χCAS for TI Nspire

Bernard.Parisse@ujf-grenoble.fr

Contents

1 Introduction

This document explains how to get started and use it effectively on TI Nspire CX calculators the computer algebra system χCAS (an adapted version of the Xcas software for this calculator) as well as the most complete to date to do math on calculators (section 13).

χCAS is a formal graphing calculator independent of the Nspire calculator with features for students who plan to continue their studies in science or math: wide variety of graphic representations, analytical geometry tick, arithmetic and cryptography (finite prime fields and extensions, polynomials, etc.), algebra, linear algebra, numerical analysis, multi-precision floating point calculus and certified (interval arithmetic), etc.

All these features are integrated, we can represent on the same graph a histogram and a function graph and a line, we can write a program to make a simulation and represent the results graphically, or use the functions mathematical terms in a Python syntax program. In addition the sessions of calculation are compatible with Xcas, Xcas for Firefox and with other calculators χCAS compatible (Numworks, Casio Graph 90 and 35eii).

Here we will spend some time learning how to use the shell and the out they edit and then make them work harmoniously together, without other limits that the memory and speed of the TI (quite reasonable for a calculation). It is therefore strongly recommended that you read this documentation for a useful optimal use of χcas, with the exception of section 18 which is intended only for programmers who want to program their calculator in C or C ++.

NB: This document is interactive, you can modify the orders and see the result of the execution of the commands proposed as an example by clicking on the button exe (or by validating with the Enter key).

2 Installation

Before installing χCAS, you will need to install ndless by following one of the tutorials following:

- Tutorial if you have an Nspire CX or CX CAS with OS 4.5.0 or lower.
- Tutorial if you have an Nspire CX or CX CAS with an OS 4.5.x (x between1 and 3)
- Tutorial if you own an Nspire CX II or CX CAS II

Then open the IT Nspire communications software and transfer

- luagiac.luax.tns and khicaslua.tns (pretty print interface)
- khicas.tns (shell interface) - possibly the Xcas examples directory.

Examinations:

- Please note that some competitions or exams prohibit the use of CAS calculators. It is the user's responsibility to verify that the formal calculators are allowed before using χCAS in an exam or competition. The authors cannot be held responsible for any use unauthorized.
- χCAS is currently not compatible with the exam mode.

Caution :

Texas Instruments does not seem to want to leave its users free to use ndless programs, and does everything possible to make ndless unusable each time update (usually under the pretext of “security”). If you update to From October 2020, you run the risk of no longer being able to use ndless so no longer be able to use χCAS. I strongly advise you never to bet update your TI Nspire without having inquired about the tiplanet site.

3 "pretty print" interface

This interface opens a calculation screen similar to that of the Scratchpad of the TI Nspire (A Main menu calculations). To be preferred if you mainly want to do calculations taking advantage of the 2d entry of expressions. If you are not on the screen home of the Nspire, press the ON / HOME key. Then type 2, then select khicaslua.

The most used Xcas commands are listed from the menu key. We can access the shell (section 4) by typing * on an empty line, and to the script editor (section 12) by typing + or + "filename" (put the name of the script without .py.tns extension).

4 Shell interface: first steps

If you are not on the Nspire home screen, tap the ON / HOME key. Then type 2, then select khicas and hit enter (you can also go in the Xcas directory and select an example then enter to open a example). This opens after ten seconds a “shell” in which you can type the formal calculation commands of Xcas. On the first run you will have to choose between the Xcas interpreter and the MicroPython interpreter. Type enter to choose Xcas (unless you are mainly interested in the Python environment, cf. section 13).

To quit χCAS and return to the explorer of the Nspire, you must type menu several times (twice from the shell).
For example, type 1/2 + 1/6 then enter, you should see the result 2/3 displayed on the line below.

You can copy a command from the history to the command line by using cursor up or down then enter, then you can change the command and execute it. For example, press the cursor up key, enter and replace 1/6 with 1/3.

You can use the result of the last command with the Ctrl-Ans key calculator (years in color above the (-) key). It is usually better define a variable as the result of a command if you want to reuse it. For this, we use one of the two assignment instructions:

- assignment to the right => is obtained with the calculator's sto → key(Ctrl then var), for example 2 => A puts 2 in variable A. You can enter then use A in a calculation, its value will be replaced by 2.

- assignment to the left =. For example A = 2 does the same as 2 => A.

To help you enter the most useful Xcas commands, χCAS has a catalog of a hundred orders, with a short description and the most frequently wind an example of execution easy to copy. Press the doc key, choose a category with the cursor, for example Algebra, type enter, then choose a command with the cursor, for example factor. A second press on the key doc shows you a short description of the command, usually with an example. By pressing tab (or enter), you copy the example from the command line. You can then validate (enter) or modify the command and validate (enter) to factorize a other polynomial than the one given as an example.

When a command returns an expression, it is displayed in na- write turelle (2-d display). You can scroll the display with the cursor keys when the expression is large. Hit esc to return to the shell.

Now try typing the plot (sin (x)) command. Indication: type doc, then select Curves, or shift-3.

When a command returns a graph, it is displayed. You can modify the graphic display window with the + or - keys (zoom in or out), the cursor keys, orthonormalize the marker (/ key) or search automatic of the scale (autoscale key *). To remove or replace the axes and graduations, tap var. Hit esc to return to the shell.

You can clear the history of calculations and variables to start a new exercise: from the menu select 9 Clear history.

You then have the choice between clearing the screen while keeping the variables (key validation to the right of the U key) or by deleting them (esc). You can view the space occupied by the variables by pressing the var key. To delete a variable to make space in memory, select the purge command in this menu, then type the name of the variable to be deleted (or select the variable from the menu var).

To exit χCAS, press the menu key twice. When you start a other application, the variables and the calculation history are saved (in the session.xw.tns file in the Xcas directory of the Nspire), they will be restored when you will come back to χCAS.

Notes:

- From the calculation shell, the keys 1 to 9, 0,., (And) preceded by shift make appear a small menu to quickly enter certain commands.
- When the cursor is on the command line just after a command name command, pressing the cursor down key displays the help on the command (if help exists) and enter an example.

- Example: type shift 2, then 3 (integrate), down arrow, then tab or enter

Modify the expression to integrate according to your needs then type enter

5 Usual formal calculation commands

5.1 Develop and factor

From the catalog, select the Algebre (2) submenu or the quick menu shift-1

- factor: factorization. Keyboard shortcut shift- * (prefixed) or => * (infixed key sto then *), for example x ^ 4-1 => *

(x - 1) (x + 1) (x 2 + 1)

. Use cfactor to factor on C.

- partfrac: expansion of a polynomial or decomposition into simple elements for a fraction. Keyboard shortcut => + (sto key then +), 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: attempt to simplify an expression. Keyboard shortcut => / (key → then /), for example sin (3x) / sin (x) => /

2 cos (2x) +1

Be careful, this command is memory intensive, and the TI has little memory, the risk reset exists.

- ratnormal: expand an expression, write a fraction in an irrational form.

