Ask Casio to keep 3rd party addins for the FXCG50 successor : Casio introduced in 2024 a new color calculator in France, the Graph Math+, and it's international equivalent (FXCG100) in 2025 as the successor of the FXCG50. **The current OS versions (1 and 2) do not support addins, therefore KhiCAS and other addins are not available on the Graph Math+ and FXCG100**. For these versions, there is a workaround but Casio might further lock the FXCG100.

χCAS(io)

Bernard.Parisse@univ-grenoble-alpes.fr

2018, 2025

Contents

Abstract

This document explains how to run efficiently χCAS on some Casio calculators (CG10, CG20, CG50 and Fx-9750GIII, Fx-9860GIII). χCAS is a port of the Giac/Xcas computer algebra system (CAS) for these calculators.

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

1 Introduction and installation

χCAS is a port of the Giac/Xcas computer algebra system (CAS) for the following Casio calculators : CG10, CG20, CG50 and Fx-9750GIII, Fx-9860GIII. Beware:

- **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.**
- **χCAS is not compatible with exam mode**.

Two versions are available for the CG50, a light version that is the same as on CG10 and CG20 (in one file), and a more complete version in two or three files. The more complete version has more Xcas commands (like geometry commands), a 3d rendering engine, some additional apps (like a formal spreadsheet or a financial application) and a port of MicroPython 1.12 with more modules than the Casio port of MicroPython 1.09, cf. section 11.

1.1 Calculator

To install or update χCAS, get on your computer

- for the FXCG50, get the two files khicas50.ac2 and khicas50.g3a.
  If you are short on storage space, you can instead get khicas5z.g3a, cas50a.mzs cas50b.mzs.
- for the FXCG10 and 20 (works also on the FXCG50 but is less complete) khicasen.g3a
- for the FX-9750GIII, Fx-9860GIII, the file khicasen.g1a
- FXCG100 support: addins are not supported. In order to restore support for addins, a modification of the OS is required, like the MPM does for the French equivalent (Graph Math+). Before installing, check your OS version, KhiCAS works only on version 2. Once addins are again supported, copy the following files mpm.bin, khicas1z.g3a, cas100a.mzs and cas100b.mzs to your fxcg100. To run KhiCAS, from the Casio main menu, press TOOLS, select Khicas1z.
  The keyboard was modified on the FXCG100, there is no more F1-F6 keys and the 6 small menus at the screen bottom weere replaced by 2 larger menus. KhiCAS will stay in the same logic as the fxcg50 (it's also coherent with other ports of KhiCAS), it will still show small menus, a key below a small menu will fire the corresponding action.
  Since F4 was replaced by up arrow key, the corresponding small menu key is empty, F4 commands on the fxcg50 are fired by pressing the CATALOG key (press shift-CATALOG to get the same menu as CATALOG on the fxcg50).
  F6 was replaced by pageup, the corresponding small menu key may either be empty or show ..., meaning that you should press the TOOLS (...) key to fire the small menu action.
  Press shift OFF to leave KhiCAS. There is no more a MENU key, the equivalent is the HOME key, located at the F2 position. For this reason, the short menu from the shell will display quit as first item, HOME EXE will also leave KhiCAS.
  When you leave KhiCAS, it may be required to run a Casio application before running another addin, otherwise a crash may happen, except if you have only KhiCAS as addin.

Connect the USB cable of the calculator, type F1 for USB key connection and copy the file(s) `khicas50.g3a` and `khicas50.ac2` or `khicasen.g3a` or `khicasen.g1a`) on the calculator USB-"key" then disconnect the calculator-key from your computer and wait a few seconds.

If you want to remove χCAS permanently : from the Main Casio Menu (MENU), select the Memory application, then F2 (Storage), select `khicas50.g3a` and `khicas50.ac2` or `khicasen.g3a` or `khicasen.g1a` (cursor keys then F1), then type F6 to delete the selected files.

If you just want to disable χCAS temporarily, for example if you have an exam where CAS is not allowed, connect the calculator to the PC, and rename the `g3a` or `g1a` file with any other extension. Then after your exam, connect the calculator to the PC and rename again with the `g3a` or `g1a` extension.

N.B. : some users have reported crashes with addin support on OS 3.80.0 (03.80.02**0**2 on fx-CG50, 03.80.12**0**2 on fx-CG50AU, 03.80.22**0**2 on Graph 90+E). If you experience some crashes, first upgrade to OS 3.80.1 Windows, Mac. If it still does not work, it is always possible to downgrade to version 3.70. Since downgrading is not allowed with the usual process, you will have to do the following (thanks to critor, TI-Planet) :

1. keep pressed the restart button and the keys F2, 4 and AC
2. release the restart button while keeping F2 4 AC pressed
3. release keys F2 and 4
4. wait one second
5. release the AC key
6. press key 9, wait one second, release key 9
7. keep key * pressed
8. the calculator should start on an "OS ERROR" screen, release the * key
9. run the 3.70 install software

1.2 Emulator

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. Once the transfert is finished, you should see the icon of Xcas in the main menu (a snowflake on the CG10/20/50).

If you want to run the complete version for the CG50, replace the file above by these 2 files khicas50.882 and emucas50.g3a.

