LBL--16171

DE83 014904

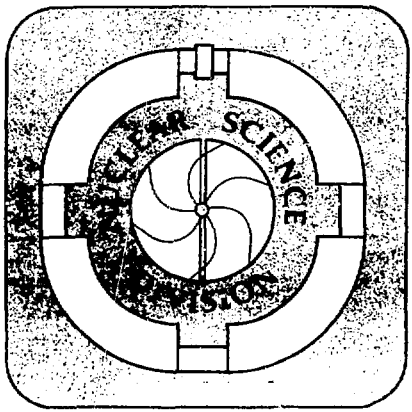
Lawrence Berkeley Laboratory
UNIVERSITY OF CALIFORNIA

Presented at the Conference on Real-Time Computer Application in Nuclear and Particle Physics, Lawrence Berkeley Laboratory, Berkeley, CA, May 15-19, 1983

LULU ANALYSIS PROGRAM

H.J. Crawford and P.J. Lindstrom

June 1983



Prepared for the U.S. Department of Energy under Contract DE-AC03-76SF00098

DISCLAIMER

LULU Analysis Program

H.J. Crawford and P.J. Lindstrom
Nuclear Science Division, Lawrence Berkeley Laboratory
and Space Sciences Laboratory
University of California, Berkeley, CA 94720

I. Introduction

As we were designing experiments at the Heavy Ion Spectrometer System (HISS)[1] at the Lawrence Berkeley Laboratory BEVALAC, it became clear that a new approach to real-time data analysis was necessary. Experimental configurations consisting of many different detector types and sizes were envisioned. The primary difficulty with analysis code available at that time was that all such code required a fixed word length event. Since HISS is operated as a facility with many different experiments being performed by different groups of physicists, we felt that it was necessary to provide analysis code capable of handling arbitrary detector configurations in which each detector may place an arbitrary number of data words into the event stream. The package we constructed to meet our needs is called LULU, and it possesses the characteristics shown below.

- On- and off-line analysis--
  raw and preprocessed data
- Variable word length events
- User subroutines
- Run time functions through RPN
- Graphics device independence
- Automatic data statistics
- Real-time output to other programs

We show in Fig. 1 some of the detector systems presently in use at the HISS facility. These include drift chamber (DC) arrays consisting of 1300 sense wires, a time-of-flight (TOF) scintillator array consisting of 140 photomultiplier tubes, a 49 element NaI array, and a multiple sampling ionization chamber (MUSIC). LULU has been used to debug, on-line monitor, and off-line analyse data from all these detector systems.

The basic data flow through LULU is shown in Fig. 2. Data are provided from a number of sources in a uniform fashion. They can come as an on-line data stream through the route shown at the right: detectors to CAMAC to the Memory Module through an MBD into a PDP 11/45 and onto an intermediate disk. Or it can be loaded onto another disk from magnetic tape for off-line analysis. LULU is able to read any data that are stored in the form shown in Fig. 3, which illustrates the concept and form of the Uniform Data Interface.[2] Using this data form, we can generate Monte Carlo data before an experiment and easily check the complete analysis scheme before staging an experiment. The power of this byte count, fiducial, data scheme is attested to by the fact that LULU has been pointed at engineering data from temperature and pressure sensors on the superconducting dipole as well as "normal" detector data streams.

II. Basic LULU Structure

The program LULU is a housekeeping, sorting, and plotting package that operates on word lists generated by a series of user subroutines called ANALYSERS. By executing a series of commands LULU lets you look at raw and digested data, form simple statistics on specified data words, and selectively fill histograms and multidimensional scatter plot array. LULU takes advantage of the very large arrays allowed by the VAX VMS operating system to save every individual data point in a scatter plot rather than forming two-dimensional histograms. These saved arrays are then passed to a plot package where they are turned into the appropriate display. Thus the full range of data is always available to the plotter so that maximum resolution can be attained as the plot arrays are manipulated.

LULU can be run either interactively or in batch mode. In batch mode plots can be printed as they are ready through use of the PLOT command, or the whole plot array can be written out in the SAVE command and perused later using the program PLOTREV.

ANALYSERS are grouped together for loading and calling in a subroutine called LUANA. Each ANALYSER is expected to manipulate data and place them in an output array that is effectively equivalenced to a portion of a large array called the VALS array. LULU then works on words in the VALS array.

