LABTOOLS version 2.0

I: Introduction.

The MATRIXWRITER of the hp 48 is a convenient tool for collecting data when conducting any kind of experiments. The calculator also offers features such as scatterplots with axes and labels, best-line aprox. and the possibility to export grobs to computers/printers. To use the 48 for collecting, processing and presentation of data may therefore seem like a good idea. However, a matrix is a rather rigid construction, so manipulation with the collected data is difficult. Graphical presentation of such data is also limited in a number of ways. Meeting these obstacles, most people will probably find the gain less than the effort, and abandon the 48 for such use. The aprox. 7.8 kB of RPL in this library is designed to be quick to learn, easy to use, yet still provide some powerful tools for fighting these limitations.

The user-interface of LABTOOLS consists of 5 programs for processing and presenting data from experiments. "data" is here to be understood as numerical values collected into a matrix. The programs in LABTOOLS can be applied on any such matrix. This is an upgrade of LABHELP 1.3. I felt LABTOOLS told more about what this is, namely a collection of tools. I also felt I should raise the version-number due to reprogramming most of the code, some of it into sys-rpl for faster execution and expanded capabilities. New features in version 2.0:

```
-An always well-behaved graphical domain-setting.
-A smoother, more consistent and overall better user-interface.
-New mark, type diamond, available.
-LABTOOLS 2.0 is (sometimes much) faster.
-APLEX can now take a program or global as input.
-LINEFIT leaves the best-line expression on the stack.
-Safer; to my knowledge no chance of losing data.
```

Plus a lot of minor changes intended to improve performance. A few bugs have been removed. This doc has been rewritten for hopefully greater clarity.

II.a: Description of the programs.

Programs: APLEX, LINEFIT, INTRPOL, MARK & POLISH.

The programs operate on the matrix in ∑DAT. The supplied CST-menu offers easy access to the tools and ∑DAT. When they do plotting related to the data-matrix, the programs use XCOL/YCOL, which must be valid for the current matrix. To insure proper execution of these programs, keep LAST STACK enabled.

APLEX (APLly EXpression) ('A+B'-> in CST)

```
(Expression ---> Expression)
(Program    ---> Program   )
(Global     ---> Global    )
```

Implements a spreadsheet-like function of applying an expression or program to any of the real numbers in the data-matrix.

As a running example, let's say you are measuring on an amplifier, putting the data into a matrix: In column 1 you put the resistance R of a pot.meter. Turning the pot.meter makes the voltage in, Vin, vary. This value you put in column 2. Finally, you put the voltage out, Vout, in column 3. If you now ENTER this matrix and store it in ∑DAT, you can easily make a scatterplot of, say, Vout vs. R and then perform a curvefit. But wouldn't it be nice if you now could enter the expression 20*LOG(Vout/Vin), the amplification In dB, into your HP, have it compute that ampl. for all your data-points and put the results into a new column of your matrix?

Generalised the problem is: Given a matrix with data where each column contains values for a specific physical quantity, and an equation connecting any number of those quantities to produce a result(a real number), how do you "feed the matrix to the equation", extracting the right data to output the right result, all within the '48? APLEX does just this for you with a few keystrokes.

First you must supply an input-expression which specifies the operation you want to perform. This can be any expression, just remember this: Substitute any variable-name(s) that have a corresponding column of values with a special name "column-name" that APLEX can interpret as that column. The column-names are 'A' for column 1, 'B' for column 2, 'C' for column 3,...and so on. Not too hard to remember, I hope :-) Our running example to clarify it: The values for Vout are in column 3, and those for Vin are in column 2. The corresponding column-names are C and B. Thus the input-expression should be '20*LOG(C/B)'. This also illustrates that any equation you would like to apply must be on the form 'result=expression'. Strip off result= and substitute column-names to get a valid input-expression. More about the input later. Start APLEX with such an expression in level 1, and this will happen: First you set the column where the results (F in our example) will be placed.

```
|Resultvalues placed|
|in column nr:      |
|::::n<             |
---------------------
```

The default number n is the rowsize + 1. If you now press ENTER, one column is added, and the result-values are placed there. You can, however, specify any other positive integer. If n is larger than the rowsize, columns are added as needed, and if n is equal to or less than the rowsize, the result-values will overwrite the old values.