Note that χCAS is not compatible with the simulator distributed by Casio on USB keys in some countries, you must run the emulator. Once installed, the Windows version of the emulator can be run under Linux with wine by the following command

```
wine "C:\Program Files (x86)\CASIO\fx-CG Manager PLUS Subscription for fx-CG50se
```

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.

If you are running the 2 files version and see the message "Unable to load ram part", upgrade your calculator OS (version 3.30 or greater).

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

Hint : If you can not find a character on the calculator keyboard, press shift INS.

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 $\rightarrow$), for example `A:=2` does the same as `2=>A`.

The most popular Xcas commands are available from F1 (algebra) and F2 (calculus), from various shortcuts (cf. section 9), or from the cmds (F4) or shift 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)`

$$1.1780972451$$

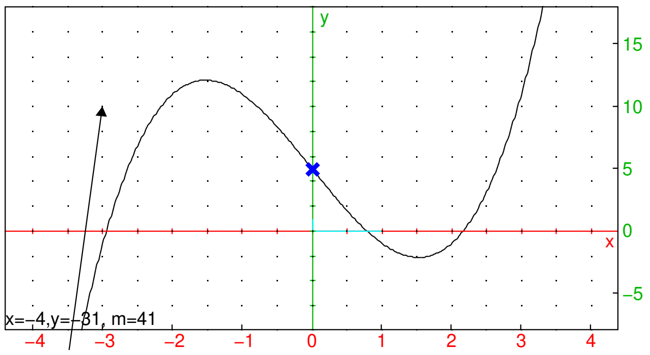
- `limit` : limit of an expression. Example `limit((cos(x)-1)/x^2,x=0)`

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

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

$$\begin{bmatrix} x & -\infty & \text{“”} & -\frac{\sqrt{21}}{3} & \text{“”} & 0 & \text{“”} & \frac{\sqrt{21}}{3} & \text{“”} & +\infty \\ y'=\left(x+\frac{\sqrt{21}}{3}\right)\left(3x-\sqrt{21}\right) & +\infty & \text{“+”} & 0 & \text{“-”} & -7 & \text{“-”} & 0 & \text{“+”} & +\infty \\ y=x^3-7x+5 & -\infty & \text{“}\uparrow\text{”} & \frac{14\sqrt{21}+45}{9} & \text{“}\downarrow\text{”} & 5 & \text{“}\downarrow\text{”} & \frac{-14\sqrt{21}+45}{9} & \text{“}\uparrow\text{”} & +\infty \\ y'' & -\infty & \text{“- ()”} & -2\sqrt{21} & \text{“- ()”} & 0 & \text{“+ ()”} & 2\sqrt{21} & \text{“+ ()”} & +\infty \end{bmatrix}$$

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 $\rightarrow$ 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)
  ```

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

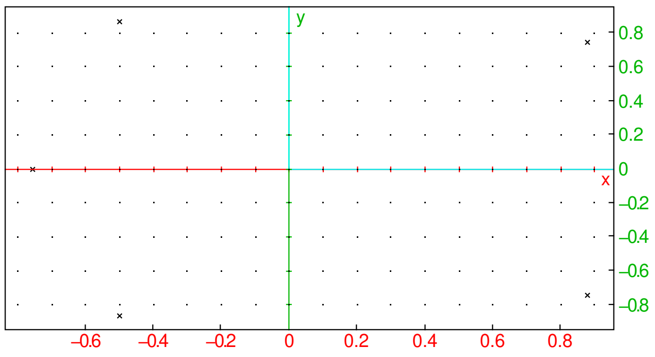
- `proot(P)` : approx. roots of P (all roots, real and complex)

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

  $[-0.754877666247, -0.5 - 0.866025403784\mathrm{i}, -0.5 + 0.866025403784\mathrm{i}, 0.877438833123 - 0.744861766662\mathrm{i}, 0.$

  Graphic representation

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



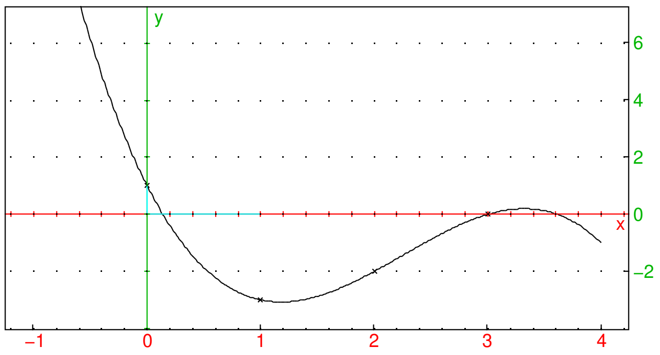
- `interp(X,Y)` : for two lists of the same size, returns the interpolating polynomial P such that $P(X_i) = Y_i$.