Communication with LULU is done through a series of commands and subcommands. There are 20 commands available, each having an arbitrary number of subcommands. These 20 commands are:

CLER	SORT	SHOW	PICT	HIST
SCAT	GO	EXIT	STAT	OPER
CNST	WRDS	LUNM	CONT	ANAL
GIOC	SAVE	FILE	ECUT	MULT

For illustration, we show in Fig. 4 the subcommand menu presented when the command FILE is executed. For instance, the subcommand COMD allows the user to enter the name of a file holding the command series for a particular analysis scheme; this is used on-line to load a series of cut and/or plot specifications and used off-line to load the entire command sequence for a complete analysis task to be run as a batch job. Other files are available for holding calibration constants or the results of intermediate analysis that may be read back into LULU to facilitate the analysis scheme being pursued presently. The subcommand DATA is used to specify the data file on which this run is to work. The FILE subcommand SAVE allows the user to specify a file into which processed data are written through the main command SAVE. This allows intermediate or final results to be saved in a form that can then be opened later as a DATA file for LULU to allow chaining of analysis tasks.

Before presenting a detailed description of the data flow through LULU, we point out a few of the features that it shares with other analysis programs. Working in LULU it is possible to form up to 100 scatter plots and 100 histograms simultaneously. (The limit of 100 is artificial.) These can then be added together, subtracted, normalized, cut, displayed with offsets, scaled with an arbitrary run-time produced function, fit to arbitrary functions, and sent to graphics, CRT, or hardcopy devices for viewing and saving. Displays generated from data passing through the complete LULU scheme can be viewed in a continuously updating fashion (useful in debugging detectors) or viewed event by event as well as viewing in the normal fashion when an analysis sequence is completed. Data can be viewed at any intermediate stage in the analysis through the SHOW command, whose subcommands allow the user to select the portion of interest.

A more detailed view of the structure of LULU and its modules is shown in Fig. 5. This diagram shows the arrays that are used, from the RAW data array (input array) through the analyser output array, VALS, and through the plot arrays, SCAT, to the final array that is displayed as a graph, XY.

III. Data Organization

As mentioned above, one of the primary reasons for developing this program was to be able to handle events of arbitrary length, arbitrary in the sense that the event length changes from event to event. The structure of an event from our recent run at HISS is shown in Fig. 3. An event may have, for instance, data from 3 TOF slats and 40 DC wires while another event may have 6 TOF slats, 60 DC wires, and 4 segments of the NaI detector. We could, of course, map each slat, wire, or crystal to a position in a large array and then address each array location with a specific word number. The problem would be that each event would then be a very sparsely populated array of 20-30k words. Rather, we have chosen to use a two-dimensional array structure, which results in small, densely populated arrays of data that are addressed by a 4 word specifier as shown in Fig. 6. This storage and retrieval method is at the heart of LULU.

To understand this structure we must discuss in more detail the role of user subroutines or ANALYSERS, up to 10 of which are allowed in any given LULU task. Just as an EVENT recognizes and is organized around the time correlation in a set of detector signals, an ANALYSER is meant to organize the raw data into groups of correlated words. ANALYSERS can be characterized by the complexity of the correlation grouping they perform. As the simplest example we show in Fig. 7 the "RAW" drift chamber analyser, which takes raw data in the form of crate number, module number, subaddress, and TDC value returned by the LeCroy 4290 system drift chamber TDC hardware and, after consulting a map, puts out data in groups of two words each consisting of the wire number and TDC values. A subsequent ANALYSER would then look at this array and group the wires into planes (paired wires), sum the adjacent TDC values, and compute x,y locations from the S,T,U planes in each drift chamber. As a second example, consider the two PMT's viewing each slat of the TOF wall. Signals from each PMT are ADC'd and TDC'd so that we get charge and time for every particle passing through the slat. The "RAW" analyser for the TOF wall then reads raw ADC and TDC values with addresses, passes these addresses through a map, and returns groups of five words each consisting of slat number, TDC1,ADC1,TDC2,ADC2, where 1 and 2 refer to the top and bottom PMT viewing each slat. The output of the DC RAW analyser would be an array 2xN where N is the number of wires that fired in the event, while the output of the TOF RAW analyser would be an array 5xM where M is the number of slats having signals in the event. Such RAW analysers, once debugged for map errors, etc. may then lose their status as analysers and become simple subroutines that are called by more complex calculating ANALYSERS. Or they may be used in a first pass LULU run to reduce the raw data to this intermediate form, with more complex anaysis being done with this first analysis as input.

