LGD Calibration Routines

S. Teige
Department of Physics, Indiana University, Bloomington IN 47405

Code to calculate the calibration constants for the 71×43 LGD from fitted photons or momentum analyzed electrons is described below. This set of routines differs from the P2 routines in that new routies have been created that interface more directly to the results created by the constrained fitter. The code can be found in /home/lemond/e852/source/LGD/s.cal3053.f and is currently reserved to me.

Routines starting with "e.." are based most directly on the orignal P2 code and are most useful for calibrating with momentum analyzed electrons. These routines may be used for photon calibration by calling them once per fitted photon with the photon enegy replacing the electron energy but this is not the most convenient method.

Routines starting with "g.." were created from the original P2 calibration code with the addition of internal loops over the photon list with knowledge of the results of the constrained fitter.

The calibration algorithm depends on the inversion of an $n \times n$ matrix where n is the number of channels in the LGD. This matrix is quite large in the present case but is very nearly empty because of limitations on the size of the clusters. This fact has been exploited to compress the storage for the matrix. This, of course, impacts on the indexing and loop control structure used during inversion. The benefit to using this compressed storage is that the code can load on machines without large amounts of RAM and the number of operations used for inversion can be dramatically reduced because the loops can be truncated. Timing tests, with and without compressed storage have shown that execution time is reduced by at least a factor of 50.

These benefits do not come without costs. The routines described below *require* that the clusters be no larger than 5×5 blocks and that the LGD array be indexed from 1 to 3053 with the missing corner and hole blocks having an appropriate index.

In order to allow use of the full capabilites of the optimizing compiler on the SGI several uninteresting variables must be declared by the calling routine (preferably the main program) and passed to this package. This package has been successfully tested using u-code optimization *without* the -static flag. The storage required is discussed below with the descriptions of the individual routines. I have used the variable names consitantly throughout these descriptions so that a variable once named, may be assumed to be the same variable described for a previous routine.

I begin by describing the routines appropriate for electron calibration. The package is initialized with:

- subroutine ecal0(ncc,store,x,u)

This routine is called exactly once at the beginning of a calibration calculation. The calling routine must provide the following storage space:

- integer ncc(3053)
- real store(-288:288,3053)
- real x(3053)
- real u(3053)

These variables must not be modified by the calling routine until after the new calibration constants have been determined.

The data to determine the calibration constants is accumulated and stored in the appropriate arrays by calling

- subroutine ecal(efit,nbclus,ebclus,mbclus,ip,cc,ncc,store,x,u)

The new variables are:

- real efit
- integer nbclus(8,25)
- real ebclus(8,25)
- integer mbclus(8)
- integer ip
- real cc(3053)

efit is the energy of the electron as determined by the beam spectrometer. nbclus,ebclus and mbclus are defined as they are for consistancy with the cluster finder. nbclus contains the indicies of the blocks in the clusters. In this case there will usually be only one cluster so only nbclus(1,..) will be interesting and needs to be filled. ebclus contains the energies of the blocks in the clusters. mbclus contains the number of blocks in the clusters. ip tells the routine which cluster to use, again in this case, almost always equal to 1. cc contains the current estimate of the calibration constants and must contain the values used to calculate ebclus from the pulse heights.

Once all the events have been analyzed and loaded the new estimate of the calibration constants can be calculated by calling

- subroutine esolv(cc,ccnew,ncc,store,x,u)

The new calibration constants will be returned via

- real ccnew(3053)

Calibration with fitted photon pairs proceeds similarly. The package is initialized with a call to

- subroutine gcal0(ncc,store,x,u)

with variables as described for ecal0.
The arrays for determination of the constants are filled with a call to

- subroutine gcal(eprime,nbclus,ebclus,mbclus,ipoint,npair,cc,ncc,store,x,u)

The new variables are:

- real eprime(8)
- integer ipoint(4,2)
- integer npair

The variables are defined as they are for consistancy with the constrained fitter. eprime(j) is the fitted energy of the j'th cluster. ipoint contains pointers to the pairs of clusters that were fitted to a massive parent. npair is the number of such fitted pairs of clusters.

Once the accumulation arrays have been filled the new calibration constants can be calculated by calling

- subroutine gsolv(cc,ccnew,ncc,store,x,u)

with variables as described for esolv.