5.2 Analysis

From the catalog (doc), select the Analysis submenu (4) or the menu fast shift-2

- diff: derivation. We can also use the notation '(shift- *) to differentiate by relative to x, so diff (sin (x), x)

cos x

and sin (x) '

cos x

are equivalent. To differentiate more than once, add the number of derivations per example diff (sin (x ^ 2), x, 3)

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

- integrate: primitive if 1 or 2 arguments, for example integrate (sin (x))

-cos x

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

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

Definite integral calculation if 4 arguments, for example integrate (sin (x) ^ 4, x, 0, pi)

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

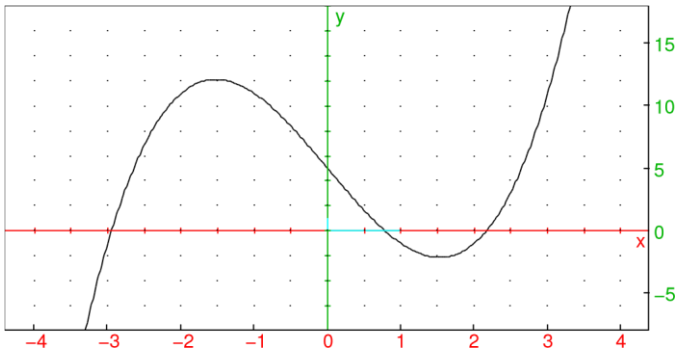
For $\int_0^{\pi} \sin(x)^4\, dx$ Put one of the integration terminals in approximate form if one wishes an approximate calculation of definite integral, for example integrate (sin (x) ^ 4, x, 0.0, pi)

1.1780972451

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

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

- tabvar: array of variations of an expression. For example tabvar (x ^ 3-7x + 5) we can check with the plot plot (x ^ 3-7x + 5, x, -4,4)



- taylor and series: Taylor development (or limited development or asymptotic). for example taylor (sin (x), x = 0.5)

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

- sum: discrete sum. for example sum (k ^ 2, k, 1, n)

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

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

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

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

calculates the sum and writes it in factored form.

5.3 Solve

From the catalog, select the Solve submenu (doc then press ln)

- solve allows to solve in an exact way an equation (reducing to a polynomial equation). The variable must be specified if it is not x for example solve (t ^ 2-1 = 0, t)

[-1, 1]

.

If the exact search fails, the fsolve command can be used to resolve approximate solution, or by an iterative method starting from an initial value fsolve (cos (x) = x, x = 0.0)

0.739085133215

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

[0.739085133215]

.

To have complex solutions, use csolve.

We can make assumptions about the variable we are looking for, for example assume (m> 1)

m

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

[2]

- solve also allows to solve simple polynomial systems, we give 1st argument the list of equations, in 2nd argument the list of variables. 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]]

- linsolve allows to solve linear systems. We give him the list equations and the list of variables (by convention an expression is equivalent to the equation expression = 0). for example linsolve ([x + 2y = 3, xy = 7], [x, y])

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

linsolve returns the general system solution (including if the solution is not unique).

desolve allows to solve in an exact way certain different equations, for example to solve y = 2y, we type desolve (y '= 2y).

An example where we indicate an initial condition, the independent variable and the unknown function:

desolve ([y '= 2y, y (0) = 1], x, y) Use odesolve for a resolution approximation and plotode for a graphical representation of a calabutment in an approximate manner.

- rsolve allows to solve in an exact way certain relations of recurrences $u_{n+1} = f(u_n, ...)$, for example arithmetic-geometric sequences, for example $u_{n+1} = 2u_n + 3$, $u_0 = 1$

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

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

5.4 Arithmetic

When necessary, a distinction is made between the arithmetic of integers and that of polynomials by the existence of the prefix i (like integer) in a com- name command, for example ifactor factors an integer (not too large) while factor factors a polynomial (and cfactor factors a polynomial over complexes). Cersome commands work for both integers and polynomials, for example gcd and lcm.

5.4.1 Integers

From the catalog, select the Arithmetic, Crypto submenu (doc 5)

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

iquo (23.13), irem (23.13)

1, 10

- isprime (n) tests whether n is a prime number. The test is probabilistic for large values of n.

isprime (2 ^ 64 + 1)

false

- ifactor (n) factors a not too large integer (up to approximately 128 bits). By example

ifactor (2 ^ 64 + 1)

67280421310721 · 274177

Keyboard shortcut keys → then * (=> *)

- gcd (a, b), lcm (a, b) GCD and PPCM of two integers or of two 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 and b,

  with | u | <| b | and | v | <| a |. u, v, d: = iegcd (23.13); 23u + 13v

  [4, -7, 1], 1

- ichinrem ([a, m], [b, n]) when possible, returns c such that c = a(mod m) and c = b (mod n) (if m and n are coprime, c exists).

  c, n: = ichinrem ([1.23], [2.13]); irem (c, 23); irem (c, 13)

  [93, 299], 1, 2

- powmod (a, n, m) computes a^n (mod m) by the fast power algorithm modular. powmod (7,22,23)

  1

The asc and char commands are used to convert a string of characters into a list of integers (between 0 and 255) and vice versa, which makes it easy to cryptography with messages in the form of strings.

5.4.2 Polynomials

From the catalog, select the Polynomials (8) submenu. The variable is by default x, otherwise it must generally be specified as the last argument, for example degree (x ^ 2 * y) or degree (x ^ 2 * y, x) returns 2, while degree (x ^ 2 * y, y) returns 1

- coeff (P, n) coefficient of x^n in P, lcoeff (P) dominant coefficient of

  P, for example

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

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

- degree (P) degree of the polynomial P.degree (x ^ 3)

  3

- quo (P, Q), rem (P, Q) quotient and remainder of the Euclidean division of P by QP: = x ^ 3 + 7x-5; Q: = x ^ 2 + x; quo (P, Q); rem (P, Q)

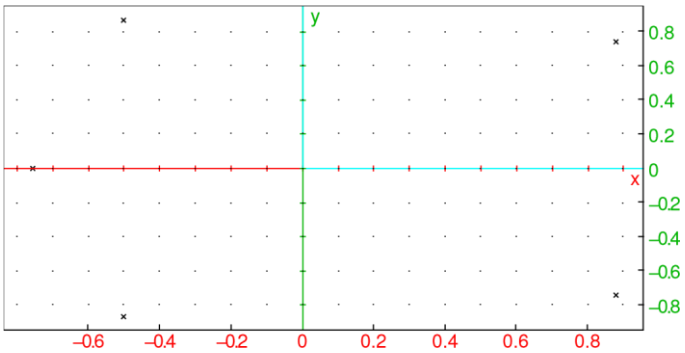
  $x^3 + 7x - 5, x^2 + x, x - 1, 8x - 5$

- proot (P): approximate roots of P (real and complex)

  proot (x ^ 5 + x + 1)

  [-0.754877666247, -0.5 - 0.866025403784i, -0.5 + 0.866025403784i, 0.877438833123 - 0.744861766662i, 0.87743883312 Graphic

  Representation : point (proot (x ^ 5 + x + 1))



- interp (X, Y): for two lists of the same size, interpolation polynomial notsant by the points (X_i, Y_i).

