Erable 3.2

Bernard Parisse*
Institut Fourier (CNRS UMR 5582)
Université de Grenoble I
F-38402 St Martin d'Hères Cédex
Bernard.Parisse@ujf-grenoble.fr

November 3, 1998

Abstract

This is the manual for the Erable computer algebra system for the HP48 calculators. Section 1 describes the end user license. Section 2 describes how to install Erable on your HP48. Section 3 gives an overview of the capabilities of Erable, compared to other computer algebra systems available on calculators. Section 4 explains the user interface of Erable. The rest of the manual gives more details on Erable functions, topic by topic, and the appendix, starting with Frequently Asked Questions (section A). gives some references.

This manual is intended more as a reference manual than as a tutorial. It is not intended as a replacement for a textbook or as remedial instruction.

Special thanks to Randolph J. Herber who helped me a lot in porting the manual from version 3.024 to version 3.2

Contents

1 License.

Erable v3.2 © 10/1998 by Bernard Parisse, with code from ALG48, © 1998 by Claude-Nicolas Fiechter and Mika Heiskanen,

The Erable package includes the ERABLE library itself and the add-ons libraries ARIT, GEO, LIN and PREP. A part of the ERABLE library is a work based on ALG48 (using long integer routines) hence is subject to the license of ALG48 (see the file license.txt). See eqstk.doc in eqstk.zip for license information about eqstk and readme.txt for information about minwriter. The rest of the package is subject to the ERABLE license (see the file copying.doc).

2 Installation.

This section describes the installation of Erable with using the Kermit or X-Modem file transfer protocol. You must choose one of the methods described in section 2.1 or in section 2.2. The first method will install Erable and a "pretty print" stack displayer (a modified eqstk), which is recommended for new users. Choose the second method if you want to install Erable using X-modem or kermit, this method is not recommended for beginners.

2.1 Simplified installation.

Erable now provides a simplified installation procedure which should make it much easier for new users. Thanks to Mika Heiskanen and Claude-Nicolas Fiechter for letting me modify and distribute their eqstk library, to Jean-Yves Avenard (for the minwriter), and to Andre Schoorl (for ufl102) for letting me include their programs in the Erable distribution.

1. Go to the directory where you have unzipped erable.zip.
2. Run Kermit or some other Kermit file transfer protocol capable communication program or terminal emulator on your computer. If you don't have Kermit, you can get it at, e.g.:

   ftp://kermit.columbia.edu,
   http://www.columbia.edu/kermit/index.html
   or at a mirror site.

   If your version of Kermit does not work under Windows, restart your computer in MS-DOS mode and try again.
3. Set the line to the HP48, e.g. type:

   ```
   set port 2
   ```

   if the HP48 is connected to the second serial port COM2 (for Linux type set line /dev/cua1).
4. To insure binary file transfer, type:

```
set file type binary
```

5. Set the speed of `Kermit` to 9600 baud, type:

   ```
   set speed 9600
   ```

6. To put `Kermit` in server mode, type:

   ```
   serv
   ```

7. Now take your HP48. If you do not have any ram card, you must have at least 120K free in your HP48. If you do not have important data in your memory, press the `ON` key, then the `A` key, then the `F` key and release them, then press the `F` key to answer `NO` to `Try to recover memory`. Otherwise, if you are running a stack replacement like `java` or `eqstk`, switch to the HP48 built-in stack handler.

8. Type the left shift key followed by the 1 key, type the `B` menu key to select `IOPAR`, press the `A` menu key until line 1 looks like:

   ```
   IR/wire: wire
   ```

9. Press the α key, and type:

   ```
   S E T U P  K G E T
   ```

   release the α key and press the `ENTER` key. this will get the program `SETUP` from the computer to the HP48. If an error occurs, verify the configuration of `Kermit` and try again.

10. Press the `VAR` key, then the `A` menu key (for `SETUP`), this will get and install the binaries for you. If you don't have enough memory, an `Insufficient memory error` will occur. You may get a `Object in use` error if you are running a stack replacement like `java` or `eqstk`, in this case, exit them (for `java`, type `JAVA` and for `eqstk`, type simultaneously `ON` and `C`). Otherwise, after about 10 minutes your HP48 will reboot.

11. After the reboot, press the `VAR` key, then the `E` menu key (again for `SETUP`). This will continue the installation and reboot the calculator after about 1 minute.

12. Eventually press the `VAR` key, then the `A` menu key (for `INIT`). This will launch the modified `eqstk` stack handler. Congratulations, you're done!!

The installation is complete. All should work smoothly. You can go to section 3.

2.2 Personalized installation.

You must first get the binary either from a computer or from another HP48 where `Erable` is installed. Then you have to install the binary and eventually install the user keys redefinition. Four binaries are provided, you must choose one of them depending how you will install `Erable`:

- `ERABLE0.LIB`: must be installed at the end of port 0: it is intended for G+ users or GX users without ram cards.
- `ERABLEMK.LIB`: must be installed at the end of port 0 (like `ERABLE0.LIB`) but for `Metakernel` users (version 2 or higher)
- `ERABLE1.LIB`: must be installed at the beginning of port 1 (GX users) or port 2 (SX users).
- `ERABLE.LIB`: like `ERABLE1.LIB` but using the built-in polynomial factoring command, it saves 2.5K of RAM but will sometimes not work on a SX.

Before installing `Erable`, you must empty the port where you will install it. For port 1, this can be done by typing:
`1 PVARS`
look at the list at level 2. If it is empty, you are done, otherwise you must purge the libraries and backup objects (you can save them in variables before), e.g. by exploding the list at level 2:

```
DROP EVAL
```

then delete the libraries on the stack:

```
DUP DETACH PURGE
```

For port 1 (or 2 on a SX), the best method is to merge the card (`1 MERGE`), get `ERABLE1.LIB`, then free the card (`{ :0:788 } 1 FREE`).

2.2.1 Getting the binaries from a computer.

Prepare your computer to send data:

- `Kermit` users: run `kermit` in server mode.
- `X-modem` users: run your client (e.g. `minicom` for the `Linux` operating system), select `send file`, write the names of the files to transfer. Note that the computer must be ready to send files before you type the HP48 `XRECV` command.

Connect your HP48 to your computer and type

```
{ ERABLEx.LIB GXKEYS } KGET
```

for `Kermit` or

```
{ ERABLEx.LIB GXKEYS } XRECV
```

for `X-modem` where `ERABLEx.LIB` is one of the four flavours of `Erable`. You can get some of the optional libraries (`ARIT.LIB GEO.LIB LIN.LIB PREP.LIB`) and/or directories (`other crypt decrypt`) as well.

2.2.2 Getting the binaries from another HP48.

This method works only if are able to use the same flavour of `Erable` on both HP48s. Check that both HP48s are in `IR` mode for `I/O` transfers. If `Erable` is installed in a RAM card, check that the RAM card is writable (otherwise the `RCL` function below will fail). Put the receiving HP48 in server mode. On the sending HP48 (where `Erable` is installed), call the `SENDIR` program if you have it or type the following little program:

```
<< :&:787 RCL :&:788 RCL RCLKEYS ->
   ERABLE.LIB UKEYS
  << { ERABLE.LIB UKEYS } SEND
  >>
>>
```

and `EVAL` it.

2.2.3 Installing the binaries

The `Erable` library must be installed in the correct port (0 for `ERABLE0.LIB` and `ERABLEMK.LIB` or if your ram card is merged, otherwise 1 for a GX or 2 for a SX), type:

`'ERABLEx.LIB' DUP RCL SWAP PURGE 0 STO` for port 0
`'ERABLEx.LIB' DUP RCL SWAP PURGE 1 STO` for port 1

where `ERABLEx.LIB` stands for the name of the flavour you have selected. The remaining libraries may be installed in any port, for example in port 2:

```
'ARIT.LIB' DUP RCL SWAP PURGE 2 STO
```

2.2.4 Installing the user keys redefinition.

This step is optional but recommended. If you got `algb` or `algbg` from your computer, go in the `algb` or `algbg` folder and hit `INIT`. This will assign keys for the user mode. If you got `GXKEYS` directly, type:

```
'GXKEYS' DUP RCL SWAP PURGE STOKEYS
```

to assign keys for user mode.

2.2.5 Improving your installation

If you have the `Metakernel`, you should put the following program in the `STARTEQW` variable:

```
<< 27 SF MAIN EVAL 27 CF>>
```

to enable the use of the main `Erable` commands in the `EQW` environment by hitting the `CST` key.

If you have enough memory, you can download the optional components of `Erable` like the `other` directory or the on-line help (`fr` or `us`).

If you choose not to install `EQSTK.LIB`, you must unassign the left shift-down arrow key by typing

35.2 DELKEYS

2.2.6 Abstract of the installation commands

This is the list of commands in the `algb` or `algbg` directory:

- `GETALL`: down loads directories with some programs using `Erable`
- `INIT`: down loads the `Erable` libraries if they are not installed and reboots. Otherwise, installs `GXKEYS` as user keys redefinition and reboots.
- `PURG`: purge the `algb` or `algbg` directory
- `SENDIR`: used to transfer the `Erable` libraries and the user keys assignment from one HP48 to another HP48. Before calling `SENDIR`, you must check that both HP48 are in `IR mode` (in the `I/O` menu) and you have to put the receiving HP48 in server mode.
- `SETFR`: sets some flags: e.g. European date display, or radian mode, ...

3 Introduction.

3.1 Overview.

`Erable` is a computer algebra system for the HP48. The main features are simplifications (including complex arithmetic and square roots), integration, first order differential equations, partial fraction decomposition, Laplace and inverse Laplace transform, limits, Taylor and asymptotic series, row reduction to echelon form of matrices, linear system (including over and under determined), eigenvalues and eigenvectors, quadratic forms, permutations, variables substitution, ... With `Erable` you will be able to solve most problems solved by a TI92 and some problems which are not solved by a TI92: some integrals (the Risch algorithm is not implemented in the TI92), some Taylor series, arithmetic, diagonalization of matrices, change of variables, ... **If you have both `alg48` and `Erable`, then you have the most complete computer algebra system currently available on a calculator (HP48, TI and CASIO).**

Examples:

$$\ln(1+i) \quad \texttt{EXPAND} \quad \frac{\ln(2)}{2}+\frac{1}{4}\pi i$$

$$\frac{1}{e^x-1} \quad \begin{matrix}\texttt{INTVX}\\ \texttt{EXPAND}\end{matrix} \quad -x+\ln(e^x-1)$$

$$\lim_{x\to 0}\frac{y^x-1}{x} \quad \texttt{LIMIT} \quad \ln(y)$$

$$\begin{pmatrix} 1 & 1 & a \\ 1 & a & 1 \\ a & 1 & 1 \end{pmatrix} \quad \texttt{rref} \quad \begin{pmatrix} 2-a-a^2 & 0 & 0 \\ 0 & -2+a+a^2 & 0 \\ 0 & 0 & 2-a-a^2 \end{pmatrix}$$

$$y' = xy^2 \quad \begin{matrix}\texttt{DESOLVE}\\ \texttt{EXPAND}\end{matrix} \quad y = \frac{y_0}{e^{-1/2x^2+1/2x_0^2}}$$

Examples not solved natively by a TI89:

$$\sin(x)/(e^x-1), x, 4 \quad \texttt{SERIES} \quad 1 - \frac{1}{2}x - \frac{1}{12}x^2 + \frac{1}{12}x^3 + O(x^4)$$

$$\cos(x)^2 \quad \texttt{LAP} \quad \frac{1}{2x} + \frac{1}{4(x-2i)} + \frac{1}{4(x+2i)}$$

$$\frac{1}{(x^2+1)(x+1)} \quad \texttt{ILAP} \quad \frac{-1}{2}\cos(x) + \frac{1}{2}\sin(x) + \frac{1}{2}e^{-x}$$

$$\int_0^1 (1+2x^2)e^{x^2}\,dx \quad \texttt{EXPAND} \quad e$$

$$\int_a^b \frac{e^x}{x}\,dx, \quad e^x = y \quad \texttt{EXEC} \quad \int_{e^a}^{e^b} \frac{1}{y}\cdot\frac{y}{\ln(y)}\,dy$$

$$\begin{pmatrix} 0 & 1 \\ -1 & 2 \end{pmatrix} \quad \texttt{JORDAN} \quad \texttt{Char :1:}(1,0); \texttt{Eigen :1:}(-1,-1)$$

3.2 Warnings.

- Using a computer algebra system does not mean that you don't have to think. Most of the time, all works perfectly and you get the answer within 30 seconds. But sometimes, after 1 or 2 minutes, you don't get the answer or you get a `Insufficient memory error`. In this case, you should think "Is there a different way to get the answer? Is there a way which will be easier for the system?" And most of the time, there is a better way! Think of double integrals where you can reverse the integration order or define integrals where you may do a variable substitution to have less variables, or linearity in inverse Laplace transforms, ... You should learn mathematics and algorithms to get the best use of any computer algebra system. And a system is never complete, you will need to program sometimes!
- Most of the problems in the real life don't have exact answers but can only be solved approximately. Think of integrals, differential equations, large matrices (say e.g. 80×80), ... Before learning how to solve exactly a problem, I strongly recommend that you learn how to solve numerically a problem. Then for a real life problem, you will know when you must stop finding an exact solution and begin to use a numerical algorithm.

3.3 `Erable` and `alg48`.

`Erable` is partially derived from the `alg48` package. The arithmetic functions of `Erable` are derived from those of `alg48`. `Erable` and `alg48` have some other

common features like simplifications, partial fraction expansion or integration. The main differences are:

- `Erable` handles complexes and square roots natively. It means that for example $\sqrt{5}$ is considered as a constant not as an unknown, hence: `GCD1(X^2-5,X-2*sqrt(5)*X+5` will return `X-sqrt(5)`.
- `Erable` accepts strings embedded in symbolics, this means that if you `EXPAND` $(5x+12)^{16}$ with `Erable` you'll get the exact answer. You may also do arithmetic operations on strings representing integers, e.g., try:

  ```
  "123456789123456789" DUP MULT
  15241578780673678515622620750190521
  ```

- `Erable` handles numerical data (reals which are not integers and complexes with non integral real and/or imaginary parts)
- `Erable` has a partial implementation of the Risch integration algorithm: it handles most of the common integrals.
- `alg48` implements the complete factorization algorithm over the rationals, `Erable` finds only first order factors of the square-free factorization and then, for 1-variable polynomials, calls the numeric solver if necessary and tries to rebuild 2nd order factor. Hence, `alg48` factors the expanded form of $(x^4+x^3+1)(x^4+x+1)$ but not x^4+1 and `Erable` does not factor the first example but factors the second one.
- The main specific feature of `alg48` is the Gröbner base computation. The main features of `Erable` are eigenvalues and eigenvectors of matrices, differential equations (first order: linear, Bernoulli, homogeneous; linear with constant coefficients), limits, Taylor and asymptotic series, quadratic forms, permutations, variable substitutions.

