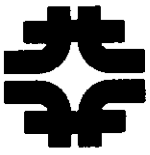

Fermilab

TM-1400
2300.000

A VIEW OF SOFTWARE FOR HEP EXPERIMENTS

H. Johnstad, P. Lebrun, E. S. Lessner, and H. E. Montgomery

May 1986

A VIEW OF SOFTWARE FOR HEP EXPERIMENTS

H. Johnstad, P. Lebrun, E. S. Lessner, H. E. Montgomery

ABSTRACT

A view of the software structure typical of a High Energy Physics experiment is given and the availability of general software modules in most of the important regions is discussed.

The aim is to provide a framework for discussion of capabilities and inadequacies and thereby define areas where effort should be assigned and perhaps also to serve as a useful source document for the newcomer to High Energy Physics.

Contents

1 INTRODUCTION

In modern high energy physics experiments the effort expended in the analysis of the data often exceeds that to construct the hardware for the experiment. Historically the HEP community has been slow to recognize this issue and solutions to software problems for individual experiments have often been taken on an ad hoc basis, with little consideration for the existing software tools and packages.

While the probability that for any given experiment it makes sense to use only general packages is small, it is nonetheless important that there be an awareness of what is and is not available. Further, it is important that within the Computing Department we have a clear view of the subject, as a foundation from which one may attempt to address the issues by some innovative technique which becomes feasible as advances in both hardware and software are made.

The series of CERN Schools of Computing (1) have covered many aspects of the subject, although there has, of course, been significant progress from the situation described in some of the earlier schools.

This document is a discussion of the components of the suite of software necessary for a typical experiment. The individual sections are delineated in terms of the particular view represented by the Flow Chart shown in Fig. 1.

With some physics idea in mind, the physicist starts from a physics modelling program which probably uses Monte Carlo techniques. Having understood the kinematics and the interesting features, he starts to model some apparatus, again using Monte Carlo techniques. This program which aids him now in design will eventually develop into a complete simulation of the apparatus, enabling the generation of simulated data which are then passed through the analysis software chain in parallel to the real data.

The real data, usually on several thousand 6250 bpi magnetic tapes, are first of all organized into different streams based on hardware trigger information; they are unpacked from the more or less arbitrary format of the online data tape and are structured in suitable fashion for the subsequent analysis. The data (both real and modelled) are then passed in turn through the pattern recognition, track fitting and vertex fitting stages of the analysis. Following stages contain analog data analyses, which depend on the results of the former, before the Data Summary Tape is written. In many experiments, this is the point at which the maximum information per event is available; subsequent stages of analysis reduce gradually both the number of surviving events and the information for each, to a point usually controlled by local computing conditions. The data is then examined for physics results from many different viewpoints and, with each move, the Monte Carlo data is treated the same way to provide the deconvolution functions for the combined operation DETECTION----►ANALYSIS. Having obtained some model independent results, there remains the pleasant task of killing current models, speculating widely, convincing the collaboration and preparing the publication.

This document, while wide ranging, will attempt to be brief and will contain little or no information on the actual use of the software cited. In its discussion stage conclusions may be reached which after a more sober discussion may turn out to be too extreme.

For details of local implementation of many software packages, the prospective user should consult the Fermilab Computing Facility Program Library (2).

2 GENERAL ENVIRONMENT

2.1 Languages

FORTRAN has dominated all other programming languages in the field of scientific and engineering computation for more than two decades. It has consistently shown itself to be attractive to scientific users because its basic simplicity and power of expression appeal to non-specialists.

The history of FORTRAN was not always glorious. It has benefited largely from the lack of competition from other languages, which failed to develop appropriate standards and universality to become an attractive alternative to FORTRAN, especially when it comes to ease of use and efficiency of object code.

Most of the world community of FORTRAN programmers has now changed from FORTRAN 66 to FORTRAN 77 standard (3,4). With this changeover it was also demonstrated that the use of FORTRAN in absolute terms was actually growing. It is difficult to measure the use of different programming languages, as very few statistics are available, and also it is unclear whether the relevant parameter for comparison is the number of lines of code written, or the number of compiler calls, or the total execution time of the programs, or other. It is probably important to notice that the use of FORTRAN programming has spread into microcomputers. Also it is a fact that the big manufacturers, like CDC, IBM, VAX, etc. are strongly supporting FORTRAN and also further development of FORTRAN compilers. This is probably a response to a perceived user demand.

In spite of this strong position of FORTRAN among compiler languages, FORTRAN is not regarded as being an ideal tool for scientific programming. In the following we shall mention some of the FORTRAN deficiencies:

1. Storage association in Fortran is essentially done through the COMMON and EQUIVALENCE statements. This makes it impossible to define any scope of variable other than local or global; it is impossible to restrict the scope of a variable to a group of subprograms only. The EQUIVALENCE statement allows potentially dangerous aliasing of locations, for example between variables of different data types.

2. FORTRAN lacks structures; it allows only for arrays and COMMON blocks. Arrays are restricted to elements of a single data type, and COMMON blocks cannot be manipulated as an entity. The high-energy physics community has produced several memory and data management packages to overcome this problem. See elsewhere in this document for a more detailed discussion on data structures.

3. The FORTRAN source form is inappropriate for entering and editing source code from a terminal.

4. FORTRAN has poor or no error recovery from the hardware conditions such as overflow and underflow, although it has some built-in error recovery functions, such as error recovery from parity and end-of-file conditions on external files.

5. FORTRAN cannot handle bit type data, although the bit is the most fundamental data type and has to be manipulated by many programs, especially in high energy physics; for example, the true/false values of physics conditions are often represented by bits packed into bytes or words, and are an essential part of many particle physics programs, such as histogramming packages, organization of data structures, etc. One usually has to resort to a local assembler language, or to specific non F77 standard routines provided by the computer manufacturer.

6. There are no means in FORTRAN to ensure that computer arithmetic is stable across a range of computer architectures, and this has often been a tremendous problem, for example, in track fitting and matrix inversion algorithms.

7. FORTRAN has essentially no query functions; it has, for example, no ability to pose questions about the run-time environments of a program. Again the users had to resort to home-grown software packages to resolve this problem. Simpler questions, like time-of-day, have become available in most FORTRAN compilers.

Some of the above deficiencies can be cured by use of Non Standard F77 extensions, which are intrinsically machine dependent. For instance, VAX Fortran supports structures. However, code transportability becomes seriously affected when using such extensions.

The implementation of new standards in FORTRAN 8x, as defined by the ANSI (American Standards Institute) committee, known as X3J3, will produce a total revision of the FORTRAN language, backward compatible with the current standard, making FORTRAN 8x a superset of FORTRAN 77. FORTRAN 8x will, however, have several so called deprecated features; for example the EQUIVALENCE and COMMON statement will be deprecated, and so will BLOCK DATA, DIMENSION, DOUBLE PRECISION, and DO statements, among others.

Major new features in FORTRAN 8x are its array processing syntax and associated features. This area contains a multitude of new features documented in numerous publications (5).

The goal with FORTRAN 8x is to introduce features which have at least been demonstrated in the context of other languages, even if not in FORTRAN dialects themselves. This should make FORTRAN 8x modern, reliable and portable.

Many other compilers than FORTRAN continue to be used, also in high energy physics programming.

PASCAL is used in many small-scale applications, and it is also heavily used as a teaching language. PASCAL has many limitations which make it inappropriate for large-scale scientific programming: it has poor I/O, it has no extended precision, it has no complex arithmetic, no exponentiation, and so forth.

The C language is also used in small-scale applications. It is used in connection with the UNIX operating system.