  ```
  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

$$A = \left(\begin{array}{cc} 1 & 2 \\ 3 & 4 \end{array}\right)$$

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

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

$$\left[\begin{array}{cc} 1 & 2 \\ 3 & 4 \end{array}\right]$$

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))
```

$$\left(\begin{array}{cc} 1 & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{3} \end{array}\right)$$

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)
```

$$\left(\begin{array}{cccc} 0.0881637986749 & 0.536643749103 & 0.360248812009 & 0.630750506185 \\ 0.0543795586564 & 0.819389336277 & 0.251521021593 & 0.0686232172884 \\ 0.178203014191 & 0.346399624366 & 0.852082391735 & 0.614078260958 \\ 0.714221911505 & 0.784336711746 & 0.453634891659 & 0.433968523517 \end{array}\right)$$

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

$$\left(\begin{array}{cccc} -0.57241508091 & -0.751538700822 & 0.989770363266 & -0.00575141421859 \\ 0.902209004052 & 0.455987131429 & -1.85967614286 & -0.243668831329 \\ -1.61539092837 & 0.558232005133 & -0.159613435564 & 1.35165685556 \\ -0.16425534864 & 1.27150836747 & 0.245690624229 & -0.595841236292 \end{array}\right)$$

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)`

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

  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()`

$$0.38078169385$$

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

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

$$\text{“Done”}, 6$$

(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)
```

$[-2.28261976775, 0.652261860753, -0.690651824321, -0.323190436625, 0.157460267236, -0.324617014178, -0.3$

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

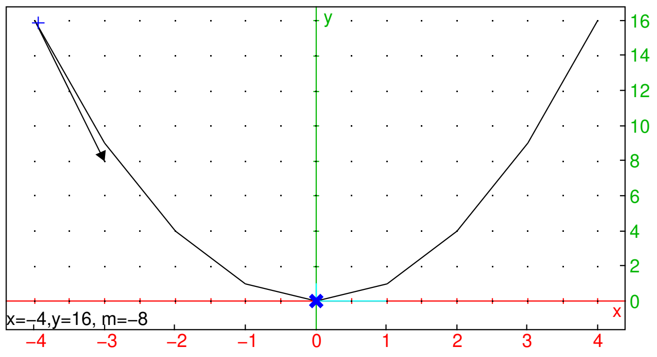
From F4 catalog, select `Statistics`:

- `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` (or type the plot shortcut F3) and choose a graphic command. If you want to have more than one curve on the same graphic, enter several commands with `;` as separator.

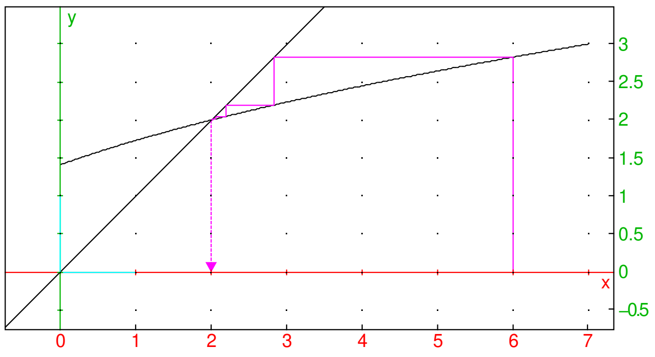
- `plot(f(x),x=a..b)` plot expression $f(x)$ for $x \in [a, b]$. Discretization option: `xstep=`, for example `plot(x^2,x=-4..4,xstep=1)`



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

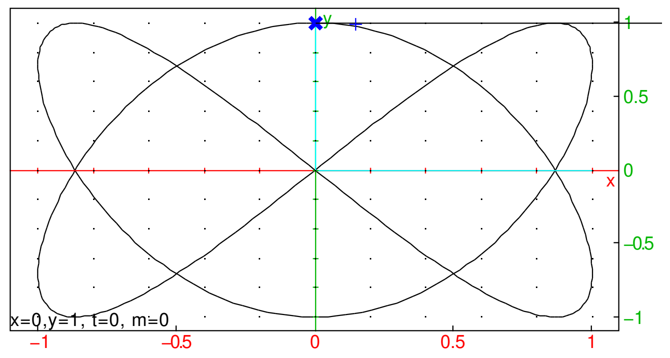
- `plotseq(f(x),x=[u0,a,b])` webplot for a recurrent sequence $u_{n+1} = f(u_n)$ of first term u_0, for example if $u_{n+1} = \sqrt{2+u_n}, u_0 = 6$, with a plot on $[0, 7]$