If you have enough memory, do like me: keep both on your calculator and choose the right instruction!

3.4 Implementation notes.

This software is written in `ML` and `SysRPL` with HP48 supported entries and standard instructions (and a few unsupported but static entries), hence it should work on all versions of the HP48. Of course, you should backup your calculator before trying it: no software is bug free!

This package was written on a 90MHz Pentium PC running under Linux (RedHat 5.0), the GNU emacs editor, my patched version of the x48 emulator (with almost instantaneous file transfers) the `JAZZ` `SDB/DB` debuggers, the gtools package (HP48 GNU compiler), Kermit and Minicom (sx/rx).

3.5 Next upgrades.

The latest versions are available by anonymous FTP at the URL:

`ftp://fourier.ujf-grenoble.fr/pub/hp48/`

If you have a WEB browser (such as Netscape), you may prefer to go to my professional home page:

`http://www-fourier.ujf-grenoble.fr/ parisse/english.html`

or to my personal home page:

`http://perso.wanadoo.fr/bernard.parisse/english.html`

4 Getting started.

Like any computer algebra system (CAS), `Erable` provides functions which may behave differently depending on the current status that may be configured. Like other CAS, `Erable` can use algebraic notation, but in addition you may use the power of the postfixed RPN notation. We begin by exploring the most important features of `Erable` that may be configured, then we will explain the main functions of `Erable`.

Warnings:

- If you have choosen the simplified installation, check that you have did step 12, if not type the `VAR` key then hit the menu key which corresponds to `INIT` and go to section 4.1.
- If you have choosen the personalized installation method, please check that you have installed the user keys redefinition as explained in section 2.2.4. If you have installed `EQSTK`, type `ASTK` to activate it.
- Verify that your calculator is in user mode. Look at the status lines, if `USER` appears then you are done. Otherwise (or if `NORM` appears when using `JAVA`), type the left shift key followed by the α key once or twice (try once and type a key, if `USER` disappears then type the left shift- α key sequence twice).

4.1 Current variable.

The current variable is the name of the unknown that is used for common operations like derivation, integration, factorization, series expansion, ... When you start `Erable` (and each time you reset it by typing `VER` followed by the `ENTER` key), the current variable is set to `'X'`.

The `DEL` key is assigned to `X`. It is therefore easy to enter a symbolic expression depending of `X` because you do not need to press the α key.

4.2 Complex and real modes

`Erable` has two major modes: complex and real mode. This mode affects the way `Erable` outputs results. For example, factorization or partial fraction decomposition is made over $\mathbb{C}$ or over $\mathbb{R}$ depending of the current mode. If you see an unwanted i on the stack, this means that `Erable` is probably in complex mode.

4.3 Exact and numeric mode.

Most of the time, you will use `Erable` in exact mode, where no approximation are made. But sometimes, exact solutions can not be computed on the HP48 (e.g. if you want to factor $x^5 + x + 1$), and `Erable` will switch to the numeric approximation mode where e.g. $\sqrt{2}$ is handled as 1.414235....

4.4 Main functions of Erable.

The HP48 arithmetic operations ($+$, $-$, $\times$, ...) have a corresponding `Erable` command (`ADDT`, `SUBT`, `MULT`, ...) that you can easily access by preceding the usual HP48 key with the $\boxed{\alpha}$ and right-shift keys. For example, the `ADDT` addition function of `Erable` is obtained in `USER` mode by hitting three keys: $\boxed{\alpha}$, $\boxed{\text{Right Shift}}$, $\boxed{+}$. More precisely the $+$, $-$, $*$, $/$, y^x, $\sqrt{x}$, $\pm$, $1/x$, ∂, $\int$ keys are redefined (after $\boxed{\alpha}$ and $\boxed{\text{Right Shift}}$) as `ADDT`, `SUBT`, `MULT`, `DIV1`, `POWER`, `SQRT`, `CHS`, `inv`, `DERVX`, `INTVX`. These commands are `Erable` commands for addition, subtraction, multiplication, division, power, square root, change sign, derivation and integration with respect to the current variable [1].

Many functions of `Erable` may be reached by the `MTH-Erable` menu or sometimes by a key preceded by α-right shift. To launch the `MTH-Erable` menu, type the `MTH` key, use the white `A-F` menu and `NXT` keys to select a topic and select the desired command again by hitting a menu key.

Remark 1 *A beginners menu of the HP48GX is also available by hitting the* `PRG` *key and may be useful for beginners. Note that the former* `MTH` *and* `PRG` *key definition are reached by α-right shift-*`MTH` *or α-right shift-*`PRG`*. If you want to delete the* `PRG` *key assignment, type* `22.1 DELKEYS`*.*

An easy way to configure `Erable` is the `ERCFG` command which may be selected from the `MTH` menu of `Erable`. You can alternatively configure `Erable` using flag setting commands (`SF` and `CF`) or storing desired values for the current variable (`VX`), etc.. For example, you can change the content of the `VX` variable to `'Y'` using the `ERCFG` command or go to the `{ HOME }` directory, and type:

```
Y 'VX' STO
```

[1] This variable is contained in 'VX', it is by default set to 'X', note that you can easily type X by hitting the DEL key

You can directly reach a topic of `Erable` by hitting left shift followed by the number of the topic in the `MTH-Erable` menu. For example, if you hit left shift-1, you will get the `BASE ALGEBRA` menu, that is the most useful commands of `Erable` that are illustrated with an example below:

- `EXPAND` (expand an expression):

  $\frac{(X+1)^4}{X^2-1}$ `EXPAND` gives $\frac{X^3+3X^2+3X+1}{X-1}$

- `FACTOR` (factor an expression):

  X^4-1 `FACTOR` gives $(X^2+1)(X+1)(X-1)$

- `EXEC` (to make a substitution at stack level 2, you must specify the substitution at level 1, e.g. `'X=1+Y'`):

  $\ln(X^2+1)+\arctan(X)$, $X=2$ `EXEC` gives $\ln(2^2+1)+\arctan(2)$

- `DERVX` (derive an expression with respect to the current variable):

  $\ln(\ln(X))^2$ `DERVX` gives $2\frac{1}{X}\frac{1}{\ln(X)}\ln(\ln(X))$

- `INTVX` (integrate an expression with respect to the current variable):

  $e^X\sin(X)^2$ `INTVX` gives $e^X\frac{1}{2}+e^X((-\sin(2X)\frac{1}{5}+\frac{-1}{10}\cos(2X))$

- `LIMIT` (of the expression at stack level 2, you must specify the variable and the limit point at level 1, e.g. `'X=0'`):

  $(1+\frac{1}{X})^X$, $X=\infty$ `LIMIT` gives e

- `TAYLOR0`: Taylor expansion of order 4 at $x=0$ (where w is the current variable):

  $\sin(X)$ `TAYLOR0` gives $X-\frac{1}{3!}X^3$

- `SERIES`: Taylor expansion with respect to a variable at a specified order, example:

  $\sin(X)$ X 4`SERIES` gives $X-\frac{1}{3!}X^3$

- `SOLVEX`: tries to find the current variable values that zero the expression at level 1:

  X^4-1`SOLVEX` gives `{ 'X=1' 'X=-1' }`

- `PLOTSTK`: plot stack level 1

- `PLOTADD`: adds the graph of a function to the graph of previous functions.

Some of these commands are directly assigned to a shortcut:

- the `EXPAND` command is assigned to the α-right shift-`SPC` sequence,

- the `EXEC` command is assigned to the α-right shift-`EVAL` sequence,

- i is assigned to the `CST` key,
- X is assigned to the `DEL` key,
- ∞ or $+\infty$ is assigned to the α-right shift-`DEL` sequence,
- $-\infty$ is assigned to the α-left shift-`DEL` sequence,

Here is a brief description of all the functions of `Erable` which may be launched this way:

- left shift-1: basic algebra commands:

  `EXPAND`, `FACTOR`, `DERVX`, `INTVX`, `LIMIT`, `TAYLR`, `EXEC`, `SOLVEX`
  `PLOTSTK`, `PLOTADD` (see above)

- left shift-2: complex:

  `inv` `abs` `neg` `re`, `im`, `conj`, `arg`

  Note that you can get i if you hit the `CST` key.

- left shift-3: trigonometry:

  `EXPLN` (convert trigonometric expressions to exp and ln)
  `SINCOS` (convert complex exp and ln to trigonometric expressions),
  `TLIN` (trig. linearization),
  `TCOLLECT` (linearization and sin/cos combinations)
  `TRIG` (applies $\sin^2 + \cos^2 = 1$ to simplify an expression),
  `TRIGCOS` (same as `TRIG`, returns only cosines if possible),
  `TRIGSIN` (same as `TRIGCOS` but returns only sines if possible),
  `TAN2SC` ($\tan(x) \rightarrow \sin(x)/cos(x)$) `HALFTAN` (convert to tan of the half-angle),
  `TAN2SC2` (convert tan to sin and cos of the double angle) and
  `ATAN2S` (convert arctan to arcsin)
  `ASIN2T` (arcsin to arctan)
  `ASIN2C` (arcsin to arccos)
  `ACOS2S` (arccos to arcsin)

- left shift-4: matrices

  `NORM` (norm of a vector)
  `CROSSP` (cross product of 3-d vectors)
  `DOTP` (dot product of vectors)
  `IDENTITY` (symbolic identity matrix),
  `TRAN` (transpose of a matrix),
  `TR` (trace of a square matrix),
  `det` (determinant),
  `rref` (row reduction to echelon form),
  `REF` (rank of matrix),
  `JORDAN` (diagonalization of matrices),

PCAR (characteristic polynomial),
SYST (solves a linear system),
SOLGEN (returns all solutions of a linear system),
RDET (determinant by row reduction),
LCXM (build a matrix),
HILBERT (build Hilbert matrix) and
VANDERMONDE (build Vandermonde matrix)

- left shift-5, calculus:

  DERVX (derivative),
  INTVX (integration),
  LIMIT (limits),
  TAYLOR0 (Taylor at 0, order 4),
  SERIES (asymptotic series expansion),
  SOLVEX (solve for X),
  PLOTSTK (plot stack),
  PLOTADD (add stack to plot),
  IBP (integration by parts),
  PREVAL (primitive evaluation),
  RISCH (integration, Risch algorithm),
  DER (derivative),
  DESOLVE (ordinary first order differential equation solver),
  LDEC (linear differential equation with constant coefficients solver),
  LAP (Laplace transform) and
  ILAP (invert Laplace transform)
  TEXPAND (transcendental expand)

- left shift-6, integer and polynomial arithmetic:

  DIV2 (Euclidean division),
  GDC1 (usual greatest common denominator),
  GCD3 (extended gcd),
  LGCD (list GCD),
  LCM1 (least common multiple),
  SIMP2 (simplification of stack levels 2 and 1),
  PARTFRAC (partial fraction decomposition),
  PROPFRAC (integer part/fractional part decomposition),
  PTAYL (Taylor method for polynomials),
  HORN (Horner evaluation),
  ISPRIME (Pseudo-prime test),
  NEXTPRIM (next prime),
  PREVPRIM (previous prime),

- left shift-7, solve and factorization:

  SOLVE (isolate a variable in an equation),
  ZEROS (same idea as SOLVE, different return format),

FROOTS (roots and poles of a fraction with multiplicity),
FCOEF (reverse of FROOTS),
FACTOR (factorization),
DIVIS (list of divisors),
SQRT (square root)

- left shift-8, exponentials and logarithms:

  TEXPAND (transcendental expand)
  LIN (linearization of exponentials),
  LNCOLLECT (collect logarithms),
  EXPLN (convert sines and cosines to exponentials),
  SINCOS (convert complex exp and ln to trigonometric expressions),

4.5 The beginners menu.

If you hit the PRG key in user mode, you will get a menu for the HP48 as well as for Erable. This menu will help new users of the HP48. Experienced users can remove this key assignment, saving some memory (using the command 22.1 DELKEYS). Here is a brief description of this main menu:

1. EDITORS:
   - EDIT LEVEL 1:
     To modify the object at stack level 1 (shortcut: down arrow)
   - VIEW LEVEL 1:
     To view (not modify) the object at stack level 1 (shortcut: left shift-down-arrow)
   - EDIT STACK:
     To Delete, copy or move object from one stack level to another stack level (shortcut: uparrow or right shift-uparrow)
   - SPEC. CHAR.:
     To get special characters (shortcut: right shift-PRG)
   - NEW EQUATION:
     To enter a new equation in the EquationWriter environment (shortcut: left shift-ENTER)
   - NEW TEXT:
     Enter a new string with the Minwriter editor. See the Minwriter documentation for editing keys.
   - NEW MATRIX:
     Enter a numeric array in the MatrixWriter environment (shortcut: right shift-ENTER)
   - NEW LIST:
     Enter a new list with the Minwriter editor.

- NEW PROGRAM:
  Enter a new program with the Minwriter editor.
- PICTURE:
  Modify the graphic object (e.g. the graph of a function) in the PICTURE environment (shortcut: left shift-right-arrow)

2. VAR, I/O:
   - MEMORY:
     The memory handler of the HP48: see your variable and their contents (shortcut: right shift-VAR)
   - SEND/GET:
     The Input/Output menu of the HP48, to exchange files with a computer or another HP48 (shortcut: right shift-1)
   - PORTS:
     Management of your RAM/ROM card extensions (shortcut left shift-1)
3. PROGRAMS:
   - USER PROG:
     The programmatic menu of the HP48 (shortcut in user mode: α-right shift-PRG)
   - LIBRARIES:
     The libraries installed (shortcut right shift-2)
4. PHYSICS
   - EQ. LIBRARY:
     The Equation Library of the HP48 (shortcut: right shift, 3)
   - CONSTANTS:
     Physical constants and other utilities (shortcut left shift-3). Choose COLIB (B menu key) to get either the library of constants (CONLIB) or the CONST function (convert e.g. 'c' at level 1 to the speed of light)
   - DATE & TIME:
     To set or modify the date and the time, and compute with dates and time (shortcut: left shift-4)
5. SETUP
   - CALC MODES:
     Modes of the HP48: system flags, angle format, clock display, ... (shortcut right shift-CST)
   - RESET ERABLE:
     Like the VER command of Erable: restore standard state (no shortcut)

- REAL MODE:
  Put Erable in real mode, like the command 13 CF
- COMPLEX MODE:
  Put Erable in complex mode, like the command 13 SF
- INTEGER ARIT:
  Assumes now that you are doing arithmetic on integers, not polynomials, like the command 10 SF
- POLYN. ARIT:
  Assumes now that you are doing arithmetic on polynomials, not integers, like the command 10 CF
- NUMERIC MODE:
  Assumes now that your integer inputs are not exact but numeric, like the command XNUM
- SYMBOLIC MODE:
  Assumes now that your integer inputs are exact, not numeric approximations, like the command XQ
- TIME SETUP:
  The TIME menu of the HP48 to set, browse alarms and date/time (shortcut right shift-4)

6. STATISTICS:
   The STAT menu of the HP48: single-variable, frequencies, fit data, summary statistics (shortcut: right shift-5)

7. UNITS:
   The UNITS menu of the HP48: length, area, volume, time, speed, mass, force, energy, power, pressure, temperature, electricity, angle, light, radioactivity, viscosity (shortcut: right shift, 6)

8. NUMERIC:
   The built-in numerical functions of the HP48:

   - MATH FNCS
     vectors, matrices, list, hyperbolic, real functions, base probability, Fast Fourier Transform, complex, constants: the built-in MTH menu (shortcut in user mode: α-right shift-MTH)
   - SOLVE EQN
     The SOLVE menu of the HP48: numeric or semi-numeric solutions for equations, differential equations, polynomial, linear systems, financial problems (shortcut: left shift-7)

9. GRAPH:
   If you want to plot the function at stack level 1, choose PLOT LEVEL 1, otherwise choose PLOT MENU

10. ERABLE:
    To launch the Erable main menu if you don't remember that the MTH key does it directly. See description above.