ADA compilers are now available and are becoming popular in real-time applications and process control. ADA has powerful numerical capabilities. It is unlikely, however, that ADA will reduce the FORTRAN community by any significant amount, in particular if the new definitions of FORTRAN 8x will have a rapid and successful implementation. ADA is more difficult to learn and use than FORTRAN, which can be used by non-specialists almost without training. ADA may possibly become an interesting language for experts in large specialist applications during the next decade.

Finally we shall mention PL1 (Program Language 1) which was designed by IBM to be the language to end all languages. However, the language never caught on, due to its huge size and unwieldly syntax. Its use is now mainly in high-level systems programming on IBM systems.

2.2 Source Code Management Systems

Particle physics software is usually quite bulky, and it is typically developed by a large number of users on several different types of computer architectures.

The user has a continuous need to store and update several libraries of source materials, not only subroutines and complete programs, but also data sets of long term constants, data sets, JCL (Job Control Language), etc. It is important for the user to keep track of changes made to the source library, and also to be able to make temporary changes on selected parts of the libraries without interfering, or having to interfere, with other users also working on developments on the same library.

There are several code management packages which will perform these services on specific computer architectures, such as CDC, IBM, VAX, UNIVAC, etc., but there were earlier no packages available which would perform across different type main-frames in a smilar fashion, and which would allow the user to transport the libraries from one type main-frame to another type without effort, modifications, special adaptions, etc.

A software package to achieve these goals was developed in the high energy physics community more than two decades ago, and is currently being used for almost any kind of distributed software in particle physics. Since this package has most of the features available in almost any specific main-frame maintenance tool, the package is also to a large extent used for local development and maintenance.

The package is called PATCHY (*), which actually consists of 1 main utility, the YPATCHY program, plus 10 auxiliary utilities, YTOBCD, YTOBIN, YTOCETA, YFRCETA, YEDIT, YSEARCH, YSHIFT, YLIST, YINDEX, AND YCOMPAR.

YPATCHY itself performs the important selection of source material to be used, for example, for input to a compiler on a given type computer, or also selection of data sets to be used by a program, etc. It allows for local editing of the source library on a line-by-line basis without altering any part of the source library itself; thus it is a powerful tool in multi-user, multi-programming, and multi-computer environments.

The PATCHY auxiliaries perform services like editing source libraries, index and listing of contents, transport between different mainframes, etc.

HISTORIAN is a recent commercial software package that allows the user to store and update a library of source materials, while simultaneously keeping track of all changes made to the library. Like PATCHY, HISTORIAN allows for multi-computer usage and maintenance of the source library, and for the transport of source libraries between mainframes of same or different type computers. HISTORIAN is available for most of the larger mainframes but as yet is not widely used in HEP.

2.3 Data Structures and Management

Most off-line software in high-energy physics is written in FORTRAN, which is the most universal high-level language available. However, as mentioned previously, FORTRAN has no dynamic data structuring facilities, except for arrays of homogeneous elements and COMMON blocks, which cannot be manipulated as entities, nor defined dynamically at execution time; further, there are no pointers available to link these structures together at a higher level.

* References to programs cited are listed at the end of the article. Whenever available the Fermilab Computing Department Library numbers are included with each product reference.

A careful design of the organization and handling of data is much needed during the development of particle physics software. The flexibility needed to follow changes in the hardware and changes in the analysis methods is very important in this organization. The data structures should also be automatically documentable so that the content of individual arrays can be easily communicated between several users and program units.

The high-energy physics community has produced several memory and data management packages to achieve these goals and to compensate for the FORTRAN deficiency of lack of manageable data structures.

Such systems allow for a truly dynamic creation of data-structures and the routing of such data structures to and from an external storage medium. The user communicates with the system from a FORTRAN subroutine by CALL to the library of various service routines in the system. The user has to follow a minimal set of rules and conventions.

Once the user program has been written in this fashion, it becomes simple to read and understand and to use and further develop the program. Addons to the program can usually be done without having to re-write or even re-compile those parts of the program already ready for production running, thus extensions to the program can be done without side-effects to the program itself.

The cost of overhead in time and memory resources is typically insignificant for such a system. Time is sometimes gained by not wasting parallel development of related features of the program which might have to be done using only conventional FORTRAN facilities.

The most ambitious systems developed so far are the HYDRA (1972), ZBOOK (1974), and more recently the ZEBRA (1985) system. Other systems like YBOS and others, also developed in the high energy physics community, enjoy much less support, development and popularity.

An attempt is being made to standardize the development of software in particle physics, and ZEBRA was written to unify the most valuable features in most previously existing systems. ZEBRA became operational in 1985 and the conversion of several distributed application software packages relying on the service of the ZEBRA routines is underway; for example, GEANT3 (the detector description and simulation tools for optimization of the design of detectors and test of the analysis programs) has been converted to use the ZEBRA system.

2.4 Histogramming

Histogramming is an important part of the particle physics software. This part of the analysis may consume up to half of the cpu cycle resources of a typical experiment, and it is therefore important that this be done in an efficient and organized manner. This becomes even more important as the absolute volume of data in a typical experiment increases, programs containing histogramming algorithms may have to grind through several thousands of magnetic tapes with raw data, and it is preferable that this be done only once during the lifetime of the experiment.

Histogramming is a common need in several applications, and a large number of packages exist; most mathematical and statistical libraries, like IMSL, NAG, etc., contain such service routines.

Several packages have been developed within the high energy physics community during the last two decades, like SUMX, KIOWA and HBOOK.

SUMX being the oldest of these packages has certain deficiencies with respect to today's software environment; in particular, it has no provision for graphics output. SUMX is, however, still the most powerful tool for in-depth physics analysis on the DST-level. A further discussion of the SUMX capabilities is presented in Section 8 of this document.

KIOWA is a program similar to SUMX, but much simpler in its operations, and all user interfacing is done via subroutine CALL to the KIOWA routines. Like SUMX, KIOWA has no graphics interface. KIOWA is more or less obsolete in today's environments.

HBOOK is a much simpler graphics package, practically without any internal structure, and, so far, no organization of data structures or memory management. It is in some sense, an efficient package; due to its simplicity, it is quite powerful for those features it provides, although they are much fewer than in the SUMX system.

HBOOK is a FORTRAN-callable histogramming facility, whose purpose is to define, fill and edit histograms, scatter plots, and tables. The input to HBOOK is always a subroutine CALL. The output consists of resulting histograms, scatter plots, or tables, edited on the line printer file. In more detail, the HBOOK user entries provide for the creation of histograms and scatter-plots with various statistical options; projections of 2D distributions onto x- and y-axes; representation of functions of 1 or 2 variables; filling (incrementation of data) of histograms; access to parameters for individual histograms in memory; arithmetic operations between histograms; least square fitting of parametric functions to the histogrammed data; smoothing of histograms using spline fits or other algorithms; random number generation; output of histograms to mass storage and editing of histograms with various printing options.

A very important additional feature of HBOOK is its excellent graphics interface, the independent HPLOT package, which gives the user advanced capabilities for graphics representation, in 2D and simulated 3D, of the data, almost without any effort whatsoever.

HPLOT consist of FORTRAN subroutines and the user input to HPLOT is always a subroutine CALL. HPLOT output is designed to produce drawings and slides, and it does not produce all the numerical information of the HBOOK output routines. HPLOT relies on the low level routines of other graphics packages, of which there are many, GD3, PLOT10, PIGS, MGKS. Recently links to commercial implementations of graphics packages such as GK2000, DI3000, written with GKS (6) or CORE (7) standards have become available.

Viewing of pictures can also be done in a semi-interactive way via the higher level package HTV, which in turn relies on the services of HBOOK and HPLOT routines.

