB
```

An example of an input subroutine is shown below.

```
        SUBROUTINE INPUT(SCANFILE,TEST,SCAN,REVIEW,LOOP,FIRST,LAST)

        IMPLICIT NONE

        CHARACTER SCANFILE*(*)
        LOGICAL TEST,SCAN,REVIEW,LOOP
        INTEGER FIRST,LAST

C This routine reads and interprets the event buffer from interrupt
C FIRST upto LAST and sends events to
C subroutines SCANSUB or REVSUB to either execute desired cuts in spectra
C or get statistics on the data, respectively.
C      SCANFILE........ the name of the event file to be scanned
C      TEST ........... a logical which, if true, tells the routine to
C                       simply try to open the file, close it, and return
C      SCAN ..........  a logical which, if true, tells the routine to
C                       call for each event the routine SCANSUB
C      REVIEW ........  a logical which, if true, tells the routine to
C                       call for each event the routine REVSUB
C      LOOP             a logical indicating whether the routine INPUT
C                       is being called in a loop and therefore the input
C                       file should not be opened at each entry.
C      FIRST            first interrupt to be analyzed
C      LAST             last interrupt to be analyzed
```

```
        INTEGER NRECORD
        PARAMETER(NRECORD=500)

C The data will be read into the following array

        INTEGER IBLOCK(NRECORD)

C To make the example simple, suppose we have fixed records of 500 I*4
C words per record and events with a constant length of 100 words.  The
C event format will be (1) the number of words in the event, (2) the event
C type, (3) the event hitbit pattern, (4) 32 ADC words, each corresponding
C to a bit in the hitbit pattern, (5) 32 TDC words, each corresponding to
C a bit in the hitbit pattern, (6) 33 electron words, where the first electron
C word is the on-line calculated fine momentum and other electron words may
C be drift times, ...etc.  Each event will be placed into the following
C common block to be passed to function VARIABLE

        INTEGER NWIE,TYPE,HITBIT,ADC(0:31),TDC(0:31),ELE(0:31),FINE_MOM
        COMMON/EVENT/NWIE,TYPE,HITBIT,ADC,TDC,ELE,FINE_MOM

C miscellaneous declaration
        INTEGER IWORD,JEVENT,J,NR_EVENT
C open the event file to be scanned
        OPEN(1,FILE=SCANFILE,FORM='UNFORMATTED',STATUS='OLD',READONLY,
     &  ERR=1000)
C if the "test" command is in effect, then it is enough to open the file and
C make sure it is there; in this case, close and return.
        IF (TEST) THEN
           CLOSE(1)
           RETURN
        ENDIF
        NR_EVENT=0
C read in an event record
 10     READ(1,END=100,ERR=2000) (IBLOCK(J), J=1,NRECORD)
        IWORD=0
C decode the 5 events in the record and "scan" or "review" each event
        DO JEVENT=1,NRECORD/100
           NR_EVENT=NR_EVENT+1
           IF (NR_EVENT.GE.FIRST) THEN
              NWIE=IBLOCK(IWORD+1)
              TYPE=IBLOCK(IWORD+2)
              HITBIT=IBLOCK(IWORD+3)
              DO JDETECTOR=0,31
                 ADC(JDETECTOR)=IBLOCK(IWORD+JDETECTOR+4)
                 TDC(JDETECTOR)=IBLOCK(IWORD+JDETECTOR+36)
                 ELE(JDETECTOR)=IBLOCK(IWORD+JDETECTOR+69)
              ENDDO
              FINE_MOM=IBLOCK(IWORD+68)
C The event is decoded, now scan or review:
              IF (SCAN)   CALL SCANSUB
              IF (REVIEW) CALL REVSUB
C move onto the next event in the record
              IWORD=IWORD+NWIE
           ENDIF
        ENDDO
C now read another record
        IF (NR_EVENT.LT.LAST) GOTO 10
C
  100   RETURN
```

```
C the following calls are to a routine in the MLIB library.
 1000   CALL DERROR(' Error opening the event file.')
        RETURN
 2000   CALL DERROR(' Error reading the event file.')
        END
```

REAL FUNCTION VARIABLE(IVAR,IDET,INDX,NWORDS,EVENT_HBIT,SETBIT,WEIGHT)

