χCAS(io)

Bernard.Parisse@univ-grenoble-alpes.fr.fr

2018

Contents

Abstract

This document explains how to run efficiently χCAS on some Casio calculators (Prizm and Fx-9750GIII). χCAS is a port of the Giac/Xcas computer algebra system (CAS) for these calculators.

CAS are not allowed during exams in some countries, it is the user responsability to check the rules before running χCAS in an exam. The authors shall not be held responsible for misuse of χCAS in exam conditions.

This document is interactive, you can modify and run commands by clicking in the ok button or by hitting Enter.

1 Installation

To install or update χCAS, get on your computer the file khicasen.g3a (Prizm) or khicasen.g1a (Fx-9750GIII).

Connect the USB cable of the calculator, type F1 for USB key connection and copy the file `khicasen.g3a` (or `khicasen.g1a`) on the calculator-"key" then disconnect the calculator-key from your computer and wait a few seconds.

If you test on the emulator, (PC, Mac), from the main menu of the calculator (MENU), go to Memory then F3 (Import/Export), then F1 (Import files), select the file `khicasen.g3a` (or `khicasen.g1a`), type F1 to save to the calculator root directory, confirm with F1 if you upgrade. Be patient, the transfert will take several minutes (about 5).

Once the transfert is finished, you should see the icon of Xcas in the main menu (a snowflake on the Prizm).

2 First steps

From the main menu (MENU), move the cursor to the Xcas icon and hit EXE. This opens the "shell" (or history) where you can write most Xcas commands.

For example, type `1/2+1/6` then EXE, you should see the result `2/3` displayed below.

You can copy a level from the history of commands by hitting the up and down arrow keys (once or more) and EXE. Then you can modify the command and run it with EXE. For example, up arrow twice, EXE, replace `1/6` by `1/3` and hit EXE.

The last result is stored in `ans()`, hit the `Ans` calculator key (shift `(-)`) to get it. It is recommended to store the result in a variable if you want to reuse a result later. There are two ways to store a value in a variable

- right-store with `=>` using the $\rightarrow$ key, for example `2=>A` stores 2 in variable A. Now, every time you write `A` in a computation, it will be replaced by 2.
- left-store with `:=` (shift INS), for example `A:=2` does the same as `2=>A`.

The most popular Xcas commands are available from F1 (algebra) and F2 (calculus), then from the Catalog, where they are shortly explained with an example. Hit F4 (cmds), choose a submenu, for example `Algebra`, hit EXE, move the selection to a command, for example `factor`. Now F6 will display a short help with an example. Hit F2 to copy the example in the commandline. You can run it as is (EXE) or modify it and run it (EXE) if you want to factor another polynomial.

When a command returns an expression, it is displayed in 2d mode. You can move with the pad if the expression is larger than the display. Type shift-F3 or ALPHA-F3 to modify the fontsize. Type EXIT to go back to the shell. The 2d view is in fact a 2d editor that will be explained later.

Now, try to type the command `plot(sin(x))`. Hint: type F4 (cmd), then select `Graphs`.

When a command returns a graph, it will be displayed in a 2d frame. You can modify the displayed area with + or − (zoom in or out, `(-)` does a partial zoomout along Oy), the cursor keys, `/` (orthonormalisation of the frame), `*` (autoscale), VAR or OPTN is a switch to display or hide axis. Type F1 (menu) to modify the graphic window settings Xmin, Xmax, Ymin, Ymax. Type EXIT to go back to the shell.

The KhiCAS File menu (F6) has an item `Clear` that will erase the display. This will not clear the variables, to achieve that type `VARS`, select the last item (`restart`) and confirm with EXE.

Hit MENU to leave χCAS. If you launch another application, the variables and history will be saved, they will be restored if you come back to χCAS. First time save is sometimes slow (10 to 20 seconds), next save will run faster.

3 Common CAS commands

3.1 Expand and factor

From F4 commands catalog, select `Algebra`, or type F1.

- `factor` : factorization. Shortcut `=>*` ($\rightarrow$ key then *), for example `x^4-1=>*`

  $$(x-1)(x+1)\left(x^2+1\right)$$

  . Run `cfactor` to factor over $\mathbb{C}$.

- `partfrac` : expands a polynomial or performs partial fraction expansion over a fraction. Shortcut `=>+` ($\rightarrow$ then + key), for example `(x+1)^4=>+`

  $$x^4+4x^3+6x^2+4x+1$$

  or `1/(x^4-1)=>+`

  $$\frac{1}{4(x-1)}-\frac{1}{4(x+1)}-\frac{1}{2(x^2+1)}$$

  .

- `simplify` : tries to simplify an expression. Shortcut `=>/` ($\rightarrow$ key then /), for example `sin(3x)/sin(x)=>/`

$$2\cos(2x)+1$$

- `ratnormal` : rewrite as an irreducible fraction.

3.2 Calculus

From F4 commands catalog, select `Calculus`, or type F2

- `diff` : derivative. Shortcut `'` for derivative with respect to x, example

  `diff(sin(x),x)`

$$\cos x$$

  and `sin(x)'`

$$\cos x$$

  are equivalent. For nth-derivative, add n, for example 3rd derivative `diff(sin(x^2),x,3)`

$$-8x^3\cos\left(x^2\right)-12x\sin\left(x^2\right)$$

  .

- `integrate` : antiderivative (1 or 2 or 4 arguments) for example

  `integrate(sin(x))`

$$-\cos x$$

  or `integrate(1/(t^4-1),t)`

$$\frac{\ln|t-1|}{4}-\frac{\ln|t+1|}{4}-\frac{\arctan t}{2}$$

  for $\int \frac{1}{t^4-1}\,dt$
  Defined integration with 4 arguments, for example `integrate(sin(x)^4,x,0,pi)`

$$\frac{3}{8}\pi$$

  computes $\int_0^\pi \sin(x)^4\,dx$. For an approximate computation, enter one boundary as an approx number, for example