The HBOOK/HPLOT histogramming/graphics package has become very popular in the high energy physics community, and is currently the most frequently used of the various histogramming packages.

2.5 Graphics

The importance of graphics (8,9) is growing at all levels in HEP computing. Both on- and off-line applications are becoming more sophisticated. In the course of planning and analyzing an experiment, graphics will help with the design of the detector by allowing the physicist user to follow the development of simulated events. Also the quality of the reconstruction program can be checked out by graphics tools. High-resolution vector devices with the capabilities of rotating the viewed event in space, or enlarging specified regions, allow the user to check the fine details of the analysis programs. Also 3D graphics are being implemented. Further, portable packages, such as HTV, allow semi-interactive plotting of histograms and functions.

Earlier graphics packages have been developed locally. Portability is not so easily achieved in graphics software which often depends on local environments and available hardware interfaces. Several packages have been developed at CERN, but very few of these had truly portable capabilities. Even GD3, which has been distributed from CERN for more than a decade, would essentially only run relatively problem-free on CDC and IBM computers.

Recently, the emphasis on graphics software has shifted towards commercial products which are based on adopted international standards for graphics, such as GKS and GKS Metafile systems. It is hoped that graphics based programs will become more portable in these new environments. The goal of current efforts is to develop both 2D and 3D standard GKS graphics

packages with an associated standardized transportable Metafile-based data base.

At this time, it is clear that none of the local or semi-portable graphics packages used so far, such as GD3, PLOT10, PIONS at CERN, or DIGS and GD3 at Fermilab, have the potential needed for the basis of a modern graphics package. CERN has recently decided to develop an entirely new graphics systems based on standard GKS and Metafile systems. It is optimistically estimated that such a package will become available at the end of 1986. DI3000 is a Precision Visual implementation of non-standard CORE based 3-D graphics. A local Fermilab interface of HPLOT and GEANT3 to DI3000 is under development.

2.6 Interactive Capabilities

As an example of a graphics application to high energy physics software we shall mention the common problem of determining how many hits of a particular type of event will be registered by a certain part of the detector and how many background events will produce the same signal. Monte Carlo calculations will establish the level of the expected background. When the background is subtracted, the observed signal becomes clearer. These calculations often require large amount of computing time, and therefore the programs are normally run in batch mode, producing metafiles for graphical output. The metafiles can be inspected by displaying the generated pictures, or by plotting them on a hardcopy device. This may result in modifications to the detector, the generation of new Monte Carlo events, or the display of results. Given enough computing power, a session in front of an interactive graphics terminal, in closed loop with the programs, could speed up this process considerably.

The example given illustrates the need for a workstation in high energy physics programming, and also the need for a unified graphics system which will perform the same functions whether the application is in batch mode or highly interactive mode. Such a universal graphics package is not yet available, but the need for such a graphics package has been recognized, and several working groups have approached the problem, and one has more or less arrived at a definition of requirements and standards which outline the performance of such a package.

3 FIXED DATA HANDLING

The state of an experiment at the time of data taking is quantitatively described by a large number of fixed data constants. These range from the allignment data for the physical pieces of the apparatus through the data describing the state of the trigger electronics to the calibration data for different detectors and magnetic fields. The accumulation techniques for these data are also diverse and have widely differing timescales. The calibration data for a calorimeter may be the result of months of data taking in a test beam followed by months of detailed analysis, whereas the information on the state of the electronics trigger may be obtained as part of the normal data taking stream.

While termed constants these data may have a time dependence which demands hourly update which then must track through subsequent analysis. All this diversity potentially implies complete confusion and each experiment requires some organization of these data to make them accessible in appropriate manner to most, if not all, of the program modules in the analysis chain. The more recent development of intelligent machines embedded in data acquisition systems leads to a need for some subset, at least, of the calibration constants to be available on-line, to permit sensible decision making by such modules and also by higher level processors in real time.

The general rule for experiments has been to design and construct their own fixed data handling techniques, often doing some parsing of free format data to permit a degree of flexibility in the presentation of the data to the system. In the more advanced cases the handling program also structures the data into the chosen Data Structure format (YBOS,ZBOOK,HYDRA...) for that experiment. There is no current, generally accepted program package, although the HYDRA title package TQ is available. As with many HYDRA applications packages and as implied by the name "TITLE PACKAGE", it was developed for the world of bubble chamber analysis with its important but perhaps limited problem of the handling of optical mapping constants.

The problem in general would seem to be a natural for treatment by data base techniques and it is perhaps worth evaluating the use of some commercial package in this role. Note that some data e.g., allignment, may not be presented in machine readable form and often convenient interactive facilities for interactive examination of the data are necessary. On the other hand, the interfacing to the data structure of the experiment is a necessity. There are in fact some efforts by experiments in this area and since traditionally "handling the allignment data" has been treated as a menial chore, it is perhaps ripe for a general treatment.

As mentioned above, the variety of data which form the input to such a data base is wide; we will therefore only discuss one particular subset, magnetic fields, where we think some benefits from a general treatment are available.

Monte Carlo and particle tracking programs require a detailed knowledge of the existing magnetic field. Accurate measurements of the field are necessary for the determination and subsequent correction of field imperfections that may arise from inhomogeneities in the magnetic materials, presence of magnetic materials in the surrounding structure of the magnet and asymmetries in the coil assemblies. Sensing methods most commonly used in magnetic field measurements are those based on magnetic resonance, Hall effect, magneto-resistance effect and search coil and integration.

For a given magnet and a set of magnetic field measurements, field mapping is a series of operations consisting of the measurements per se, analysis of the data and final representation of the field, under the constraint that it does not violate Maxwell's equations.

The purpose of the analysis is, therefore, to provide (10):
- removal of inconsistencies in the data and check for smoothness;
- fitting of the data, allowing for a consistency check (the data should satisfy Laplace's equations); smoothing and reduction of the individual errors on each data point and ability to predict the field at points other than the measured ones;
- representation of the field suitable for use in Monte Carlo and particle analysis programs.

The overall fit can be done by minimizing a family of harmonic polynomials or of trigonometric and hyperbolic functions. The latter method is implemented by the program MAGNET, which evaluates a magnetic field component of a magnet from boundary observations only (11). One limitation is that values cannot be extrapolated outside the volume of computation, a possible problem for points near the pole faces; also, the number of terms in the fit is not easily controllable.

At Fermilab, an automatic field mapping device for high energy experimental magnets, known as ZIPTRACK, has been generally adopted as the magnetic field measuring system (12). ZIPTRACK maps the field by integrating the field induced current in three mutually orthogonal search coils as these are moved through the magnetic field.

The fact that each experimental group would produce its own analysis and representation of the field, for the particular magnets used in the experiment, has suggested that a general package should be written which should comply with the analysis aims cited previously. Such a general package is under development by the Physics Software Projects Group and will treat raw data integrity, drift correction, rotations and data smoothness. Finally, a representation of the field for use in particle analysis programs will be provided, so as to enable fast evaluation of the field at any point in the physical domain of the magnet.

4 HEP Modeling and Monte Carlo Programs

Experiments in HEP need reliable Monte Carlo simulations, (see, for instance, (13) not only at the design phase of the apparatus, but also during the analysis phase, in order to monitor acceptance, event reconstruction efficiency and resolution. Modeling may refer to the theory of the basic phenomena studied in the experiment (e.g., the production of a hadron jet, charmed particles and so forth) as well as to the physics engineering (magnetic field, scattering, simulation of the electronics, etc).

A good Monte Carlo program must have (at least some of) the following qualities:

- Reliability: correct within the required accuracy. The degree of accuracy needed in particle tracking or shower simulations often dictates the refinement needed in modeling the detector and, also, the amount of cpu time required to run the Monte Carlo. Therefore, a good estimation of such accuracy should be made prior to encoding or appropriate flexibility built in to the code.