```
INTEGER IVAR,IDET,INDX,NWORDS
LOGICAL*1 EVENT_HBIT(*)
LOGICAL SETBIT
REAL    WEIGHT
```

This routine is the function which returns event variables.

- `ivar` specifies the variable type to return by indicating the index in character array `goodsp` below. The variable type `adc`, for example, would be named in `goodsp` and assigned within the routine as `variable = adc(idet)`, where the `adc` array is contained in the event common block. There are two special cases. If `ivar=0`, `intvar` should return the number of words in the event. If `ivar` is less than 0, `intvar` should return the hitbit pattern.
- `idet` is the detector number.
- `intvar` returns the integer variables which specify the hit pattern and the number of words in the event.
- `setbit` is an output logical which tells the calling routine if the bit `idet` was set in the hit pattern for the event. Additionally, special cuts may be accomplished by using SETBIT.
- `weight` is the weight to give the event in a histogram.

The variables returned by this routine are defined in the common `defspec`, which is the only common block shared with the main program COOLHANDS. This common block is contained in the include file `defspec.h`.

```
structure /var_type/
   character*12 name
   integer intbin
   logical array
   logical subdet
   logical*1 all_det(32),all_det_opt(32)
   logical*1 enf_det(32)
   real range(2)
end structure

structure /det_type/
   character*80 string
   integer number
end structure

record /det_type/ defdet(32)
record /var_type/ defvar(10)
character var_txt(10)*80,det_txt(32)*80

common /defvars/ defdet,defvar,var_txt,det_txt
```

- `goodsp` defines the names of the spectra as they will be identified in input by the user.

- `iadet` is an array of integers which specifies allowed detectors for each spectrum; if detector 7 is allowed to be associated with spectrum 4, then bit 7 of iadet(4) will be set.
- `iadetopt` is an array of integers which specifies allowed detectors for each spectrum which can be optimized.
- `range` provides low and high limits for each spectrum. The spectrum range is used only in by the *review* facility and as a window for the spectrum when *auto-window* is true. Out-of-bounds variables are NOT corrected to the nearest window.
- `intbin` is an array which types each defined spectrum as an integer one or a real one. If the i^{th} element of `intbin` is 1, the corresponding spectrum will be declared to be an integer one and thus integer binning will be done. Otherwise, real binning will be used. If the spectrum is set up as an integer one, the values will still be returned in the real `variable` but should be truncated to be "integer reals". See the COOLHANDS User Manual for a description of each type of binning.

The variable `ivar` in the calling sequence corresponds to the index of dimension `NSTYPES` for each of these arrays. Sample data statements to define the spectra are:

```
      DATA GOODSP /'adc','tdc','fm','mom',26*' '/
      DATA IADET  / -2 , -2 , 1 , 1, 26*0 /
      DATA IADETOPT  / 30*0 /
      DATA INTBIN /1,1,1,0,26*0/
      DATA RANGE  / 0.,2047. , 0.,2047. , 0.,2047. , 0.,5. , 52*0. /
```

The above data statements define 4 variables:

- `adc` is an integer variable within an allowed range of 0 to 2047; it may correspond to detectors 1-31 but not to detector 0
- `tdc` is an integer variable within an allowed range of 0 to 2047; it may correspond to detectors 1-31 but not to detector 0
- `fm` is an integer variable within an allowed range of 0 to 2047; it may correspond to detector 0 but not to detectors 1-31
- `mom` is a real variable with an allowed range of 0. to 5.; it may correspond only to detector 0

An example of the function is

```
      REAL FUNCTION VARIABLE(IVAR,IDET,INDX,NWORDS,EVENT_HBIT,
     >                                                  SETBIT,WEIGHT)