  X, Y: = [0.1,2.3], [1, -3, -2.0]; P: = interp (X, Y) => +

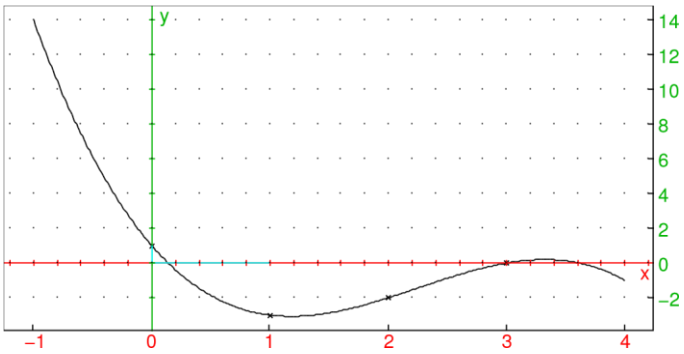
  $-4x^3 + 27x^2 - 47x + 6$

  [0, 1, 2, 3], [1, -3, -2, 0],

  6

  Graphic Representation

  scatterplot (X, Y); plot (P, x, -1.4)



- resultant (P, Q): resulting from polynomials P and Q

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

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

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

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

5.4.3 Z / nZ and finite fields

To work with modulo n classes, use the a mod n notation, e.g. example sqrt (2 mod 7). This also applies to work on pre- finite fields.

miers Z / pZ. To work on finite fields that are not prime, we must first define the body with GF, then we use a polynomial in the generator of the body.

5.5 Linear algebra, vectors, matrices

Xcas does not differentiate between vector and list. For example to make the product scalar of two vectors, we can enter:

v: = [1.2]; w: = [3,4]

dot (v, w)

To enter a matrix element by element, press shift-7 (M key as matrix) then matrix (or menu i (edit matrix) .You can then create a new matrix or edit an existing matrix from the list of proposed variables.

For small matrices, you can also enter a list of lists of the same size. For example to define the matrix

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

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

or [[1,2],

[3,4]] => A

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

It is strongly advised to store the matrices in variables to avoid enter them several times.

To enter a matrix whose coefficients are given by a formula, we can use the matrix command, 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 whose coefficient row j and column k is equal $_{j+k+1}$ (be careful with indices start at 0).

The identity matrix of size n is returned by the idn (n) command, while ranm (n, m, law, [parameters]) returns a matrix with random coefficients of size n, m. for example

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

$$\begin{pmatrix} 0.0881637986749 & 0.536643749103 & 0.360248812009 & 0.630750506185 \\ 0.0543795586564 & 0.819389336277 & 0.251521021593 & 0.0686232172884 \\ 0.178203014191 & 0.346399624366 & 0.852082391735 & 0.614078260958 \\ 0.714221911505 & 0.784336711746 & 0.453634891659 & 0.433968523517 \end{pmatrix}$$

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

$$\begin{pmatrix} -0.57241508091 & -0.751538700822 & 0.989770363266 & -0.00575141421859 \\ 0.902209004052 & 0.455987131429 & -1.85967614286 & -0.243668831329 \\ -1.61539092837 & 0.558232005133 & -0.159613435564 & 1.35165685556 \\ -0.16425534864 & 1.27150836747 & 0.245690624229 & -0.595841236292 \end{pmatrix}$$

To execute a command on matrices, if it is basic arithmetic (+, -, * inverse), we use keyboard operations. For other orders.

from the catalog, select the Matrices submenu (doc sin)

- eigenvals (A)

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

eigenvects (A)

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

return the eigenvalues and eigenvectors of a square matrix A.

- P, D: = jordan (A)

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

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

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

- Ak: = matpow (A, k)

- computes the k-th power of a matrix A with k a formal variable.

- rref performs the reduction in scaled form of a matrix A (pivot of Gauss)

- lu calculates the LU decomposition of a matrix A and returns a permutation of matrix P and two lower triangular L and upper triangular U matrix such that PA = LU. The result of the command

P, L, U: = read (A)

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

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

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

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

- qr computes the QR decomposition of a matrix A and returns two matrices Orthogonal and upper triangular R such that A = QR.

- svd (A) computes the singular value factorization of a matrix A, and return path U orthogonal, S vector of singular values, Q orthogonal such that

A = U * diag (S) * tran (Q). The ratio of the greatest singular value of

S on the smallest gives the number of conditions of A relative to the norm Euclidean, the greater this number, the more precision is lost when solving a system Ax = b when b is not known exactly.

6 Probabilities and statistics

6.1 Random draws

From the catalog, select the Probabilities submenu (doc 9) then select rand ()

0.38078169385

(real according to the uniform law in [0, 1]) or
n: = 6 :; randint (n)

"Done", 6

(integer between 1 and n). Many other random functions exist, with like prefix rand, followed by the name of the law, for example randbinomial (n, p) returns a random integer according to the binomial distribution of parameters n, p. To create a vector or a random matrix, use the ranv or ranm command (Alglin menu, Matrix), for example for a vector of 10 components according to the reduced centered normal law ranv (10, normald, 0,1)

[-2.28261976775, 0.652261860753, -0.690651824321, -0.323190436625, 0.157460267236, -0.324617014178, -0.36112711845

6.2 Laws of probability

From the catalog, select the Probabilites submenu (9). The laws proposed in the catalog are the binomial law, the normal law, the exponential law and the uniform law. Other laws are available from All: chisquared, geometric, multinomial studentd, fisherd, fish.

To obtain the cumulative distribution of a law, we enter the name of the law and the suffix

_cdf (select cdf in the Probabilities submenu catalog and type F1). For

obtain the inverse cumulative distribution, we enter the name of the law and the suffix _icdf (select cdf in the Probabilities submenu catalog and type F2).

Example: calculation of the centered interval I for the normal distribution of mean 5000 and of standard deviation 200 such that the probability of being 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

6.3 Descriptive statistics 1-d

These functions act on

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

From the catalog, select the Statistics submenu (doc log)

- mean (l)

11

: arithmetic mean of a list

- stddev (l)

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

: standard deviation of a list

Use

stddevp (l)

$$\sqrt{\frac{}{14}}$$

to have an unbiased estimator of the standard deviation of a population whose l is a sample - median (l)

10.0

, quartile1 (l)

9.0

,

quartile3 (l)

13.0

return respectively the median, the 1st and 3rd quartiles of a list For 1-d statistics of lists with counts, we replace l by two lists of the same length, the 1st

list is the list of values of the statistical series, the 2nd list is the list of staff. See also the commands of the menu shift-3 histogram and bar plot.

6.4 Descriptive statistics 2-d, regressions.

Enter the two lists of data in two variables, for example X: = [1,2,3,4,5] and Y: = [3,5,6,8,11], or in a matrix variable with 2 columns, from the shell with the shift-6 8 matrix menu (.

From the catalog, select the Statistics submenu (doc log keys), for regressions, from the shell, type shift-6.

- correlation (X, Y) calculates the correlation between 2 lists of the same size.
- covariance (X, Y) calculates the covariance between 2 lists of the same size.
- the _regression (X, Y) suffix commands compute adjustments by least-squares regression, commands with the _regression_plot suffix plot the curve representative of the regression (These commands display

  plus the coefficient R_2 which quantifies the quality of the fit (more R_2 is close to 1, the better the fit). Khicas has orders for do linear, exponential, logarithmic, power, polynomial and logistics.
- For example linear_regression (X, Y) returns the coefficients m, p of the linear regression line y = mx + p.