By convention at HISS, all ANALYSERS with very general utility such as the DC RAW or TOF RAW analysers described above are written in a form that is independent of the main program; that is, relevant input and output arrays are passed as simple arguments so that these subroutines can be used in main programs other than LULU, should the user so desire.

The main concept of ANALYSERS is that they input arrays of correlated variables, calculate or mix them with other arrays, and output an array of digested data in correlated groups. Thus, a secondary analysis ANALYSER may take input from the RAW TOF and RAW DC arrays, see what DC hits point to which TOF slats, and combine these in groups that include slat number, charge, velocity, and x,y hit locations. This output may then be passed to the ANALYSER that converts track x,y locations to rigidity and pathlength by integration through the magnetic field.

We now discuss the way in which ANALYSERS are called by LULU. ANALYSERS are called through a routine called LUANA at any of four points in LULU. The point in the event analysis at which a specific ANALYSER is called is set at run time by setting a flag. This idea is shown in Fig. 8. It is often the case that we want to perform complete analysis on only a subset of all the events available. For instance, during on-line analysis we would not want to fully reconstruct all events because of the CPU overhead of track fitting ANALYSERS. We might then set the ANALYSER flags to call RAW DC and RAW TOF each event, then call charge computation ANALYSER for events having a certain multiplicity, then call complete reconstruction for events having a charge sum of six, multiplicity of three with good planes in all DC's. In passing over the data then, the track routines are called only for events passing this set of cuts. The output of this analysis can then be displayed through LULU or sent to a very specific graphics routine that actually draws these events in a cave view. An example of such output is shown in Fig. 9. This feature of calling sequence is very useful in monitoring specific event channels during an experiment, allowing the CPU to keep up with current data even while performing very complex calculations.

IV. PLOT Package

Moving downstream in the analysis flow we find the next point is entry to the PLOT package (see Fig. 5). First a short word about histograms: data can be cut before going in, the number of bins can be up to 200k, the population per bin can be 2**32, you can make 100 of them simultaneously, they can then be cut and displayed at will, added or subtracted in arbitrary numbers, normalized, pedestaled, etc. We spend most of our time analysing scatter plot arrays, which emphasize rather than hide correlations and provide much greater power in understanding our data. Histograms can be formed from the scatter plot arrays after correlations have been seen and understood.

The ability to form and save up to 100 multidimensional scatter plots, each having a different dimensionality, gives as much power in the plot package as you have in the main analysis sections of LULU. By selecting variables suspected of being correlated and placing them in groups in the SCAT array, you can then pick any two variables, cut on other values in the group or in the event (event structure is built into the SCAT array), and then display these in scatter plots or histograms. We show in Fig. 10 the subcommand menu for the PLOT command when multidimensional, as opposed to two-dimensional, arrays have been set up in SCAT. We shall spend the remaining time discussing two of the options available in this part of LULU.

One option in this package is to "select/define functions". Entering this menu option makes available the reverse polish notation (RPN) function generator whose menu is shown in Fig. 11. With this menu you can take any words out of each group and use them as variables in an arbitrary function, which is keyed in just as you would in a hand calculator. Such a function then becomes a subroutine through which all data passing previously specified cuts is passed. In Fig. 12 we show the utility of such an option in on-line analysis where we first collect the raw signals from the TOF wall and plot them as ADC product (proportional to fragment charge) vs TDC sum (a measure of time of flight). We see in Fig. 12 that the time is pulse-height dependent because we used leading edge rather than constant fraction discriminators. To remove this pulse-height dependence we make up a function that corrects time as

a function of pulse height. To get the best possible set of values for the constants in this function we use the "minimize functions constants" option of the PLOT menu and end up with the on-line correction to TDC values shown in Fig. 12. This makes it simple to see fragment velocity distributions on-line. In later analysis, once we have settled on the form of this function, we build it into the calculation ANALYSER for the TOF wall and minimize constants over a much larger data base, using the same PLOT option.

Another useful feature of the final plot section, the module entered after the x,y values to be displayed have been selected, is illustrated in Figs. 13 and 14. The TOF wall scintillator array is used to determine fragment charge as well as fragment velocities. To convert the ADC signals from each slat into charges requires calibration of the slat ADCs; relativistic protons give signals near channel 100 while relativistic carbon particles give signals near channel 1600 in the TOF ADC's. We accomplish the calibration of this detector system in the following way. Set up a scatter plot in which the x axis is the slat number and the y axis is the square root of the ADC product for each slat. Then cut the y axis to keep only those values lying between 0 and 300 channels (the proton region, gains having been set to place all proton signals at nearly the same pulse height).