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



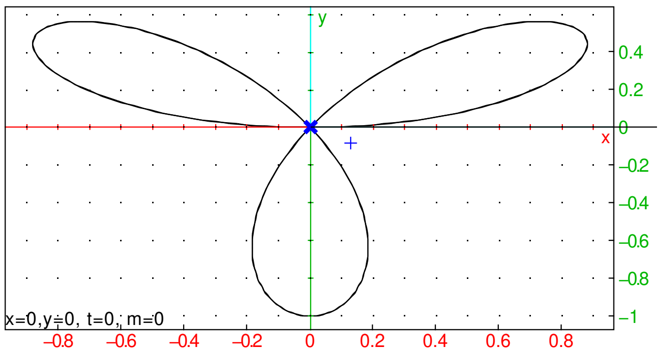
- `plotparam([x(t),y(t)],t=tm..tM)` parametric plot $(x(t), y(t))$ for $t \in [t_m, t_M]$. Discretization option: `tstep=`. Example

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



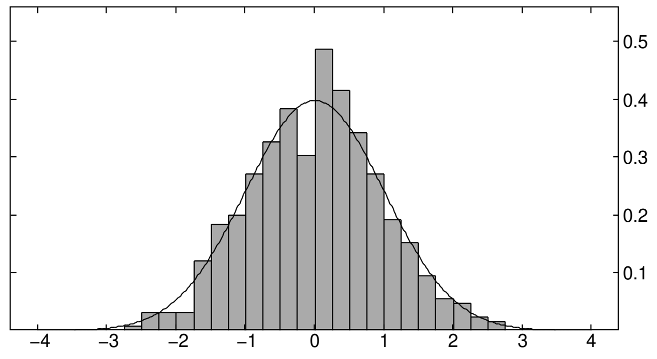
- `plotpolar(r(theta),theta=a..b)` polar plot of $r(\theta)$ for $\theta \in [a, b]$, for example

  `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

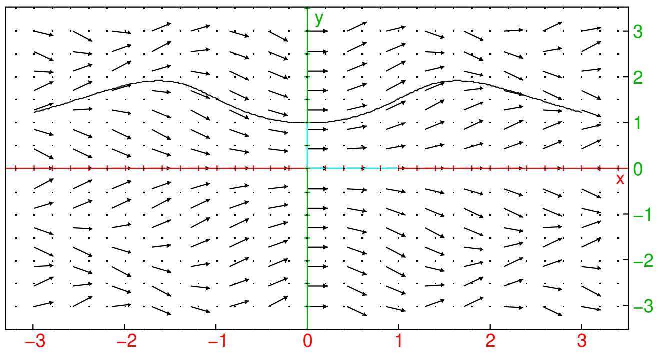
```
l:=ranv(500,normald,0,1); histogram(l,-4,0.25
); plot(normald(x),x,-4,4)
```



- `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 `;`

If a command returns an arc of curve, the trace mode is active by default on the addin in 2 parts for the FXCG50 or on the addin for the monochrom model. In this mode, typing the left or right key will move a pointer on the active arc of curve, and display the pointer coordinates (and the parameter value for parametric curves), and a tangent vector (speed). If there are several curves, press up or down to change active curve. Type F2 to get info on the active curve, then EXE to see a table of value.

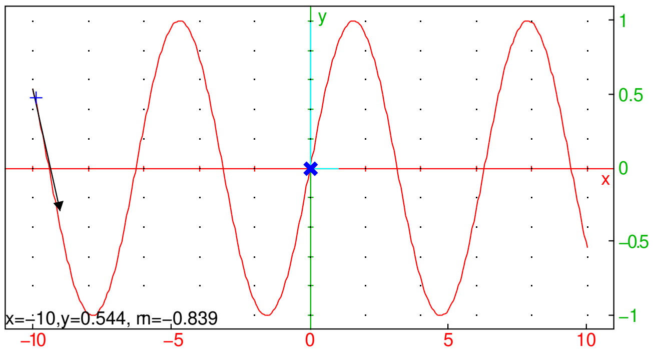
From F1 menu, if you select the curve study item (shortcut x, θ, t key), you can add a normal vector pointing to the center of curvature (F4), and/or the osculating circle and radius of curvature. From the curve study menu, you can also move the pointer to a location, or to a remarkable point: root, horizontal or vertical tangent, inflexion point, intersection with another curve arc (limited to intersection of function graphs on monochrom Casio). You can also compute an arc length or the area under the curve between pointer and mark.

When a curve is active, the variables `X0,X1,X2,Y0,Y1,Y2` are set with the expressions of position, speed and acceleration. When you search a root, horizontal tangent, inflexion, arc length or area under curve, variables are set with the last value.

For display options, press the OPTN key:

- `display=color` color option: select a color then press F2, for example

  `plot(sin(x),display=red)`

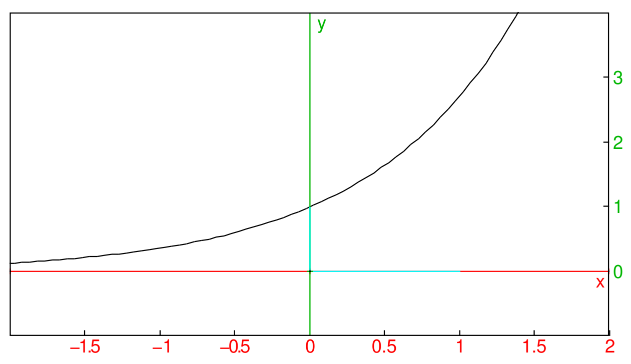


x=–10,y=0.544, m=–0.839

- `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 SIN.

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;
```

```
         ⎛ { local D;                        ⎞
(P,N) ↦  ⎜   D:=1.96*sqrt(P*(1-P)/N);        ⎟
         ⎜   return([P-D,P+D]);              ⎟
         ⎝ }                                 ⎠
```

