1 Introduction

At the 17-18 March 2001 collaboration meeting, we were asked (by Larry Dennis) to provide any insights as to how the Hall D analysis would be done. This document details my view as to how things will be done. I hope it serves as a target for discussion and criticism.

I see this as problem in two parts: we must develop code that will analyze one event then we must develop facilities to repeat that achievement 10^N times. I have little expertise in the second part so will not comment on it.

2 Some general assumptions

I see us breaking the data into manageable pieces I'll call runs. I assume a run exists as a disk file (at least temporarily) that can be efficiently accessed by a CPU. I assume the CPU can write to a disk file and that there is about twice as much space available on the disk as taken up by the raw data file.

3 Global initialization

An analysis, in principle, could span many runs. The first task for an analysis job is to define and initialize any quantities that exist for the entire analysis job. Histograms are an example, perhaps output files need to be defined at this stage also.

To begin analysis, the state of the apparatus appropriate for the run being analyzed must be defined. This requires that the "run number" or some such equivalent information is available.

4 Event reading

The next requirement is a tool that gets one entire event from the data file and places it into arrays/structures/objects/whatevers in a well defined way. This is the "event reader" and its structure is tightly coupled with the online system. The online system is subject to constraints of CPU time, bandwidth, hardware configuration, etc. and should write the data in whatever format is most efficient for the configuration we choose when all is said and done. The impact on the offline system is one of priority, the online system defines the raw data format, the offline system deals with that definition and any considerations imposed by the offline system that are not easily implemented in the online system should be ignored.

5 Components of the analysis

At this stage a series of tasks must be performed on each event. Some must be executed serially, some could be executed in parallel or even during a "second pass" through the data. Needless to say, the number of passes through the data should be minimized to avoid impact on the "second problem", that is, the development of facilities to analyze 10^N events.

5.1 Charged particle pattern recognition

Charged particle tracks must be reconstructed from hits in the drift chambers. This pattern recognition (PR) problem is almost certainly the most CPU intensive part of the analysis. Once a series of hits are assigned to a track, the track momentum is determined by fitting using a measured magnetic field map. Once individual tracks are fitted the primary vertex is determined by a closest approach method using the parameters of the fitted tracks and their error matrices. Secondary vertices can be searched for next. Tracks that did not fit well to a single vertex hypothesis become candidates for detached vertices.

5.2 Calorimeter cluster finding

Cluster reconstruction is a disjoint task. Two-dimensional pattern recognition in the LGD and barrel produces a position and an energy. This is the only output of this analysis. This task is extremely fast, RadPhi and E852 photon reconstruction is accomplished at rates of about 3000 events per second on current CPUs.

5.3 Beam reconstruction

Another disjoint task is beam reconstruction. Basically, tagger hits are searched for a correlation in time with some other part of the event in question, the accelerator RF, the time in the start counter, etc. Once these hit(s) are found the beam energy is known.

5.4 Start counter reconstruction

Space points must be reconstructed from the start counter hits. I haven't thought this through so won't say anything more.

5.5 Output of reconstructed information

I assume that each of these tasks will add information to the raw event record. For example, the LGD PR would add a position, an energy and a list of blocks in the cluster for each cluster found. This information should be stored in such a way that it is accessible by later analysis passes, that it can be extracted from the data file and written into "data summary tapes" and is as compact as possible. Beam and charged particle PR have different requirements as to the details of what must be stored but the access requirements are the same.

6 Global reconstruction

Next, "global reconstruction" starts. This is that part that requires both raw and reconstructed information and so impacts how the information from the previous steps is stored and transmitted. Information from more than one sub-system of the detector is also required for this stage of the processing.

For example, particle ID is part of global reconstruction. The dE/dx information from hits on tracks can be used. Particle tracks can be extrapolated to the cerenkov counter, TOF wall and LGD. If tracks are found to correlate with clusters in the LGD, flags must be set in the cluster information to "tag" these clusters as not due to photons. Similarly, if the energy of the cluster is close to the momentum of the track, both the track and the cluster could be tagged as an electron candidate. If other parts of the particle ID algorithm produce information as to particle type (pion, kaon, proton) , this information must be added to the output.

The positions of clusters in the LGD (not tagged as associated with charged tracks), along with the position of the primary (or secondary) vertex is used to create the photon momenta. These momenta could be added to the list of "tracks" found above although this is redundant and increases the output record size.