We then enter the x,y peak finding algorithm defined as a menu option in the plot module. The action of this option is illustrated in Fig. 13. You are allowed to choose the size of x and y bins and then, for each x bin, the program histograms all y values in this bin, finds the peak, FWHM, etc. for each bin, displays each x bin histogram if desired, and stores the peak, FWHM, etc. in a separate save file that can be read by the TOF ANALYSER as a set of constants to convert ADC value to charge. Raw ADC products are shown as a function of slat number in Fig. 14a with the resulting charge distribution for each slat shown in Fig. 14b.

V. Conclusions

Our analysis program LULU has proven very useful in all stages of experiment analysis, from prerun detector debugging through final data reduction. It has solved our problem of having arbitrary word length events and is easy enough to use that many separate experimenters are now analysing with LULU. The ability to use the same software for all stages of experiment analysis greatly eases the programming burden. We may even get around to making the graphics elegant someday.

Acknowledgments

We would like to thank Chuck McParland, Mark Bronson, and Ernestine Baleal for making the data file handling routines clear and simple to use. We thank Doug Greiner, James Symons, and Mike Webb for their suggestions and help in making LULU more versatile. We thank Hester Yee for her help in preparing this manuscript. This work was supported by the Director, Office of Energy Research, Division of Nuclear Physics of the Office of High Energy and Nuclear Physics of the U.S. Department of Energy under Contract DE-AC03-76SF00098 and National Aeronautics and Space Administration grant number NGR 03-003-513.

References

1. H.J. Crawfnrd, "Study of ^{12}C Interactions at HISS", Lawrence Berkeley Laboratory report LBL-15447, 1982
2. Mark Bronson, "A Uniform Data Interface", this conference

DETECTOR SYSTEMS

1. SOLID STATE DETECTOR TELESCOPES
   1 - 50 ELEMENTS ≤ 100 WORDS
2. TIME OF FLIGHT SCINTILLATOR ARRAYS
   2 - 200 ELEMENTS ≤ 400 WORDS
   TDC AND ADC
3. MULTIPLE DRIFT CHAMBER ARRAYS
   20 - 4000 ELEMENTS ≤ 4000 WORDS
4. MULTIPLE SAMPLING IONIZATION CHAMBER
   128 ELEMENTS ≤ 12000 WORDS

Fig. 1. Detector systems in use at HISS and the number of words each detector may place in an event.

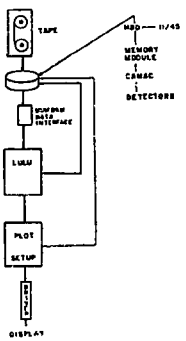


Fig. 2. Data flow through LULU showing both on-line and off-line data sources.

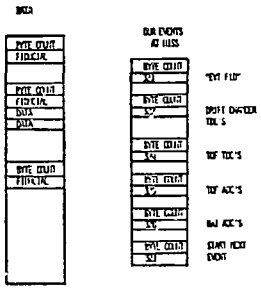


Fig. 3. The uniform data interface structure; general pattern on left, typical HISS data event on right.

FILE SECTION

COMMAND	FILE NAME	DESCRIPTION	
DATA		DATA INPUT FILE	HDF OR LUN 10
STAT		SUBROUTINE OUTPUT	LUN 18
COMD		COMMAND INPUT FILE	LUN 1
LIST		LISTED OUTPUT FILE	LUN 21
PLOT		PRINTER PLOT OUTPUT	LUN 21
SAVE		SELECTED PORTION OF VALS	HDF
PLSV		SAVED PLOT ARRAYS	LUN 23
REWI N		REWIND FILE N	
OPEN		OPEN ONE OF THE ABOVE FILES	
CLOS		CLOSE ONE OF THE ABOVE FILES	

ENTER COMMAND — OR TAKES YOU TO THE MAIN COMMAND MENU

Fig. 4. Main command FILE subcommand menu.