or in Python syntax:

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

"Done"

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

- CG10,20,50 : shift-left or right arrow key
- fx-9860GIII: 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.

9.1 KhiCAS 50 and 90 versions (2 files)

These shortcuts are valid inside the shell and text programming editor. With default configuration:

- shift INS: table of ASCII characters
- F1-F6, shift-F1 to shift-F6, alpha-F1 to alpha-F6: see legend at screen bottom
- OPTN: fast menu for color options
- shift-OPTN: programming commands
- VARS: variables list
- shift-PRGM: fast menu for programming
- MENU: back to main Casio menu
- shift-SETUP: setup
- EXIT: switch from shell to editor
- shift-EXIT: display logo turtle screen
- ALPHA-EXIT: if alpha is not locked, displays last 2d or 3d graph
- shift angle: fast menu for geometry
- fraction: `%` inside shell, indentation in editor, force sheet reeval in spreadsheet
- shift-fraction: fast menu for poynomial arithmetic in shell, completion in editor
- touche S$\leftrightarrow$D: additional apps (spreadsheet, finance, ...). Inside spreadsheet, display sheet graphs.
- shift `,`: `;`
- shift $\rightarrow$: `:=` or `:` depends on active interpreter Xcas or Python
- AC/ON: cancel selection or cancel search/replace
- shift CAPTURE: save session or file
- shift CLIP: begin selection or copy selection to clipboard
- shift PASTE: paste clipboard
- shift CATALOG: list of all Xcas commands
- shift FORMAT: programming commands
- shift 6: fast menu with <>_! and `comb, rand, binomial,`normaldl

- shift List: fast list menu
- shift Mat: fast matrix menu
- shift 3: menu rapide algèbre linéaire
- shift EXE: next line in editor

You can modify fast menus shortcuts by editing the file `FMENU.py`. Delete the file from Memory Casio application to reset to default configuration.

9.2 KhiCAS short version (1 file))

- 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
- cursor down from shell or shift fraction key from program editor (G key): completion/online help
- `(-)` 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

On color models, you must first press MENU before shutting down the calculator with shift ON. When you press ON again, press MENU to go back in KhiCAS.

If you connect a color model to your PC as a USB disk, you will have to press a key after you have pressed F1, otherwise nothing happens.

If KhiCAS is inside a long computation, you should be able to interrupt it by pressing AC/ON. If it does not interrupt, you may have to press the reset button.

If KhiCAS crashes, you will see a message SYSTEM ERROR etc.. Try to press the MENU key and open any other application. If you are lucky, this will save your session and you can go back and reopen KhiCAS without having to reinitialize the calculator.

The memory available for computations is about 500K with the color g3a addin on the CG50, and 58K on the monochrom g1a addin. On the CG50, it is recommended to check periodically the remaining free memory by pressing the VAR key. If it is less than 100K, press MENU, open any other app, press MENU again and reopen KhiCAS with your session and a fresh unfragmented memory.

11 More complete version for the CG50

The light version in one file is not a full version of Xcas because the maximal size for a Casio add-in is too small (2 Mo). A more complete version of χCAS for the FXCG50 is distributed in 2 files (one of them is run from a section of the RAM of the FXCG50 that is currently not used by Casio).

This more complete version has more Xcas commands (like geometry commands), a 3d rendering engine, some additional apps (like a formal spreadsheet or a financial application) and a port of MicroPython 1.12 with more modules than the Casio port of MicroPython 1.09.

See section 1 to install.

11.1 MicroPython 1.12

You can select the shell interpreter by typing shift SETUP. The menu background color of the shell reflects the active interpreter (yellow=MicroPython, magenta or cyan for Xcas native or Xcas Python compatible).

Available modules: turtle (more complete version, with filled objects), graphic (more complete than casioplot), matplotl, arit (integer arithmetic), linalg/numpy (linear algebra, matrices), ulab (scipy compatibility), cas (CAS from Python).

11.2 3d and 4d graphs

One can plot 2 variables function or parametric plots, cones, solids, planes, ... For example type `F4 * 5 F2 EXE` to draw a cube or `shift F4`1| to select the plot command, and enter `x^2-y^2`.

For fonctions plots from $\mathbb{C}$ to $\mathbb{C}$, run the `plot` command with argument a complex valued expression of 2 variables (real and imaginary parts) like for example `plot((x+i*y)^2-9).` For expression depending directly on the complex variable (without requiring real/imaginary part), one can use the `plot3d` command, for example `plot3d(x^2-9)` (the `plot(x^2-9)` command would not work, because it is already used for a graph from $\mathbb{R}$ to $\mathbb{R}$). The modulus is represented along the Oz axis and the argument using raimbow colors: from $-\pi$ in blue magenta to 0 in green (through yellow orange) and from 0 to π via cyan.

If you need precise options, run the `plotfunc` command, for example
`plotfunc((x+i*y)^3-1,[x=-2..2,y=-2..2],nstep=500)`
will plot $z \to z^3 - 1$ from a square in the complex plane centered at origin, size 4 with a 500 small reectangles discretization.

Beware, the 3d rendering engine is slow on the calculator, therefore the drawing precision is set to a medium value by default (this impacts mostly objects with angles, like polyhedrons). You can modify the rendering precision with F2 (faster, less precise) or F3 (slower, more precise). Tip : you can increase the CPU speed by running the Ptune3 addin.

If you just want one time a higher precision rendering, type ^ and be patient. If you want to interrupt this rendering, press DEL. Use the cursor keys to change the viewpoint, 5 to reset to default viewpoint, and - or + to zoom in/out. While a cursor key is kept pressed, the precision is lower, when the key is released the last position is redrawn with the default precision. Type F4 to show/hide a second hidden objet, F5 to show/hide intermediate points, F6 to show/hide polyhedron edges.

11.3 Interactive geometry application

The geometry application lets you construct figures in the 2d plane or in 3d space. It is possible to move one point of the figure and observe how the construction evolves, illustrating some properties (dynamic geometry). Pure geometric constructions may be mixed with function graphs and other analytic constructions. This application has two view: the graphic view where the figure is rendered and the symbolic view where you see the Xcas instructions to construct the figure. The philosophy is similar to Geogebra, but with Xcas commands instead.