5 Simplifications.

Two kinds of simplifications are provided: full rational simplification (EXPAND) and transcendental presimplification (TEXPAND, LIN, TLIN, TRIGSIN, TRIGCOS, ..., LNCOLLECT, TSIMP). In many situations, full rational simplification achieve the whole simplification, but sometimes you will need to detect relations between exponentials and logarithms; in this situation you should call TEXPAND, TLIN, ..., TSIMP, followed by EXPAND or FACTOR to finish the simplification.

5.1 Rational simplifications instructions.

EXPAND does a complete simplification of an expression viewed as a rational fraction, FACTOR tries to factor a symbolic. For convenience, arithmetic operations of Erable perform automatic rational simplifications: e.g. ADDT (shortcut: α-right shift-+) is equivalent to + EXPAND.

5.2 Presimplification instructions

To simplify non rational expressions, you will most of the time apply identities like $\ln(xy) = \ln(x) + \ln(y)$ or conversely and afterwards you will call EXPAND.

5.2.1 Linearization.

Exponential and trigonometric linearization are implemented via:

- LIN: $e^x e^y$ gives e^{x+y} and for integral powers $(e^x)^n = e^{nx}$
- TLIN: $\sin(x)\sin(y) = \frac{1}{2}(\cos(x-y) - \cos(x+y))$ and similar identities, $\sin(x)^n$ and $\cos(x)^n$ for n integer
- TCOLLECT: like TLIN but in real mode TCOLLECT combines sin and cos of the same argument, e.g. $\sin(x) + \cos(x)$ becomes $\sqrt{2}\cos(x - \pi/4)$.

5.2.2 Development.

The TEXPAND instruction applies the following identities:

- $e^{x+y} = e^x e^y$ and $e^{nx} = (e^x)^n$ for n integer,
- $\ln(xy) = \ln(x) + \ln(y)$.
  Warning: this identity is only valid modulo $2i\pi$
- $\sin(x+y) = \sin(x)\cos(y) + \sin(y)\cos(x)$ and $\cos(x+y) = \cos(x)\cos(y) - \sin(x)\sin(y)$

5.2.3 Trigonometry

The remaining trigonometric simplifications instructions are:

- `EXPLN` and `SINCOS` to apply Euler identities in both directions
- `TRIGCOS`, `TRIGSIN`: replace $\sin^2$ [resp. $\cos^2$] by $1-\cos^2$ [resp. $1-\sin^2$]
- `TRIG`: replace complex logarithms with arctan functions and then does whichever of `TRIGCOS` or `TRIGSIN` had been done last
- `TAN2SC`: replace tan by sin / cos, depending upon previous use of `TRIGCOS` or `TRIGSIN`
- `TAN2SC2`: replace $\tan(x)$ by $\sin(2x)/(1+\cos(2x))$ or $(1-\cos(2x))/\sin(2x)$.
- `HALFTAN`: replace $\sin(x)$, $\cos(x)$ and $\tan(x)$ in terms of $\tan(x/2)$
- `ATAN2S`: replace $\arctan(x)$ by $\arcsin(x/\sqrt{x^2+1})$
- `ASIN2T`: replace $\arcsin(x)$ by $\arctan(x/\sqrt{1-x^2})$
- `ASIN2C`: replace $\arcsin(x)$ by $\pi/2-\arccos(x)$
- `ACOS2S`: replace $\arccos(x)$ by $\pi/2-\arcsin(x)$

5.2.4 The `TSIMP` instruction

`TSIMP` is used to minimize the number of rational "variables". It may be used if `INTVX` fails because it returns an expression which is "weak-normalized". Note that `TSIMP` considers trigonometric functions as complex exponentials, and simplifies them this way and that the output of `TSIMP` is affected by the state of the flag 13 (complex flag): if flag 13 is cleared, then complex logarithms and exponentials are converted to arctan and sin/cos functions.

5.3 Recurse flag.

If flag 21 is set, "variables" of an expression are simplified recursively (global name are evaluated, integrals are evaluated by a call to `INTVX`, etc.).

6 Limits, Taylor and asymptotic series.

The program `SERIES` computes Taylor series, asymptotic development and limit at finite and infinite points. It should cover many weird limits, even some that are not handled by the TI89 (not surprising!) nor by maple (more surprising!) like:

$$\lim_{x\to 0} sin(1/x+x)-sin(1/x)$$

The `LIMIT` instruction may be used if you need only the limit. Note that `SERIES` handles more limits than `LIMIT`, but is a lot slower for trivial cases. `SERIES`

can not be used with non-exact arguments (like 0.1) and should not be used with parameters. `LIMIT` may give strange results with parameters. If you see a warning message, you will get a binary integer as answer, this means that `SERIES` or `LIMIT` was not successful, the binary integer is an error code (see section F).

Syntax of `SERIES`:
Put on the stack the following arguments in this order:

- the function $f(x)$
- the variable if the limit point is 0 or an equation $x = a$ if the limit point is a (and the variable is x). This entry is optional if the stack as only 1 argument.
- the order for series expansion (optional), by default 4 (minimum 2, maximum 20). If the order is a positive integer, the series expansion is made from the right, if the order is a negative integer from the left. For bidirectional series expansions, give the order as a binary integer (e.g. `#5d`).

Type `SERIES`, this computes the bidirectional limit at level 3. At level 2, you get a list of two elements: the series expansion and the rest order. They are expressed in terms of a small parameter h. At level 1, h is expressed in terms of the initial variable (hence calling `EXEC` would return the series expansion and the rest in terms of the initial variable).

Remark 2
- *Note that the series expansion is not always fully truncated, don't forget to look at the rest. If you need to truncate the series expansion, split the list on the stack (hit the* `EVAL` *key) and call the* `TRUNC` *function.*
- *Sometimes* `SERIES` *will not be successful and returns an error code. You can look at section F for more information about the failure.*
- *For mono-directional series expansion, either specify the order as a positive or negative integer, or for default order, put an equation $x = a + 0$ for a left-directional series expansion at a or $x = a - 0$ for a right-directional series expansion.*
- *For limits at infinity: you may use the ∞ symbol (from keyboard type α-right shift-*`DEL` *in user mode or α-right shift-I in normal mode).* To get $X = +\infty$, type $X = \infty$, then hit `EVAL`.

Examples:
'$1/x$' '$x = \infty$' `SERIES`
'$1/x$' '$x = +\infty$' `SERIES`
'$1/x$' '$x = -\infty$' `SERIES`
'$\sin(x)/x$' 'x' `SERIES`
'$\sin(x)/x$' '$x = +\infty$' `SERIES`
'$\sqrt{(2+x)}$' 'x' '5' `SERIES`
'$\sin(1/x + x) - \sin(x)$' 'x' `SERIES`
'$(\ln(-\ln(x + x^2)) - \ln(-\ln(x))) * \ln(x)/x$' '$x = 0 + 0$' `SERIES`

The syntax of LIMIT is similar: put the function and the the specification of the limit variable and the limit point, 'variable=limit_point', on the stack. Note that you can not force the order for series expansions and LIMIT handles only bidirectional limits (except at infinity). LIMIT returns only the limit at stack level 1.

Erable can handle relatively complex limits, like the example below (extracted from the Mupad on-line help):

$\frac{\exp\left(\frac{e^{-x}}{e^{-x}+e^{\frac{-2x^2}{x+1}}}x\right)-e^x}{x}$, $X=\infty$, LIMIT returns $-e^2$.

In addition Erable provides the:

- TRUNC instruction which truncates a series expansion at level 2 with respect to the rest at level 1.
  For example, to truncate $x+x^5$ with respect to x^4, try:
  'X+X^5' 'X^4' TRUNC
  that will return x since x^5 can be neglected with respect to x^4.

- DIVPC instruction which make a division in ascending power up to an integer order. The numerator is at level 3, the denominator at level 2 and the order at level 1.
  For example, if you want to compute the Taylor expansion of $(1+x)/(1-x)$ at $x=0$ at order 4, you type '1+X' (numerator) then '1-X' (denominator) then the order 4 and DIVPC returns:
  '1+2*X+2*X^2+2*X^3+2*X^4

These instructions may be used to understand series expansions without cumbersome calculations.

7 Derivation and integration.

7.1 Derivation

The Erable derivation instructions are DER and DERVX, they compute the derivative of a (list of) function(s) like the built-in instruction but do not evaluate numeric expressions (like $\sqrt{2}$ or $\frac{1}{2}$). DERVX is used for derivation with respect to the variable contained in VX and takes only one argument (the function to derive). DER is used with 2 arguments: the (list of) function(s) to derive at level 2 and the variable with respect to which you want to derive at level 1. If level 1 is a list, DER returns the gradient of level 2:

```
2: 'X^2+2*X*LN(Y)-1/Y'
1: { X Y } DER gives
{ '2*X+2*LN(Y)' '2*X*(1/Y)+1/SQ(Y)' }
```

DER returns djZ(...) for the derivative of the user-defined function Z(...) with respect to the j-th variable of $z(x,y,\ldots)$.

Examples:

Suppose that $x \to z(x)$ is the primitive of $\sqrt{x^3-1}$. Type `'Z(X)' X DER`, you get $\partial_1 z(x)$ on the stack. Enter $\sqrt{x^3-1}$ and hit `=` then enter `DEFINE`. Now, you can type `'Z(X^2)' X DER EVAL` and get $2x\sqrt{(x^2)^3-1}$. Try `'Y(X,X^2)' X DER`.

7.2 Integration

The main integration commands are `INTVX` and `EXPAND`. The `INTVX` program accepts functions as input and (tries to) return the primitive. `EXPAND` should be called for symbolic expressions which contains the $\int$ symbol. The last computed antiderivative is stored in the variable `PRIMIT`. The variable `ERABLEMSG` contains additional information if `INTVX` returns an unevaluated antiderivative (with a $\int$ sign).

Some examples for `INTVX`:

- $\frac{1}{x^2-4}$ gives $\frac{1}{4}\ln(x-2) - \frac{1}{4}\ln(x+2)$
- $x\ln(x)$ gives $\frac{1}{2}x^2\ln(x) - \frac{1}{4}x^2$
- $\sqrt{x^2-1}$ gives $\frac{1}{2}\ln(-x+\sqrt{x^2-1}) + \frac{x}{2}\sqrt{x^2-1}$
- $1/(\sin(x)+2)$ gives $\frac{-2}{3}\sqrt{3}\arctan(\frac{-2\tan(x/2)-1}{3}\sqrt{3})$

The `INTVX` program must sometimes be used in conjunction with the `TSIMP` function to get "weak normalization". If you get `No closed form` in `ERABLMSG`, try `TSIMP` and `INTVX` again, if you get again the message `No closed form`, this does not mean that `INTVX` failed, but that your input does not admit to an antiderivative which may be expressed in terms of elementary functions.

Remark 3

- `INTVX` *is only a partial implementation of the Risch algorithm: it works with pure transcendental extensions (i.e. square root are generically not allowed), and exponential polynomial parts must not contain logarithms or other exponentials. Examples:*

$$\ln\ln(x), \quad \frac{1}{e^{x^2+1}-1}, \quad x^3 e^{\left(\frac{x+1}{x+2}\right)}$$

are allowed (and returned since they do not have an antiderivative which may be expressed with elementary functions), but:

$$\sqrt{\ln(x)^2-1}, \quad e^{\ln(x)^2+1}$$

are not allowed as input.

In addition to this partial implementation, `INTVX` *can integrate fractions of the type* $F(x, \sqrt{ax^2+bx+c})$.

- *You can not use the name of the current variable as a parameter name in an integral, for example if* `VX` *is set to* `X`*, evaluation of:*

$$\int_0^T (X^2 - Y^2)\, dY$$

does not return a correct answer, because X in the integral is a parameter. You can however use X as integration variable, e.g.

$$\int_0^T (X^2 - Y^2)\, dX$$

works.

For integrals with limits, the right instruction is `EXPAND`. Example of `EXPAND` usage (in real and symbolic mode):

$$\int_1^2 \frac{1}{x^3+1} \quad \texttt{EXPAND} \text{ gives } \frac{\ln(3) - 2\ln(2)}{6} + \frac{\pi}{18}\sqrt{3}$$

If you have computed the antiderivative e.g. with `INTVX`, you can evaluate it between two bounds using `PREVAL`. Arguments of `PREVAL` are a function $f(x)$ at level 3, lower and upper bounds a and b at level 2 and 1. It returns $f(b) - f(a)$ (x is the variable contained in `VX`).

Remark 4 *Warning:* `EXPAND` *does not detect discontinuities of the antiderivative. It blindly computes the value at both ends of the integration interval (by a call to* `LIMIT`*, hence infinite bounds are allowed) and returns the difference. For example, $\int_0^{2\pi} \frac{1}{\sin(x)+2}$ returns 0. You should always check the answer numerically and if the answers are not the same, you have to study the antiderivative for discontinuities.*

7.3 Integration by parts

Integration by parts is implemented via the `IBP` command. You have to put an integral $\int_a^b f(x)\, dx$, where x is in `VX`, at level 2 and a function $u(x)$ at level 1. Let $f(x), dx = u(x)v(x)\, dx$, then `IBP` returns

`2:` $u(x) \cdot \int v(x)\, dx$

`1:` $-\left(\int v(x)\, dx\right) \cdot \frac{du(x)}{dx}\, dx$

You may call `IBP` several times, adding an appropriate $u(x)$ each time.