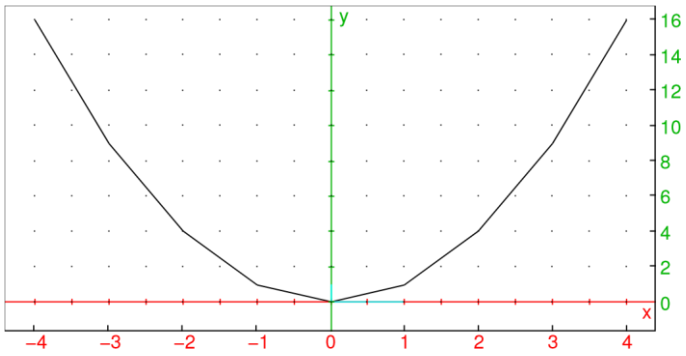
  linear_regression_plot (X, Y) draw the data-fitting line born contained in the X, Y lists of the same size.
- see also the scatterplot, polygon plot and polygon scatterplot commands to display the data on a graph. We can superimpose several curves regression on the same graph by separating them by one; from the command line. So to display the linear regression line corresponding to the data X = [1,2,3,4,5] and Y = [3,5,6,8,11], type the two commands above or edit a matrix having 2 columns with the shortcut shift-6 8. Then shift-6 enter (or doc log then select the command linear_regression_plot () complete the command by X, Y) or by the variable name of the matrix then enter.

Note: if your data is in a matrix m having 2 rows instead of 2 columns, you can use m ^ * to transpose (in fact it transconjugates, which is the same for real data).

7 Curves and other graphical representations

From the catalog, select the Curves submenu (quick access via doc 7, or shift-3 from the shell).

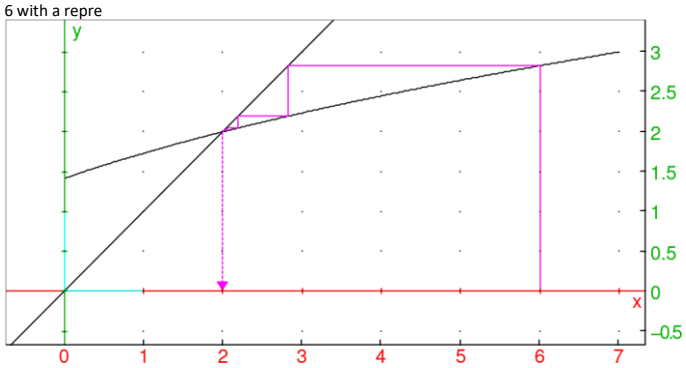
- plot (f (x), x = a..b) plot the graph of f (x) for x ∈ [a, b]. We can specify trust a discretization step with xstep =, for example plot (x ^ 2, x = -4..4, xstep = 1)



- LineTan (f (x), x, x0) draws the tangent to the graph of f (x) at x = x_0 .
- plotarea (f (x), x = a..b, n, method) plot the graph of f (x) for x ∈

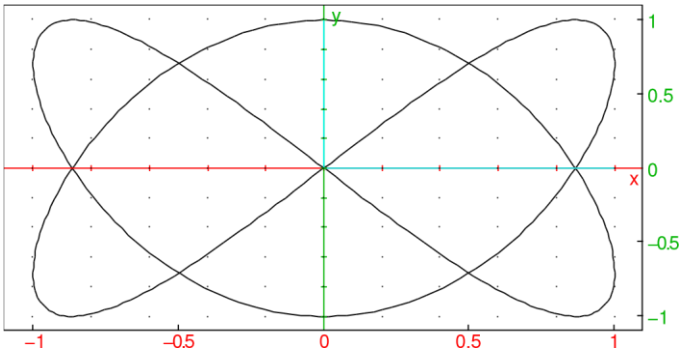
  [a, b], and darkens a portion of the plane which approaches the area under the curve, we can specify an integration method with n subdivisions among rectangle_droit, left_rectangle, trapezoids, simpson (go to doc, Options then type the beginning of the method name to enter it more quickly)
- plotseq (f (x), x = [u0, a, b]) "spider web" graph of the recursion sequence U n+1=f(Un) of first term U_0 given. For example if u_{n+1} =sqrt(2 + u_n), u_0 =

6 with a repre



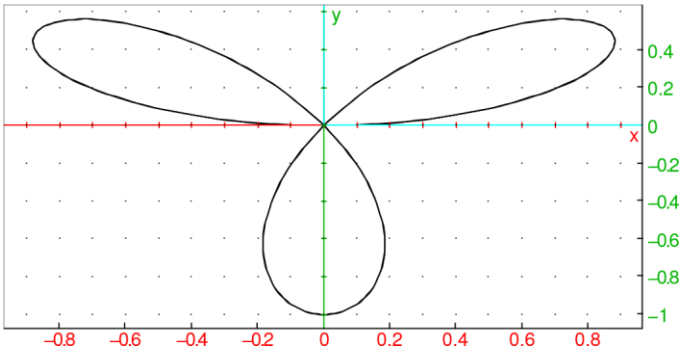
- plotparam ([x (t), y (t)], t = tm..tM) curve in parametric (x (t), y (t)) for t ∈ [t_m, t_M]. We can specify a discretization step with tstep = by example

  plotparam ([sin (2t), cos (3t)], t, 0.2 * pi)



- plotpolar (r (theta), theta = a..b) polar curve r (θ) for θ ∈

  [a, b], for example plotpolar (sin (3 * theta), theta, 0.2 * pi)



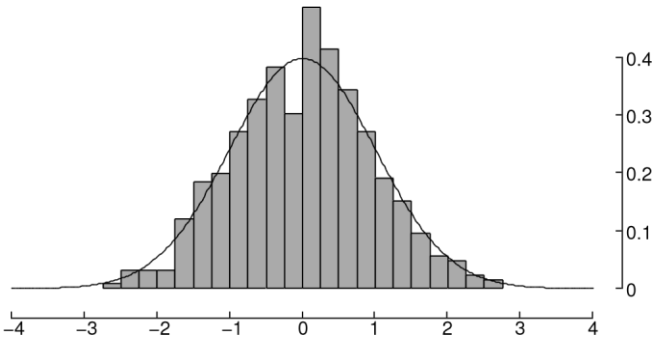
- plotlist (l) for a list l, draw the polygonal line connecting the points of coordinates (i, l_i) (index i starting at 0).

  plotlist ([X1, Y1], [X2, Y2], ...) draw the polygonal line connecting the coordinate points (X_i, Y_i)

- scatterplot (X, Y), polygonscatterplot (X, Y) for 2 lists X, Y of same size, draw a point cloud or polygonal line connecting the points of coordinates (X_i, Y_i)

- histogram (l, class_min, class_size) plots the histogram of the data born from list l with class_size class width starting at class_min. For example, we can test the quality of the random generator with l: = ranv (500, normald, 0.1); histogram (l, -4,0.25

  ); plot (normald (x), x, -4,4)

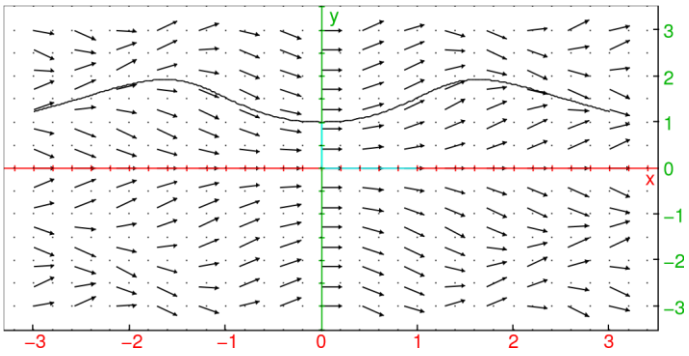


- plotcontour (f (x, y), [x = xmin..xmax, y = ymin..ymax], [l0, l1, ...]) draw the level curves f (x, y) = l_0, f (x, y) = l_1, ...