  `integrate(sin(x)^4,x,0.0,pi)`

$$1.1780972451$$

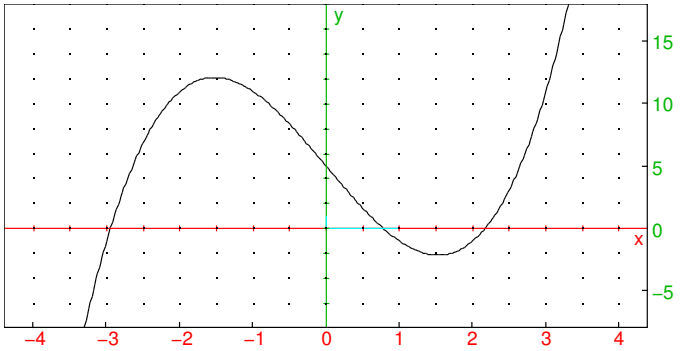
- `limit` : limit of an expression. Example `limit((cos(x)-1)/x^2,x=0)`

$$-\frac{1}{2}$$

- `tabvar` : table of variations of an expression. for example `tabvar(x^3-7x+5)`

$$\left[\begin{array}{cccccccccc} x & -\infty & "" & -\frac{\sqrt{21}}{3} & "" & 0 & "" & \frac{\sqrt{21}}{3} & "" & +\infty \\ y'=\left(x+\frac{\sqrt{21}}{3}\right)\left(3x-\sqrt{21}\right) & +\infty & "+" & 0 & "-" & -7 & "-" & 0 & "+" & +\infty \\ y=x^3-7x+5 & -\infty & "" & \frac{14\sqrt{21}+45}{9} & "" & 5 & "" & \frac{-14\sqrt{21}+45}{9} & "" & +\infty \\ y'' & -\infty & "-\ ()" & -2\sqrt{21} & "-\ ()" & 0 & "+\ ()" & 2\sqrt{21} & "+\ ()" & +\infty \end{array}\right]$$

one can check with the graph `plot(x^3-7x+5,x,-4,4)`



- `taylor` and `series` : Taylor expansion or asymptotic serie expansion, for example

`taylor(sin(x),x=0,5)`

$$x-\frac{x^3}{6}+\frac{x^5}{120}+x^6\text{order\_size}\,(x)$$

Add `polynomial` if you do not want to have the remainder term.

- `sum` : discrete summation, for example

`sum(k^2,k,1,n)`

$$\frac{2\,(n+1)^3-3\,(n+1)^2+n+1}{6}$$

computes $\sum_{k=1}^{n} k^2$,

`sum(k^2,k,1,n)=>*`

$$\frac{1}{6}n\,(n+1)\,(2n+1)$$

computes the sum and rewrites it factored.

3.3 Solvers

From F4 commands catalog, select `Solve`.

- `solve` solves an equation exactly. Takes the variable to solve for as second argument, unless it is `x`, for example `solve(t^2-1=0,t)`

$$[-1,1]$$

.
If exact solving fails, run `fsolve` for approx solving, either with an iterative method starting with a guess `fsolve(cos(x)=x,x=0.0)`

$$0.739085133215$$

, or by dichotomy `fsolve(cos(x)=x,x=0..1)`

$$[0.739085133215]$$

.
For complex solutions, run `csolve`.
It is possible to restrict solutions using assumptions on the variable, for example `assume(m>1)`

$$m$$

then `solve(m^2-4=0,m)`

$$[2]$$

.

- `solve` can also solve (simple) polynomial systems, enter a list of equations as 1st argument and a list of variables as 2nd argument, for example intersection of a circle and a line:
`solve([x^2+y^2+2y=3,x+y=1],[x,y])`

$$[[0,1],[2,-1]]$$

- Run `linsolve` to solve linear systems. enter a list of equations as 1st argument and a list of variables as 2nd argument, example:
`linsolve([x+2y=3,x-y=7],[x,y])`

$$\left[\frac{17}{3},-\frac{4}{3}\right]$$

- Run `desolve` to solve exactly a differential equation. for example, to solve $y'=2y$, type `desolve(y'=2y)`.
Another example with an initial condition:

`desolve([y'=2y,y(0)=1],x,y)`
Run `odesolve` for approx solving or `plotode` for a graphic representation of the approx. solution.

- `rsolve` solves some recurrence relations $u_{n+1} = f(u_n, ...)$, for example to solve the arithmetico-geometric recurrence $u_{n+1} = 2u_n + 3, u_0 = 1$, type:

  `rsolve(u(n+1)=2*u(n)+3,u(n),u(0)=1)`

  $$[4 \cdot 2^n - 3]$$

3.4 Arithmetic

When required, the distinction between integer arithmetic and polynomial arithmetic is done by a prefix `i` for integer commands. For example `ifactor` for integer factorization and `factor` for polynomial factorization (or `cfactor` for polynomial factorization over $\mathbb{C}$). Some commands work for integers and polynomials, like `gcd` and `lcm`.

3.4.1 Integers

From F4 catalog, select `Arithmetic, Crypto`. Shortcut shift S↔D

- `iquo(a,b), irem(a,b)` quotient and remainder of euclidean division of two integers.

  `iquo(23,13),irem(23,13)`

  $$1, 10$$

- `isprime(n)` checks whether n is prime. This is a probabilisitic test for large values of n.

  `isprime(2^64+1)`

  faux

- `ifactor(n)` factorizes an integer (not too large, since algorithms used are trial division and Pollard-ρ, there is no space left in memory for quadratic sieve), for example

  `ifactor(2^64+1)`

  $$67280421310721 \cdot 274177$$

  Shortcut → then * (=>$\star$)

- `gcd(a,b)`, `lcm(a,b)` GCD and LCM of two integers or polynomials.

```
gcd(25,15),lcm(25,15)
```

$$5, 75$$

```
gcd(x^3-1,x^2-1),lcm(x^3-1,x^2-1)
```

$$x-1, \left(x^2+x+1\right)\left(x^2-1\right)$$

- `iegcd(a,b)` returns 3 integers u, v, d such that $au + bv = d$ where d is the GCD of a et b, $|u| < |b|$ and $|v| < |a|$.

```
u,v,d:=iegcd(23,13); 23u+13v
```

$$[4, -7, 1], 1$$

- `ichinrem([a,m],[b,n])` returns (if possible) c such that $c = a \pmod m$ and $c = b \pmod n$ (if m are n coprime, c exists).

```
c,n:=ichinrem([1,23],[2,13]); irem(c,23);
 irem(c,13)
```

$$[93, 299], 1, 2$$

- `powmod(a,n,m)` returns $a^n \pmod m$ computed by the fast modular powering algorithm

```
powmod(7,22,23)
```

$$1$$

- `asc` converts a string to a list of ASCII code, `char` converts back a list to a string. These commands may be used to easily write cryptographic algorithms with string messages.