C input:  ivar,idet
c output: variable,intvar,setbit,weight

      IMPLICIT NONE

      INTEGER IVAR,IDET,INDX,NWORDS
      LOGICAL*1 EVENT_HBIT(*)
      LOGICAL SETBIT
      REAL    WEIGHT
```

```
C include the common block /defspec/

      INCLUDE 'DEFSPEC.H'
C the following common block should be shared with the input routine
      INTEGER NWIE,TYPE,HITBIT,ADC(0:31),TDC(0:31),ELE(0:31),FINE_MOM
      COMMON /EVENT/ NWIE,TYPE,HITBIT,ADC,TDC,ELE,FINE_MOM
C miscellaneous declarations
      REAL UNBIN,CALC_MOM
      INTEGER ISEED
      DATA ISEED/536543/
C define the spectra
      DATA GOODSP   /'Adc','Tdc','Fm','Mom',16*' '/
      DATA IADET    / -2 , -2 , 1 , 1, 16*0 /
      DATA IADETOPT / 20*0 /
      DATA INTBIN   /1,1,1,0,16*0/
      DATA RANGE    / 0.,2047. , 0.,2047. , 0.,2047. , 0.,5. , 32*0. /
C initialize returned values
      WEIGHT = 1.0
      VARIABLE = 0.
      INTVAR = 0.
C did the detector IDET fire for this event?
      SETBIT = BTEST(HITBIT,IDET)
C return the appropriate variable for each spectrum type
      IF (IVAR.EQ.0) THEN
         INTVAR = NWIE
      ELSE IF (IVAR.LT.0) THEN
         INTVAR = HITBIT
      ELSE IF (.NOT.SETBIT) THEN
C
C        Detector did not fire.
C
      ELSE
C
C        Detector fired.
C
         IF (IVAR.EQ.1) THEN
            VARIABLE = ADC(IDET)
         ELSE IF (IVAR.EQ.2) THEN
            VARIABLE = TDC(IDET)
         ELSE IF (IVAR.EQ.3) THEN
            VARIABLE = FINE_MOM
         ELSE IF (IVAR.EQ.4) THEN
C the variable unbin spans one channel and simply "unbins" data to
C prevent problems associated with binning discrete data with bin sizes
C not the same as the original binning done by the tdc's and adc's.
C Note that a second course of action could have been taken here.  The
C variable "fine_mom" could have set up as an integer variable by
C multiplying the actual calculated momentum by a number sufficiently
C large to gain the desired precision and then truncating it to an
C integer value.
            UNBIN = 0.5*( 1 -2.*RAN(ISEED))
            FINE_MOM = FINE_MOM+UNBIN
C the routine "calc_mom" below should be a routine to take the fine
C momentum and spectrometer field and generate the electron momentum
            VARIABLE = CALC_MOM(FINE_MOM)
         ELSE
            STOP 'Illegal spectrum type in module scan'
         ENDIF
      ENDIF
```

```
C
      RETURN
      END
```

SUBROUTINE USER_INIT(AWIND,NDIM,W1,W2,NBIN,DO_NTUPLE)

This routine is called upon program entry and simply defines default values for some program variables upon entry. Other functions (like beginning the cpu clock or defining names for the 'option' command) may be included here. The default routine which is provided in COOL_DEFAULT library is shown below.

```
      SUBROUTINE USER_INIT(AWIND,NDIM,W1,W2,NBIN,DO_NTUPLE)
c
c----------------------------------------------------------------------
c
      IMPLICIT NONE

c all passed variables are output
      LOGICAL AWIND,DO_NTUPLE
      INTEGER NDIM,NBIN(*)
      REAL W1(*),W2(*)

      LOGICAL first
      INTEGER jdim

      SAVE FIRST

      DATA FIRST /.true./

c for the 1st time through...
      IF (FIRST) THEN
c call a routine to begin accumulating cpu and real time (this routine is
c in MENU library)
         CALL CPUBGN
c set up the names for the option command
         CALL OPTION_SETUP
         FIRST = .FALSE.
      ENDIF
c assign default values for windows, # bins, and ascii logical
      AWIND = .FALSE.   ! autowindow
      DO JDIM = 1,NDIM
         W1(JDIM)   = 0.    ! lower window
         W2(JDIM)   = 2048. ! upper window
         NBIN(JDIM) = 50    ! number of bins
      ENDDO
      DO_NTUPLE = .FALSE.  ! multi-dim histograms, no ntuple format

      RETURN
      END
```

SUBROUTINE USER_EXIT

The following routine is called upon an `quit` command and may be used for any purpose. The default routine which is provided in COOL_DEFAULT library is shown below.