- plotdensity (f (x, y), [x = xmin..xmax, y = ymin..ymax]) represented by color levels of the function of 2 variables x and y in the rect specified angle (default between -4 and 4).

- plotfield (f (t, y), [t = tmin..tmax, y = ymin..ymax]) plots the field tangent to the differential equation y = f (t, y). We can add last optional parameter, plotode = [t0, y0] to simultaneously plot the solution passing through the initial condition y (t_0) = y_0. Example y = sin (ty) on the interval [-3, 3] in time and [-2, 2] in y

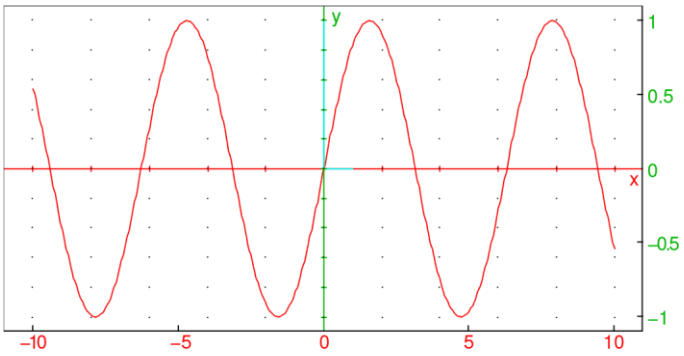
  plotfield (sin (t * y), [t = -3..3, y = -3..3], plotode = [0.1])



You can also use the plotode command outside of a plotfield command. You can draw several graphs simultaneously, just separate the commands of tracing by;

The Options menu (doc,) allows you to specify certain graphic options:

- for colors, display = color is used, for example plot (sin (x), display = red)

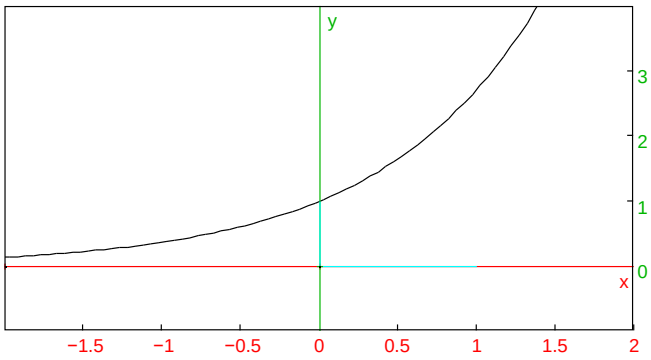


- To change the thickness of the segments (including the polygonal lines used to draw a curve), use display = line_width_2 to display = line_width_8.

  To change both the color and thickness, add the attributes, e.g.

  example: display = red + line_width_2

- The circles as well as the rectangles whose edges are parallel to the axes can be filled with the display = filled attribute (which can be added to other attributes).
- To replace the automatically calculated graphics window with values predefined, use the OPTN key, select gl_x or / and gl_y and indicate the desired x or y interval, the command must precede a command of trace. for example gl_x = -2..2; gl_y = -1..4; plot (exp (x))



- To remove the axes, select axes (axes = 0). The command must precede a plot command.

8 Analytical geometry.

For the moment, no interactive geometry application has been made, the u The use of analytical geometry commands from Xcas is therefore possible online of commands, with command names in English at the moment, e.g. ple point, segment, circle, circumcircle, incircle line, triangle, polygon, parallel, perpendicular, bisector, perpen_bisector, center, radius, distance, ...

9 Units and physical constants.

The doc menu, physical constants (shortcut doc pi) and physical units (raccourci doc sqrt) displays

- unit management commandsmksa to convert to the basic units of the international system usimplify to simplify using a single unit when possible, or a product of two.

  ufactor to force the write of one unit according to another => (shortcut key sto) to convert between two compatible units

- a non-exhaustive list of physical units
- a list of fundamental physical constants.

Examples:

60_ (km / h) => _ (m / s) mksa (_hbar_) usimplify (1_W * 1_s) ufactor (1_W, 1_J)

10 The expression editor

When a calculation returns an expression, it is displayed in full screen in the editor. expression tor 2d. From the calculation history, if the selected level is a expression, pressing shift-5 (2d) displays the expression in the 2d editor. Online from command, pressing shift-5 also opens the 2d editor, i.e. with 0 if the command line command was empty, or with the contents of the command line if this is syntactically correct.

When the 2d editor is open, the expression is displayed in full screen and a part of the expression is selected. We can then act on the selection by performing commands entered via the menus or the keyboard, the selection can also be edited (input mode 1d). This allows you to rework subexpressions or edit a expression in natural writing.

You can undo the last modification made by pressing shift-3 (undo).

Press the enter key to exit the 2d editor and copy the inline expression from command, type esc to quit without copying the expression. Example 1: we are going to enter

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

From an empty command line, type shift-5 (2d), you should see 0 selected. Hit the x key and enter, now x is highlighted. Tap on the

key trig (trig returns sin, shift trig cos and ctrl trig tan), it is sin (x) which is over 0 shine. Tap on the division key (above -), you should see $\frac{\sin(x)}{0}$ with 0 highlighted, type x then enter, you should see $\frac{\sin(x)}{x}$ with x in the denominator in highlight. Tap on the up arrow cursor to highlight $_{x}$ highlighted, then shift-2 4 (for limit). The expression is correct, you can type enter for the copy in command line and enter again to run the calculation. If we had wanted a limit in + c, it would have been necessary to move the selection with cursor to the right, then do shift-1 7 (oo) enter.

Example 2: we are going to enter

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

From an empty command line, type shift-5 (2d), then shift-2 3 (integrate), you must see

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

with x selected. It is therefore necessary to change the 1 of the upper limit and the 0 to integrate. To modify 0, cursor left to select it then 1 / (x ^ 4 + 1) enter, then cursor left shift-1 7 enter. Press enter to copy to the line of command then enter to perform the calculation, the result is displayed in the 2d editor, inter quits the editor with the integral and its value in the history (in algebraic syntax 1d).

Example 3: we will calculate and simplify

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

From an empty command line, type shift-5 (2d), then shift-2 3 (integrate), you must see

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

Move the cursor to the 0 of the lower bound of the integral and press the key DEL you must see

∫0dx

with everything selected. Use the cursor down to select 0 and type 1/ (x ^ 4 + 1) enter then enter to copy in command line then enter to run the calculation, the result is now displayed in the 2d editor. You can then select with the cursor keys for example the argument of a arctangents and execute shift-1 enter (simplify) to simplify partial result, then start over with the other arctangent.

We can simplify even more, by gathering the logarithms. For that we must first swap two of the arguments of the sum. Select one of the logarithms with cursor movements, then type right or left shift-cursor, this swaps the argument selected with its right or left sibling. Then type ctrl cursor to the right or to the left, this increases the selection by adding the sibling of right or left. Once the two logarithms have been selected, menu shift-1 2 enter (factor), then move the selection down to the sum or difference of logarithms, go in the doc menu then enter (All), type the letters l, n, c which moves to the first command starting with lnc, select lncollect, validate and finally hit enter (eval).

11 Calculation sessions

11.1 Editing the history.

Using the cursor up / down key, you move through the history of calculations, the current level is highlighted.