Examples:

For: $\int \arcsin(x)^2\, dx$ by entering

```
2: 'ASIN(X)^2'
1: 'T'
```

`IBP` gives

```
'X*ASIN(X)^2'
'-(X*(2*ASIN(X)*INV(√(1-SQ(X)))))'
```

adding `'√(1-X^2)` to the stack and doing `IBP` gives

```
3: 'X*ASIN(X)^2'
2: '√(1-x^2)*(2*ASIN(X)*INV(√(1-SQ(X))))'
1: '-(√(1-X^2)*(2*INV(√(1-SQ(X)))))'
```

doing `+ + EXPAND` gives

```
'-(2-2*√(1-X^2)*ASIN(X)-X*ASIN(X)^2)'
```

which is the antiderivative of $\arcsin(x)^2$.

Try: $\int \exp(x)\sin(2x)\,dx$ with $\exp(t)$ `IBP` twice. Note, after doing `EXPAND`, that the last line is -4 times the original integral. Replace the last line with $-4*I$. Do `+ +`. Put I at the bottom of the stack. Do `=`. Put I again on the bottom of the stack. Do `SOLVE`. The result is the antiderivative of $\exp(x)\sin(2x)$.

8 Ordinary differential equations.

8.1 Linear differential equations with constant coefficients

You can solve with `LDEC` either a linear differential equation of any order or a first order linear system of differential equations. The syntax is somewhat similar to the syntax of a division: put the second member on stack level 2 and the characteristic equation or the matrix of the system at level 1.
Examples:

- to solve $y'' + 4y = x^3$ type:
  `'X^3' 'X^2+4' LDEC`
  you will get:

$$(y_0 * \cos(2x) + \frac{8y_1+3}{16}\sin(2x) + \frac{2x^3-3x}{8}$$

  this is the solution such that $y(0) = y_0$ and $y'(0) = y_1$.

- suppose we want to solve the following system:

$$\left\{ \begin{array}{rcrcrcl} y_1'(x) & = & y_1(x) & - & y_2(x) & + & 1 \\ y_2'(x) & = & 2y_1(x) & + & 4y_2(x) & + & e^x \end{array} \right.$$

  Type:
  `{ 1 'EXP(X)' } { { 1 -1 } { 2 4 } } LDEC`
  you will get the solution such that $y_1(0) = v_1$ and $y_2(0) = v_2$:

$$\left(\begin{array}{c} (2v_1+v_2+2)e^{2x} - \frac{1}{2}e^x - \frac{2}{3} + -\frac{6v_1+6v_2+5}{6}e^{3x} \\ -(2v_1+v_2+2)e^{2x} + \frac{1}{3} + \frac{6v_1+6v_2+5}{3}e^{3x} \end{array} \right)$$

`Erable` solves these types of equations using the Laplace transform defined by:

$$Y(s) = \mathcal{L}(y)(s) = \int_0^\infty e^{-st}y(t)\,dt$$

Example: solve $y' + 2y = \cos(x)$. Apply $\mathcal{L}$, since:

$$\mathcal{L}(y')(s) = s\mathcal{L}(y)(s) - y(0)$$

we get:

$$(s+2)\mathcal{L}(y)(s) = \mathcal{L}(\cos(x))(s) + y(0)$$

hence:

$$y(x) = \mathcal{L}^{-1}\left(\frac{\frac{s}{s^2+1} + y(0)}{s+2}\right) = \mathcal{L}^{-1}\left(\frac{\frac{s}{s^2+1}}{s+2}\right) + y(0)\mathcal{L}^{-1}(\frac{1}{s+2})$$

since $\mathcal{L}(\cos(x))(s) = s/(s^2+1)$.

The program `LAP` takes the function f as argument and returns $\mathcal{L}(f)$ (Laplace transform is performed with respect to the variable contained in `VX`).

The program `ILAP` performs the inverse Laplace transform of rational fractions. Example: for $x/((x^2+1)(x+2)$, type `'X/(X^2+1)' 'X+2' / ILAP` to get the answer:

$$y(x) = \frac{1}{5}(2\cos(x) + \sin(x)) + \frac{-2}{5}e^{-2x}$$

Remark 5 *The name of the Laplace variable is the same name as the normal variable (and is contained in* `VX`*).*

8.2 First order equations.

The `DESOLVE` program recognizes and solves the following equation types:

- $y'(x) = f(y(x))$,
- $y'(x) = f(x, y(x))$ with f homogeneous,
- $y'(x) = g(x)y(x) + h(x)y(x)^\alpha \quad \alpha \in I\!R$ (Bernoulli type),
- $y'(x) = f(x)g(y)$ (separable, if f and g are rational fractions) and
- $y'(x) = f(x)y(x) + g(x)$ (linear)

The input is the function $f(x, y(x))$ or an equation like `'d1Y(X)+Y(X)=2'`.

Examples:

`Y(X)^2+Y(X) DESOLVE` which is incomplete
`X*Y(X)+1-X^2 DESOLVE` which is linear
`(Y(X)-X)/(Y(X)+X) DESOLVE` which is homogeneous
`Y(X)^2+X*Y(X) DESOLVE` which is Bernoulli

The output may be y as a function of x or x as function of y or x and y as a function of t (parametric solution) for an homogeneous ode. The equation type is stored in the `ODETYPE` variable.

9 Substitution, change of variables: `EXEC`.

The `EXEC` programs checks the object type at stack level 1 and performs the corresponding action:

- one algebraic substitution:
  If stack 1 is an equation (`'objA=objB'`), replace `objA` by `objB` in stack2. The syntax is `'old_name=expression' EXEC`.

  `oldname` may be a global name, an expression (in this case, the first global name in this expression will be isolated) or an user-defined function.

  Examples:

  - `'X^2+2*X+5' 'X=1' EXEC`: evaluate an expression at $x = 1$.
  - `'X=Y^2' EXEC`: change of variables, works in integrals too
  - `'2*Z(X)-X*d1Z(X)' 'Z(X)=X^2 EXEC`: in a differential equation, replace the function $z(x)$ by x^2.
  - `'Z(X)+d1Z(X)' 'Z(X)=EXP(-X)*Y(X)' EXEC`:
    change of function in a differential equation.
  - `'X^2+X*COS(X)' 'X^2=1-Y' EXEC`: replace x^2 by $1 - y$ and replace x by $\sqrt{1-y}$.

- multiple substitutions:
  If stack 1 and 2 are lists, replace each object of list2 in stack level 3 by the corresponding object of list1.

  The syntax is

  `{ old_name_1 ... old_name_n } { expr_1 ... expr_n } EXEC`

  Note that here `EXEC` does only substitutions.

  Examples:

  - `'SIN(X)^2+SIN(X)*COS(X)' {'SIN(X)^2'} {'1-COS(X)^2'}`
    `EXEC` replaces $\sin(x)^2$ by $1 - \cos(x)^2$ without replacing $\sin(x)$ by $\sqrt{1-\sin(x)^2}$.
  - ```
    'COS(X)+i*SIN(X)'
    { SIN COS }
    { << i * EXP DUP INV - i 2 * / >>
      << i * EXP DUP INV + 2 / >>
    }
    EXEC
    ```
    replace sin and cos by complex exponentials. If you call `EXPAND`, afterwards you get e^{ix}.

- variable isolation:
  If stack 1 is a symbolic but not an equality, `EXEC` tries to isolate stack level 1 in stack level 2. Example:

  ```
  'X^2-5'
  X
  EXEC
  ```
  ```

returns 'X=$\sqrt{5}$'.

- *doall* function:
  If stack 1 is a program, EXEC executes program at stack level 1 recursively on the components of a list object at stack level 2. Example:

  ```
  { 1 2 3 } << NEG >> EXEC
  ```

  is the same as

  ```
  { 1 2 3 } CHS
  ```

10 Arithmetic.

10.1 Complex arithmetic

- re: real part
- im: imaginary part
- conj: conjugate
- abs: absolute value (modulus)
- arg: argument. Warning: for expressions containing variables, the returned argument is only valid modulo π.

Remark 6 *If flag 13 is cleared (real mode), all global names and all non rational functions are considered as real with respect to the instructions* RE, IM, CONJ. *This could lead to false simplifications if a global name stays for a complex, or if a non-rational inverse function is called with an usually forbidden real argument, like* LN(-1) *or* ASIN(2)

Solution: either replace your global name, say 'Z', *by* 'X+iY' *or set flag 13 (shortcut α-right shift-*CST *or* 13 SF*)*

10.2 Integer and polynomial arithmetic

You may force integer arithmetic by setting flag 10 (shortcut α-right shift-CST or 10 SF). Otherwise, polynomial arithmetic is assumed. This is important for instructions like GCD3 or ABCUV.

- DIV2: Euclidean division. Stack 2 is the quotient, stack 1 the remainder.
- GCD1: returns the greatest common divisor d of two objects a and b (integers, Gauß integers, polynomials). Examples:

  'X^2+2*X+1' 'X^2+3*X+2' GCD1 returns 'X+1'
  25 15 GCD1 returns 5

  If flag 12 is clear it returns 1.

- `LCM1`: lowest common multiple (`GCD1`$(a,b)\times$`LCM1`$(a,b) = a\times b$). Examples:

  `'X^2+2*X+1' 'X^2+3*X+2' LCM1` returns `'(X^2+2*X+1)*(X+2)'`

  `25 15 LCM1` returns `75`

- `GCD3`: extended gcd algorithm, given x and y returns d , u and v such that:

  $$ux + vy = d$$

  (d is a multiple of the *gcd* of x and y by an invertible, i.e. an integer in the univariate case)

- `ABCUV`: (Bezout identity) solve the equation $c = ax + by$ Examples:

  `'X^2+2*X+1' 'X^2+3*X+2' 'X+1' ABCUV` gives `-1 1 1`
  `'X^2+2*X+1' 'X^2+3*X+2' 1 ABCUV` gives `0`

  This means for the first case that:

  $$(X + 1) = (X^2 + 2X + 1) * (-1) + (X^2 + 3X + 2) * 1$$

  as in the second case there is no solution because the gcd of $x^2 + x + 1$ and $x^2 + 3x + 2$ does not divide 1.

- `LGCD`: returns the gcd of a list of objects.

- `SIMP2`: simplifies two objects by dividing them by their gcd. Sets flags 12, 14 and 15. Example:

  ```
  2: 9
  1: 6
  SIMP2 gives
  2: 3
  1: 2
  ```

- `DIVIS`: gives a list of divisors of an object.

  Example:

  `21 DIVIS` gives `{ 1 7 3 21 }`

- `fact` and `comb`: like the built-in `FACT` and `COMB` instructions but for long integers.

- `EULER`: Euler indicatrix
  Given an integer n, returns an integer e: the number of integers lower than and prime with n.

  Example:

  for $n = 25$, $e = 20$ because 1, 2, 3, 4, 6, 7, 8, 9, 11, 12, 13, 14, 16, 17, 18, 19, 21, 22, 23, 24 are prime with 25.

- `PA2B2` (`kernel` library):
  Given a prime p which is 1 modulo 4, returns a complex $z = a + ib$ such that $|z|^2 = a^2 + b^2 = p$. Used for factorization of Gauß integers.

- `XFRC`:
  Same as $\rightarrow$`Q` but handles quadratic irrationals (recognizes a quadratic irrational if its expansion in continued fractions is ultimately periodic of period less or equal to 3).

  Examples:

  1.20710678118 gives $\frac{1}{2} + \frac{1}{2}\sqrt{2}$
  1.5 gives $\frac{3}{2}$

- `ORND`:
  round object at stack level 2, stack level 1 is the expected denominator of all rationals of the object.

  For example, $.49999999999 + .2000000001 * x$ 10 `ORND` returns $.5 + .2 * x$.

10.3 Infinities.

The arithmetic operations of `Erable` accept infinity arguments. The $+\infty$ symbol may be obtained from the keyboard by α-right shift-I in normal mode or α-right shift-`DEL` in user mode followed by `ENTER`. The $-\infty$ symbol is obtained by hitting the quote key, the - key and α-right shift-I or α-right shift-`DEL` followed by `ENTER`. ∞ and $-+\infty$ are understood as unsigned infinity (with the following exception: ∞ is understood as $+\infty$ in a bound of an integral or in a `LIMIT` instruction). The ? symbol means that the result of an operation is undefined. All operations involving ? will return ? after simplifications.

Remark 7 *The current release returns* `?` *for some expressions which are not undetermined, like* $(+\infty)^{-\infty}$.

Be aware of the fact that arithmetic operations on infinities may return false answers because `Erable` *can not check for nullity if non rational expressions are encountered. For example* $(\sin(2x) - 2\sin(x)\cos(x))/0$ *will return* ∞ *instead of 0. This remark applies to the* `LIMIT` *instruction too (call* `SERIES` *if the answer of* `LIMIT` *seems false).*

10.4 Modular arithmetic

You must first store an integer n in the `MODULO` variable (by default n is set to 3). All computations are made modulo this integer. The `kernel` library provide the following commands `MODADD`, `MODSUBT`, `MODMULT`, `MODDIV`, `MODPOW` and `MODINV` for the usual operations. `MODADD`, `MODSUBT`, `MODMULT`, `MODDIV` now accepts polynomials and Gauß integers as well as integers as input. `MODPOW` accepts a polynomial as first argument, the second argument must be a positive integer.

Remark for `alg48` users: these functions are analog to `AADD`, `ASUB`, ... applied to mod-polynomials (like `'MOD(X^2+3*X+7,11)'`) but without `MOD` notation. I reserve the `MOD` notation for a future use with a polynomial as second argument so that it will be possible to compute e.g. in $\mathbb{Z}/3\mathbb{Z}[X]/(X^2+1)$.

11 Factorization. Solving equations.

11.1 Summary of the instructions.

- `FACTOR`: factor a symbolic fraction (returns a symbolic). Factorization may be incomplete, but is square-free and all first order factors are detected.
- `LNCOLLECT`: collect logarithms, e.g. $\ln(2)+\ln(3)$ `LNCOLLECT` returns $\ln(2\times 3)$.
- `SOLVE`: tries to isolate a variable name at stack level 1 in the symbolic expression located at level 2. Returns the variable name at level 2 and a list of solutions at level 1. Examples:
  - `'X^4-1' X SOLVE` returns:

    if real (13 CF), `{ 'X=1' 'X=-1' }`
    if complex (13 SF), `{ 'X=1' 'X=-1' 'X=i' 'X=-i' }`
  - `'2*SIN(X)^2-3*SIN(X)+1' X SOLVE` returns

    ```
    { 'X=(π-4*π*n2)/2'
      'X=(π+4*π*n2)/2'
      'X=(5*π-12*π*n1)/6'
      'X=(π+12*π*n1)/6' }
    ```

    where `n1` and `n2` represent integers.