- Reasonable speed: although the amount of cpu time required for Monte Carlo simulation may be small compared to the total amount needed to complete the off-line analysis, there are exceptions, particularly in electromagnetic/hadronic shower simulations, where the number of traced particles increases drastically.

- Flexibility: easy to write or modify, especially during the design phase of the experiment.

- Modularity: the user must be able to perform only a specific part of the simulation, while using results from the previous phase of the modeling. For instance, using a data file containing lists of particles, when studying the tracking, instead of regenerating every time the kinematics of the particles, saves a lot of time. Also, the tracing through the detector does not need to be repeated when the number of wires in a particular chamber changes, since the tracking and digitization are clearly two separate functions.

- Good "general computing environment": Good graphics, I/O, interactive versions, user friendliness, transportability and so forth.

Although all detectors are different, many HEP simulations share some basic knowledge: decay of particles, tracking in magnetic field, showering, etc. Thus, in order to save programming effort, some experiments successfully use specific Monte Carlo programs to understand different aspects of their detector. The following list of such programs is probably incomplete, but mentions the most used packages.

Basic HEP theory:

- NVERTX, SAGE: Generation of particles produced in a collision, according to phase space distributions.

- ISAJET, LUND: Generation of hadron in HEP collision, according to the naive parton model or Quantum Chromodynamics with added models of hadronization.

Experimental HEP, phenomenology and engineering:

- TRANSPORT, TURTLE, HALO: Beam transport and muon halo simuation.

- EGS: Electromagnetic shower simulation.

- CASIM, GHEISHA: Hadron shower simulation program.

- GUIDE7: Simulation of light ray propagation in refractive or reflective materials to be used in the design of scintillating counters, Cerenkov counters.

- GEANT: As experiments increase both in physical size and complexity, "organized" physicists felt a need not only for specific purpose programs, but also for a sound programming environment, with intelligent data structures, where the detector model can be easily maintained as the experiment evolves. GEANT1 was written at CERN for that purpose. Based on the memory management ZBOOK, it is interfaced with I/O systems (FFREAD,EPIO) with a histogram package (HBOOK). The code is maintained through PATCHY. GEANT2 has the same facility but has more specific purpose physics tools, such as handling of tracks in a magnetic field and multiple scattering. Finally, GEANT3 emerged as, to our knowledge, the best "expert system" in HEP Monte Carlo calculations: all of the required features mentioned above are more or less satisfied; it has a very wide knowledge base, including electromagnetic and hadronic showering, which is interfaced with the LUND Monte Carlo. Using the "rather crude" graphic package PIGS, the user is able to see interactively how his detector can be built from a wide variety of elementary volumes. Among the possible improvements to such a package let us mention:

- Implementation of a richer data structure/memory management package: move from ZBOOK to ZEBRA. Hopefully this latter will be achieved soon at CERN.

- Increase and refinement of the physics knowledge. For instance, Cerenkov counters are poorly represented since a GEANT3 medium ignores index of refraction, optical absorption and so forth.

- Implementation of a better graphics package, for instance, of DI3000 at Fermilab.

Magnetic Field Modelling

Again, magnetic field modelling and design is often a large part of the early phase of an experiment, deserving extra emphasis.

The high requirements for the design of magnets used in high energy experiments have to be met by accurate algorithms and codes for the calculation of magnetostatic fields. The several methods of approach to solving the Poisson equation, the governing equation of magnetostatics, can be loosely separated into three major groups: differential, integral and coupled methods. In addition, the scalar potential or the vector potential can be chosen as the dependent variable. An alternative approach is to solve the integral form of the field equations in terms of the magnetic field components.

- Integral methods are based on the evaluation of the integral form of the field equations. Their major advantage is that discretization is required in permeable regions, only, thus avoiding the necessity of imposing the far field boundary conditions. However, the solution depends on the inversion of a dense matrix, whose computer solution time can be quite high (14-16).

- Differential methods are based on the differential formulation of the defining equations. These methods can be quite accurate and are generally faster than the integral methods. Their major disadvantage is that the solution depends significantly on the position of the far field boundary (17).

- Coupled methods apply differential operators in non-linear regions and integral operators in linear ones (18).

In any of the above methods, the choice of dependent variable is important. Many algorithms use the reduced scalar potential (19,20), The gradient of which is the field generated by the magnetized regions. It is single-valued everywhere, but numerical cancellations between the current source potentials and the calculated potentials may occur in regions of high permeability. This difficulty may be overcome by using a total scalar potential that includes contributions from the magnetization and source fields. However, that potential is multivalued in regions containing sources. Reduced vector potentials can be used also as the dependent variable, but they are plagued with cancellation problems similar to the reduced scalar potential ones. An alternative is the use of total vector potentials that are continuous everywhere. The cost is in computer memory, since, for 3-dimensional geometries, 3 unknowns are required per mesh point. Finally, algorithms exist based on the integral formulation of the magnetization vector, for which problems of ill-conditioning and convergence may occur in regions of high permeability.

In summary, there is no ideal algorithm that could be applied universally. It is the general consensus that an efficient code should use a scalar potential, reduced for source regions, total potential, elsewhere; differential operators for non-linear regions and integral operators for linear ones. Perhaps a newer analysis of applicability and efficiency of the aforementioned methods is in order, in face of the existence of large memory computers (appropriate for vector potential formulations) and of high speed computers (appropriate for integral formulations).

The following is a comprehensive list of available magnet design computer codes.

- ANSYS - 3-dimensional code; uses the reduced scalar potential together with finite elements methods.

- AOS/Magnetic: integral formulation of the vector potential. Non-linear materials and cylindrical geometries allowed.

- BIM2D and BIM3D: integral formulation for 2-dimensional and 3-dimensional geometries, respectively. The dependent variable can be the reduced or reduced and total scalar potential and reduced or total vector potential.

GFUN: integral formulation of the magnetization vector for 3-demensional geometries; problems of ill-conditioning for high permeability materials.

LINDA: 2-dimensional problems with simple interfaces of iron with finite non-uniform permeability.

NASTRAN/MSC: highly accurate finite elements routines for 2 and 3 dimensions, which can be coupled to reduced scalar potential formulations.

PE2D: differential formulation of reduced scalar, reduced vector or total vector potential. Non-linear materials allowed.

POISSON: 2-dimensional code; it uses finite difference methods for the solution of either potential or magnetic field components. It can be used for non-linear materials.

TOSCA: 3-dimensional code for problems consisting of coil carrying known currents and regions of hard or soft permeable magnetic materials. It uses both the reduced and scalar potentials with a finite elements method.

5 STREAMING, DECODING,STRATIFICATION, FORMATTING

The data generated event by event in an experiment has, in general, characteristics peculiar to the hardware of the experiment and the organization of the data taking.

The data, as accumulated in the on-line computer, will in general have a different number of bytes (or bits) associated with different pieces of information. There may be one element of information per channel of hardware or there may be only data which is significant (not null) with associated address information; there may or may not be markers associated with counter or detector boundaries.

A typical experiment has several (tens of) triggers and each may cause the readout of different components of the apparatus. At different times different combinations of these triggers will be active.

Often constraints of tape economy force the packing of data, bits into bytes, bytes into words, bytes spanning word boundaries. All these factors complicate the situation at the start of analysis, since the on-line information on how to chain down an event and unpack it, or how to minimize the unpacking to the relevant pieces, may be minimal.

The first necessary task is to reorganize the data based on trigger types (Physics A, Physics B,... Calibration A, Calibration B,...) which require significantly different subsequent treatment. The task is to read the tape(s) and depending on the trigger type, usually found in an event header record, redistribute them to other tapes or disk files. Since the number of tapes may be very large, this is often a significant logistical problem.

