CLAS-NOTE 2007-016

Energy loss corrections for charged particles in CLAS

E. Pasyuk
Arizona State University

July 26, 2007

Abstract

This note describes how to apply energy loss correction for charged particles in CLAS.

Introduction

The momentum of the charged particles in CLAS is determined by tracking them in the magnetic field with drift chambers. The present of CLAS tracking code does not account for energy losses along the track. It returns some effective momentum between Region 1 and Region 3 drift chambers. The energy losses could be quite substantial, especially for low momentum particles. And more so for the runs with photon beams where we have start counter installed which cause particles to loose more energy.

The software package has been developed to account and correct for some of those losses. For given reconstructed particle momentum it finds out what was the momentum at the vertex. It is a set of subroutines that finds path-length of the particle in each of the materials along the track starting from air gap between region 1 drift chambers, start counter, scattering chamber, target cell wall and target material between the wall and the vertex. The code works for any charged particle with a charge equal to 1 and heavier than electron. Valid momentum range $0.05 < p/m < 50$.

1 Where and how to get it

The source code is located in CLAS CVS repository. To get it type:
`>cvs co packages/eloss`.
To build it you will also need `eloss.h` include file which is located in `packages/include/` directory. The `README` file is included in the package. It gives a brief description of usage.
To build a library go to `eloss` directory and do
`>make lib`
To link it with you analysis code add `-leloss` in your `Makefile`.

2 Description of the package

2.1 Initialization

As with the most of the CLAS detector packages `eloss` needs to be initialized first. The initialization is done by function `InitEloss`.
From FORTRAN code':

```
      CALL INITELOSS(runno)
```

From C code:

```
      InitEloss(RunNo);
```

You have to add the line in your code:

```
#include <eloos.h>
```

It has a function prototype declared as:

```
void InitEloss(int RunNo);
```

It must be called at the beginning of the run, like all other initialization routines. It reads

Start counter and target positions from the GEOMETRY table in caldb. If it fails to get information from the map, both positions will be set to default, which is CLAS center.

2.2 Main routine

Main subroutine is `MOMCOR`. It calls all the others.

2.2.1 Calling sequence in FORTRAN code.

```
CALL MOMCOR(POUT, PMASS, VERTEX, IFLAG, ICELL, PIN)
```

Input parameters:

`POUT(3)` (real) – 3-vector of particle momentum after start counter.

`PMASS` (real) – particle mass

`VERTEX(3)` (real) – vertex coordinates.

`IFLAG` (integer) - target material flag:

IFLAG	Target material
0	empty target. For IFLAG=0 no energy losses in target
1	LH2 target
2	LD2 target
3	L3He target
4	L4He target

`ICELL` (integer) - target cell type:

ICELL	cell geometry
0	no target cell
1	g1a/g1b/g6a/g6b cell
2	g2a cell
3	g1c cell
4	g3 cell
5	g8a/g6c cell
6	g10a cell
7	g11a/g8b/g13ab cell
8	eg3a cell (geometry of the cell is the same as 7)

There is one output parameter:

`PIN(3)` (real) – 3-vector of particle momentum at vertex.

This is what wee need for kinematics calculations.

2.2.2 Calling sequence from C-code.

The C function returns a four vector and is merely a wrapper for the FORTRAN call. It uses vector type to make the calling sequence easy.

- Include the prototype definition with:

  `#include <eloss.h>`

  which has the following line in it:

```
vector4_t c_momcor(vector4_t p_out,
                   float pmass,
                   vector3_t vertex,
                   int iflag,
                   int icell);
```

- Then in your analysis code just make the call as :

  `pf_in = c_momcor(pf_out,pmass,mvrt->mvrt[0].vert,iflag, icell);`

  4-vector `pf_in` is what you want – particle momentum at vertex.

Warning: People often get confused with notation for `PIN` and `POUT`. `pout` is the *input* parameter and `pin` is the *output* parameter! The logic behind this is very simple. We track particle back trough the absorber. We measure the particle momentum when it gets OUT of the absorber. The goal is to find out what was the momentum when the particle got IN.

2.3 `LOSS` function

Function `LOSS` could be used as stand-alone routine. For given momentum after absorber it calculates particle momentum before the absorber. Currently it works for liquid hydrogen, carbon, plastic scintillator, aluminum, liquid deuterium, liquid ^{3,4}He and air.
Calling sequence:

```
Ierr = LOSS (mate, thick, pmass, pout, pin)
```

Input parameters:

`POUT(3)` - 3-vector of particle momentum after absorber
`PMASS` - particle mass
`THICK` - material thickness
`MATE` (integer) - material index:

IMATE	material
1	liquid hydrogen
2	plastic scintillator
3	carbon
4	liquid deuterium
5	liquid 3He
6	aluminum
7	liquid 4He
8	air (gas)

Output parameters:

`PIN(3)` - 3-vector of particle momentum after absorber.
`LOSS` (integer) - return status:
`LOSS = 1` normal completion
`LOSS = 0` unknown material
`LOSS = -1` momentum out of range

If LOSS is not equal to 1, then PIN=POUT, no energy correction is done.
If you need other materials, let me know, I'll put them in.

3 External routines.

There is one external function which is not a part of `eloss` package. It is function `DIVDIF`[1] from `cernlib`. This function does interpolation using third order polynomial.

References

[1] http://wwwasdoc.web.cern.ch/wwwasdoc/shortwrupsdir/e105/top.html