To change the order of levels in the calculation history, type ctrl cursor upwards or downwards. To erase a level, press the DEL key (the level is copied to the clipboard).

To modify an existing level, you type on shift-5 or on shift-4. In the first case, it is the 2d editor which is called if the level is an expression, in the second In this case, the text editor is called. Type esc to cancel the changes or enter to validate. If the modifications are validated, the command lines located below the modified line will be automatically calculated, taking into account the modifications, for example if you modify a level like A: = 1, the lines located below depending on A will be updated.

This process can be automated using a cursor, which can be created with an assistant, from the menu menu, Parameter. Once created you can edit a cursor by pressing the + or - keys when the level containing the command assume or parameter is selected (type * or / for a more modification fast).

11.2 Variables

By pressing the var key you display the list of variables which have a value, as well as commands for handling variables. Move the cursor to a variable then enter to copy it to the command line, DEL copies online from commands the command to delete the variable (then confirm with enter). The restart command allows you to delete all the variables. The command assumes allows to make an assumption on a variable, for example assume (x> 5) (> se found in the shift-PRGM menu).

11.3 Backup and compatibility

You can save a session by pressing ctrl save or menu 2. You can open a session by pressing shift save or menu 4. The sessions are saved. kept with a file name ending in .xw.tns. You can open them on your computer with Xcas or Xcas for Firefox. Conversely, the File menu, Excarry as allows you to save a session in this format, or in another format to use your session on a χCAS-compatible calculator. In Xcas for Firefox, you must select in the settings Nspire CX as a calculator then export.

NB: you can save the content of the script editor independently of the session, the scripts have a file name ending in .py.tns. Simply rename the script by removing the .tns to use this script with any Python compatible interpreter, for example a calculator from another manufacturer. posing a MicroPython interpreter. Conversely, you can import a script Python by making a copy of the file by appending .tns to the filename and then transferring to the TI Nspire CX. You can then open it in the Script Editor.

12 Programming

The χcas programming environment is quite complete: an editor, the Xcas interpreter with all its commands (partial compatibility with the modules Python math, cmath, random, turtle, numpy, scipy, giacpy, matplotlib and a surwhole kandinsky module), with a debugging tool allowing the execution step by step. This same environment can use the MicroPython interpreter instead of Xcas (cf. section 13).

You can program using the French command structures of

Xcas or using Python syntax compatibility. Very short programs

(in one line) can be entered directly from the command line. The programs

longer or that you want to save will be entered in the program editor on the calculator, or transferred from a PC or another calculator.

Programs are installed in the Xcas directory. You can view them from the shell by typing menu 4 (Load), and run them from the shell by typing ctrl r.

12.1 Getting started (programming)

A first example from the command line:

a function defined by an algebraic expression. We enter function_name (parameters): = expression

For example, to define the perimeter of a circle of radius r, we can type peri (r): = 2 * pi * r

$$r \mapsto 2\pi r$$

(to type :, press ctrl then,) then we can calculate peri (1)

$$2\pi$$

.

Another example, to calculate the confidence interval of a second knowing a frequency p and an effective n, we type

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

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

then we test F (0.4,30)

[0.217425814165, 0.582574185835]

Another example: with the tortoise of Xcas

The tortoise of Xcas is a small robot which moves according to orders given to it. born leaving a trace of its passage. The turtle controls are accessible from the last item in the doc menu, using the doc x 2 shortcut . Enter the order advance in this menu then validate, you should see the turtle (symbolized by a triangle) advance 10 pixels. Type esc to return to the command line. Grab the command turn_left and validate, the turtle has turned 90 degrees. Repeat 3 times these two commands to display a square.

To erase the drawing and return the turtle to the origin, enter the erase command.

To make turtle drawings, it is advisable to use the program editor (cf.

below).

Another example: an “oneliner” loop in Xcas syntax.

Open the Programs menu (doc cos), then select the example of for (cursor on for then Years)

```
for j from 1 to 10 do print (j, j ^ 2); ffor;
```

0

hit enter, you should see the squares of the integers from 1 to 10.

Exercise: make the turtle square using a loop.

Using the editor

Let's modify this example to display squares from 1 to n using the syntax Python compatible and program editor. Press esc to switch from shell to the program editor. If you are asked for program or Turtle, enter. This opens the editor with a function mockup def f (x): Check that the syntax Python is enabled (menu menu), otherwise enable it (8). Replace x with n, then replace the cursor at the end of the line and go to the line (key to the right of the key U). Type Shift-2 then enter (for), place a j between for and in range (then a n between the parentheses of range (). On the next line type shift-3 7 (print) and validate (enter), then type j, j ^ 2). You should have the following source text:

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

Replace x with n in return (or delete the line). NB: for the power, we can use ^ or ** in KhiCAS (you must use ** in Python).

Now type enter. If all is well you should see Success in the line state. Otherwise, the line number of the first error is indicated as well as the word that caused the error. The cursor is positioned on the line where the error was detected (it may happen that the error is located before but detected only a little further). If you are using the syntax in Python, note that the programming structures are translated into Xcas language, the errors displayed are compared to this translation (therefore end of structure keywords like end may have been added).

If the program is syntactically correct, you can save it from the menu menu. To run it, return to the command line by typing the esc key, type f (10) for example, you should see squares 1 to 10 displayed.

Be careful in the program editor, pressing the enter key interprets the publisher content. To go to the next line, press the validate key. dation (to the right of the u key).

3rd example: Calculation of the terminal confidence interval

In Xcas syntax. You can enter it on the command line

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

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

Redundant calculations can be avoided by using a local variable (use doc Programs to enter function, local, return and ffunction)

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

Exercise Create a carre.py file containing a script to display a square with the turtle. Create a carren.py file to display a square of n pixels, in using a function with argument n and the repeat instruction.

Solution From the script editor, press menu 5 (Delete). Then shift-4 clears. Add 4 times in advance; left_turn ;. Press enter to test. Safekeep (menu 3 Save as).

Erase again (menu 5 erase) Then shift-4 erases. Add before line erase

```
def f (n):
    for j in range (4): advance (n) left_turn
```

then after the line erases; type for example f (40) then enter. Then do menu 3 (Save as).

An example of a non-algebraic function: the computation of the GCD of 2 integers. Use enter to move to a new line. In Xcas syntax

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

We check pgcd (12345,3425)

5 . The same in Python

syntax

```
def gcd (a, b):
```

```
while b! = 0:
    a, b = b, a
return a
```

Focus

The debug command is used to execute a function in step-by-step mode, ie visualize the evolution of variables instruction by instruction, for example debug (pgcd (12345,3425))

12.2 Some examples

The Xcas directory of the archive contains some sample programs (with frequently a graphic representation) including:

- frequency in a sample, interval of fluctuations (freq.xw)
- the paradox of the Duke of Tuscany (toscane.xw)
- solving quadratic equation (deg2.xw)
- dichotomy (dicho.xw)
- rectangle method (integr.xw),
- Mandelbrot fractal (mandel.py)
- a benchmark used by tiplanet to measure the speed of the interpreter (qcc.xw)

12.3 Usable commands

Unlike the adaptations of MicroPython proposed by the manufacturers (including that of the TI Nspire), programming in (simili-) Python in KhiCAS is not an independent app. So you can use all types of Xcas (for example the rationals) and apply all the commands of Xcas in your programs. This corresponds more or less to a Python environment with modules math, cmath, random (more complete than the urandom module provided by the constructors), scipy, numpy, a small pixel graphics module (set_pixel (x, y, c), set_pixel () to synchronize the display, clear (), draw_line (x1, y1, x2, y2, c), draw_polygon ([[x1, y1], [x2, y2], ...], c), draw_rectangle (x, y, w, h, c), draw_circle (x, y, r, c), color + thickness + fill c is a parameter optional, draw_arc (x, y, rx, ry, t1, t2, c) allows to draw an arc of ellipse). and to replace matplotlib we can use the graphics commands in a χCAS coordinate system (point, line, segment, circle, barplot, histogram and the plot commands ...). In addition, you can work with expressions and do formal calculation on it. For the complete list of commands and an overview detailed, we refer to the documentation of Xcas.