Now there's one thing left before APLEX can start computing, and that is setting the domain, i.e. which rows in the data-matrix (stored in ∑DAT, remember?) APLEX should apply the expression to. This domain-setting is described separately below, as it is common to all the programs, except POLISH.

When you have sat the domain, APLEX evaluates the expression for the specified rows and places the result-values in the specified column. Try it, it's easy.

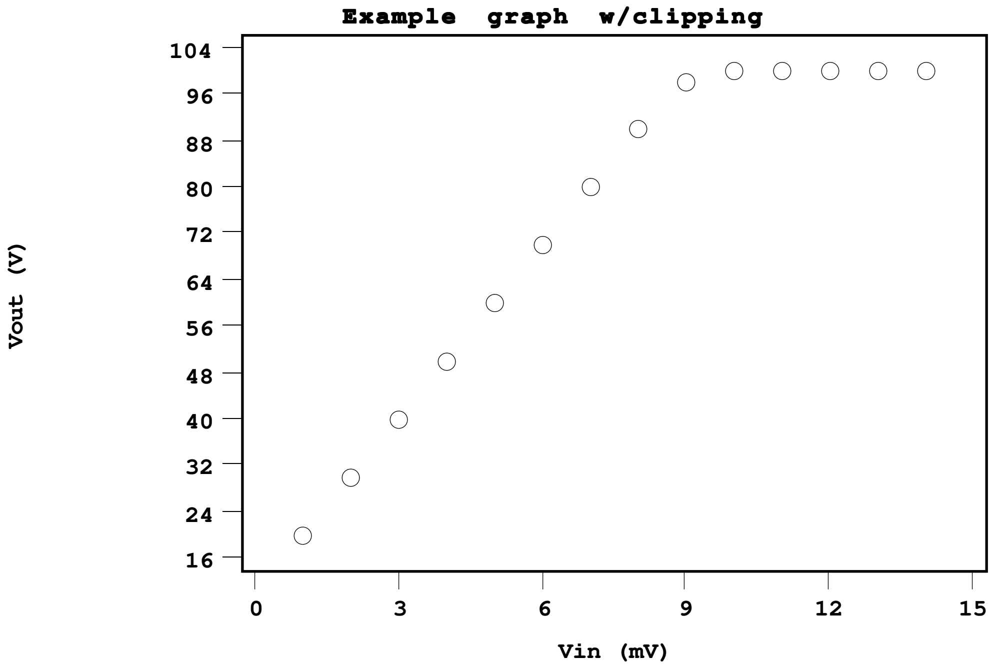
About the input: You can also supply a program as input, as indicated in the stack-diagram. Everywhere the word "expression" is used in the preceding and following, "program" applies as well.

The expression must contain at least one valid column-name, and APLEX assumes that any formal variable-name (doesn't exist along the current path) is a column-name, and uses the first letter in the name to determine which column it corresponds to. '20*LOG(CVout/BVin)' would thus work equally well in the example, and is recommended for legibility. Any global variables in the expression will be used correctly, but must of course be able to produce a num. value. Specifically, if such a global variable contains an expression, that expression can contain column-names and globals (which may contain expressions ...... ad infinitum), and they will be used correctly. A supplied global must contain an expression.

LINEFIT (An exp.line and the word "FIT" in CST)

```
( ---> Expression)
```

This program is an extension of the built-in curvefitting. LINEFIT lets you do a bestline-fit on part of a scatterplot. This is useful when you have a matrix with data from measurements on a system that changed characteristics during the data-recording. For example, our amplifier will start clipping when Vin exceeds a certain value. A scatterplot of Vout vs. Vin. might look like this:



Other such systems could be fluid-flow, stress/strain or really any system with a chance of some kind of overflow. The problem is that there is no way to tell the built-in curve-fitting to ignore the "flat" part of the plot, and therefore it can't tell you what's the best line through the linear raising part.