3.4.2 Polynomials

From F4 catalog, select `Polynomials`. The default variable is x, otherwise you can specify it as last optional argument. For example `degree(x^2*y)` or `degree(x^2*y,x)` return 2, `degree(x^2*y,y)` returns 1.

- `coeff(P,n)` coefficient of x^n in P, `lcoeff(P)` leading coefficient of P, for example

  `P:=x^3+3x; coeff(P,1); lcoeff(P)`

$$x^3+3x, 3, 1$$

- `degre(P)` degree of polynomial P

  `degree(x^3)`

$$3$$

- `quo(P,Q)`, `rem(P,Q)` quotient and remainder of euclidean division of `P` by `Q`

  `P:=x^3+7x-5; Q:=x^2+x; quo(P,Q); rem(P,Q)`

$$x^3+7x-5, x^2+x, x-1, 8x-5$$

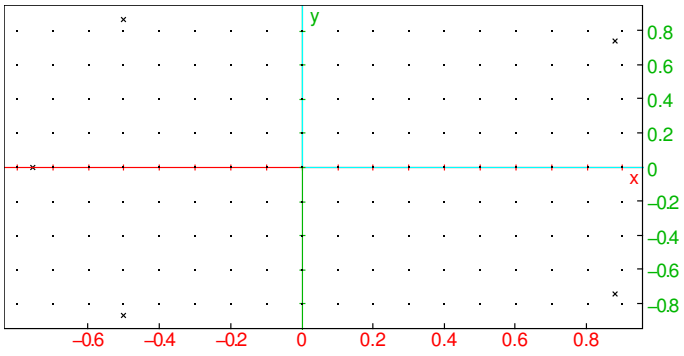
- `proot(P)` : approx. roots of P (all roots, real and complex)

  `proot(x^5+x+1)`

$$[-0.754877666247, -0.5-0.866025403784\mathrm{i}, -0.5+0.866025403784\mathrm{i}, 0.877438833123-0.744861766 62\mathrm{i}, 0.$$

  Graphic representation

  `point(proot(x^5+x+1))`



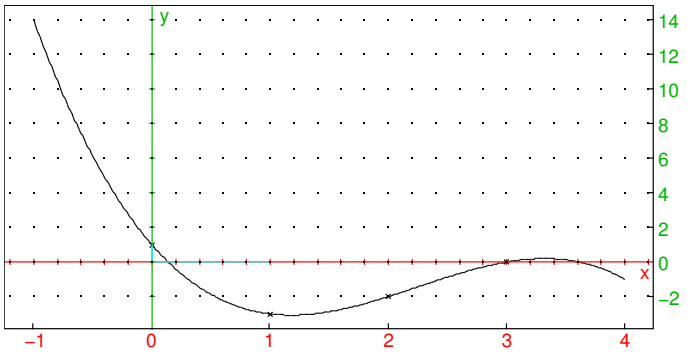
- `interp(X,Y)` : for two lists of the same size, returns the interpolating polynomial P such that $P(X_i)=Y_i$.

  `X,Y:=[0,1,2,3],[1,-3,-2,0]; P:=interp(X,Y)=>+`

$$[0,1,2,3], [1,-3,-2,0], \frac{-4x^3+27x^2-47x+6}{6}$$

Graphic representation

```
scatterplot(X,Y); plot(P,x,-1,4)
```



- `resultant(P,Q)` : resultant of polynomials P and Q

  ```
  P:=x^3+7x-5; Q:=x^2+x; resultant(P,Q)
  ```

  $$x^3+7x-5, x^2+x, 65$$

- `hermite(x,n)` : n-th Hermite polynomial (orthogonal for the density $e^{-x^2}dx$ on $\mathbb{R}$)
- `laguerre(x,n,a)` : n-th Laguerre polynomial
- `legendre(x,n)` : n-th Legendre polynomial (orthogonal for the density dx on $[-1,1]$)
- `tchebyshev1(n)` and `tchebyshev2(n)` Tchebyshev polynomials of 1st and 2nd kind defined by :

$$T_n(\cos(x)) = \cos(nx), \quad U_n(\cos(x))\sin(x) = \sin((n+1)x)$$

3.5 Linear algebra, vectors, matrices

Xcas does not make distinction between vectors and lists. For example,

```
v:=[1,2]; w:=[3,4]
```

defines 2 vectors v and w, then `dot` will compute the scalar product of v and w:

```
dot(v,w)
```

11

A matrix is a list of lists of the same size. You can enter a matrix element by element using the matrix editor (shift-MATR EXE or F6 0). Enter a new variable name

to create a new matrix or the name of an existing variable to edit a matrix. The , key may be used to insert a line or column, and the DEL key erases the line or column of the selection (press UNDO if you want to go one step back). For small matrices, it is also convenient to enter them directly in the commandline, for example to define

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

```
A:=[[1,2],[3,4]]
```

or `[[1,2],[3,4]]=>A`

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

It is recommended to store matrices in variables!

If a matrix is defined by a formula, then it's better to use the `matrix` command (shift-MATR EXE AC), for example:

```
matrix(2,2,(j,k)->1/(j+k+1))
```

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

returns the matrix where coefficient line j and column k is $\frac{1}{j+k+1}$ (beware, indices begin at 0).