You will find here a short description of this application, cf. here for a more complete documentation of the geometry application, with a few screenshots.

11.3.1 Modes, graphical and symbolic view

Type F6 1 (or S↔D) to display the list of additional applications, then EXE then select an existing figure or create a new 2d or 3d figure. You may also open the geometry application from an existing graph displayed from the shell (for example after running `plot(sin(x)))` by typing F6 then Save figure.

At startup, you are in graphical view in frame mode. The cursor key will modify the viewpoint (move the frame in 2d, rotate viewpoint in 3d). Press F4 to change mode. Type EXE to switch to symbolic view or back. For example, type F4 3 to enter point mode, move the pointer and press EXE where you want to create a point. Or type F4 5 to enter triangle mode move the pointer to each vertex and press EXE on each vertex. You can move the pointer with the keypad (use shift cursor key for a fast move), or if there is an existing point, type the name (e.g. type ALPHA A to move the pointer to the point A if it has already been defined).

In a 3d figure, objects will be created in the yellow plane. Press 4 or 6 to move this plane. It is recommended to keep a viewpoint with Oz vertical (therefore change viewpoint only with the right and left cursor keys that make a rotation of axis Oz).

Dynamic geometry howto: switch to pointer mode (F4 2), move near an exising point and select it with EXE, then move the point with the cursor keys, you will see how the whole figure depends on that point, this helps conjectures or illustration of some geometric properties, like the fact that 3 lines of a triangle have a common intersection.

If you type EXIT in the graphical view, it will reset mode to frame mode if you were not in frame mode, or it will switch to symbolic view if you were in frame mode. In the symbolic view, you can modifiy existing commands or create new geometric objects with new commandlines (one line per object). You can save the construction in text format from F6 menu, note that the file generated will have a `.py` extension despite the fact that it is not a Python script. Type EXIT again to leave the geometry application.

When leaving the geometry application, the figure is saved in an Xcas variable that has the same identifier than the filename displayed in the symbolic view. You can erase this Xcas variable if you want to clear the figure.

Example : circumcircle.
From KhiCAS shell, type F6 1 then select new figure 2d EXE. Type F4 5 Triangle, EXE to create the first vertex, move the pointer with cursor keys and type EXE for the second vertex, move the pointer again and type EXE to create the 3rd vertex and the triangle.

Long version, construction of the center: Type F4 7, select 8 Perpen bisector, move the pointer so that only one edge of the triangle is selected (this is displayed at the bottom right, something like `perpen_bisector D5,D`, type EXE will create the perpendicular bisector. Move the pointer to another edge of the triangle, type EXE to create the 2nd bisector. You may optionnaly create the 3rd bisector. Then type F4 6, select 4 Single intersection. Move the pointer to a perpen bisector, type EXE, move the pointer to another perpen bisector and type EXE. This will create the circumcircle center. Type F4 4, move the pointer to the center (with the cursor keys or by typing ALPHA H or the center name if it is not H), then EXE, move to one of the triangle vertex and press EXE.

Short version with the `circumcircle` command: Type F4 9, select circumcircle, select each vertex with pointer move + EXE ((ALPHA A EXE ALPHA B EXE ALPHA C EXE, replace A, B, C with the vertices names).

Symbolic view version: If you are in the graphical view, type EXIT to move to the symbolic view. Move to the script end and add a newline if required (shift EXE). Enter `c:=circonscrit(A,B,C)` EXE

3d Example

Type F6 1 from the shell, then select new 3d figure. Then EXIT or EXE to switch to the symbolic view. Then F5 c ALPHA shift = F4 uparrow twice, select 3D, EXE 5 for cube, F6 for help on the cube command, press F2 to copy+paste the first example in the symbolic view. You should see
`c=cube([0,0,0],[1,0,0],[0,1,0])`
Type EXE, this display a small cube, type + a few times to zoom in. Then EXE to switch back to symbolic view. Type shift EXE to begin a new commandline. Now we define the vertices of the cube with `A,B,C,D,E,F,G,H=` (type ALPHA A , ALPHA B etc.). Then type F4 and uparrow 3 times to select Geometry then uparrow 4 times to select `sommets` EXE, put c as argument to get `sommets(c)`. Type EXE to display the cube and the vertices with their names. Type EXE again to go back to symbolic view. Then shift EXE to enter a new commandline, that will define the plane ABC. Type ALPHA P = then F2 to open the fast menu lines and 8, this will copy `plane(`. This command takes 3 points as argument (or a cartesian equation), here A, B, G, `P=plan(A,B,G,`. We now add a color to the plane with the F3 disp fast menu `display=filled+green`. Check by EXE EXE. Go to the next line (shift EXE) and create segment DE ALPHA S = F2 select segment command with EXE, type D, E, then F3 and give a color
`S=segment(D,E,color=cyan)` The whole construction should be

```
c=cube([0,0,0],[1,0,0],[0,1,0])
A,B,C,D,E,F,G,H=sommets(c)
P=plan(A,B,G,display=filled+green)
S=segment(D,E,display=cyan)
```

Type EXE to display it and use the keypad to change the viewpoint. Type EXE or EXIT to go back to symbolic view and EXIT to leave the geometry application. Type F1 to save the figure if you did not save it from the F6 menu. You can access analytic geometry information from KhiCAS shell, for example `equation(P)` (F4 select Geometry submenu) will return the cartesian equation of the plane P, or `is_orthogonal(P,S)` (F4 Geometry) will confirm that the plane P is orthogonal to the segment S (this should be apparent from 3d rendering).