  `SOLVE` will be successful if the symbolic expression is a polynomial of a function of the variable or a product of polynomials. Otherwise it will fail. Hence, it may be useful to rewrite the symbolic expression using `LNCOLLECT` or/and `FACTOR` before calling `SOLVE`. Example:

  `'LN(X-1)+LN(X+2)=2' X SOLVE` doesn't work
  but `'LN(X-1)+LN(X+2)=2' LNCOLLECT X SOLVE` works
- `SIGNE`: returns the sign of a mono-variate rational fraction as a list of signs $+$ or $-$ separated by roots or poles of the rational fraction, starting from $-\infty$ at the left to $+\infty$ at the right of the list.

  Example:

  ```
  '(X^2-1)/(X^3-7*X^2+16*X-2)' SIGNE
  ```

  returns:

  ```
  { '-∞' + -1 - 1 + '+∞' }
  ```

hence $\frac{x^2-1}{x^3-6x^2+11x-6}$ is negative for $x \in (-\infty,-1) \cap (1,2) \cap (2,3)$, positive for $x \in (-1,1) \cap (3,+\infty)$, and zero or infinite for $x \in \{-1,1,2,3\}$.

- `FROOTS`: given an object as input, outputs the list of variables (stack level 3), the list polynomial (2), and the list of root/multiplicity (1) (each root is followed by its multiplicity). Examples:

```
* 'X^3-6*X^2+11*X-6' FROOTS gives
  3: { X }
  2:'X^3-6*X^2+11*X-6'
  1: { 2 1 3 1 1 1 }
* '1/X^2' FROOTS gives
  3: { X }
  2: '1/X^2'
  1: { 0 -2 }
* 'X^2+2*X*Y+Y^2' FROOTS gives
  3: { X Y }
  2: 'X^2+2*X*Y+Y^2'
  1: { '-Y' 2 }
```

  For a symbolic, `FROOTS` factors the symbolic expression with respect to the variable contained in `VX`, or if the symbolic is independent of `VX` with respect to the first variable of `LVAR` applied to the symbolic.

  (*For the SX version only*): If stack 1 is a real integer, `FROOTS` computes the roots of a list polynomial with numeric coefficient using Bairstow method (real coefficients) or Laguerre method (complex coefficients). During iterations, you can modify some parameters:

  - `E*10`: ($\varepsilon \times 10$) multiplies the test value by 10, use this when there are multiple roots.
  - `E/10`: divides the test value/10, for accurate precision (use this after you have found all multiple roots)
  - `RAND`: reset current iteration (restart with random initial value)
  - `STOP` abort iteration (for the next one or two roots)

  Displayed are the last found roots and the current test value (to compare with the ε value). Before starting the program, you must specify the ε value and the number of test-successfully iterations using the following stack input:

  - 3: list polynomial,
  - 2: test value (positive real),
  - 1: iteration number (real integer)

  Example:

```
{ 1 -21 183 -847 2196 -3024 1728 }
1E-4
3
FROOTS gives
approximatively { 3 3 3 4 4 4 }
```

The result is bad since 3 and 4 are multiple roots. (End of specificity of the SX version)

- `FACTORS`: same stack as `FROOTS` but returns a list of "prime" factors instead of roots. Example:

```
* 'X^3-6*X^2+11*X-6' FACTORS gives
  3: { X }
  2: 'X^3-6*X^2+11*X-6'
  1: { '(X-2)' 1 '(X-3)' 1 '(X-1)' 1 }
* 21 FACTORS gives
  3: { }
  2: 21
  1: { 3 1 7 1 }
```

- `FCOEF`: input is a list of roots/multiplicity, output is a fraction or polynomial with leading coefficient 1 having this roots (and poles). Example:

```
{ 1 1 2 1 3 1 } gives '(X-3)*(X-2)*(X-1)'
{ A -1 2 1 } gives '(X-2)/(X-A)'
```

11.2 A word about factorization.

You should skip this section for a first reading. Factorization of polynomial is very important in several mathematical functions, like symbolic integration or matrix diagonalization. It is important to understand the mechanism used by `Erable` to perform this tasks.

Let's begin by recalling some mathematical facts:

- Theorem (d'Alembert):
  A polynomial of degree n has exactly n complex roots (counted with multiplicity).

- Formula exists to get the solution of polynomials up to order 4 but Galois proved the following theorem last century:

  > There is no formula for solving a generic polynomial of degree ≥ 5 (by algebraic operations and extraction of n-th roots)

This means that you can not compute the roots of a multivariate polynomial of order ≥ 5 (for such polynomials, systems like Maple, Reduce, Axiom, Mathematica or Mupad use algebraic extension), and that you can only compute the roots numerically of an arbitrary univariate polynomial of order ≥ 5. Note that the generic solution of a polynomial of order 3 is still complicated and of order

4 very complicated. I think that it is not possible to handle the generic solution of polynomials of order 3 or 4 on the HP48 in a reasonable amount of time. Hence, only polynomial of order 2 are generically solved by `Erable`.

However, in some situations, you can root exactly polynomials of order ≤ 3, by searching multiple roots and by finding obvious roots (or obvious factors). The rooting algorithm of `Erable` searches first for multiple roots by computing the *gcd* of the polynomial and its first derivative (this is the `SQFFext` algorithm in the source of `Erable`). Of course flag 12 must be set for this step to be done. If an univariate polynomial has only integer (or rational) coefficients, you can find all rational solutions of this polynomial by testing a finite set of rationals (of the form numerator/denominator where numerator is a divisor of the constant coefficient and denominator a divisor of the leading coefficient). This is implemented in `Erable` by the null named XLIB `EVIDENText` which is called if flag 14 is set. Hence, `Erable` detects all 1st order factors of a symbolic.

If the method above fails, `Erable` calls the numeric solver for univariate polynomials (which is the HP48GX `PROOT` function in `ERABLEG.LIB` or the Bairstow or Laguerre algorithm in `erable.lib`) and tries to find second order polynomial with integer coefficients by coupling 2 approximate solutions (this was an idea of Mika Heiskanen implemented in `POLYLIB`). Hence `Erable` should find all rational and quadratic irrationals roots of an univariate polynomial (unless the polynomial is badly conditioned).

For multivariate polynomials, the two first steps are achieved (`EVIDENText` and `SQFFext`). `Erable` should find all rational multivariate roots of a polynomial (1st order factors). Unfortunately, `Erable` does not implement the exhaustive search of all 2nd order (or greater) multivariate rational factors. This can be performed using the `FCTR` function of the `ALG48` library.

Abstract of the SysRPL `XLIB`s (include `erextdec.h` and `erhash.h` in your source code to use them) to factor:

- `EVIDENText`: finds rational roots
- `SQFFext`: finds square-free factorization of a polynomial
- `SOLVext`: roots an univariate polynomial numerically and tries to rebuild quadratic irrationals roots

Abstract of the user commands to factor:

- `FACTOR`:
  for symbolic input calls `SQFFext`, then `EVIDENText`, does **not** call `SOLVext`, and for list input calls only `SQFFext`
- `FROOTS`: calls `SQFFext` then `EVIDENText` then `SOLVext`,
- `FACTORS`: calls `SQFFext` then `EVIDENText`, does not call `SOLVext`

Flags 12 and 14 may be cleared to skip respectively `SQFFext` and `EVIDENText`.

12 Linear algebra.

Lists of lists are used to represent symbolic matrices, in other words a symbolic matrix is entered like a numeric matrix, replacing [by { and] by }. Symbolic vectors are allowed as well (represented as lists).

12.1 Building a matrix

To build a matrix, you may type it as usual with { and } instead of [and] or you may use one of the following instructions:

- `IDENTITY`: build a symbolic identity matrix I_n (if n is at level 1)
- `LCXM`: build a matrix $A = (a_{ij})_{1 \leq i \leq l, 1 \leq j \leq c}$. The command takes 3 arguments: l, c and a program building a_{ij} from i and j. Example:

  ```
  2 4 << SQ + >> LCXM
  ```

  returns a 2×4 matrix with $a_{ij} = i + j^2$
- `VANDERMONDE` returns a Vandermonde matrix given a list of objects
- `HILBERT` returns a Hilbert matrix given an integer.

12.2 Operations

`Erable` provides the arithmetic usual operations on matrices and vectors (`add`, `SUBT`, `MULT`, `CHS`) and:

- `STUDMULT`: (`MATR` directory) student multiplication of matrices (term by term)
- `TR`: trace of a matrix
- `TRAN`: transpose of a matrix (true transpose, no conjugation)
- `DOTP`: scalar product of two vectors
- `CROSSP`: cross product of two 3-d vectors.

12.3 Gauß-Jordan row reduction.

Summary of the instructions:

- `rref`: row reduction to echelon form. At level 2, the list of pivoting coefficients is given, this is useful to treat particular cases.
- `REF`: rank of matrix using half row reduction
- `det` and `RDET`: determinant (using respectively the $O(n * n!)$ algorithm or row reduction)

- `inv`: inverse of a matrix using row reduction
- `LU2`: given a square matrix, returns L^{-1} and U such that $A = LU$ (i.e. $A =$`stk2` $^{-1}\times$`stk1`) where L and U are lower and upper triangular (maybe with respect to to a permutation matrix, this means that computing the inverse of L or U is trivial). For comparison, the built-in `LU` returns three matrices L, U and P such that $A = PLU$.
- `SYST` and `SOLGEN`: solution of a linear system.

Note that all instructions using row reduction show intermediate steps if flag 1 is set (`1 SF`). If flag 1 is cleared (`1 CF`), you get directly the results.

12.3.1 Solving a linear system.

Suppose you want to find (x, y) such that:

$$\begin{cases} mx & + & y & = & -2 \\ mx & + & (m-1)y & = & 2 \end{cases}$$

where m is a parameter. Type a list containing the linear equations and as last element put the list of unknowns. Here:

```
{ 'M*X+Y=-2' 'M*X+(M-1)*Y=2' { X Y } }
```

Then call `SYST` or `SOLGEN`. For `SYST`, you get the solution at level 1, the list of particular cases at level 2 and the original system at level 3. For `SOLGEN` you get the same results but at level 2, 3 and 4 and the parameterized solution at level 1.

On the above example, we get at level 1:

```
:X:'-2/(M-2)' :Y:'4/(M-2)'}
```

At level 2, you get the list of pivots. The result returned by `SYST` and `SOLGEN` is incorrect if one of the pivot is 0. Here level 2 is:

```
{ 'M^2-2*M' '-M+2' -1 'M+-2' }
```

Using `'M' SOLVE`, we see that we have to solve for the particular cases $m = 0$ and $m = 2$. The commands `SYST` and `SOLGEN` create a variable named `SYSTEM` to help solving particular cases. To solve for $m = 2$, recall `SYSTEM` on the stack, type `'M=2' EXEC`, and call `SYST`.

For systems, the `SOLGEN` program provides another way of writing the solution as an affine space of solutions. Recall the matrix on the stack (simply hit `SYSTEM`), type:

```
'M=0' EXEC SOLGEN
```

you get at level 2:

```
If { }, { X Y }=:{ X -2 }
```

(level 1 is the same as the result of `SYST`). This means that $(x, -2)$ is solution for every x. The `If` statement shows necessary conditions for the system to have solutions (here no condition, but if we try $m = 2$ instead of $m = 0$, the system has no solution: the `If` statement is `If { '0=-1'}` never fulfilled).

Another way to solve the system is the enter the matrix of the system

```
{{M 1 -2}{M 'M-1' 2}}
```

and call `rref` to reduce it. You get at level 1:

```
{{'M^2-2*M' 0 '-2*M'}{0 'M-2' 4 }}
```

This means that:

$$(m^2 - 2m)x = -2m, \qquad (m-2)y = 4.$$

The reduction is correct iff all the coefficients in the list at level 2 are non 0. You should have at level 2:

```
{1 'M-2'}
```

The second coefficient vanishes if $m = 2$. You have to solve for this particular case again. To do this, you can use the variable named `MATRIX` (which is created if the argument contains at least one parameter). Recall this matrix and type:

```
'M=2' EXEC
```

This replaces all occurrences of `M` by 2 in the original matrix. Now type `rref` again, you get:

```
{{2 1 -2}{0 0 4}}
```

The last line means that:

$$0x + 0y = 4$$

which is clearly impossible; the system has no solution.

12.3.2 Inversion

The `inv` implements the Gauß method to invert matrices.

```
{ { '1/2' -1 } { 1    '2/3' } }
```

inv returns

```
{ { '1/2' '3/4'  }{ '-3/4' '3/8' } }
```

12.3.3 Determinant

The `RDET` instruction implements Gauß row reduction to compute determinant.

```
{ { 1 T T T }
  { 1 K T T }
  { 1 T K T }
  { 1 T T K } }
RDET
```

gives

```
'(K-T)*(K-T)*(K-T)'
```

12.4 Kernel and image of a linear application.

To get the kernel of a linear application f with matrix A, enter the matrix A and type `KERN`. This will return the parameterized equations of the kernel like `SOLGEN`.

To get a basis of the image of f, enter the matrix A, type:

`<< TRAN rref >>`

the basis is made of the non-zeros lines of this matrix.

12.4.1 Other examples.

- LU decomposition example:

$$A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

`LU2` returns:

$$L^{-1} = \begin{pmatrix} 1 & 0 \\ -3 & 1 \end{pmatrix} \qquad U = \begin{pmatrix} 1 & 2 \\ 0 & -2 \end{pmatrix}$$

We have $A = LU$.

- Rank of a matrix:

$$\begin{pmatrix} 1 & 2 & 4 & 6 \\ -1 & 3 & 5 & 7 \\ 2 & 1 & 0 & 1 \\ 2 & 6 & 9 & 14 \end{pmatrix},$$

hit `REF`, and look at the matrix:

$$\begin{pmatrix} 1 & 2 & 4 & 6 \\ 0 & 5 & 9 & 13 \\ 0 & 0 & -13 & -16 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

the rank is 3, the number of non zero lines (By the way, you also get a half reduced matrix)

- Linear relations between vectors
Suppose we want to know the rank and linear relations existing between $v_1(1,2,0)$, $v_2(-2,-1,1)$, $v_3(0,3,1) \in I\!R^3$:

```
{ { 1 2 0 V1 }
  { -2 -1 1 V2 }
  { 0 3 1 V3 } }
```

then `REF`, we get:

```
{ { 1 2 0 V1 }
  { 0 3 1 '2*V1+V2' }
  { 0 0 0 '-(2*V1)-V2+V3' } }
```

The family is of rank 2 (the 3rd line is 0) and $-2v_1 - v_2 + v_3 = 0$.