Or maybe you want to rule out the "bad influence" of points that are obviously in the wrong place. This is impossible with the built-in commands, but LINEFIT provides a solution to these problems by letting you choose a suitable domain for the best-fit. Pressing "ALL" is equivalent to using the built-in curvefit. "INCL" tells the program you want to perform a curvefit on part of the data-matrix, and "EXCL" tells it you want to rule out, or ignore, the effect of "bad" points. Having sat the domain, you next choose the linefitmodel. Select a model by pressing one of the corresponding keys. The curvefit is performed. LINEFIT is ended, leaving you in graphical env. The best-line-expression is put on the stack, and is also available via EQ (*Not* from ∑LINE).

INTRPOL (INTeRPOLate) (Three adjoined lines in CST)

```
Nothing from or to stack.
```

INTRPOL draws lines between the points in a scatterplot. The domain-setting described below is used: The effect of pressing "ALL" or "INCL" should be obvious. Pressing "EXCL" has the effect of bypassing the points you specify when you set the limits.

MARK (Three different marks in CST)

```
Nothing from or to stack.
```

As you may have noticed, the points in a scatterplot often get overdrawn when for ex. a curvefit is performed. This makes your data in effect disappear! Not very elegant. MARK marks your points with diamonds, boxes or circles (circles produces an error on the 48G series) to make'em visible. If you access MARK from the LIBRARY-menu, you choose between the three in the next-appearing menu. If you access MARK via the CST-menu, however, you choose by pressing the unshifted for diamonds, leftshifted for boxes and rightshifted menu-key for circles, corresponding to the position of each mark in the label-grob.

POLISH (A frame and axes in CST)

```
Nothing from or to stack
```

Press POLISH, and four menu-labels will appear.

"DONE" returns to the last menu.

"AXVAL" places values along the axes with a separation you specify. Give it in tick-marks (= 10 pixels) for each axis and press ENTER. The axes are marked and small arrows are put at the end of each axis. The values are rounded to 4 significant digits to limit space-requirements and codesize.

"FRAME" draws a frame around PICT. Points lying next to the frame are erased.

"AXLBL" labels the axes. "AXLBL" + "AXVAL" is meant to replace the built-in LABEL-command. The ":Size:"-prompt refers to the->GROB command, see owners manual if necessary. To properly label the Y-axis, the program rotates the GROB made from your text 90 degrees counterclockwise. Set flag 55 if you can't wait the max. 10 seconds it'll take. The algorithm used to determine where to place the labels is not very clever, but should work well in most cases. If you want full control over the label-placing, install the graphics editor PixPhics. AXLBL will then use it to let you move and place the labels where you want'em, see PixPhics doc. If you choose not to install PixPhics, you need to have aprox. 20 pixels under the X-axis and 30 pixels to the left of the Y-axis to properly place the labels there. (PixPhics was to be released autumn -93, but unfortunately never made it public, got 85% finished)

II.b: The domain-setting.

The domain-setting is a sub-routine which lets you pick out the portion of the data-matrix you want to work on. It starts by displaying a menu: "ALL" selects the entire matrix. If you only want to work on part of the matrix, press "INCL" or "EXCL", in which Case you are brought to a second menu. There you are asked how you would like to specify the limits; by rownumbers "STAC" or by scatterplot "SCATR". If you pressed "INCL" in the previous menu, the limits define a portion that is to be the domain, and if you pressed "EXCL"the limits define a portion that is to be excluded from the domain, which will be the rest of the data-matrix.