Run `idn(n)` to get the identity matrix of order n and `ranm(n,m,law,[parameter])` to get a matrix with random coefficients with dimensions n, m. for example

```
U:=ranm(4,4,uniformd,0,1)
```

$$\begin{pmatrix} 0.383762286976 & 0.825535877142 & 0.0712354816496 & 0.640399608761 \\ 0.277037233114 & 0.385660002008 & 0.967445645481 & 0.955524944235 \\ 0.658353145234 & 0.393765959423 & 0.940668717492 & 0.881703319494 \\ 0.168567307759 & 0.833197717089 & 0.52638866799 & 0.38078169385 \end{pmatrix}$$

```
N:=ranm(4,4,normald,0,1)
```

$$\begin{pmatrix} 0.438672615623 & -1.08072554736 & -1.87165197475 & 0.869559690319 \\ 1.15057649806 & 2.08750998169 & -1.19023312065 & 0.366829848087 \\ -1.65374124495 & -0.276421576494 & -0.393102505248 & -1.42117901973 \\ 1.99331539221 & -1.11210372428 & -0.581409118539 & -0.42763568061 \end{pmatrix}$$

For basic arithmetic on matrices, use keyboard operators (`+ - *`, inverse). Otherwise, open catalog and select `Matrices`

- `eigenvals(A)`

$$\frac{\sqrt{33}+5}{2}, \frac{-\sqrt{33}+5}{2}$$

`eigenvects(A)`

$$\begin{bmatrix} \sqrt{33}-3 & -\sqrt{33}-3 \\ 6 & 6 \end{bmatrix}$$

eigenvalues and eigenvectors of matrix A.

- `P,D:=jordan(A)`

$$\begin{bmatrix} \sqrt{33}-3 & -\sqrt{33}-3 \\ 6 & 6 \end{bmatrix}, \begin{bmatrix} \frac{\sqrt{33}+5}{2} & 0 \\ 0 & \frac{-\sqrt{33}+5}{2} \end{bmatrix}$$

finds the Jordan normal form of matrix A, returns matrices P and D such that $P^{-1}AP = D$, with D upper triangular (diagonal if A is diagonalizable)

- `Ak:=matpow(A,k)`

$$\begin{bmatrix} \frac{1}{66}\left(\sqrt{33}-3\right)\left(\frac{\sqrt{33}+5}{2}\right)^k\sqrt{33}-\frac{1}{66}\left(-\sqrt{33}-3\right)\left(\frac{-\sqrt{33}+5}{2}\right)^k\sqrt{33} & \frac{1}{132}\left(\sqrt{33}-3\right)\left(\frac{\sqrt{33}+5}{2}\right)^k\left(\sqrt{33}+11\right)- \\ \frac{6}{66}\left(\frac{\sqrt{33}+5}{2}\right)^k\sqrt{33}-\frac{6}{66}\left(\frac{-\sqrt{33}+5}{2}\right)^k\sqrt{33} & \frac{6}{132}\left(\frac{\sqrt{33}+5}{2}\right)^k\left(\sqrt{33}+11\right) \end{bmatrix}$$

computes matrix A to the k-th power, where k is symbolic.

- `rref`: row reduction to echelon form
- `lu`: LU factorization of matrix A, returns a permutation P and two matrices L (lower) and U (upper) such that $PA = LU$. The result of

  `P,L,U:=lu(A)`

$$[0,1], \begin{bmatrix} 1 & 0 \\ 3 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 2 \\ 0 & -2 \end{bmatrix}$$

  may be passed as an argument to the command `linsolve(P,L,U,v)`

$$\left[0, \frac{1}{2}\right]$$

  to solve a system $Ax = b$ by solving two triangular systems (in $O(n^2)$ instead of $O(n^3)$).

- `qr` QR factorization of matrix A, Q is orthogonal and R upper triangular, $A = QR$.
- `svd(A)` singular value decomposition of matrix A returns U orthogonal, S vector of singular values, Q orthogonal such that `A=U*diag(S)*tran(Q)`.
  The ratio of the largest and the smallest singular value of S is the condition number of A relative to the Euclidean norm.

4 Probabilities and statistics

4.1 Random numbers

From F4 catalog, select `Probabilities` then `rand()`

$$0.206383689772$$

(real in $[0, 1)$) or

```
n:=6:; randint(n)
```

$$\text{"Done"}, 5$$

(integer between 1 and n). Other commands with prefix `rand` are available, followed by the name of the law, for example `randbinomial(n,p)` returns a random integer according to binomial law of parameters n, p. For a random vector or matrix, run `ranv` or `ranm` (from `Alglin, Matrice` submenu), for example for a vector with 10 random reals according to normal law (mean 0, stddev 1), type

```
ranv(10,normald,0,1)
```

$$[-1.0292213846, 0.549673205703, 0.353153821095, 1.03060992213, -0.632884555692, 1.03592365468, 0.29980763$$

4.2 Probabilities

From F4 catalog, select `Probabilities` (8). There you will find a few distribution laws: `binomial`, `normald`, `exponentiald` and `uniformd`. Other distribution must be keyed in: `chisquared`, `geometric`, `multinomial`, `studentd`, `fisherd`, `poisson`.

To get the cumulated distribution function, enter the law name then the `_cdf` suffix (shortcut: select `cdf` in the catalog at the end and press F1). Inverse cumulated distribution function follows the same principle with `_icdf` suffix (shortcut: select `cdf` in the catalog and press F2).

Example : find the centered interval I for the normal law of mean 5000, standard deviation 200, such that the probability to be outside I is 5%

```
M:=5000; S:=200; normald_icdf(M,S,0.025);normald_icdf
(M,S,0.975)
```

$$5000, 200, 4608.00720309, 5391.99279691$$

4.3 1-d statistics

The statistic functions are taking lists as arguments,

```
l:=[9,11,6,13,17,10]
```

From F4 catalog, select `Statistics`:

- `mean(l)`

  $$11$$

  : arithmetic mean of a list

- `stddev(l)`

  $$\frac{\sqrt{105}}{3}$$

  : standard deviation of a list.
  Run

  `stddevp(l)`

  $$\sqrt{14}$$

  to get an unbiaised estimate of the standard deviation of a population from a sample `l`

- `median(l)`

  $$10.0$$

  , `quartile1(l)`

  $$9.0$$

  ,
  `quartile3(l)`

  $$13.0$$

  returns respectivly the median, first and third quartile of a list.

For 1-d statistics with frequencies, replace `l` by two lists of the same length, the first list being the values of the serie, the second list the frequencies. For graphic representations, open catalog, `Graphic` and select `histogram` or `barplot`.