12.4.2 Stack input/output for reduction instructions.

Program	Input	Output
LU2	1: matrix $A = LU$	3: pivots, 2: L^-1, 1: U
REF	1: matrix	2: pivots 1: half-reduced matrix
rref	1: matrix	2: pivots, 1: `rref`-ed matrix
RDET	1: matrix	1:determinant
inv	1: matrix	1: inverse
SYST	1: { equations { unknowns } }	3: original system 2: list of pivots 1: list of tagged algebraics
SOLGEN	1: { equations { unknowns } }	4: original system 3: list of pivots 2: result 1: list of tagged algebraics,

12.5 Diagonalization

The diagonalization instructions are:

- `MAD`: given a square matrix, returns the determinant, the formal inverse, a list polynomial and the characteristic polynomial. The list polynomial P_A is a matrix coefficient polynomial defined by the relation:

$$(xI_n - A)P_A(x) = M(x)I_n = M(x)I_n - M(A) \tag{1}$$

  where M denotes the characteristic polynomial of A.

- `PCAR`: characteristic polynomial using `det`

- `JORDAN`: compute eigenvalues and eigenvectors (cf. infra)

Given a square matrix A, `JORDAN` returns 6 levels:

- 6: $\det(A)^{-1}$
- 5: A^{-1}
- 4: list of eigenvalues (with multiplicities)
- 3: characteristic polynomial
- 2: minimal polynomial M (it divides the characteristic polynomial)
- 1: list of characteristic spaces tagged by the corresponding eigenvalue (either a vector or a list of Jordan chains, each of them ending by a "Eigen:"-tagged eigenvector)

Examples:

1.

$$A = \begin{pmatrix} 1 & -1 & 0 \\ 0 & 1 & -1 \\ -1 & 0 & 1 \end{pmatrix}$$

returns:

```
6: 0
5: { { inf inf inf }
     { inf inf inf }
     { inf inf inf } }
4: {0 1 '3/2+i/2*V3' 1 '3/2-i/2*V3' 1 }
3: 'X^3-3*X^2+3*X'
2: 'X^3-3*X^2+3*X'
1: {    :0: {1 1 1}
        :'3/2+i/2*V3': {1 '-1/2-i/2*V3' '-1/2+i/2*V3'}
        :'3/2-i/2*V3': {1 '-1/2+i/2*V3' '-1/2-i/2*V3'} }
```

This means that A has 3 eigenvalues $\frac{3\pm\sqrt{3}i}{2}$, and a basis of eigenvectors is:

$$\{(1,1,1),(1,\frac{-1\mp i\sqrt{3}}{2},\frac{-1\pm i\sqrt{3}}{2})\}$$

corresponding to $0, (3+\sqrt{3}i)/2, (3-\sqrt{3}i)/2$. The characteristic and minimal polynomial are identical (this is generically the case) X^3-3X^2+3X. The matrix is not invertible and has a 0 determinant.

2. For the identity matrix I_2 (`2 IDENTITY`), we get:

```
6: 1
5: { { 1 0 } { 0 1 } }
4: {1 2}
3: 'X^2-2*X+1'
2: 'X-1'
1: { :1, Eigen: { 0 1 } :1, Eigen: { 1 0 } }
```

The minimal polynomial is $X-1$, which differs from the characteristic polynomial $(X-1)^2 = X^2-2X+1$.

3.

$$\begin{pmatrix} 1 & 2 & 1 \\ 2 & 0 & 0 \\ 1 & 0 & 3 \end{pmatrix}$$

4. An example with 1 parameter:

```
{ { 1 A }
  { A 1 } }
```

5. When the dimension is greater than 2, the factorization routines may fail. For this reason, you may have to call `MAD`, factor the characteristic polynomial (e.g. by trying the `FCTR` instruction of `ALG48`) before calling `JORDAN`. If you have `ALG48` installed, try this:

```
{ { 1 1 A }
  { 1 A 1 }
  { A 1 1 } }
MAD FCTR JORDAN
```

   Note that this example is solved by typing `JORDAN` directly but it may fail in other situations.

6. Jordan cycles example:

$$A = \begin{pmatrix} 3 & -1 & 1 \\ 2 & 0 & 1 \\ 1 & -1 & 2 \end{pmatrix},$$

   returns:

```
6: -4
5: : { { '1/4' '1/4' '-1/4' }
       { '-3/4' '5/4' '-1/4' }
       { '-1/2' '1/2' '1/2'  } }
4: { 2 2 1 1}
3: 'X^3-5*X^2+8*X-4'
2: 'X^3-5*X^2+8*X-4'
1: { :2, Char: {2 2 1} :2, Eigen:{1 1 0} :1: {0 1 1} }
```

   This means that 2 has multiplicity 2, but the corresponding eigenspace is only 1-dimensional (spanned by $(1,1,0)$ the last vector of the Jordan chain). The first vector $(2,2,1)$ is only a characteristic vector, its image by $(A-2I)$ is the eigenvector $(1,1,0)$.

Remark 8 *You can not use the current variable name as a parameter of a symbolic matrix that you want to diagonalize. This would lead to incorrect results. For example, if* `VX` *is set to* `X`*, you can not diagonalize the following matrix:*

```
{ { 1 1 X }
  { 1 X 1 }
  { X 1 1 } }
```

Workaround: make a change of variable, e.g. `'X=A' EXEC`.

12.6 The `MMULT` instruction.

This multiplication takes 3 arguments: 2 objects at levels 3 and 2, and a real at level 1: the product type:

- 0: matrix, matrix
- 1: matrix, vector
- 2: matrix, scalar,
- 3: vector, scalar
- 6: scalar, matrix
- 7: scalar, vector

It is not intended for interactive mode use (if you plan to write your own program using `Erable`, you may need to use `MMULT` to switch to internal mode data representation for speed).

13 Multivariate analysis.

`Erable` implements the following functions:

- `DER` with a list of variables at level 1 returns the gradient of the expression at level 2 with respect to these variables.

  Example:

  `'X+2*Y' { X Y } DER` returns `{ 1 2 }` $= (\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y})$.

- `DIV` returns the divergence of a list-vector at level 2 with respect to a list of variables at level 1.

  Example:

  ```
  { 'X+2*Y' 'X^2+3*Y^3' } { X Y } DIV
  ```

  returns

  `'1+9*Y^2'` $= \frac{\partial f}{\partial x} + \frac{\partial f}{\partial y}$

- `CURL` returns the rotational (same arguments as `DIV`). Note that `CURL` is only defined in 3-dimensional space

- `LAPL` returns the Laplacian of a symbolic expression at level 2 with respect to a list of variables at level 1 (same arguments as `DER`, `LAPL` is simply a shortcut for `DER DIV`)

- `HESS` returns at level 1 the Hessian of a symbolic expression with respect to a list of variables (same arguments as `DER`). Level 2 is the gradient. This is useful to find local extrema of a function: you first find the solutions of gradient=0 (you may use the Gröbner basis program of `ALG48` to simplify this system and use the `SOLVE` instruction to find all solutions), then you compute with `EXEC` the Hessian at these critical points, and you find the signature of the critical point using `GAUSS` (in the `other` directory: see section 14).

Example:

$$f(X,Y) = X^4 + XY + Y^3$$

`{ X Y } HESS` returns at level 2:

$$(4X^3 + Y, X + 3Y^2) = (\frac{\partial f}{\partial X}, \frac{\partial f}{\partial Y})$$

and at level 1:

$$\begin{pmatrix} 12X^2 & 1 \\ 1 & 6Y \end{pmatrix}$$

To find critical points, you have to solve level 2=(0,0):

$$(4X^3 + Y, X + 3Y^2) = (0,0)$$

hence $X = -3Y^2$. Swap level 2 and 1, type `'X=-3*Y^2' EXEC`, then `1 GET` to have the first coordinate:

$$4(-3Y^2)^3 + Y$$

then `Y SOLVE`. This equation has two real solutions: 0 and approximately 0.392026340842 giving two critical points. For $(0,0)$, the Hessian is:

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

hence $(0,0)$ is not an extremum (signature $(1,1)$). For the second point, $(Y = 0.392..., X = -3Y^2)$, the Hessian is:

$$\begin{pmatrix} 1.84421582297 & 1 \\ 1 & -2.76632373445 \end{pmatrix}$$

and hence it is not an extremum.

14 Quadratic forms.

The main program is `GAUSS` (located in the `other` directory) to perform reduction of a quadratic form q. There are two ways to use `GAUSS`:

- symbolic input:
  Input: a quadratic form q (symbolic) at level 1 or the quadratic form q at level 2 and the list of variables at level 1.

  Output:

  - 5: D the list of diagonal coefficients (only the number of positive and negative coefficients is characteristic of q)
  - 4: P (the columns vectors of P^{-1} form a q-orthogonal basis of A at level 3)

– 3: A (A is the matrix of q in the dual base of the coordinates-forms at level 2, we have $A = P^t DP$ where P^t denotes the transpose of P)
– 2: list of variables
– 1: symbolic as a sum of independent squares

Examples:

Example 1:

```
'X^2+4*X*Y-2*X*Z+4*Y^2+6*Y*Z+7*Z^2' GAUSS
5: { 1 '-25/6' '1/6' }
4: { { 1 2 -1 } { 0 1 0 } { 0 5 6 } }
3: { { 1 2 -1 } { 2 4 3 } { -1 3 7 } }
2: { X Y Z }
1: '1/6*(6*Z+5*Y)^2+ -25/6*Y^2+(-Z+2*Y+X)^2'
```

Example 2: same example but with the variables in the reverse order

```
'X^2+4*X*Y-2*X*Z+4*Y^2+6*Y*Z+7*Z^2' { Z Y X } GAUSS
5: { '1/7' '7/19' '-25/19' }
4: { { 7 3 -1 } { 0 '19/7' '17/7' } { 0 0 1 } }
3: { { 7 3 -1 } { 3 4 2 } { -1 2 1 } }
2: { Z Y X }
1: '-25/19*X^2+7/19*(17/7*X+19/7*Y)^2+1/7*(-X+3*Y+7*Z)^2
```

Example 3: if you want to orthogonalize with with one or more parameters, you need to enter the list of variables of the quadratic form to identify them:

```
'X^2+2*A*X*Y' { Y X } GAUSS
5: { '-A^2' 1 }
4: { { 1 0 } { A 1 } }
3: { { 0 A } { A 1 } }
2: { Y X }
1: '(X+A*Y)^2-A^2*Y^2'
```

- matrix input:
  Input (stack level 1): the formal matrix A of the quadratic form q

  Output: at stack level 2 D the diagonal coefficients list and at stack level 1 the transition matrix P. We have $A = P^t DP$ where P^t denotes the transpose of P. Note that to obtain a q-orthogonal basis, one can take the columns of the inverse P^{-1} of P.

  Example:

  The matrix of q defined by $q(x, y) = 4x^2 + 2xy - 3y^2$ is:

$$A = \begin{pmatrix} 4 & 1 \\ 1 & -3 \end{pmatrix},$$

  (to get the matrix of q, enter `'4*X^2+2*X*Y-3*Y^2'`, then the list of variables `{ X Y}` and hit `QXA`). Call `GAUSS` which returns:

```
2: { '1/4' '-13/4' }
1: { { 4 1 } { 0 1 } }
```

This means that:

$$A = \begin{pmatrix} 4 & 0 \\ 1 & 1 \end{pmatrix} \times \begin{pmatrix} '1/4' & 0 \\ 0 & '-13/4' \end{pmatrix} \times \begin{pmatrix} 4 & 1 \\ 0 & 1 \end{pmatrix}.$$

This means that:

$$q(x,y) = 4x^2 + 2xy - 3y^2 = \frac{1}{4}(4x+y)^2 - \frac{13}{4}y^2.$$

The other utilities are `QXA` and `AXQ` to switch between algebraic and matrix representation of a quadratic form (quadratic as symbolic to array). `QXA` switches from algebraic representation and accepts an optional list of variables at level 1. `AXQ` switches from matrix representation.

15 Customization and other utilities.

15.1 Data types.

Data handled by `Erable` have two representations: the user representation which you see most the time and the internal representation (used internally).

List of data types:

True data	Example	User	Example	Internal	Example
Integer	5	real, hex, string	5	hex	#5
Float	5.02	real	5.02	long real	%% 5.02
Gauß integer	$1+2i$	symbolic	`'1+2*i'`	secondary	`:: #1 #2;`
Complex	(1.1,2.3)	complex	`(1.1,2.3)`	long complex	`C`
Fractions	$\frac{2}{3}$	symbolic	`'2/3'`	symbolic	`'#2/#3'`
Irr. quadr.	$1+2\sqrt{5}$	symbolic	`'1+2*`$\sqrt{3}$`'`	program	`<< #1 #2 #3 >>`
Unknowns	$a, x \ldots$	variables	`A X`	list	*variable*
Symbolics	$a+x^2$	symbolic	`'A+X^2'`	list	*variable*
Lists	{1 i}	list	`{1 'i'}`	list	`{#1 :: #0 #1;}`
Array	`[1 2 ]`	array	`[[1 2] [3 4]]`	array	`[[1 2] [3 4]]`
Symbolic array	`{1 2}`	array	`{{1 2} {3 4}}`	array	`{{1 2} {3 4}}`

15.2 Flags

HP48 user flags are used to control the behavior of `Erable` and are documented in section E

15.3 Conversions

- `AXL`: array and list conversion (transforms { } to [ ] and conversely).
- `EPSX0`: strip leading zeros in list-polynomials, replace objects by 0 if their absolute value is less than `EPS`.
- `FXND`: splits a fraction in numerator (stack 2) and denominator (stack 1). Example:

  `'(X+1)/A' FXND` gives `2:X+1, 1:A`
- `NDXF`: reverse of `FXND`. Example:

  `1 2 NDXF` gives `'1/2'`

  Works for all data types (warning: you can get strange symbolics with `NDXF`).
- `XNUM`: convert level 1 to a numeric format like the build-in →`NUM`, but accepts lists (this was not the case on S/SX models). Clears flags 12, 14 and 15.
- `XQ`: convert level 1 to rational format, like the build-in →`Q`. Sets flags 12, 14 and 15.
- `SXL` (`other` directory, obsolete): Used for conversion to internal data type representation)
  - `VX` variable-fraction representation conversion. Switches from algebraic to list-polynomials or fractions. Example:

    `'(X+1)/(3*X-2)'` is switched between `'{1 1}/{3 -2}'` (which is displayed as `'UNKNOWN/UNKNOWN'`)
    `'X+3'` is switched between `{1 3}`
  - General stack object conversion. Example:

    ```
    'X+3*SIN(X)'
    { { 1 '5*X' } { 'SIN(X)' 1 } }
    { 'X^2+7*X' '3*SIN(X)' }
    #3h  SXL gives { 'SIN(X)' X }
    { 3 { 1 0 } }
    { { 1 { { 5 0 } } } { { 1 0 } 0 } }
    { { { 1 7 0 } } { 3 0 } }
    ```

    To go back, type `{ #0 #1 #2 } SXL`
- `S2L` (`other` directory, obsolete): convert an algebraic polynomial to a list polynomial. Example:

  `'1+2*A' A S2L` gives `{ 2 1 }`

  It accepts lists. Example:

`{ '1+A' '2*A-3' } A S2L` gives `{ { 1 1 } { 2 -3} }`

- `L2S` (`other` directory, obsolete): It converts a list polynomial to an algebraic. `L2S` may be used for multiple variable polynomial evaluation. Example:

  ```
  { { 1 2 3} {4 5 6} } { X Y } L2S   gives
  '(Y^2+2*Y+3)*X+(4*Y^2+5*Y+6)'
  ```

15.4 Other functions

- `HORNER` executes an Horner scheme. The syntax is:

  ```
  2: P
  1: r
  ```

  gives

  ```
  3: P div (X-r)
  2: r
  1: P(r)
  ```

  Example: `'X^2+2*X+3' 5` gives `'X+7' 5 38` This means that $X^2+2*X+3 = (X+7)(X-5)+38$.