11.3.2 Cursors

Cursors are parametric values that live in a given inteval and may be moved by 1% steps from the graphical view. They are created with the `element` command in the symbolic view or from F4 menu in the graphical view.

Example : quadratic explorer
This example demonstrates how the curve of the parabola of equation $y = ax^2 + bx + c$ depends on the value of a, b, c. Create 3 cursors from F4 menu of the graphical view (for each cursor F4 uparrow 4 times EXE EXE). In the symbolic view you should have something like

```
a:=element(-1..1)
b:=element(0..1,0.5)
c:=element(-1..1)
```

then add the parabole graph, from graphical view, type F4 0 (for 10 Curves) and select plot, or from the symbolic view shift EXE shift F6 and select plot fill inside the parenthesis with `a*x^2+b*x+c` (beware, do not forget the $\star$), then EXE. From the graphical view, you should see 3 cursors `a`, `b` and `c` and the corresponding graph. You can now modify the value of a, b, c and see how it affects the shape of the parabola. Type F4 2 (pointer mode), then F5 a (or b or c) EXE to select a (or (b or c). Type EXE then the right and left arrow keys, EXE again to stop moving the cursor.

A lot of variations may be done, some of them simpler with one or two cursors and a curve depending on one or two parameters. For example a line equation explorer with `line(y=a*x+b)` or a trigonometic explorer with `plot(sin(a*x+b))`.

11.3.3 Measures and legends

From the graphical view, type F4 and select 13 Measure. You can compute and display a measure at some point of the figure. For example after creating a triangle, one can display the perimeter of the triangle or its area. Type F4 then uparrow twice EXE. Move the pointer near the triangle, type EXE, move the pointer where you want to display the measure and type EXE.

From the symbolic view, you can display a legend with the `legende()` command. The first argyment of legend may be a point of the figure, or a vector of 2 integers giving the absolute position in pixels (measured from the top left corner). The second argument is the legend, it may be a string or any expression.

If the legend is a numeric value, it can be used as a numeric parameter for commands that require such an argument, like transformations (angle of rotation, homothety ratio...)

```
r:=legend([20,40],"2")
homothety(A,extract_measure(r),B)
```

11.3.4 Traces

The `trace()` command lets you keep track of all the positions of a geometric object when the figure is recomputed while moving a point

Example Enveloppe of the normals to a parametric curve (here an ellipse)

```
E:=plotparam([cos(t),2*sin(t)],t=-pi..pi)
a:=element(-pi..pi)
M:=element(E,a)
T:=tangent(M)
N:=perpendiculaire(M,T)
trace(N)
```

If you change the value of a (F4 2 for pointer mode, F5 a), you will see a curve that separates the area of normals to the curve and a free area, this is the enveloppe of normals, it is also the evolute of the ellipse
`evolute(E,color=red)`
You can remove the traces in the graphical view from the F6 menu (last item).

11.4 CAS spreadsheet...

The file menu has an application item that lets you select additional applications: a formal spreadsheet, a finance app, a periodic table, ... Shortcut: type S↔D from the shell or programming editor.

Unlike Casio spreadsheet, the CAS spreadsheet can handle exact or symbolic values. You can compute cells whose values are fractions, square roots or expressions containing variables like x, y....

A cell can contain any valid Xcas value, numbers, strings, etc. If you enter a list of values, or an Xcas command returning a list of values, the list will fill consecutives cells (downwards or to the right, according to the setup). For example type F1 `range(` 10 EXE, this will fill 10 cells with numbers from 0 to 9.

Defining a cell content with reference to other cells is similar to other spreadsheet, begin with =, and enter an expression that may contain cell references (characters `:` and `$` are available from F3 menu, `:` is also accessible with shift →). While editing the cell content, you can select another cell by pressing the up or down cursor key followed by any other cursor key. To select a range, move to the begin of selection cell, press shift-CLIP then move to the end of selection and type EXE.

While defining a cell, any Xcas commands may be used (you can get them from F4 menu, or fast menus (F1, F2, shift F1-shift F6, alpha F1-alpha F6 or shift CATALOG). Programming Xcas structures may also be used as well as Xcas functions that you have defined. Beware that MicroPython functions are not supported.

A cell can be defined with a command returning a graphic result. Type the S↔D key to display the graphic corresponding to the graphical output of the whole spreadsheet.

12 Copyright and Thanks to.

- Giac and χCAS, computing kernel (c) B. Parisse et R. De Graeve, 2022.
- 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 13).
- Parts of the code is under MIT license
  - MicroPython 1.12 (c) Damien P. George
  - QR code generator, project Nayuki
- 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.