13 Integrated MicroPython interpreter

χCAS now comes with its own MicroPython interpreter (which is similar to that provided by Numworks). To switch from χCAS to MicroPython and vice versa you can type the command python or xcas in the shell on an empty line. These commands are accessible in the menu fast shift).

Attention, if you launch MicroPython after having worked with Xcas or / and with the spreadsheet, there may not be enough memory for the MicroPython heap, in this case you risk losing your work session, remember to save it. Default 40K are reserved for MicroPython. You can change this value up to 64K in configuration (Home key then In key). If you choose a high value, it may need to exit χCAS and reopen it for the heap to be allocated in a contiguous memory area (heap fragmentation problem otherwise).

Note: when the Xcas interpreter is active, if you pass as an argument to the xcas or python command the variable name of a function you provided programmed, this displays its source text as a string in syn-

Xcas or Python tax. You can therefore program in Xcas syntax and translate into Python if you have to comply with rules that impose this language.

13.1 Standard modules: math, cmath, random

These are the native modules provided by MicroPython (urandom has been renamed random), and which conform to the standard. Other standard MicroPython modules that are not intended for math are available. We can type halp ('modules') to see the list of available modules. See the MicroPython online help.

13.2 The case module

It gives access from MicroPython to the native commands of Xcas. It contains a single caseval command, which takes as argument either a character string which will be evaluated by the Xcas interpreter, i.e. a character string (containing the name of the command to be executed) then arguments of any type (representing arguments). The result is a string of characters. We can of course call eval to turn the return value into a Python object. Examples:

```
caseval ("sin", 0) caseval ("sin (0)") caseval ("integrate", "1 / x", "x") caseval
("integrate", "1 / x", "x", 2,3) a = [1,2,3] caseval ("a: =", a) caseval ("a")
```

Note: caseval has two synonyms xcas and eval_expr.

13.3 The graphic module

This is a native MicroPython module that exports plot functions from Xcas pixelated. Some of the commands are accessible from the quick shift menu. This module has synonyms, kandinsky and casioplot in order to facilitate the use Python scripts for Numworks (Epsilon) and Casio calculators.

13.4 The matplotl module

This is a native MicroPython module that exports plot functions from Xcas spotted and aims at a certain compatibility with the matplotlib module (or one of its submodules) on PC. Some of the commands are accessible from the quick menu shift 0.

13.5 The arit module

It is a native module which exports integer arithmetic functions from Xcas: test of primality and next prime number (by Miller-Rabin), factorization of integers too big (detection by Pollard-rho of the smallest prime factor, so up to about 9 figures), gcd and identity of Bèzout, indicator of Euler (if we know how to factorize). I've been also included two list to string conversion functions to facilitate teaching a bit of cryptography.

13.6 The linalg module

It allows you to handle lists as vectors and lists of lists as matrices. Unlike Xcas, Python is not a specifically adapted language in math, you have to use commands prefixed add, sub, mul to perform the basic arithmetic operations + - * on the vectors and matrices represented by lists.

The linalg module is a native module, which uses Xcas to do the quasiall calculations.

13.7 The numpy module

It is an overlay of the linalg module which defines an array class to represent vectors and matrices. We can then use + - * to do the operations base on vectors and matrices.

The numpy module is not a native module, it is source text written in Python. Importing it therefore consumes RAM memory. You can write your own version of numpy.py and store it in the scriptstore, it will then take precedence on the version used by default. The latter aims to ensure a minimum of compatibility with the module of the same name on PC. Although not native, this module is available in exam mode (the default source text is integrated into the source code of the MicroPython interpreter).

```
import linalg import math class array:
    def __init __ (self, a): self.a = a

    def __add __ (self, other): return array (linalg.add (self.a, other.a))

    def __sub __ (self, other): return array (linalg.sub (self.a, other.a))

    def __mul __ (self, other):
        if type (self) == array: if type (other) == array: return array (linalg.mul
            (self.a, other.a))
            return array (linalg.mul (self.a, other))
        return array (linalg.mul (self, other.a))

    def __rmul __ (self, other): if type (self) == array: if type (other) == array:
                                                    return array (linalg.mul (self.a, other.a))
            return array (linalg.mul (self.a, other))
        return array (linalg.mul (self, other.a))

    def __matmul __ (self, other):
        return __mul (self, other)

    def __getitem __ (self, key): r = (self.a) [key] if type (r) == list or type (r) == tuple:
            return array (r)
        return r

    def __setitem __ (self, key, value): if (type (value) == array):
            (self.a) [key] = value.a else:
            (self.a) [key] = value return None

    def __len __ (self):
        return len (self.a)

    def __str __ (self): return 'array (' + str (self.a) + ')'

    def __repr __ (self): return 'array (' + str (self.a) + ')'

    def __neg __ (self): return array (-self.a)

    def __pos __ (self):
        return self

    def __abs __ (self): return array (linalg.abs (self.a)) def __round __ (self):

        return array (linalg.apply (round, self.a, linalg.matrix))

    def __trunc __ (self): return array (linalg.apply (trunc, self.a, linalg.matrix))

    def __floor __ (self): return array (linalg.apply (floor, self.a, linalg.matrix))

    def __ceil __ (self): return array (linalg.apply (ceil, self.a, linalg.matrix))

    def T (self): return array (linalg.transpose (self.a))
```

```
def real (x):
	if type (x) == array:
		return array (linalg.re (xa))
	return x.real

def imag (x):
	if type (x) == array:
		return array (linalg.im (xa))
	return x.imag

def conj (x):
	if type (x) == array:
		return array (linalg.conj (xa))
	return linalg.conj (x)

def sin (x):
	if type (x) == array:
		return array (linalg.apply (math.sin, xa, linalg.matrix))
	return math.sin (x)

def cos (x):
	if type (x) == array:
		return array (linalg.apply (math.cos, xa, linalg.matrix))
	return math.cos (x)

def tan (x):
	if type (x) == array:
		return array (linalg.apply (math.tan, xa, linalg.matrix))
	return math.tan (x)

def asin (x):
	if type (x) == array:
		return array (linalg.apply (math.asin, xa, linalg.matrix))
	return math.asin (x)

def acos (x):
	if type (x) == array:
		return array (linalg.apply (math.acos, xa, linalg.matrix))
	return math.acos (x)

def atan (x):
	if type (x) == array:
		return array (linalg.apply (math.atan, xa, linalg.matrix))
	return math.atan (x)

def sinh (x):
	if type (x) == array:
		return array (linalg.apply (math.sinh, xa, linalg.matrix))
	return math.sinh (x)

def cosh (x):
	if type (x) == array:
		return array (linalg.apply (math.cosh, xa, linalg.matrix))
	return math.cosh (x)

def tanh (x):
	if type (x) == array:
		return array (linalg.apply (math.tanh, xa, linalg.matrix))
	return math.tanh (x)

def exp (x):
	if type (x) == array:
		return array (linalg.apply (math.exp, xa, linalg.matrix))
```

```
        return math.exp (x)

def log (x):
        if type (x) == array:
                return array (linalg.apply (math.log, xa, linalg.matrix))
        return math.log (x)

def size (x):
        if type (x) == array: return linalg.size (xa)
        return linalg.size (x)

def shape (x):
        if type (x) == array:
                return linalg.shape (xa)
def dot (a, b): return a * b

def transpose (a): if type (x) == array: return array (linalg.transpose (xa))

def trn (a):
        if type (x) == array:
                                                        return array (linalg.conj (linalg.transpose (xa)))
        return linalg.conj (linalg.transpose (xa))

def zeros (n, m = 0): return array (linalg.zeros (n, m))

def ones (n, m = 0): return array (linalg.ones (n, m))

def eye (n): return array (linalg.eye (n))

def det (x):
        if type (x) == array: return linalg.det (xa)
        return linalg.det (x)

def inv (x):
        if type (x) == array:
                return array (linalg.inv (xa))
        return linalg.inv (x)

def solve (a, b):
        if type (a) == array: if type (b) == array:
                        return array (linalg.solve (aa, ba))
                return array (linalg.solve (aa, b))
        if type (b) == array:
                return array (linalg.solve (a, ba))
        return linalg.solve (a, b)

def eig (a):
        if type (a) == array: r = linalg.eig (aa) return array (r [0]), array (r [1]) return linalg.eig
        (a)
def linspace (a, b, c): return array (linalg.linspace (a, b, c))

def arange (a, b, c = 1): return array (linalg.arange (a, b, c))

def reshape (a, n, m = 0):
        if type (n) == tuple: m = n [1] n = n [0]
        if type (a) == array: return array (linalg.matrix (n, m, aa))
        return linalg.matrix (n, m, a)
```