Press "STAC", and you are asked to give the limits this way:

```
|Specify limits |
|by rownumbers  |
|:From:<        |
|:To:           |
-----------------
```

Give the limits and press ENTER. For ex. if you first pressed "INCL", then "STAC", and supplied :From:2 :To:14 , the domain will be row 2 to row 14 in the data-matrix. If you first pressed "EXCL", the domain will be all the rows except rows 2 to 14. Notice that if you want the upper limit to be the last row, but you're not sure how many rows there are, you can just supply a high number, known to be larger than the number of rows.

Or press "SCATR". (Reset XCOL/YCOL first if necessary). A scatterplot is now drawn, using XCOL and YCOL, and control is given to the graphical environment. Now move the haircross so that it is nearest (meaning nearer in x-value than to any other point in the scatterplot) to the point representing the row you want as the lower/upper limit and press ENTER, then move it to where it is nearest to the point representing the row you want as the other limit and press ENTER. The presses represent limits, so always press at least twice (your two last presses are used). Then press ATTN (ON).This returns control to the main program.

III: Presenting dataplots.

You may have two intentions when you fill your '48 with experimental data: processing and presenting. The goal of the latter could be the graph from the '48 included in your report. Short manual:

Experiment. Collect data in matrix. Store in ∑DAT. Set the PICT size: The bigger, the better. A PICT 500 pixels wide will fit in on a paper (A-4). Use the built-in SCATTER to draw an autoscaled scatterplot. Reset CENTR in the PLOTR-menu to a nearby nice value, and reset SCALE to the next, bigger, nice value. You decide what a 'nice" value is, but use common sense to avoid '0.1033853333'-values and to let the axis-scales express something sensible. Set AXES where you want'em, not necessarily, but preferably somewhere with enough room to put values and labels on them. ERASE and reDRAW. This prevents a new autoscaling, and your plot looks better. (If you want several plots in the same PICT, reset XCOL/ YCOL as needed, and use DRAW in the PLOTR-menu, since SCATTER always erases/autoscales PICT.) Use LABTOOLS as needed. Export GOB to computer and paste it into your now sharp-looking report.

IV: Respond-wishes.

LABTOOLS is freeware. That means, feel free to send me comments, criticism, suggestions, opinions, questions. Any of them in any order, in any amount. If you don't have net access you can reach me at my physical address:


```
Lars A. Gundersen
Bygdoy Alle 18
0262 Oslo
Norway
```


I've spent *many* hours working on this application, so if you find it useful, drop me a line saying so, and I will be happy! "Every program can be shortened with at least one instruction and contains at least one bug. Therefore it follows by deduction that every program can be reduced to one instruction which doesn't work." Please report bugs. I would also be grateful if you would help spread this to places near you where you think they might find LABTOOLS useful.

V: (Dis)Claims and credits.

This application is a mixture of user-and system-rpl. No warranty is issued for fitness, performance or anything else! You are granted the right to copy LABTOOLS to others, provided *all* the code is copied, including the 'CST'-variable. This doc must also be passed on; LABTOOLS will be useless without it.

Thanks to Conrad Winchester for his fast ML circle-procedure. Thanks to Hewlett Pacard for the 'HP 48 Programmer's Reference Manual'. Thanks to R. Ray Depew for his book 'HP 48 Graphics'. Thanks to Nicolai for encouragement and help.

VI: Installing.

LABTOOLS 2.0 is implemented as a library, ID number 847. To install, perform these steps: Download the library to the '48. Recall the variable to the stack (and press ASC-> if necessary). Execute :n: 847 STO, where n is the port where you want LABTOOLS to reside; 0, 1 or 2. PURGE the variable you downloaded. Turn the '48 off and then on again. LABTOOLS is now ready for use. Download and copy the 'CST'-variable to directories where you want easy access to LABTOOLS.

To get rid of, perform these steps: Go to the HOME dir. Execute 847 DETACH and :&: 847 PURGE.

If you think my English sucks, you should just be glad I didn't write this in Norwegian, which is my native language...

Lars A. Gundersen

lag@fys.uio.no http://www.fys.uio.no/~lag/