4.4 2-d statistics

From F4 catalog, select `Statistics`:

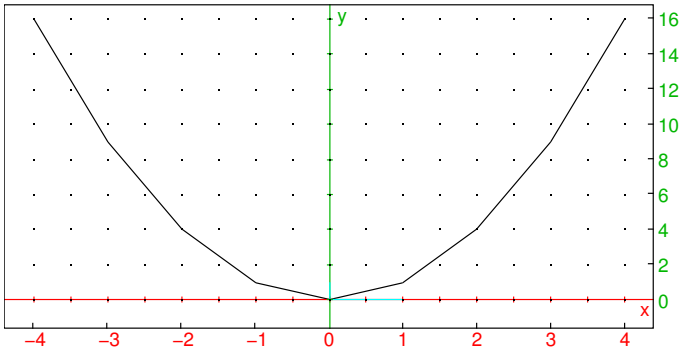
- `correlation(X,Y)` : correlation of two lists X and Y of the same length.
- `covariance(X,Y)` :: covariance of two lists X and Y of the same length.
- regression computations: run commands with suffix `_regression(X,Y)`, for example `linear_regression(X,Y)` returns coefficients m, p of the linear regression line $y = mx + p$.

- `linear_regression_plot(X,Y)` and all commands of suffix `_regression_plot` will display the line (or curve) of the regression. These commands will also print the R^2 coefficient that give information on the quality of adjustment (R^2 near 1 is good).

5 Graphics

From F4 catalog, select `Graphics` (shortcut 7).

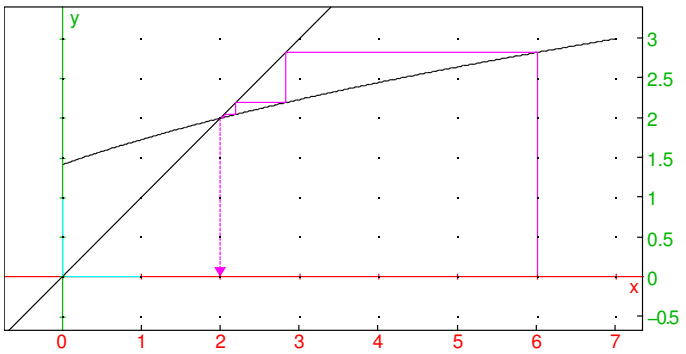
- `plot(f(x),x=a..b)` plot expression $f(x)$ for $x \in [a,b]$. Discretization option: `xstep=`, for example `plot(x^2,x=-4..4,xstep=1)`



Default is 384 evaluations per plot (one per horizontal pixel).

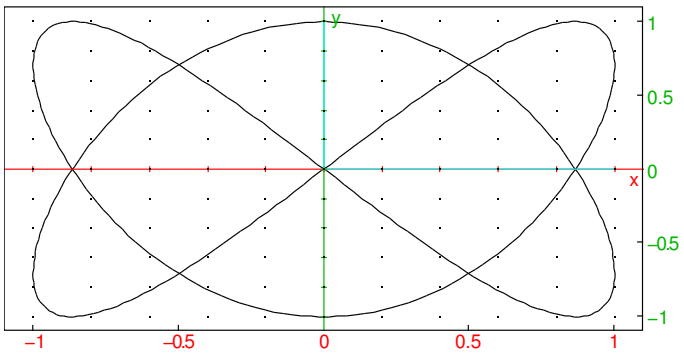
- `plotseq(f(x),x=[u0,a,b])` webplot for a recurrent sequence $u_{n+1} = f(u_n)$ of first term u_0, for example if $u_{n+1} = \sqrt{2+u_n}, u_0 = 6$, with a plot on $[0,7]$

  `plotseq(sqrt(2+x),x=[6,0,7])`



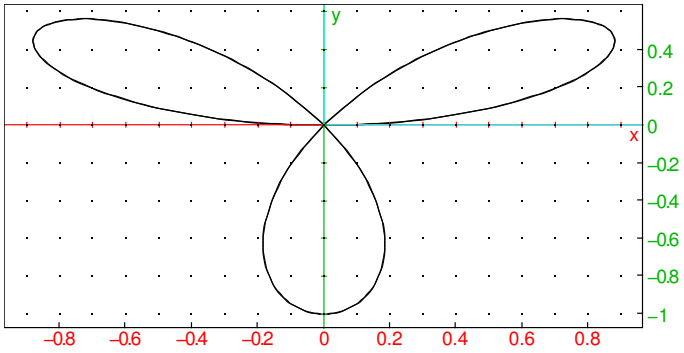
- `plotparam([x(t),y(t)],t=tm..tM)` parametric plot $(x(t), y(t))$ for $t \in [t_m, t_M]$. Discretization option: `tstep=`. Example

  `plotparam([sin(2t),cos(3t)],t,0,2*pi)`



- `plotpolar(r(theta),theta=a..b)` polar plot of $r(\theta)$ for $\theta \in [a, b]$, for example

  `plotpolar(sin(3*theta),theta,0,2*pi)`

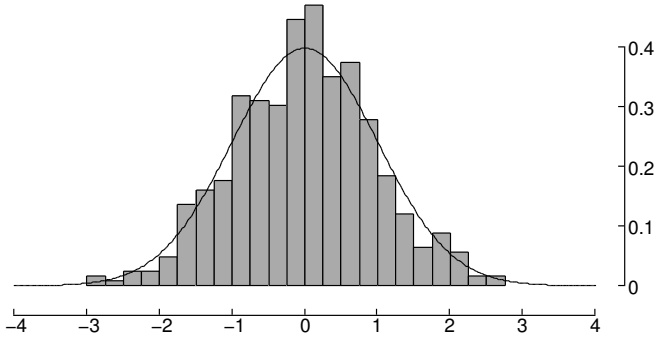


- `plotlist(l)`: plot a list `l`, i.e. draws a polygonal line with vertices (i, l_i) (index i starts at 0).
  `plotlist([X1,Y1],[X2,Y2],...)` polygonal line with vertices the points of coordinates (X_i, Y_i)

- `scatterplot(X,Y)`, `polygonscatterplot(X,Y)` for two lists `X,Y` of the same size, draws the points or a polygonal line of vertices (X_i, Y_i)