The redistribution of events may or may not be accompanied by an internal reorganization of the data. If subsequent space consideration is not important, a complete decoding of the event into a format suitable for the subsequent analysis package may be attempted at this stage. For example, the data may be organized into the relevant Data Structure (ZBOOK,ZEBRA,YBOS...). If on the other hand, circumstances dictate otherwise, sufficient ancilliary structuring must be embedded in the data to ensure efficient subsequent access to relevant pieces.

The problem can be attacked at several different levels.

- The capability of the data acquisition hardware is increasing to the extent that appropriate reformatting of the data occurs before writing to tape, either in the on-line computer or in the readout controllers. Care is required in the design to ensure that at the time of hardware configuration the desirable format is known.

- The current mode of operation relies on the availability of efficient bit and byte handling utilities, associated to a considerable effort on the part of some physicist-programmer to track (by hand) changes in readout and hardware and to insert (by hand) appropriate changes in a large packing and streaming software module.

There is as far as we are aware no general approach to this problem, but one possible scenario which should be considered is as follows.

a) Decide at the design stage the data ustructure to be supported.

b) Create/define a package which would unpack efficiently into the general form of the relevant data structure.

c) Drive this decoding from the data itself.

The implication is that the overhead associated with the relevant on-line directive data would be offset by the decrease of anguish, since any changes in data acquisition configuration would be automatically propagated through the whole system.

Such a scenario is not guaranteed to be viable, however, the potential gains suggest that at least an in depth study of the problem would be worthwhile.

6 Basic Tracking: Pattern Recognition and Elementary Fitting

Basic tracking (21,22) is the central phase of the offline analysis of a given experiment, since: occurring after data taking and prior to any new physics discovery, it influences strongly successive analysis paths and potential results; it requires a substantial fraction (of the order of 1/2) of the total amount of cpu time, as well as a strong and coherent team of physicists and programmers.

Let us introduce a rather arbitrary distinction between "analog or calorimetry" and "digital or tracking" event reconstruction. Analog means that the most elementary information is an amplitude; for instance, the pulse height recorded on a phototube base. Thus, analog pattern recognition deals primarily with energy measurements or particle identification using Cerenkov counters. Algorithms treating this kind of problem tend to be very specialized. Common concepts do exist, such as calibrations or "cluster finding", but their application varies from one experiment to the other; they will be discussed further in Section 7.

"Digital" event reconstuction deals primarly with PWC (Proportional Wire Chambers) like devices, or Drift chambers, the goal being to reconstruct particle trajectories from unrelated measured space coordinates. PWCs are very common devices and also the most simple ones in terms of data characteristics. Thus, physicists have tried for a long time to write general purpose tracking reconstruction programs.

Special tracking programs exist in the CERN library (X-package), but do not have a large audience, essentially because their scope is too limited or they are too cumbersome to use, since their COMMON BLOCK structure is somewhat obscure. Thus, most fixed target experiments rely on their own package, with little shared knowledge; the only common software tools are either a histogramming package (HBOOK, KIOWA) or a mathematical library (IMSL or CERNLIB). Then, it becomes obvious that the only programming language is straight FORTRAN.

Although called a "high level" language, FORTRAN, as discussed in Section 1, is not rich enough to support easily and efficiently the complexity of a tracking system. For instance, the simple tracking operation "consider all XUV hit combination in such a chamber" translates in FORTRAN into a bunch of DO LOOPS, index manipulations or BRANCH IF, with very little semantics between the tracking operation and the actual coding. Also, many elementary tracking operations overlap each other when translated into a FORTRAN program; when performing the above tracking operation, the FORTRAN routine may also execute a fit or skip over an inefficient detector. Thus, these codes are hard to understand and are basically unmaintainable. The addition of a new tracking device in the experiment is so cumbersome to implement that many of the routines must be rewritten from scratch.

Studies of HEP tracking semantics should be pursued. Many elementary tracking operations have been recognized, their corresponding algorithms, found virtually in every fixed target tracking programs, are now better understood. It has been recognized that these algorithms are strikingly similar to those found in RELATIONAL DATA BASE systems (23).

In fact, the input hit coordinate ("raw hits") list can be understood as a DATA BASE. The goal is to establish "RELATIONS" between such hits, i.e., to select a hit subclass satisfying some geometrical criteria. Such a related subclass is called a "track". Note that these tracks are themselves the elementary "hits", in the context of vertices. Solving the basic pattern recognition means essentially deducing the "track" lists from the "hit" list.

Elementary tracking operations expressed in HEP jargon for instance, "projecting", "setting up roads" or "arbitrating tracks", can be understood in terms of elementary relational data base operations.

While assembling tracks, two logically distinct phases should be recognized. In the first phase candidate tracks are formed, based on some raw geometrical criteria--the initial one is relational in nature, as one computes pointers to the hit list, copies, selects or transfers blocks of data. Thus, good data or memory managements tools are needed.

In the second phase, these candidate tracks are "fitted" to a particular model. If the result of such is acceptable, the track is considered as established. This latter aspect includes the "real" calculations, and thus requires complete mathematical libraries. Because of this aspect, and also because of the fast I/O requirements, commercial data base packages are inadequate.

While resolving the basic pattern recognition, the complexity of these fits is dictated, on the one hand, by the accuracy needed to distinguish between some nearly equivalent track candidates and, on the other, by practical computer limitations (memory as well as cpu time). Thus, such "reconstruction" fits are in most cases much less sophisticated than the final fits, where the tracks parameters are finally estimated.

Having recognized the elementary tracking operations, it is then possible to build a true TRACKING RELATIONAL DATA BASE SYSTEM. Such a system is tuned to the particular tracking problem, and does not attempt to provide general memory framework. Also, it is certainly possible to build such a system with a user-friendly philosophy: the graduate student should not need to know about pointer rules nor reference links subtleties. Optionally all relevant information should be transmitted through straightforward F77 subroutines calls.

The success of such a package will depend not only on its quality but also on the HEP community's response to a more modern approach to computing, where the user must integrate himself into a programming system rather than writing his own "DO LOOPS" independently of his nearby collegues. Hopefully, such conversions will occur.

The approach described above is emphasized as an example of a new approach to Pattern Recognition Problems; however, it is not claimed to be the only one. Indeed we would be remiss if we did not mention that efforts are again being made to investigate the use of algorithms tailored to vector machines (24).

7 Analysis of Analog Data

The previous section treated the problem of extracting tracks from digital data from wire chambers or other similar apparatus. There are, in addition, several aspects of data analysis which deal with analog data. For the purposes of this document we consider in this category all types of particle identification data: Time of Flight, DE/DX, Cerenkov, Lead Glass, and Transistion Radiation (TRD) data. The analysis of the data on such properties may be quite involved and often it is necessary to utilize as input the information derived from the tracking detector data. There is also, however, the situation which pertains when the analog device, for instance, a calorimeter, is the primary source of tracking information. This is the common situation at higher energies, especially in collider detectors. The two situations will be discussed separately below.

It is rare that adequate particle identification can be achieved with a single device; it is more often the situation that a given track traverses several devices, Cerenkov, TRD,..., with each producing analog data as a result In general, the problem is to compare the response for the series of detectors with those expected, given the variety of possible hypotheses for the particle identity: Electron, Muon, Pion, Kaon, or Nucleon. Often a "Maximum Likelihood" approach (25) is used which permits the ultimate combination of all data. A necessity for such an analysis is a thorough understanding of (and therefore ability to model) the detector behavior. Possible backgrounds and inefficiencies must be handled in detail. Although extensively studied there are, as far as we know, no generally available packages for this problem.