6.1 Event indexing and sorting

At this point a final state has been identified that can be characterized by a list giving the number of photons found, the number of pions found, the number of kaons found... etc. This information is contained entirely in the output record and is appended to the raw data record. Another CPU intensive task remains, global fitting. Constraints of conservation of energy and momentum can be imposed. Combinations of photons can be searched for neutral pions, etas, omegas or eta-primes and these masses can be imposed as constraints. Combinations of photons and charged particles can also provide constraints, for example, $\omega \to \pi^+\pi^-\pi^0$ or $\eta' \to \eta\pi^+\pi^-$. The results of the fit are added to the raw+processed data record as before. Time is difficult to estimate but 100 events/second is probably close to right.

7 Physics analysis

Finally, the highest level of processing can begin. Selected final states are extracted from the data files, for example, $\gamma p \to p\omega\pi^+\pi^-$, with a fully reconstructed $\omega \to \pi^+\pi^-\pi^0$ decay, a "good" chi-squared for the conservation of momentum/energy fit and no evidence of other tracks or photons in the event. Events of the selected hypothesis are output to a high level summary file and subjected to topology dependent PWA. I anticipate the extraction of events will be accomplished with some sort of standard tool but that the PWA will be done in the style of whoever is doing the analysis using standard tools to read the file of extracted events.

8 The need for modularity and documentation

I have neglected many tasks that are required to complete this analysis because I have chosen to use the offline analysis task as the "backbone" of this description. Other tasks are essential and should use parts of the analysis discussed above so as to avoid duplication of effort.

For example, the drift chambers require alignment relative to each other and relative to the experiment to achieve their ultimate resolution. This is achieved by reconstructing tracks and examining the distance from the track to a hit on the track on a wire by wire basis. The positions of the wires are then adjusted so that the means of these "residual" distributions are zero.

Clearly, the program that performs this task should use the same pattern recognition code that is used by the offline analysis program. It is also clear that the alignment program does not need to know anything about, for example, the barrel calorimeter (at least not for internal alignment).

If the pattern recognition code meets certain rather minimal requirements it can be easily "transported" between different applications:

- It must be self contained. There is a place where the code, include files, initialization data, etc. can be obtained by anyone interested in using the code.
- It must be self initializing or have an exactly specified initialization method.
- The input and output must be exactly specified to a level of detail that allows someone to read or write code intelligently.
- Error conditions and error handling are well defined.
- It is documented, all of the above information should be available at some centralized location. The above information is the minimum we should consider acceptable. Additional information, for example, a description of the algorithms used to accomplish the task, would be extremely useful

9 Conclusions and comments

I've chosen to outline my view of the analysis task as a collection of modules each accomplishing a specific task strung together to accomplish the ultimate goal of the experiment.

I think this objective can be achieved by creating a set of well documented tools. The required initialization, the input and the output must be exactly specified. The algorithm should be well described and the code that implements it well commented. I believe specification of the coding language is inappropriate, all languages can talk to each other. The system experts should not be constrained to use unfamiliar coding methods.

I include below an example of documentation which specifies part of the pattern recognition problem. Needless to say that, at this point, the documentation is not as complete as I'd like but, it is a start and perhaps will be useful for the design of the larger pattern recognition code.

The example serves to illustrate several points.

- There are references to "access database", for example. If we had documentation similar to the example for the entire analysis task, "access to database" would start to become a well defined concept.

- Input and output are specified, It is clear that the output created would be useful for the cylindrical drift chamber PR problem. If all parts of the problem were specified in this way, the actual arrays/structures/objects/whatevers that serve to connect parts of the problem could be intelligently discussed and designed.
- The problem is isolated, that is, the specification is for a very specific task and a code module could be constructed using the algorithm part as a guide. Should a better algorithm become available (or necessary) this whole section of code could be easily removed and replaced with the improvement.
- Self initialization has been omitted. To maintain the modularity required, each task should be self initializing. Once tasks are better specified, common methods of initialization should become obvious.

10 An example

Here is an example specification of a part of the charged particle pattern recognition code. This section would describe pattern recognition in the planar drift chambers.

10.1 Assumptions and input

A.) x,y,u planes in each chamber.

B.) Input is a list of (chamber,plane,index) triplets which may be packed in some efficient way. if "plane" refers to a wire, a time is also available.

C.) The transformation from (chamber,plane,index) to coordinate is known.