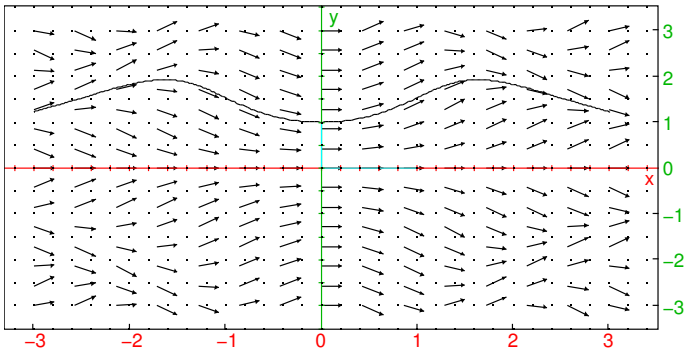
- `histogram(l,class_min,class_size)` plots the histogram of data in `l`, class size `class_size`, first class starts at `class_min`. Example: check the random generator quality

```
l:=ranv(500,normald,0,1); histogram(l,-4,0.25
); plot(normald(x),x,-4,4)
```



- `plotcontour(f(x,y),[x=xmin..xmax,y=ymin..ymax],[l0,l1,...])` plot implicit curves $f(x,y)=l_0, f(x,y)=l_1, ...$
- `plotfield(f(t,y),[t=tmin..tmax,y=ymin..ymax])` plot the field of tangents for the differential equation $y' = f(t,y)$. Add the optional last parameter `,plotode=[t0,y0]` to plot simultaneously the solution with initial condition $y(t_0)=y_0$. Example $y' = \sin(ty)$ for $t \in [-3,3]$ and $y \in [-2,2]$

```
plotfield(sin(t*y),[t=-3..3,y=-3..3],plotode=
[0,1])
```

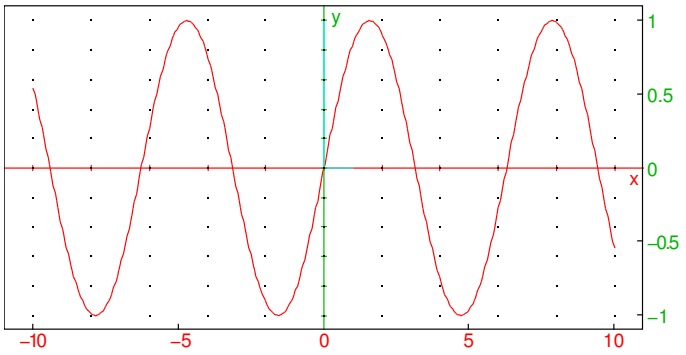


N.B.: `plotode` may be used outside of `plotfield`.

For simultaneous plots, write commands separated by `;`

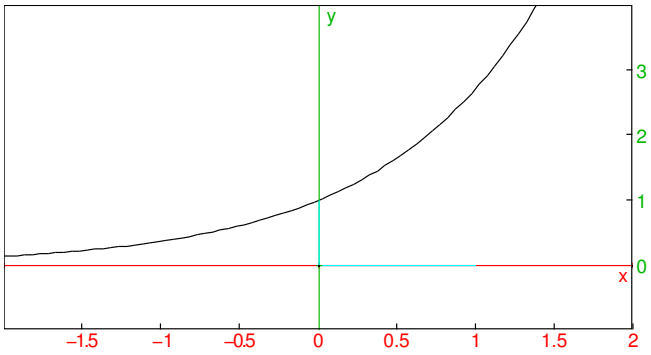
For display options, press the OPTN key:

- `display=color` color option: select a color then press F2, for example `plot(sin(x),display=red)`



- `display=line_width_2` to `display=line_width_8`: change segments width (including inside polygonal line used to plot a curve). Simultaneous display options should be added with +. For example `display=red+line_width_2`
- Circles and rectangles with edges parallel to the coordinate axis may be filled with `display=filled` (this attribute might be added to other attributes)
- If you want to define the display window (overwriting the autoscale computation), select `gl_x` or/and `gl_y` and add an x or y interval, for example

  `gl_x=-2..2;gl_y=-1..4;plot(exp(x))`



  Note that `gl_` commands must preced the plotting command.

- If you want to remove axes, select `axes` and press F2 (`axes=0`). Like `gl_` commands, `axes=0` must preced the plotting command. Axes can be removed interactively when the graph screen is displayed by pressing VARS.

6 Programs

You can program either with Xcas-like syntax (English or French) or with Python-like syntax.

Example : function defined by an algebraic expression `nom_fonction(parametres):=expression` for example simple confidence interval for a frequency p in a sample of size N

```
F(P,N):=[P-1/sqrt(N),P+1/sqrt(N)]
```

$$(P,N)\mapsto \left[\begin{array}{cc} P-\frac{1}{\sqrt{N}} & P+\frac{1}{\sqrt{N}} \end{array}\right]$$

Test `F(0.4,30)`

$$[0.217425814165, 0.582574185835]$$

Second example : more precise confidence interval for a frequency p in a sample of size N:

```
f(P,N):=[P-1.96*sqrt(P*(1-P)/N),P+1.96*sqrt
(P*(1-P)/N)]
```

$$(P,N)\mapsto \left[\begin{array}{cc} P-1.96\sqrt{P\frac{1-P}{N}} & P+1.96\sqrt{P\frac{1-P}{N}} \end{array}\right]$$

To avoid computing twice the same quantity, one can insert a local variable. The commandline is not well adapted to write these kinds of functions. For non algebraic functions, it is best to run the program editor. Press F6, select Script Editor, clear the editor if it is not empty (F6 Clear) and type with the help of test (F1), loop (F2) for programming structures the following program, in Xcas syntax:

```
function f(P,N)   local D;   D:=1.96*sqrt
(P*(1-P)/N);   return [P-D,P+D]; ffunction;
```

$$(P,N)\mapsto \left(\begin{array}{l} \texttt{\{ local D;} \\ \texttt{\ \ D:=1.96*sqrt(P*(1-P)/N);} \\ \texttt{\ \ return([P-D,P+D]);} \\ \texttt{\}} \end{array}\right)$$

or in Python syntax:

```
def f(P,N):  D=1.96*sqrt(P*(1-P)/N)  return
 [P-D,P+D]
```

undef

Type EXE to check the syntax. Once the program is correct, save it (F6 2), then type EXIT. Now you can call your program from the commandline like this
`f(0.5,30)`

Third example : a loop printing integer squares from 1 to n in Python syntax. Check that Python syntax is enabled in the F6 or shift-SETUP menu, if it is not checked, check it. Open F6 Script Editor, if there is some old script source, clear it (F6 Clear). Select `f(x):=` from F2 (or from F4, Program, function def), you should get `def f(x):`.