```
      SUBROUTINE USER_EXIT

      IMPLICIT NONE

      REAL      TIMCPU,TIMCLK
      CHARACTER TIME*8,DAY*8,TEXT*80
C find the time of day and stop the cpu clock (these routines are in MLIB)
      CALL TOD(TIME,DAY)
      CALL CPUEND(TIMCPU,TIMCLK)
C
      WRITE(TEXT,100) TIMCPU/60.,TIMCLK/60.
      WRITE(6,110) TIME,DAY,TEXT
C
 100  FORMAT('   CPU time (min):',F6.2,'   Real time (min):',F6.2)
 110  FORMAT('   Ending time: ',A,'   Ending day:   ',A,/A)
C
      RETURN
      END
```

SUBROUTINE FILTER_OPEN(IUNIT,FNAME)
SUBROUTINE FILTER_EVENT(IUNIT)
SUBROUTINE FILTER_CLOSE(IUNIT)

If one wishes to filter data based upon certain cuts, these routines can accomplish that. `filter_open` is called to allow the user to open the filter file. `filter_event` is called each time an event has passed the specified cuts for that filter. Multiple filters may be accomplished simultaneously. The default routines supplied in the COOL_DEFAULT library are shown below.

```
        SUBROUTINE FILTER_EVENT(IUNIT)

        IMPLICIT NONE

        INTEGER IUNIT
        CHARACTER*(*) FNAME

C this entry is called only when the event has passed the conditions
C placed upon the filter specified by IUNIT
        RETURN
C
        ENTRY FILTER_OPEN(IUNIT,FNAME)
C this entry opens filter files
        RETURN
        ENTRY FILTER_CLOSE(IUNIT)
C this entry closes filter files
        RETURN
        END
```

SUBROUTINE OPTION_DO(IOPTION,ARGUMENT)

If the user wished to provide himself with options given during normal COOLHANDS input, he may supply this routine to accomplish that. Calls to `option_define` (a COOLHANDS library routine) define the option names (one per call), and this routine executes them. The default routine supplied in COOL_DEFAULT library is shown below.

```
        SUBROUTINE OPTION_DO(IOPTION,ARGUMENT)

        IMPLICIT NONE

        INTEGER NOPTIONS,IOPTION
        PARAMETER (NOPTIONS=20)
        CHARACTER OPTION(NOPTIONS)*80,ARGUMENT*(*)
        INTEGER*4 JO

        DATA OPTION/20*' '/
C IOPTION gives the option number, and ARGUMENT is its argument
        RETURN
C
        ENTRY OPTION_SETUP
C This entry should be called from USER_INIT
C        DO JO = 1,NOPTIONS
C           CALL OPTION_DEFINE(OPTION(JO))
C        ENDDO
C
        RETURN
        END
```

Bibliography

[1] KUIP, CERN *Program Library Long Writeup I102*, version 2.05 ; Manual can be obtained at the anonymous ftp-site `asis01.cern.ch` in the directory `doc/cernlib` .

[2] PAW, CERN *Program Library Long Writeup Q121*; Manual can be obtained at the anonymous ftp-site `asis01.cern.ch` in the directory `doc/cernlib` .

[3] J.B. Mandeville, Ph.D. thesis, University of Illinois at Urbana-Champaign, 1993.

[4] G. Heyes *et al.*, www-site `http://alcor.cebaf.gov/coda` .

[5] *Backtracing particle rays through magnetic spectrometers: avoiding systematic errors in the reconstruction of target coordinates*, Th. Veit, J. Friedrich and E.A.J.M. Offermann, Nucl. Instr. and Meth. **A336** 572 (1993)

[6] Eddy Offermann, *White paper for calibrating the High Resolution Spectrometers in hall A*, CEBAF (1995)

[7] K.L. Brown, D.C. Carey, C. Iselin, and F. Rothacker, TRANSPORT: *A Computer Program for Designing Charged Particle Beam Transport Systems*, Stanford Linear Accelerator Center, 1991, SLAC-91, NAL-91, and CERN-73-16.

[8]

[9]