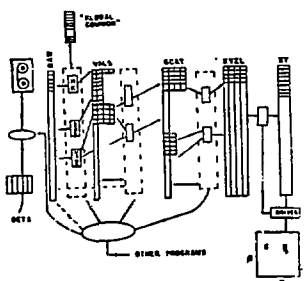


Fig. 5. Detail of array structure in LULU showing placement of user subroutines (USUs) and communication paths to disk files.

ANALYSERS"

"RAW" ORDER RAW DATA IN DETECTOR GROUPS
e.g. DRIFT CHAMBERS:
CHAIE #, MOD. #, SUB. ADDR., TDC
→ WIRE #, TDC

"FIRST PASS" PREPARE DATA FOR PHYSICS
e.g. TOF WALL
BLAT #, 11, 12, ADC1, ADC2 →
DRIFT CHAMBERS → TRACKS

"SECOND ETC." PHYSICS
E.T.P.R. → E.F. S2 ETC.

SUBROUTINES WRITTEN TO BE INDEPENDENT OF MAIN PROGRAM

Fig. 7. Examples of ANALYSER functions from the simplest "RAW" through the more complex physics calculating routines.

ANALYSER CALLING

FLAG:
0 OFF
1 CALL EACH FIDUCIAL SPECIFIED CUTS?
2 CALL EACH END OF EVENT CUTS?
3 CALL EACH END OF EVENT CUTS?

FILL PLOTS, STATISTICS, ETC.
PASS ON TO OTHER PROGRAM

Fig. 8. Run time calling command flags showing points at which "cuts" in data are made.

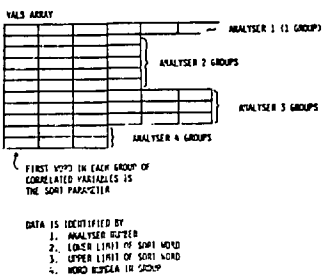


Fig. 6. Structure of the VALS array showing addressing scheme.

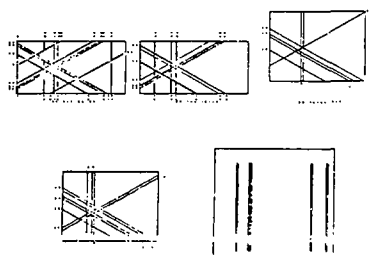


Fig. 9. View of DC wire hits and TOF slat hits produced by program RAWDLT from data selected and passed by LULU.

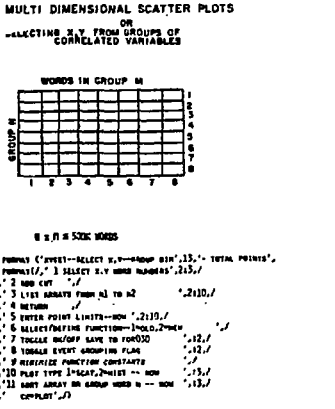


Fig. 10. Main command PLOT subcommand menu showing the structure of the SCAT array from which x and y variables are selected for display.

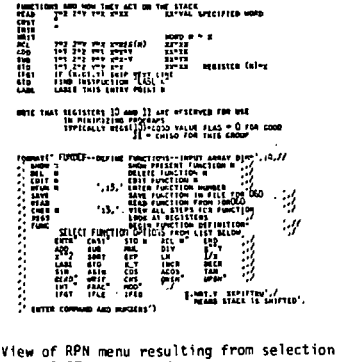


Fig. 11. View of RPN menu resulting from selection of option 6 in the PLOT subcommand menu.

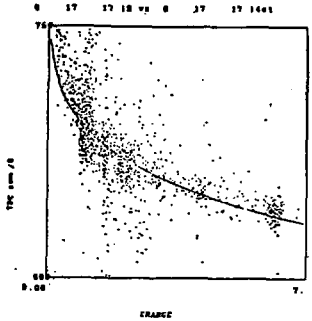


Fig. 12. Example of function found with RPN option. Function corrects time for pulse-height dependence in TOF slats. Title indicates word addressing scheme: analyser 6 sort word 17 parameter 12 vs 14 with cuts on data.

Fig. 13. Description of xy peak finding option in PLOT package.

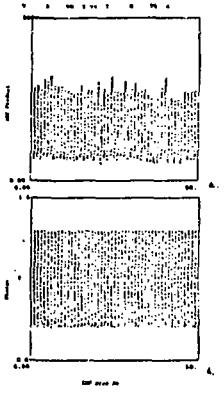


Fig. 14. Example of peak finding option used to calibrate 50 slates of TOF wall simultaneously for charge 1 fragments.