Replace `x` by `n` (press F5 to lock the keyboard in alpha lowercase), move to the end of the line and press shift-EXE to input a newline. Type Shift-PRGM then `3 for`, then F5 J space alpha, then Shift-PRGM then `6 in range(a,b)`. Type `1,n+1)` then F1 (`:`). Type shift-EXE to insert a newline then Alpha SPACE, F4 (Cmds), EXE (`1 All`), P, R select `print` with the cursor then type EXE, type `j,j^2)` then EXE.

```
def f(n):
  for j in range(1,n+1):
    print(j,j^2)
```

Inside Xcas `^` means power, `**` is also accepted like in Python.

Now, type EXE (or F6, select `1. Check syntax`). If syntax is correct, you will see `Success` in the status line. Otherwise, the first error line number and token will be displayed and cursor will be positionned at the line where the error was detected. Note that the error may be before this line but it was only detected later. Note also that if you are using Python syntax compatibility, programming structures are translated into Xcas, errors are displayed after translation, therefore you might see token errors like `end` that were added by the translator.

If the program is correct, you can save it with the F6 menu (save or save as). You can run it from the commandline by pressing EXIT then for example `f(10)` should display all squares from 1 to 10.

The turtle is a nice way to learn programming. The turtle is a small robot that you can move, it handles a pen that marks its path. Type F6, Script Editor, then F6 Clear. Type shift-QUIT select `efface` which means clear the screen. You can access to the turtle commands using shift-QUIT (move the cursor to a command and press F6 for help). For example try `avance` (forward). Checking the syntax (EXE) will display the turtle window moves. You can enter several moves in your script, and organize them inside tests, loops and functions. For example:

```
function square(n)
  repete(4,avance n,tourne_gauche);
ffunction:;

efface;
for n from 1 to 10 do
  square(10*n);
od;
```

Another example of non algebraic function: the euclidean algorithm to compute the GCD of two integers. Press shift-EXE to insert a newline. `!` is in the submenu `Programmation_cmds` (11, shortcut X, θ, T) or in the test F1 menu. Xcas syntax

```
function pgcd(a,b)
  while b!=0 do
    a,b:=b,irem(a,b);
  od;
  return a;
ffunction
```

Python syntax

```
def pgcd(a,b):
  while b!=0:
    a,b=b,a
  return a
```

Check with `pgcd(12345,3425)`

"Aborted"

If your program has runtime errors or if you want to see it run step by step, run `debug` on it, for example
`debug(pgcd(12345,3425))`

Unlike adaptations of Micro-Python by calculator manufacturers (including Casio), the Python syntax in Xcas is fully integrated. You can therefore use all Xcas commands and data types in your programs. This corresponds approximatively to importing Python modules `math`, `cmath`, `random`, `scipy`, `numpy`, `turtle`, `giacpy`. There is also a small pixelised graphic commands set (`set_pixel(x,y,c)`, `set_pixel()` to synchronize display, `clearscreen()`, `draw_line(x1,y1,x2,y2,c)`, `draw_polygon([[x1,y1],[x2,` `draw_rectangle(x,y,w,h,c)`, `draw_circle(x,y,r,c)`, the color+width+filled `c` parameter is optional, `draw_arc(x,y,rx,ry,t1,t2,c)` draws an ellipsis arc). And you can somewhat replace `matplotlib` with graphic commands of χCAS (`point`, `line`, `segment`, `circle`, `barplot`, `histogram` and all `...plot...` commands). Plus you have natural access to data types like rationnals or expressions, and you can run CAS commands on them. The complete list of commands available on the calculator is given in appendix. For documentation on commands not listed in the catalog categories, please refer to Xcas documentation.

7 The 2d editor.

If a computation returns an expression, it will be displayed in the 2d expression editor. This also happens if you press F3 when the selected level is an expression, or if you press F3 from the commandline if the line is empty or contains a syntaxically correct expression.

Once the 2d editor is open, the expression is displayed in full screen and all or part of the expression is selected. One can run a command on the selection (from the menus or from the keyboard), or edit (in 1d mode) the selection. This is an efficient way to rewrite expressions or edit them.

Example 1 : enter

$$\lim_{x\to 0}\frac{\sin(x)}{x}$$

From an empty commandline, type F3 (view), you should see 0 selected. Type x then EXE, this will replace 0 by x selected. Type SIN, now $\sin(x)$ should be selected. Type the division key (above -), you should see $\frac{\sin(x)}{0}$ with 0 selected, type x then EXE, you should now see $\frac{\sin(x)}{x}$ with x (below the fraction) selected. Type the up arrow key, now

$\frac{\sin(x)}{x}$ should be selected. Now type F2 4 (for limit). The expression is ready to eval, type EXE to copy it to the commandline and EXE again to eval it. For the same limit at $+\infty$, before leaving the 2d editor with EXE, move the selection with the right arrow key, then type F1 8 (oo) EXE.

Example 2 :

$$\int_0^{+\infty} \frac{1}{x^4+1}\,dx$$

From an empty commandline, type F3 (view), then F2 3 (integrate), you should see:

$$\int_0^1 0\,dx$$

with x selected. We must modify the 1 (upper bound) and the 0 (integrand). Press left arrow key, this will select the integrand 0, type `1/(x^4+1)` EXE, then left arrow key F1 8 EXE. Type again EXE to copy to commandline, EXE again to run the computation, the result will be displayed in the 2d editor, EXE will leave the 2d editor, with the integral and its value in the history.