- `PTAYL`: fast Taylor development for polynomials:

  `2: P(X), 1: r` gives `P(X-r)`

  Example:

  `'X^3+2*X' 2 PTAYL` gives `'X^3+6*X^2+14*X+12'`

  which means that $X^3+2X = (X-2)^3+6(X-2)^2+14(X-2)+12$

- `LEGENDRE` [resp. `HERMITE` and `TCHEBYCHEFF`]: given an integer n, returns the n-th degree Legendre [resp. Hermite and Tchebycheff] polynomial.

- `PFEXEC`: execute the program at level 1 in subexpressions between all $+$ and $-$ of the symbolic expression at level 2. For example, try:

  `'1/2/(X^2-1)+1/4/(X^2-4)' << FACTOR >> PFEXEC`

- `tEVAL` (`other` directory): evaluate object 1 and returns the time it took to evaluate it. Not as accurate as `TIM` of the hacker library.

- `LATEX`: converts a symbolic to a string, the LaTeX translation of the symbolic. To `tex` it on a computer, you must include the string in a math. environment (in `$ $` or in `\[ \]` or in an `equation` environment, and you must include the file `hp48.tex`).

15.5 Permutations

A permutation is represented as a list of images of $[1..n]$ e.g. { 5 1 2 4 3 } means $\sigma(1) = 5$, $\sigma(2) = 1$, $\sigma(3) = 2$, $\sigma(4) = 4$ and $\sigma(5) = 3$. The `P2C` instruction converts this representation to the cycle decomposition, here { { 1 5 3 2 } { 4 } } (stack level 2) and computes the signature of p (stack level 1). `C2P` converts cycle decomposition to the usual representation of permutations. `CIRC` compose 2 permutations in the usual representation (returns $\sigma_{\text{level 2}} \circ \sigma_{\text{level 1}}$).

15.6 Variables

- `LVAR`: returns the list of "variables" of an algebraic. The list is sorted by reverse alphabetic order. Example:

  `'SIN(A)+B*X+1'` gives `{ X B 'SIN(A)' }`

- `LIDNT`: list of global names of an algebraic Example:

  `'SIN(A)+B*X+1'` gives `{ X B A }`

15.7 Differential geometry

There is currently only one program available in the `other` directory, written in UserRPL by John Wilson, that I have translated into SysRPL, it is named `TNBA` for tangent, normal, binormal, acceleration. It takes a 3-dimensional vector (as a list) at level 1 (e.g. `{ '2*T' 1 'T^2' }`) and returns the position, tangent, normal and binormal vectors as well as the tangential and normal acceleration.

16 Final remarks.

Remaining things to do:

- extend the Risch algorithm to multiple exponentials?
- improve the factorization algorithm (Berlekamp method over $\mathbb{Z}[i]$).
- Z-transform?
- ....

A Frequently asked questions.

- When I call `SETUP`, I get the error `STO Error` with 0 at level 1. Why?

  You did not use `Kermit` to down load files to your HP48 and your software is case-insensitive, or your file system is case-insensitive and translated uppercase names to lowercase names. Please use `Kermit`, and unzip the `Erable` archive in a case-sensitive file system or a file system where filenames are uppercase names. `MS-DOS` should work, as well as `ext2` (Linux), but not `VFAT` (`Windows 9x` file system under Linux).

- When I call `INIT`, I get the `RCL Error: Undefined Name` with `GXKEYS` at level 1 on the stack. Why?

  The reason is the same as above, but there is a simple workaround in this case. Type the following command line:

  ```
  'gxkeys' DUP RCL 'GXKEYS' STO PURGE INIT
  ```

- How can I launch the `eqstk` stack replacement?

  Type `ASTK`.

- `ASTK` does not launch `eqstk`. Why?

  If you want to shutdown your calculator, you must press the right shift followed by the `ON` key. If you press the `ON` key too fast, this will stop `eqstk` unexpectedly and `ASTK` will not work.

  Workaround: type `ON` and `C` simultaneously, then again `ASTK`. If this doesn't work, you must reinstall `eqstk`.

- In previous versions of `Erable`, a `CST` menu was created. Is it gone?

  Yes, the user interface is now completely handled by the user key redefinition, because this is faster. You may create you own `CST` menu if you want by compiling the `SCST` file from the original package.

- How can I simplify $\sqrt{x^2}$?

  Type `28 SF` to simplify $\sqrt{x^2} = |x|$. Otherwise $\sqrt{x^2}$ remains not modified. This flag setting is not the default because setting flag 28 means that subexpressions are always simplified and this slow down most simplifications.

- How can I simplify $|x|$ if I know that $x > 0$?

  Type `29 SF`. If flag 29 is set, `Erable` will try to guess the sign of your expression: either at $x = 0^+$ or at $x = +\infty$ depending of the status of flag 24. The default is $x = +\infty$ but be aware that calling `SERIES` (directly or indirectly via `LIMIT`) sets flag 24 enabling an estimate of $x = 0^+$.

- Sometimes, `Erable` can not find the antiderivative of $\sin(x)$. Why?

You are in complex mode. In this mode, `INTVX` does not recognize trigonometric functions, you must convert them to complex exponentials (`EXPLN`) or go back to real mode (`13 CF`).

- I have `alg48` installed, I would like to use the factorization routines of `alg48` inside `Erable`. Is it possible?

  Not currently :-(

B All functions of `Erable` listed in alphabetic order.

The following symbols will be used:

- %: real
- C%: complex
- n: integer (real integer)
- [ ]: numeric array
- { l }: list
- { m }: symbolic array
- p: polynomial ({ p } for a list-polynomial),
- { v }: list of variables
- s: symbolic object
- v: variable (global name or irrational symbolic)
- f: a fraction
- N, D: numerator and denominator of a fraction
- o: object

List of all global variables in `HOME`, `algb` or `algbg`:

Name	Function	Arguments	Returns
EPS	ε		%
ERABLMSG	Risch log		string
INVLAP	Last inverse Laplace	nothing	s
MATRIX	Last matrix	nothing	m
MODULO	Arithmetic in $\mathbb{Z}/n\mathbb{Z}$		n
ODETYPE	Ordinary differential		string
	equation type		string
PRIMIT	Last primitive	nothing	s
SYSTEM	Last system	nothing	$\{mv\}$
GXKEYS	User keys string	nothing	string
VX	integration variable	nothing	v
fr	French short doc	nothing	string
us	English short doc	nothing	string

If you are short in memory, you can erase all variables in `{ HOME }` and subdirectories except `EPS`, `VX` and `MODULO`.

Functions of the `Erable`, `ARIT`, `GEO`, `LIN PREP` libraries and of the `other`, `algb` or `algbg` directories:

Name	Function	Arguments	Returns
ABCUV	Bezout	3,2,1:a, b, c	1:1 [3,2: x, y] or 1: 0
	$ax + by = c$		
ACOS2S	Arccos to arcsin	s	s
ASIN2C	Arcsin to arccos	s	s
ASIN2T	Arcsin to arctan	s	s
ATAN2S	Arctan to arcsin	s	s
AXL	array and list	[ ] or { m }	{ m } or [ ]
AXQ	array to s	{ m }	s
	quadratic form		
C2P	Cycles to	{ cycles }	p
	permutations		
CHINREM	Chinese	2: { n N }	{ o O }
	remainder	1: { m M }	
CHS	Change sign	o	$-o$
CIRC	Compose 2	2:p_2, 1:p_1	$p_2 \circ p_1$
	permutations		
COSN	$\cos, \sin(nx)$ to	$n > 0$	2: s, 1: s
	$P(\cos x, \sin x)$		
		$n < 0$	2: { p }, 1: { p }
CROSSP	Wedge product	2: x, 1: y	$x \wedge y$
CURL	Rotational	2: { $s_1 s_2 s_3$ }	{ $s'_1 s'_2 s'_3$ }
		1: { v }	
DEGREE	Order	{ p }	n
DER	derivative or	2: s, 1: v	1: s
	gradient		

DERVX	derivative	s	s
DIV	Divergence	2: { s_1 ... s_k }	s
		1: { v }	
DIV1	Usual division	2: o_2, 1: o_1	o_2/o_1
DIV2	Euclidean division	2: o_2	2: o_2 div o_1
		1: o_1	1: o_2 mod o_1
DIVIS	List of divisors	o	{ l }
DIVPC	Division in ascending power	3: s, 2: s', 1: n	s
DESOLVE	Solve $y'(x) = f(y(x), x)$	$f(y(x), x)$	$y(x)$
DOTP	Scalar product of 2 vectors	2: x 1: y	$x.y$
EPSX0	Strip expression	o	o
ERCFG	Erable configuration		
EULER	Euler indicatrix	n	$\varphi(n)$
EXEC	Substitution or *doall*	2: { l }	1: { l }
		1: program	
		2: s	s
		1: $o_1 = o_2$	
		3: s	s
		2: { l1 }	
		1: { l2 }	
EXPAND	Simplification	o	o'
EXPLN	Conversion to exp, ln	s	s
FACTOR	Factorization	s	s
FACTORS	Factorization	o	3: { v }
			2: f
			1: { $f_1 n_1 f_2 n_2$ ... }
FCOEF	roots/poles to fraction	{ $r_1 n_1 r_2 n_2$ ... }	f
FOURIER	Fourier coefficient	2: $f(x)$	$\int_0^{2\pi} f(x) e^{inx} dx$
		1: n	
FROOTS	Factorization	o	3: { v }
			2: f
			1: { $s_1 n_1 s_2 n_2$ ... }
FXND	Split a fraction	$f = N/D$	2: N, 1: D
GAUSS	Gauß quadratic form reduction	1: A	2: D, 1: P
GCD1	Greatest common divisor	2: o_2, 1: o_1	GCD(o_2,o_1)

GCD3	GCD (solves $au + bv = d$)	2,1: a , b	GCD(a,b) = d, u, v
GROBADD	Add grobs (horiz. gluing)	2: grob 1: grob	1: grob
HALFTAN	To half angle tangent		
HERMITE	Hermite polynomial	integer n	H_n
HESS	Hessian	2: s, 1: { v }	matrix
HILBERT	Hilbert matrix	integer n	$n \times n$ matrix
HORNER	Horner scheme	2:p 1: r	3:$p/(X - r)$ 2: r 1: $P(r)$
IBP	Integration by parts	$\int_a^b f(t)dt$, u	$[uv]_a^b - \int_a^b uv'(t)dt$ $v = f/u'$
IDENTITY	identity	real, integer or matrix	identity matrix
ILAP	Inverse Laplace transform	s	$\mathcal{L}^{-1}(s)$
INIT	Initialization	nothing	nothing
INTVX	Symbolic integration	s	s
JORDAN	Diagonalization	endomorphism	7 to 1: *cf.* section 12
KERN	Kernel of a linear appl.	m	4 to 1: *cf.* section 12
KEYEVAL	Execute keynum	$nn.d$	?
L2S	Evaluation	2: { p }, 1:v	$p(v)$
LAGRANGE	Lagrange . interpolation.	{ { x_1 ... x_n } { y_1 ... y_n } }	P such that $P(x_i) = y_i$
LAP	Laplace transform	2:f, 1:g	$L(f)/g$
LAPL	Laplacian	2: f, 1: { v }	Δf
LATEX	LaTeX conversion	1: s	1: string
LCM1	Least common multiple	2: o_2, 1: o_1	LCM(o_2,o_1)
LCXM	Matrix creation	3: r 2: c 1: prog	1: $r \times c$ matrix
LDEC	Linear Diff. w/Cnst. Coef.	2: { v } 1: { m }	1: solution
LEGENDRE	Polynomials	integer r	L_r
LGCD	GCD of a list	{ l }	o=GCD
LIDNT	List of variables	s	2: s, 1: { v }