D.) Transformations (x,y) $\rightarrow$ u, (x,u) $\rightarrow$ y and (y,u) $\rightarrow$ x exist

E.) All chambers are identical except for known rotations. In the following I'll use (x,y,u) as the local coordinate system for the chamber. A point in this coordinate system is transformed into the Hall D coordinate system by a rotation about the z-axis. The rotation angle depends only on the chamber index. The z-coordinate of the chamber is uniquely determined by the chamber index.

F.) A field sufficiently uniform that a "constant field" approximation is adequate.

10.2 Concepts and jargon

The important concepts are the "hit", a signal over threshold associated with a wire or cathode strip in a chamber, the "space point", a location where a charged particle passed (given by three coordinates) and the "track", a collection of space points caused by a single charged particle.

10.3 Algorithm

1.) Loop over chambers

   1a.) Transform (plane,index) to x or y or u as appropriate for the plane. (x,y,u) are coordinates local to this chamber. Need access to alignment database. Drift time is ignored at this stage. Count hits per plane.

   1b.) Initialize hit tags to "unused"

2.) Search for space points in this chamber

   2a.) Associate hits with each other. Note this algorithm does not allow for any hit to be on more than one track. Loop over all pairs of unused hits in a chamber. P pair requires a hit in one plane and a hit in another. Calculate the value of the coordinate of the third plane given the two coordinates chosen above. Search the hits associated with the third plane for a match with the calculated coordinate. If a match is found tag all three hits as used

   2b.) Loop over points found by 2a.). Transform (x,y,u) into (x',y',z) using knowledge of the z-coordinate of this chamber and the rotation appropriate for this chamber. Database access required. (x',y') are Hall D coordinates. Add the (x',y',z) to a space points list.

   2c.) End of loop over chambers.

3.) Next task is to connect points to form tracks. At this point there is a list of space points in the coordinate system of the experiment. I assume these are indexed by chamber so that there are a set of hits for each chamber, all at the same z. A track may not have more than one hit in any given chamber.

   3a.) Initialize used point tags

   3b.) Loop over chambers

3c.) Choose an unused point in this chamber, call it point(i). If there are no unused points in this chamber or all points have been tested, increment the chamber counter and goto 3c. Loop over unused points in the other chambers, lets call the current point in this "other chamber" point(j). point(i) and point(j) when projected to z=0 must lie on a circle containing the point (x',y',z)=(0,0,0) The three points point(i),point(j),(0,0,0) define a unique circle. Loop over all unused points in the remaining chambers, that is, chambers that do not contain point(i) or point(j). Lets call the current point point(k). Search the points indexed by k to determine if any of them lie within some small, as yet to be determined distance from the circle. If a point is found within this distance tag point(i), point(j) and point(k) as used and associated with a track. point(i) and point(j) need only be tagged once. Skip the rest of the points in this chamber and repeat the circle search using unused points in chambers not containing points i,j, or k. Any additional points found are tagged as used and associated with the current track. Continue until a point is found in the last chamber or all unused hits in the last chamber have been rejected.

3d.) If all points (i) in this chamber have been used or the last point in this chamber has been rejected, increment the chamber index and goto 3c.) If this was the last chamber, point linking is complete.

3e.) Project all points on this track onto the plane z=0. Find the circle that best fits these points. Save the parameters of this circle.

4.) Gather "stragglers". If any unused hits remain, they are either associated with noise or have one (or more) coordinate of the (x,y,u) triplet missing due to inefficiency.

4a.) Search each chamber for unused hits. Require at least two coordinates (an unused hit in more than one plane) to proceed. If such combinations exist form a space point using the coordinates pairwise. Perform the circle search for each track and add the two-coordinate space point to the track list if appropriate. Tag the hits used to make this two-coordinate space point as used.

4b.) Continue until all unused hits have been examined.

5.) Initial momentum reconstruction. At this stage a set of tracks has been found. A list of space points associated with each track exists or can be made from the point tags.

5a.) Loop over tracks.

5b.) Project all points on this track onto the plane z=0. Find the circle that best fits these points. The radius of this circle determines the initial value for P-perp. Pz can be determined from the chamber spacing. Choose a point on this track and evaluate the vector momentum at this point. Save this point and the vector momentum for this track.

6.) Done.

10.4 Output

The algorithm produces the following output:

a.) The number of tracks.

b.) A list of hits on each track containing the (x,y,u) coordinates of each hit and the TDC value associated with the u hit (if a u hit exists). At most, one of the three coordinates of a hit may be missing.

c.) A preliminary estimate of the vector momentum of the track.

d.) A point on the track in the where the vector momentum is given.