Example 3 : compute and simplify

$$\int \frac{1}{x^4+1}\,dx$$

From an empty commandline, type F3 (view), then F2 3 (integrate), you should see

$$\int_0^1 0\,dx$$

Move the selection to the lower bound 0 (right arrow key), type DEL, you should see

$$\int 0\,dx$$

selected. With the down arrow key, select 0, type `1/(x^4+1)` EXE, EXE copy to the commandline, EXE to run the compuation, the result is now displayed in the 2d editor. With the arrow key, select one of the arctangent, type F1 EXE (simplify), this will make a partial simplification, do the same on the second arctangent.
For a more complete simplification, we will collect the logarithms. The first step is to exchange two terms of the main sum so that the logarithms are grouped. Select one of the logarithm with the arrow keys, then type

- Prizm: shift-left or right arrow key
- Graph 35: F5 left or right arrow key, then ALPHA

this will exchange the selection with the right or left sibling. Now type ALPHA right or left arrow key to extend the selection adding the right or left sibling. Once the two logarithm terms are selected, press F1 2 EXE (factor), decrease the selection to the numerator, type F4 EXE (All), type the letters l, n, c, this moves in the catalog to the first command beginning with `lnc`, select `lncollect`, EXE and F6 (eval).

8 Managing sessions

8.1 Modifying a session

With the up/down cursor keys, you can move in the history, the current level is printed with reverse colors.

You can move one level in another position with ALPHA-up and ALPHA-down. You can delete a level with the DEL key (the level is copied into the clipboad).

You can modify an existing level with F3 or ALPHA-F3. With F3, the 2d editor is called if the level is an expression, with ALPHA-F3 the level is edited in the text (program) editor. Type EXIT if you want to cancel modifications, or EXE if you confirm the modifications. If you confirm the modifications, the commandlines below the current level will automatically be re-evaled. This way, if you modify for example a level like `A:=1`, all levels below that depend on the value of `A` will be up to date. If you want to do that several times, it is best to introduce a parameter with the F6 Parameter wizzard. Then pressing + or - on the `assume(...)` or `parameter` level will modify the value of the parameter (press * or / for faster move).

8.2 Variables

Press VARS to see which variables are assigned to a value. Select a variable name, press EXE to copy it to the commandline, DEL will input the command that erases the variable (confirm with EXE). `restart` will purge all variables at once (press `AC/ON` to clear the history and start a fresh new session). `assume` is a command to make assumptions on a variable, like `assume(x>5)` (`>` can be accessed from the shift-PRGM menu).

8.3 Archiving and exchanging with Xcas

On the calculator, go back to the history (type EXIT if you are in the programming editor or the 2d expression editor). From the F6 menu, you can save/restore sessions in the calculator flash memory. Files have the `xw` extensions. They can be copied to your computer (connect the calc, choose F1 USB key), and there they may be opened with Xcas or Xcas for Firefox. From Xcas, choose the File menu then Open file, then select all type of files and open the session file. From Xcas for Firefox, press the Load button.

Conversely you can save a session from Xcas (choose File, Export to Khicas) or from Xcas for Firefox (choose Export at the right of the session name).

9 Keyboard shortcuts.

- F1 to F3 : depends on mode (Python/Xcas) and shift/alpha state, see labels
- F4: commands catalog.

- F5: uppercase to lowercase switch. If alpha mode is not active, locks the keyboard in alpha lowercase.
- F6: File menu
- `(-)` in the text editor: returns the _ character.
- shift PRGM: programming commands or characters
- OPTN: all options
- shift-QUIT: turtle commands
- shift-List: create or edit a list, list commands
- shift-Mat: create or edit a matrix, matrix commands
- S↔D key: real number commands
- yellow shifted S↔D key: integer commands
- angle key: complex commands
- yellow shifted fraction: plot commands
- fraction key: special characters/proba in history, indentation in text editor
- red r key: `abs`
- red θ key: `arg`

In programming editor

- shift-cursor key: move to begin/end of line or file
- shift CLIP: begin selection. Move the cursor to the selection end, type DEL to remove the selection (it will be copied to clipboard) or again shift-CLIP to copy selection to clipboard without removal. Type AC/ON to cancel selection.
- EXE: if a search/replace is currently active (F6 6) find next word occurence. Otherwise parse/execute.
- shift EXE: add a newline.
- DEL: remove selection or previous character if no selection active
- shift PASTE: copy clipboard
- Shift-INS (touche DEL): remove current line and copy to clipboard
- AC/ON: cancel selection or cancel search/replace or check syntax (like F6 1)
- EXIT: leave text editor to the commandline. Type EXIT again to come back to the text editor.

10 Remarks

This adaptation is not a full version of Xcas because the maximal size for a Casio add-in is too small (2 Mo). There are more complete adaptations of Xcas for calculators, for more informations, refer to Giac/Xcas homepage.

11 Copyright and Thanks to.

- Giac and χCAS, computing kernel (c) B. Parisse et R. De Graeve, 2018.
- χCAS interface adapted by B. Parisse from Eigenmath source code by Gabrial Maia and from Xcas source code.
- χCAS license GPL2. See details in the `LICENSE.GPL2` file, inside khicasio.zip or GPL2 on the Free Software Foundation website. The source code of χCAS and of the required libraries libtommath and USTL are available in the Casio section of my webpage (see section 12).
- Thanks to the active members of tiplanet and Planete Casio for answering questions and testing during the time I developed χCAS. Special thanks to LePhenixNoir (Prizm/35+eii help), Nemhardy (Prizm), and to critor for articles, tests and advertising. Thanks to all contributors of the Prizm programming portal. Thanks to Pavel Demin for compilation tricks that spared about 135K.
- Thanks to Camille Margot for her interest in this ports, and to Casio France for sending me calculators and an emulator license.

12 Developer infos.

To build this add-in, I have installed the gcc cross-compiler for sh3eb CPU following this tutorial (French). I have configured gcc like this
`../gcc-5.3.0/configure --target=sh3eb-elf --prefix="$HOME/opt/sh3eb-elf" --disab`
Unfortunately, there is no support for sh3eb in the `newlib` (C librairy) of gcc, nor for `libstdc++`.

I installed `libfxcg.tar.gz` with a few modification (corrections of small bugs, added missing functions like qsort, ...), it is available in this folder (unarchive and compile with make). In this folder you will also find `tommath.tgz` (big integer support) and `ustl.tar.gz` (standard template library) that I also had to modify to make it work with sh3eb-elf-g++, with partial success, i.e. enough support to build Giac (vector/string/map supported, I/O on files are not supported, there is a custom iostream file for cin/cout minimal support). Unarchive and compile with make.

The file sh3eb-elf.tar.gz is a binary version of gcc/libraries for GNU/Linux debian 9.

The file giac35.tgz is the source of the port of Giac. Run the `mkxcas` script to compile Giac.