Pattern recognition in calorimeters, the establishment of appropriate clustering algorithms and the subsequent solution of events, is a relatively new field except in the particular field of Pizero detection. There are some examples of major detectors such as The Crystal Ball (26) and the GAMS detector (27) which have relied on calorimetric information at modest energies but here also with Pizero and Gamma detection as the primary aim. In recent years, the use of calorimetry for charged particles as a bonafide alternative to tracking has emerged, at very high energies. The current understanding is that the possible techniques depend critically on the segmentation available in all three spacial dimensions and the correlation or not of the data in several views. A calorimeter with interleaved longitudinal segmentation but with only two view readout, in principle, permits the solution of the well known "crossed hodoscope" ambiguities but, in practice, a modest multiplicity already poses insuperable problems for the software.

In all experiments it is important that resolution effects be properly handled so that the final results are correct, both in magnitude and in assigned error. This is a general problem for the total analysis but in the field of calorimetric pattern recognition, unfolding of responses from detectors with finite resolution is the primary aim. There has been considerable discussion of this problem and a review is given by Blobel (28).

8 Data Summary Tape Analysis

Eventually for each experiment there comes a stage at which the details of the data are thought to be understood, the systematic defficiencies adequately treated and the data reduced to a collection of 4-vectors and particle identities. What remains is the "Physics Analysis". In general it is necessary to work with at least three data sets; one is Real Data, a second is Monte Carlo Truth (Physics Model) and the third is the Monte Carlo after passage through the experiment simulation and analysis chain. Comparison of the latter two permit the understanding of how to map backwards from the Real Data to the Real Physics. Each of these data sets may contain many tens of thousands, hundreds of thousands or even millions of events and the significant effects may be quite subtle. It is the proverbial "Needle in the Haystack" problem.

In order that the maximum physics be extracted it is necessary to minimize the mechanical effort involved in this exercise, since maximum creativity is needed. It helps if the previous stages of analysis have adhered to some rational data structure, but flexibility is the key. In past times there have existed tools which were appropriate, such as SUMX or KIOWA, which permitted the application of many cuts, slices and conditions to the data and minimized the confusion.

The actual data processing in SUMX is done by a number of so called blocks or processors. The orderly sequential operation of the various processors is regulated by the master program SUMX, which is a main control subroutine in the root segment of the efficiently overlayed program SUMX. This organization provides for high efficiency both in cpu cycles and memory resources; each processor is a plug-in unit which is physically put into memory and into operation on the request of the user at execution time. The various SUMX processors provide special features for input/output; create and plot histograms, scatter plots, slices and profile histograms; find mean values and variances; provide for multi-dimensional analysis by way of a special SELECT processor which evaluates the truth-values of specified conditions; create mini-DST of selected events, or group of events, depending on the truth/false value of conditions, and store the truth/false value of these conditions in bits on the mini-DST. Other processors provide a dictionary facility for the organization of conditions, a breed facility for the creation of sub-events, for example, performing grouping of tracks by perturbation, etc.

The user provides information to SUMX either by subroutines which communicate with the SUMX processors through defined COMMON blocks, or via data given as input to the SUMX program. The data may specify conditions, book histograms, group histograms, perform arithmetic operations on one or several histograms according to conditions, specify the selection of parameters and events which will be stored on a mini-DST, etc.

Although SUMX is still the most powerful package available for DST-analysis, it lacks several capabilities needed in today's HEP software environments, such as graphics interface and interactive data handling and editing.

The appropriate environment today would seem to be that afforded by a high performance work station and this option has been discussed (28). This approach bears much resemblance to the attitude taken in the CAD and CAE arena where maximum interaction of the engineer with the projected solution is yielding high productivity. It is also consistent with the approach being taken to the development of algorithms for reconstruction of highly complicated events as demonstrated by the UA1 and UA2 experiments at CERN.

At the present time there are no active projects in this field at Fermilab and it seems to be an area with great potential for progress. It should not be forgotten however that depending on the size of the data sample there may be a need for many MIPS (Million Instructions Per Second) and much taping spinning. This aspect of the problem has led some to suggest (29) that it is necessary not only to address the development of the environment but to apply large systems of microprossessors in conjunction with Optical Disks. This approach may get under way in the next year or so since the appropriate microproccessors now appear to be available (30).

9 CONCLUSIONS

In this document we have attempted to skim the subject of High Energy Physics Software from the point of view of desired functionality as related to the phases of analysis of a experiment. It is by design lacking in detail but has already served its minimum purpose of educating its authors. The summary is the obvious, but perhaps not emphasized often enough, statement that the computing environment for a HEP experiment should permit, ideally, the close envolvement of the physicist in every stage of the data analysis, for which the availability of packages allowing for fast, easy to implement, adjustments to new methods on information, coupled to good graphics tools, is essential. We do feel that this document has identified several areas where a common approach and general software techniques might have significant impact were they available, such as treatment by the data base techniques of the fixed data handling stage; new analysis of efficiency/applicability of magnet design methods; general approach to Streaming, Decoding, Stratification and Formatting; further development of pattern recognition methods; general packages for analysis of analog data and last, but not the least, a closer look to the work station environment and graphics packages. The list is daunting; however, the motivation is correspondingly high since the estimated manpower required to generate the software for each of the current large experiments is of the order of 100 person years (31) and will clearly grow as the detectors become even more complex.

REFERENCES

1. Proceedings of the CERN SCHOOL OF COMPUTING, CERN 71-6, Cern 72-21, CERN 74-23, CERN 76-24, CERN 78-13, CERN 81-03, CERN 82- and CERN 85-09.

2. Fermilab Central Computing Facility Program Library, Fermilab Computing Department, December 1984, GC0002.3

3. M. Metcalf, "An introduction to FORTRAN 77", CERN Computer Center, DD/VS/11.

4. "Programming Language FORTRAN-ANSI x3.9 - 1978.

5. ACM Newsletter: Fortran Forum.

6. F.R.A. Hopgood et al, "Introduction to the Graphical Kernel System, GKS", Ch. 11, 145-149, Academic Press, New York, 1983.

7. See DI3000 as an implementation of CORE based graphics.

8. J. A. McDonald, J. Pedersen, "Computing Environments for Data Analysis", SLAC-PUB-3577, STAN-LCS-09, February 1985.

9. J. A. McDonald, J. Pedersen, "Computing Environments for Data Analysis", SLAC-PUB-3578, STAN-LCS-10, February 1985.

10. C. Best, A. M. Osborne, "The Analysis of the Magnetic Field of the Forward Spectometer Magnet", CERN, EMC/78/26, 1978.

11. H. Wind, "Evaluating a Magnetic Field Component from Boundary Observations Only", Nuclear Instruments and Methods, 84, 117-124,1970.

12. R. Yamada et al, "Fermilab Magnet Mapping System", Nuclear Instruments and Methods, 138, 567, 1976.

13. F. James, "Simulation in High Energy Physics", Proceedings of the 1970 CERN Computing and Data Processing School, 443-466, CERN 71-6, 1971.

14. C. W. Trowbridge, "Progress in Magnet Design by Computer", Proc. 4th Conf. Magnet Technology, Brookhaven National Laboratory, Brookhaven, New York, 555-565, 1972.

15. M. T. Newman, C. W. Trowbridge, L. P. Turner, "GFUN: An Interactive Program as an Aid to Magnet Design", Proc. 4th Conf. Magnet Technology, Brookhaven National Laboratory, Brookhaven, New York, 617-626-1972.

16. J. Simkin, C. W. Trowbridge, "On the Use of the Total Scalar Potential in the Numerical Solution of Field Problems in Electro-Magnetics", Int. J. Num. Meth. Engn, Vol. 14, 423-440, 1979.