13 Developer infos.

13.1 Giac

13.1.1 Linux install with a script

Get linux_compile_khicas.sh and run `bash linux_compile_khicas.sh`
Thanks to HexaCalc who checked, authored this script and allowed me to include it here. Here is what the script does :

```
# Make sure that the terminal is opened in the home directory

# Update system and install dependencies
sudo apt update
sudo apt upgrade -y
sudo apt install -y cmake zlib1g-dev libpng-dev wget tar

# Install libmpfr.so.4
wget https://www-fourier.univ-grenoble-alpes.fr/~parisse/casio/libmpfr.so.4
sudo cp libmpfr.so.4 /usr/local/lib/
echo "/usr/local/lib" | sudo tee -a /etc/ld.so.conf
sudo ldconfig
rm -f libmpfr.so.4

# Install casiolocal toolchain
wget https://www-fourier.univ-grenoble-alpes.fr/~parisse/casio/casiolocal.tgz
tar xzf casiolocal.tgz
rm -f casiolocal.tgz
```

```
# Add toolchain to PATH (replace 'username' with your actual username)
echo 'export PATH="/home/username/opt/sh3eb-elf/bin:$PATH"' >> ~/.bashrc && sour

# Install mkg3a
wget https://www-fourier.ujf-grenoble.fr/~parisse/casio/mkg3a.tgz
tar xzf mkg3a.tgz
cd mkg3a/build
cmake ..
make
sudo make install
cd ../..
rm -rf mkg3a mkg3a.tgz

# Install libfxcg
wget https://www-fourier.ujf-grenoble.fr/~parisse/casio/libfxcg.tgz
tar xzf libfxcg.tgz
cd libfxcg
make install
cd ..
rm -rf libfxcg libfxcg.tgz

# Download giacbf
wget https://www-fourier.ujf-grenoble.fr/~parisse/casio/giacbf.tgz
tar xzf giacbf.tgz
cd giacbf

# Modify Makefile (replace 'username' with your actual username)
sed -i 's/parisse/username/g' Makefile
# Optional: Remove Wine-related commands if you don't have Wine installed
# sed -i '/\/bin\/cp/d;/\.wine\/drive_c/d' Makefile

# Compile the SDK
make
```

13.1.2 Linux install by hand

Get libmpfr.so.4 and copy it to `/usr/local/lib`, check that `/usr/local/lib` is in the paths listed in `/etc/ld.so.conf` and run `sudo ldconfig`. Now unarchive casiolocal.tgz, this is a working cross-gcc for Casio calculators with additional libraries (libc, ustl, tommath). You will also need to install mkg3a to build addins for the FXCG. Then get giac2.tgz or giacbf.tgz, unarchive and run `make`. For the light version get giac90.tgz

If something goes wrong, here are some details. You must install 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++`.

The libc is replaced by libfxcg (for CG50) (it comes from the original SDK with a few modifications, corrections of small bugs, added missing functions like qsort, ...), In the same 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.

For the monochrom Fx-9860GIII, get giac35.tar.bz2 and run `make` in the `giac35/src0` directory.

13.2 Debugger

Install the cross-gdb tool for the Casio like this

1. install wine
2. install Casio emulator (available from Casio site) with a command like this
   `wine /path_to/fx-CG_Manager_PLUS_Subscription_for_fx-CG50series_Ver.3.40.exe`
3. install gdb-server equivalent

   ```
   cd .wine/drive_c/Program\ Files\ \(x86\)/CASIO/fx-CG\ Manager\ PLUS\ Subscri
   mv CPU73050.dll CPU73050.real.dll
   wget https://www-fourier.univ-grenoble-alpes.fr/~parisse/casio/CPU73050.dll
   cp CPU73050.dll CPU73050.dbg.dll
   ```

   source and other releases for the dll : `https://github.com/redoste/fx-CG50_Manager_PLUS-gdbs`
4. compile gdb for sh3:

   ```
   wget https://ftp.gnu.org/gnu/gdb/gdb-11.1.tar.gz
   cd casio (ou autre repertoire de build)
   tar xvfz ../gdb-11.1.tar.gz
   mkdir sh3eb-gdb
   cd sh3eb-gdb
   ../gdb-11.1/configure --srcdir=../gdb-11.1 --target=sh3eb-elf
   ```

   (add –prefix=path if you do not have write access to /usr/local/bin). Then

   ```
   make
   sudo make install
   ```

   Create two scripts, `casioemu` for normal emulation

```
#! /bin/bash
cd ~/.wine/drive_c/Program\ Files\ \(x86\)/CASIO/fx-CG\ Manager\ PLUS\ Subsc
/bin/cp CPU73050.real.dll CPU73050.dll
cd
wine "C:\Program Files (x86)\CASIO\fx-CG Manager PLUS Subscription for fx-CG
```

and `casiodbg` for debug mode

```
#! /bin/bash
cd ~/.wine/drive_c/Program\ Files\ \(x86\)/CASIO/fx-CG\ Manager\ PLUS\ Subsc
/bin/cp CPU73050.dbg.dll CPU73050.dll
cd
wine "C:\Program Files (x86)\CASIO\fx-CG Manager PLUS Subscription for fx-CG
```

5. For normal emulation run `casioemu`, for debug emulation run

```
casiodbg
sh3eb-elf-gdb
target remote localhost:31188
```

Or in one command if your debug infos are in emucas.elf
`sh3eb-elf-gdb -i=mi -ex "target remote localhost:31188" emucas.elf`

You can set a breakpoint with `b` or `hb` (`hbreak` (hardware break, this is required for the first breakpoint).