14 Additional applications.

Accessible by menu menu 1 or the shift-ANS keyboard shortcut. There are currently the periodic table of the elements (according to Maxime Friess), and four examples addin (the source code of which can serve as a model for experienced programmers wishing to program addins for χCAS): a formal spreadsheet, the continuation of Syracuse (very simple), the Mandelbrot fractal, a mastermind set (which can also serve as a game),

15 Keyboard shortcuts.

- the validation key to the right of the u key does not always act like the enter key, in particular in the program editor, it allows to go to the line and in the expression editor it evaluates the selection
- the trig key and its shift / ctrl give access to the sin, cos, tan functions
- you can type ctrl- * to type "and shift- * to type '
- the \ character is accessible via ctrl- /,% by shift- /.
- ; and: are in shift / ctrl of,
- shift-1 to shift-6: depending on the mode (Xcas or Python) see the captions
- Xcas mode: shift-7 matrices, 8 complexes, 9 arithmetic, 0 probability and others, .reals, (-) polynomials, (lists,) programming
- Python mode: shift-7 and 8 matrices, 9 arithmetic, 0 spotted graphs,. graphs pixelated, (-) colors, (lists,) programming
- interpreter change: shift) 8
- => followed by a physical unit performs a unit conversion, => followed by a function is used to execute actions:

  => * write as a product (for a unit, write using the units fundamentals of the MKSA system),

  => + write in the form of a sum (for a unit, write in the most

  simple possible),

  => / write in the form of a quotient,

  => sin, => cos, => tan conversion to sines, cosines or tangents.

  =>, used to time an evaluation,

- 16 => => write the following integers in base 16, 10 => => in base 10, 8 => => in base 8
- ctrl then p programming,
- ctrl then o gives access to the options menu
- ctrl followed by R in the shell allows to re-run the current session (run)
- ctrl followed by S in the shell allows access to the configuration (setup)
- ctrl followed by Z usually undoes the last modification in the program editor:
- shifted cursor keys: move to start / end of line / file.
- ctrl followed by C allows you to start a selection, redo ctrl then C to copy, ctrl then X or del to cut, ctrl then V to paste
- enter: if a word search / replacement is active (after having made menu6), searches for the next occurrence of a word. Otherwise, just wrap.
- DEL deletes the previous character or the selection.
- ans (shift (-)): toggle between the editor and the turtle figure
- esc: quits the editor and returns to the command line. We can then go back to the editor by typing esc again.

16 Switching off, reset, clock.

Like all software, χCAS is not free from bugs. If a calculation blocks, then start by trying to interrupt it by pressing the ON button. If that's not enough, you have to resolve to press the RESET button on the back of the calculator. For be able to restart χCAS, you must first reactivate ndless (from 2. My documents then in the ndless directory).

To reset the internal calculator clock that is displayed in the shell, connect your calculator with your computer, or type a command from the type hh, mm =>, for example 16.05 =>, for 16:05.

On CX models only, the calculator backlight turns off at the automatically after a few tens of seconds if nothing is done, it suffices press ON to switch on again. You can also do ctrl ON to force the shutdown backlight. Be careful, this is not a real extinction of the calculator (there is not enough information available on the Texas Instruments hardware to do it). If you really want to turn off the calculator, you must exit χCAS by pressing the menu key as many times as necessary (between 2 and 4 times). To avoid a battery discharge too quickly, on nspire CXs, χCAS self-discharges when after 2 hours of inactivity (by saving the current session under the usual name session.xw.tns). The calculator will then turn on briefly before turning off.

Actually. In case of prolonged inactivity of several days, the calculator reboots when it is on again, you must reactivate ndless before you can use χCAS.

17 Copyright, licenses and acknowledgments

- Giac and χCAS, kernel of calculation (c) B. Parisse and R. De Graeve, 2019. The calinteger assays are performed with GMP, multi-precision floats with MPFR, interval arithmetic with MPFI
- χCAS interface adapted by B. Parisse from the user interface of the source code of Eigenmath created by Gabriel Maia and the user interface of Xcas.

  License to use CAS: GPL2. (see the details of the conditions in the LICENSE.GPL2 file or on the GPL2 page of the Free Software Foundation).
- Thanks to Fabian Vogt for firebird-emu, ndless. Thanks to any the ndless team. Thanks to the tiplanet team, in particular Xavier Andréani, Lionel Debroux and Adrien Bertrand, for the discussion forum and all travailcas development work (archive storage, articles, contests).
- the periodic table of the elements is a port of the application of Maxime Friess with its distribution authorization under the GPL license.

18 Development in C ++ with χCAS and Ndless

You must first install the SDK of ndless, under linux by the command git clone --recursive https://github.com/ndless-nspire/Ndless.git

Then you have to compile it, which takes one to several hours depending on the power of the golddinator:

cd ndless / ndless-sdk / toolchain / ./build_toolchain.sh

Then you have to install the source code of Giac / Xcas.

wget https://www-fourier.univ-grenoble-alpes.fr/~parisse/giac/giac_stable.tgz tar xvfa giac_stable.tgz cd giac-1.6.0 / micropython-1.12 / nspire

open the mklib file and adjust the path to copy the libmicropy.a library, then

sh mklib

cd ../../src cp config.h.nspire config.h cp Makefile.nspire Makefile

adjust the paths in the Makefile then type make.

You are finished.