17. O. C. Zienkiewicz et al, "The Coupling of the Finite Element and Boundary Solution Procedures", Int. J. Num. Meth Engn, Vol. 11, 355-375, 1977.

18. T. W. Daniel, R. B. Fernandez,P. R. Root, R. B. Anderson, "An Accurate Scalar Potential Finite Elements Method for Linear, Two-dimensional Magnetostatic Problems", Int. J. Num. Meth. Engn, Vol. 19. 725-737, 1983.

19. O. C. Zienkiewicz, T. Lyness, D. R. J. Owen, "Three-dimensional Magnetic Field Determination Using a Scalar Potential - a Finite Element Solution", I. E. E. E. Trans. Mag. MAG-13, 1649-1656, 1977.

20. J. Simkin, C. W. Trowbridge, "Three-dimensional Non-linear Electro-magnetic Field Computations, Using Scalar Potentials", I. E. E. E. Proc., pt. B, no6, 368-374, 1980.

21. L. Bugge, J. Myrcheim, "Tracking and Track Fitting", Nuclear Instruments and Methods, 179, 365-381, 1981.

22. H. Eichinger, M. Regler, "Review of Track-fitting Methods in Counter Experiments", CERN 81-06, Data Handling Division, June 1981.

23. T. Nash, S. Bracker, I. Gaines, "Fermilab's Advanced Computer R&D Program", Fermi National Accelerator Laboratory, FN-383, 2380. 000, April 1983.

24. C. H. Georgiopoulos, J. H. Goldman, D. H. Leventhal, M. F. Hodous, "A Non-numerical Method for Track Finding in Experimental High-Energy Physics Using Vector Computers", The Florida State University Supercomputers, Computations Research Institute, FSU-SCRI 85-11, 1985.

25. L. Lyons, W. Allison, "Maximum Likelihood or Extended Maximum Likelihood?", Nuclear Physics Laboratory, Oxford, 1985.

26. E. D. Bloom, Proc. 1979 International Symposium on Lepton and Photon Interaction at High Energies, Fermilab, 1979. Ed. T. B. W. Kirk and H. Arbanel, Fermilab, 1980.

27. Binon et al, "Hodoscope Gamma Spectometer GAMS200", Nuclear Instruments and Methods, 188, 507-516, 1981.

28. V. Blobel, "Unfolding Methods in High Energy Physics

Experiments", Proceedings of the 1984 CERN School of Computing, CERN 85-09, 1984.

29. T. Nash, Private Communication.

30. T. Nash et al, "The Fermilab Advanced Computer Program Multi-Microprocessor Project", presented at the 1985 Computing in High Energy Physics Conference, Amsterdam, Netherlands, June 1985.

31. "Computing for Particle Physics, Report on the HEPAP Subpanel on Computer Needs for the Next Decade", pp97-108, U. S. Department of Energy, DOE/ER-0234, August 1985.

PROGRAM REFERENCES

ANSYS - G. J. De Salvo, J. A. Swanson, "ANSYS-Engineering Analysis System User's Manual", Swanson Analysis Systems, Inc., Houston Pennslyvania.

AOS/MAGNETIC - A. O. Smith Data Systems, Milwaukee, Wisconsin.

BIM2D, BIM3D - M. J. Newman, "BIM2D User Guide", Rutherford Appleton Laboratory, RL-79-088, Rutherford England.

CASIM - A. Van Ginneken, "CASIM-Program to Simulate Transport of Hadron in Cascades in Bulk Matter", Fermilab, FN-272, 1100.050, January 1975.

DIGS - J. Ingebretsen, "Device Independent Graphics System", Fermi lab, SP0012, May 1985. Fermilab SP0012.

DI3000 - "DI3000 User's Guide", Precision Visuals, Inc. Boulder, Colorado, September 1984. Fermilab SP0018.

EGS - R. L. Ford, W. R. Nelson, "The EGS Code System", SLAC PUB-210, June 1985. Fermilab PM0060.

EPIO - H. Grotel, I. Mc Laren, "EPIO Manual", CERN Computer Center Program Library, CERN I-101, DD/EE/81-2, May 1982. Fermilab SU0025.

FFREAD - R. Brun, R. Hagelberg, M. Hansroul, J. C. La Salle, G. Misuri, "FFREAD - Format Free Input Processing User Guide and Reference Manual", CERN Computer Center Program Library, CERN I-302, DD/EE/78-2, DD/US/71, June 1982. Fermilab SU0026.

GEANT2 - R. Brun, F. Carena, M. Hansroul, J. C. La Salle, "GEANT User Guide and Reference Manual Version 2.00", CERN Data Handling Division, DD/US/86, June 1982. Fermilab PM0049.

GEANT3 - R. Brun, F. Bruyant, A. C. Mc Pherson, "GEANT3 User's Guide", CERN Data Handling Division, DD/US/86, DD/EE/84-1, February 1985. Fermilab PM0062.

GD3 - R. Miller, "GD3 User Guide", CERN Computing Center Program Library, DD/US/9, February 1985. Fermilab SP0015.

GFUN - A. G. Armstrong, C. J. Collie, N. T. Diserens, M. J. Newman, J. Simkin, C. W. Trowbridge, "GFUN3D User Guide", Rutherford Appleton Laboratory, RL-76-029/A, 1976.

GKS - "Guide to the Use of GKS at CERN", Draft Manual, Version 1.4. CERN Data Handling Division, September 1985.

GUIDE7 - T. Massam, "GUIDE 7, A General Program for Evaluating the

Properties of Scintillation and Cerenkov Counter Optical Systems", CERN Experimental Physics Division, EP-76-21, December 1976. Fermilab PM0047.

HALO - Ch. Iselin, "HALO - A Computer Program to Calculate Muon Halo", CERN Experimental Area Group, CERN 74-17, August 1974. Fermilab PM0033.

HBOOK - R. Brun, I. Ivanchenko, P. Palazzi, "HBOOK User's Guide", CERN Computer Center Program Library, CERN Y-250, DD/EE/81-1, DD/US/22. Fermilab PM0039.

HPLOT - R. Brun, H. Watkins, "HPLOT User's Guide", CERN Computer Center Program Library, CERN Y-251, DD/EE/80-2, DD/US/17. Fermilab SP0014.

HTV - R. Brun, "HTV - The Interactive Version of HBOOK + HPLOT", CERN Computer Center Program Library, DD/EE/80-5, DD/US/87.

HISTORIAN - F. James, "Introduction to HISTORIAN", CERN Computer Center Program Library, DD/US/49.

HYDRA - J. Zoll, "Hydra Topical Manuals", CERN Computer Center Program Library. Fermilab PU0045.

IMSL - "IMSL User's Manual", IMSL Lib-0009, Edition 9.2, Houston, Texas, November, 1984. Fermilab SM0002.

ISAJET - F. Paige, S. Protopopescu, "ISAJET 5.10, A Monte Carlo Event Generator for p-p and pbar-p Reactions", Brookhaven National Laboratory, Upton, New York. Fermilab PM0059.

KIOWA - J. Friedman, "KIOWA", SLAC Computer Group, CGTM No. 136. Fermilab PM0016.

LINDA - A. Nelson, "LINDA", Fermilab, IM0012, April 1971. See also. J. S. Colonias, "Particle Accelerator Design": Computer Programs", Academic Press, 1974. Fermilab PM0012.

LUND - T. Sjostrand, H. U. Bergtsson, G. Ingelman, "The LUND Monte Carlo Programs", CERN Pool Programs W5035, October 1985. Fermilab PM0050.