LIMIT	Limit	3:s, 2:v, 1:n	s
LIN	Linearization of exp	s	s
LNCOLLECT	Collect log	s	s
LU2	LU decomposition	M	L^-1, U
LVAR	list of variables	o	{ v }
MAD	inverse, char. polyn., etc.	o	4: det, 3: $1/o$, 2: { p }, 1: { p }
MAIN	Main menu of `Erable`		
MENUXY	list of `Erable` commands	2: n_1 1: n_2	1: { }
MMULT	special product	3: o_2, o_1, n	"$o_2 \times o_1$"
MULT	product	2: o_2, o_1	$o_2 \times o_1$
NDXF	create a fraction	2: N, 1: D	$f = N/D$
ORND	Round an object	2: o, 1: D	o
P2C	Permutation to cycles	p	3: p 2: cycles 1: signature
PA2B2	Prime factorization	1: p $p \equiv 1[4]$	1: $\frac{a+ib}{a^2+b^2}$ $= p$
PCAR	Characteristic polynomial	endomorphism	s
PARTFRAC	Partial fraction	f	$\sum_i f_i$
PFEXEC	EXEC between + and −	2:$\sum_i f_i$ 1: prg	$\sum_i \text{prg}(f_i)$
PFSTEP	Step-by-step partial fraction	f	$\sum_i f_i$
POTENCE	Euclidean division	2: n (or list-poly) 1: n (or list-poly)	
PLOTADD	Add stack to plot	s	s
PLOTSTK	Plot stack	s	s
POWER	integral power	2: o, 1: n	o^n
PREVAL	Evaluation	3: primitive 2,1:bornes	s
PROMPTSTO	Prompt and store	1: name	nothing
PTAYL	Taylor for polynomials	2: $P(X)$, 1: o	$P(X - o)$
PURG	Purge algb(g)	nothing	nothing
QXA	s quadratic form	2: s, 1: { v }	{ m }

	to array		
		s	2:{ m }, 1: { v }
RDET	Determinant	endomorphism	2: { m }
	(rref)	{ m }	1: determinant
REF	Sub-diagonal reduction	{ m }	2: special cases 1:{ m }
RISCH	Symbolic integration	s	s x
S2L	Symbolic to list	2: o, 1: { v }	2: { v },1:{ p }
		2: o, 1: v	{ p }
SCROLL	Scrolls a grob	grob	
SERIES	Series	3: s, 2: v, 1: n	6: 6-1: s
SETFR	Set French Flags	nothing	nothing
SIMP2	Simplification	2: o_2, 1: o_1	2: o_2', 1: o_1'
SINCOS	Exponential to	s	s
SIGNE	Sign of a rational fraction	s	tagged list
SOLGEN	Solves a linear system	{ eqns { v } }	*cf.* section 12
SOLVE	Solve	2: s, 1: x	solutions
SOLVEX	Solve for x	s	solutions
SQRT	Square root	n or C% or s	n or C% or s
STUDMULT	"students" $\times$ of matrices	M, M'	"$M\dot{M}'$ "
SUBT	Subtraction	2: o_2, 1: o_1	$o_2 - o_1$
SXL	Conversion	Internal [user]	User [internal]
SYST	Solves a linear system	{ eqns { v } }	*cf.* section 12
TABVAL	Table of values	2: s, 1: { }	{ }
TABVAR	Variation table	1: s	
TAN2SC	Tangent to sine/cosine		
TAN2SC2	Tangent to sin/cos 2θ		
TAYLOR0	Taylor at $x = 0$ order 4	s	s
TCHEBY-CHEFF	Polynomials	integer r	T_r
TEXPAND	Expand transcendent functions	s	s
TLIN	Trigonometric linearization	s { p }	s
TNBA	Tangent, nor-	{v}	

	mal, ...		
TR	trace	[ ] or { m }= $(a_{ij})_{1\leq i,j\leq n}$	$\sum_{i=1}^{n} a_{ii}$
TRAN	transpose	[ ] or { m }	[ ] or { m }
TRIG	i, ln & exp to sin, cos, arctan	s	s
TRIGCOS	Trigonometry: $\sin^2$ to $1-\cos^2$	s	s
TRIGSIN	Trigonometry: $\cos^2$ to $1-\sin^2$	s	s
TRUNC	Truncate an asymptotic expansion	2: s, 1: rest s'	s
TSIMP	Simplification (transcendental)	s	s
VANDER-MONDE	Vandermonde matrix	list of objects	matrix
VER	Version	nonthing	% 3.2
XFRC	To quadratic irrational	o	o
XGROB	to grob	o	graphic object
XNUM	to numeric	o	o
XQ	to rational	o	o
ZEROS	Solve	2: s, 1: x	solutions
abs	Absolute value	s	s
add	Addition	2: o_2, 1: o_1	o_2+o_1
arg	Argument	1: s	1: s
comb	Combinations	2: n, 1: n'	$C_n^{n'}$
conj	Conjugate	o	$\overline{o}$
det	Determinant (expand)	endomorphism	determinant
fact	Factorial	n	$n!$
im	imaginary part	o	$\Im(o)$
inv	Inversion	o	o^{-1}
re	real part	o	$\Re(o)$
rref	Row reduction	M	{ s }, reduced matrix
tEVAL	Execution time	... 1: o	EVAL(o), 1: time

Modular functions of the `ARIT` library. Don't forget to set an integer n in the variable `MODULO` (by default $n=2$):

Name	Function	Arguments	Returns
`{KERNEL.LIB}`	(`0:788`)		

ADDTMOD	Modular addition	2: n_1, 1:n_2	$(n_1 + n_2)$ mod n
SUBTMOD	Modular subtraction	2: n_1, 1:n_2	$(n_1 - n_2)$ mod n
MULTMOD	Modular multiplication	2: n_1, 1:n_2	$(n_1 * n_2)$ mod n
DIV1MOD	Modular division	2: n_1, 1:n_2	(n_1/n_2) mod n
POWMOD	Modular power	2: n_1, 1:n_2	$n_1^{n_2}$ mod n
INVMOD	Modular inversion	1: n_1	n_1^{-1} mod n
GCD1MOD	Modular gcd	2: n_1, 1:n_2	$gcd(n_1; n_2)$
EXPAMOD	Modular expand	1: o	o mod n

C User Keys.

From top left corner to bottom right corner, α -[Right Shift]-ed keys:

Princ. Key	Reminder	Function
MTH		MAIN
PRG		(normal PRG)
EVAL		EXEC
SIN	∂	DERVX
COS	$\int$	INTVX
$\sqrt{}$	$\sqrt{}$	SQRT
y^x	y^x	POWER
$1/x$	$1/x$	inv
$\pm$	pm	CHS
DEL	∞	
$\div$	$\div$	DIV1
$\times$	$\times$	MULT
-	-	SUBT
SPC		EXPAND
+	+	ADDT

[Left Shift]-ed keys:

Princ. Key	Reminder	Function
7	(solve, factor)	SOLVER
8	(exp and ln)	EXP&LN
4	(matrices)	MATR
5	(diff. calc)	DIFF
6	(arithmetic)	ARIT
1	(basic algebra)	BASE
2	(complex)	CMPLX
3	(trigonometry)	TRIG

Other redefined keys:

- G/GX only: `PRG` key (runs a HP48GX main menu)
- `CST` key: i
- `DEL` key: `X`
- right shift-`DEL` key: $-\infty$
- left shift-down arrow (35.2): calls `AGROB` followed by `SCROLL`. Needs `EQSTK` or `JAVA`. Type:

  ```
  35.2 DELKEYS
  ```

  if you do not have `EQSTK` or `JAVA`
- S/SX only: 33.2 (`XQ`) and 33.3 (`XNUM`)
- G/GX only: 33.2 (`XNUM`) and 35.6 (`XQ`).

D Erable 3.024 compatibility.

Several functions have been renamed to improve compatibility with Texas Instruments calculators:

Previous	Current
COLC	FACTOR
DSOLVE	DESOLVE
EXPA	EXPAND
EXPLIN	LIN
FACTO	FACTORS
FSIGN	SIGNE
HORN	HORNER
IPP	IBP
INVL	inv
LNCOLC	LNCOLLECT
PF	PARTFRAC
RANG	RANK

Previous	Current
SOLV	SOLVE
TCHEB	TCHEBYCHEFF
TEXPA	TEXPAND
TRIGCOLC	TCOLLECT
TRIGLIN	TLIN
VAND	VANDERMONDE
WEDGE	CROSSP
XY	DOTP
cross	CROSSP
der	DER
der1	DERVX
idn	IDENTITY

Many functions have been added or modified: see the `CHANGES` file.

E User flags.

List of the flags used by `Erable`. The sign * after the flag number means that the flag is cleared if `VER` is called, # means that the flag is set if `VER` is called, some flags (e.g. flag 1) are not modified by `VER`

- 01: if set then verbose mode (details of some algorithms are shown) else quiet mode.

- 10: if set then `Erable` performs integer arithmetic otherwise `Erable` performs polynomial arithmetic
- # 11: internal use, cleared if a non-rational algebraic is found
- # 12: if clear then `GCD` returns always 1 (hence algebraics are not simplified and multiple roots of polynomials are not detected)
- # 13: if set then complex mode, else real mode (modifies the way of simplifying expressions with `re`, `im` and `conj` and the way of rooting polynomials)
- # 14: if set then searches formal first order factors
- # 15: if set enables construction of integer fractions and square roots of integers
- * 16: enables/disables calls of `TSIMP` inside `SERIES`.
- * 17: cleared for non modular computations
- * 18: internally used for Euclidean division of polynomials
- * 19: internally used by `INTVX`
- * 20: "Large data" flag. Set this flag if your data are large e.g. if you want to invert a 10×10 matrix.
- # 21: if set then recursive simplification for `EXPAND` and `TSIMP`
- * 22: if set then the rule $i^2 = -1$ is not applied
- * 23: if set then `INTVX` does not try linearity
- * 24: if set then positivity of expressions are tested at $x = 0$ instead of at $+\infty$.
- * 25: if set then the rule $\sqrt{x}^2 = x$ is not applied
- * 26: if set then `TRIG` tries to return only sines, otherwise it returns cosines
- * 27: if set `MAIN` launches a chooser, else a menu
- * 28: determines whether embedded quotients are immediately simplified or not (for example $2/4 + 3/6$ may be simplified first to $1/2 + 1/2$ and then to 1 or directly to $24/24 = 1$).
- * 29: "sloppy flag": if set then $|x|$ is simplified to x for every "variable" (as returned by `LVAR`)

To set a flag (e.g. flag 13), type `13 SF`. To clear this flag, type `13 CF`.
You should only modify user flags 1,10,13,15 and 20 to 29.

Remark 9 *System flag 27 (*`-27`*) affects the way symbolic complex numbers are displayed. System flag 2 affects the way symbolic constants are evaluated (for example π is returned as a symbolic constant or as a numeric approximation). This flag is set or cleared by* `Erable` *according to the current mode of* `Erable` *(numeric or symbolic).*

F Error codes for the `SERIES` command.

- 1: can not determine series expansion for $\arctan(x)$ function with current argument x.
- 2: $\arcsin(x)$ not defined for infinite argument.
- 3: no series expansion for $\exp(x)$ at $x = \infty$ if sign is unknown.
- 4, 5, 6, 7: failed to compare 2 variables.
- 8: can not determine order for the current rest.
- 9: negative argument for logarithm function.
- 10 (`<Ah>`): insufficient order. (You can try again with a larger order)
- 11 (`<Bh>`): can not find sign of argument of `ABS` function
- 16 (`<10h>`): numeric input are not allowed

G Thanks to ....

Many people helped me during the creation and distribution of `Erable`:

- Claude-Nicolas Fiechter and Mika Heiskanen for letting me use their long integer routines for `Erable`. Special thanks to Mika for explanations about the source code of `ALG48`.
- Some math teachers, particularly Renée de Graeve and Scott Guth who made tests, suggestions and bug reports.
- Randolph J. Herber for additions, corrections and documentation formatting
- Some of my students and net-surfers tested various versions of `Erable` and encouraged me to improve it: particularly Maurice Al-Khaliedy, Christophe Burdin, Craig Clifford, Jerome Coss, David Czinczenheim, Benoit Darcy, Ludovic Dumaine, Eduardo (a.k.a. `maciasval@mx2.redestb.es`), Frederic Hermann, Eric Gorka, Stephane Monboisset, Lionel Pilot, Eric Saya, Quan Tong Duc, Camillo Toselli, Samy Venin, John Wilson, ... Special thanks to Gilles Virone who showed me first what an HP28/48 is able to do.

- all anonymous ftp sites administrators, particularly those of

  fourier.ujf-grenoble.fr (André Voutier),
  ftp.funet.fi,
  cbs.cis.com, hplyot.obspm.fr,
  hpcvbbs.cv.hp.com and
  wuarchive.wustl.edu,

- I used the following software to create `Erable`: the `EQSTK`, `JAVA` stack displays ([7], [17]), the `TED` and `Miniwriter` editors ([13], [1]), the `JAZZ` debugger ([12]), the `Metakernel` ([14]), various compilers (`JAZZ`, the `HP48` tools ([2]), the `RPL` based tools ([16]) and eventually the GNU `tools` ([15]).

- I looked at the following books and software: [8], [3], [6], [4], [5], [10], [11], [9] . One of the best reference is certainly [4] and references therein. M. Heiskanen's WWW-homes-page has a lot of interesting math links.

References

[1] J.-Y. Avenard. `miniwriter`. `http://www.epita.fr/~avenar_j`, 1997.

[2] H. P. Corvallis. `TOOLS.EXE`. `hpcvbbs.cv.hp.com ftp.funet.fi wuarchive.wustl.edu`, 1991.

[3] P. Courbis and S. Lalande. *Voyage au centre de la HP 48 S/SX*. Angkor, 1993.

[4] J. Davenport, Y. Siret, and E. Tournier. *Calcul formel: Systèmes et algorithmes de manipulations algébriques*. Masson, 1989.

[5] J. Ferrard. *"Mathez" la HP 48 G/GX*. D31 Diffusion (HP48), 1993.

[6] C. Ferraro. `POLY46SX, POLY46GX, SMATH, SMATHGX`. `ftp.funet.fi ftp.cis.com hplyot.obspm.fr`, 1993.

[7] C. N. Fiechter and M. Heiskanen. `EQSTK92.ZIP`. `http://www.hut.fi/~mheiskan`, 1997.

[8] C. N. Fiechter and M. Heiskanen. `ALG48V42.ZIP`. `http://www.cs.pitt.edu/~fiechter/hp48` `http://www.hut.fi/~mheiskan`, 1998.

[9] B. Fuchssteiner. `MuPAD`. `ftp://ftp.inria.fr/lang/MuPAD` `http://www.mupad.de`, 1998.

[10] F. Gantmacher. *Théorie des matrices*, volume 1. Dunod, 1966.

[11] M. Heiskanen. `POLYLIB.ZIP`. `http://www.hut.fi/~mheiskan`, 1992,1995.

[12] M. Heiskanen. `JAZZV65.ZIP`. `http://www.hut.fi/~mheiskan`, 1996.

[13] M. Heiskanen. `TED31.ZIP`. `http://www.hut.fi/~mheiskan`, 1997.

[14] Maubert Development Group. `Metakernel, 48+`. `http://www.epita.fr/~avenar_j`, 1998.

[15] M. Mikocevic. `GNUTOOLS`. `ftp://srcm1.zems.fer.hr:/pub/hp48/tools2.1.9.zip`, 1995.

[16] D. Müeller and R. Hellstern. `RPL48V20.ZIP`. `ftp.funet.fi cbs.cis.com hplyot.obspm.fr`, 1993.

[17] R. Steventon, A. Schoorl, and W. Laughlin. `JAVA34.ZIP`. `ftp://ftp.cis.com/pub/hp48g/uploads/java34.zip` `http://www.engr.uvic.ca/~aschoorl`, 1998.