MAGNET - R. Messerli, C. Petit, "Magnetic Field Analysis Program", CERN/EP/DHR 76-7, 1976. Fermilab PM0051.

NAG - "The Finite Element Library" Release 2.0, Numerical Algorithms Group, Inc., Downers Grove, Illinois.

NASTRAN/MSC - R. H. Mc Neal (ed.), "The NASTRAN Theoretical Manual", NASA SP-221(01).

NVERTX - A. E. Brenner, D. C. Carey, R. Pordes, J. H. Friedman,

"A Computer Program for Monte Carlo Phase Space with Importance Sampling and Histogram Display", Fermilab PM004, April 1977. Fermilab PM0004.

PATCHY - H. J. Klein, J. Zoll, "PATCHY Reference Manual", CERN Computer Center Program Library, October 1983. Fermilab PU0013.

PIONS - J. Bettles, D. R. Myers, "The PIONS User Guide", CERN Data Handling Division, CERN DD/EE/85/1.

PE2D - C. S. Briddlecombe, N. J. Discerens, C. P. Riley, J. Simkin, "PE2D User Guide, 2D/Axisymmetric Static and Dynamic Electro Magnetic Analysis Package", Version 6.3, Rutherford Appleton Laboratory, RL-81-089, 1983.

PLOT10 - C. S. Curran, "Introduction to Using PLOT-10 /TCS/AGII, or CDC /MFZ/MFA/MFB", CERN Computing Center Program Library, DD/US/61.

POISSON - R. F. Holsinger, Ch. Iselin, "The CERN-POISSON Program Package (POISCR) User Guide", CERN Computer Center Program Library, T604, 1983. Fermilab PM0054.

SAGE - J. Friedman, "SAGE - A General System for Monte Carlo Event Generation with Preferred Phase Space Density Distributions", Lawrence Berkeley Laboratory, Group A Programming Note No P-189, January 1971. Fermilab PM0022.

SUMX - F. Beck, J. Zoll, "SUMX User's Guide", CERN Computer Center Program Library, CERN Y-200, 1973. Fermilab PM0007.

TOSCA - C. P. Riley, J. Simkin, A.G.A.M. Armstrong, "TOSCA User Guide, 3D Static Electro-Magnetic/Electrostatic Analysis Package", Version 3.1, Rutherford Apleton Laboratory, RL -81-070, 1982. Fermilab PM0055.

TRANSPORT - K. L. Brown, D. C. Carey, Ch. Iselin, F. Rothacker, "TRANSPORT: A Computer Program for Designing Charged Particle Beam Transport Systems", CERN Super Proton Synchroton Division, CERN 80-04, 1980. Fermilab PM0008.

TURTLE - D. C. Carey, "(Trace Unlimited Rays Through Lumped Elements), A Computer Program for Simulating Charged Particle Beam Transport Systems", Fermilab NAL-64, December 1971. Fermilab PM0009.

YBOS - D. Quarrie, "Programmers Reference Manual Version 3.00", CDF Computing Group, Fermilab, CDF Note No. 156 (V3.00), November 1984.

ZBOOK - R. Brun, F. Carena, M. Hansroul, J. C. La Salle, W. Wojcik, "ZBOOK User's Guide and Reference Manual", CERN Computer Center Program Library, CERN Q210, DD/US/73. Fermilab PU0032.

ZEBRA - R. Brun, M. Goossens, J. Zoll, "ZEBRA User Guide", CERN Computer Center Program Library, CERN Q100, DD/EE/85-6. Fermilab PU0046.

ZIPTRACK - A. Ito, W. Bosworth, J. Rutherford, A. Lynch, L. Tung, W. Yang, "ZIPTRACK Manual", Fermilab TM1200, 2311.00, 1983.

FLOW DIAGRAM OF ANALYSIS OF A TYPICAL HIGH ENERGY PHYSICS EXPERIMENT

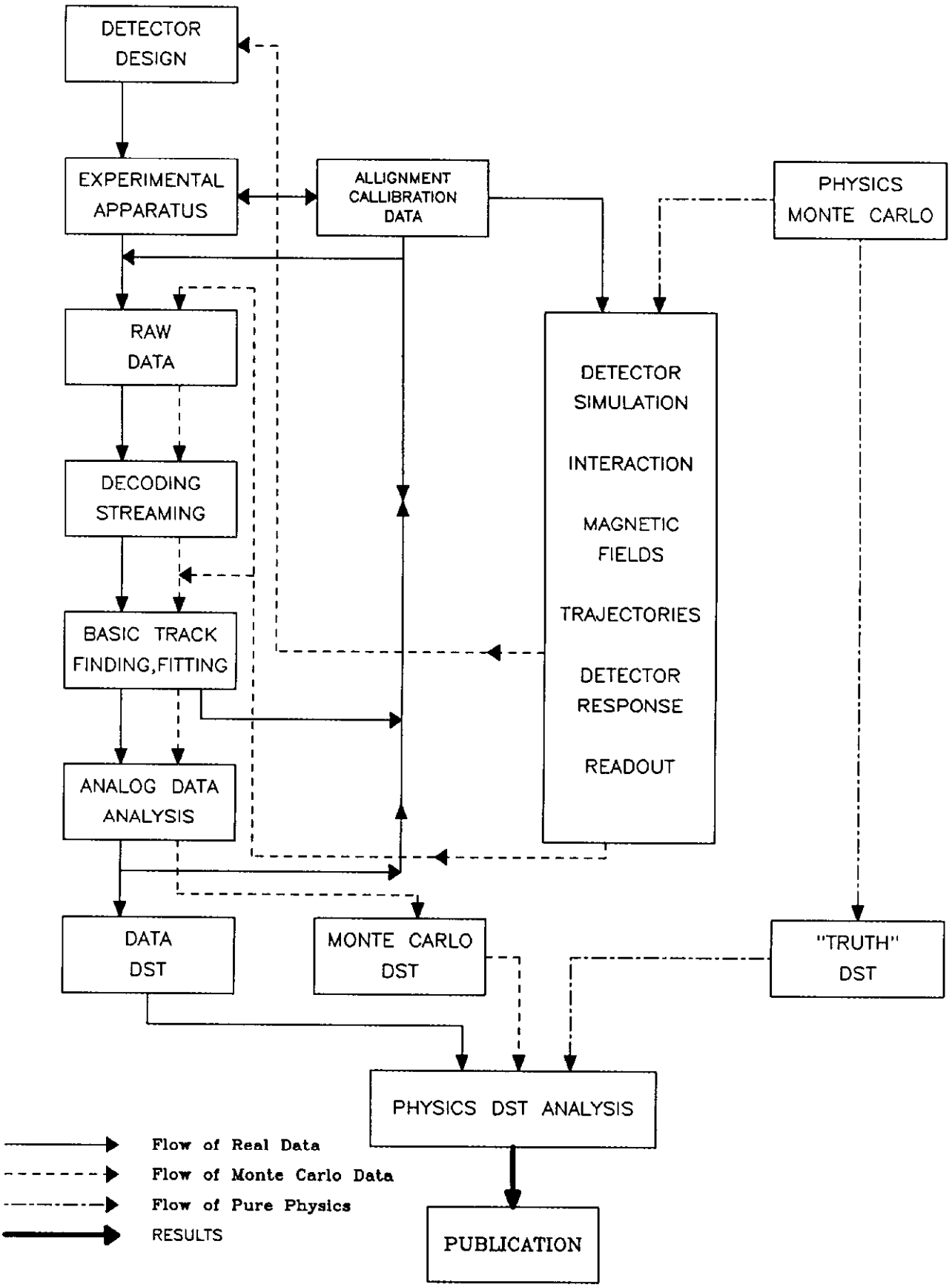

Fig. 1.