Scientific Calculator

by Michele Vacatello

A free, lightweight, simple to use and versatile calculation system for scientists

Università di Napoli Federico II
Dipartimento di Scienze Chimiche

Prof. Michele Vacatello
Università di Napoli Federico II
Dipartimento di Scienze Chimiche
Via Cinthia, 12 – 80126 Napoli (Italy)

michele.vacatello@unina.it
http://micvac.dichi.unina.it

Università di Napoli Federico II
Dipartimento di Scienze Chimiche

Scientific Calculator (SC) is:

lightweight :

- a single .exe file less than 200 KB
- peak memory used less than 15 MB
- no installation (run from all writable media)

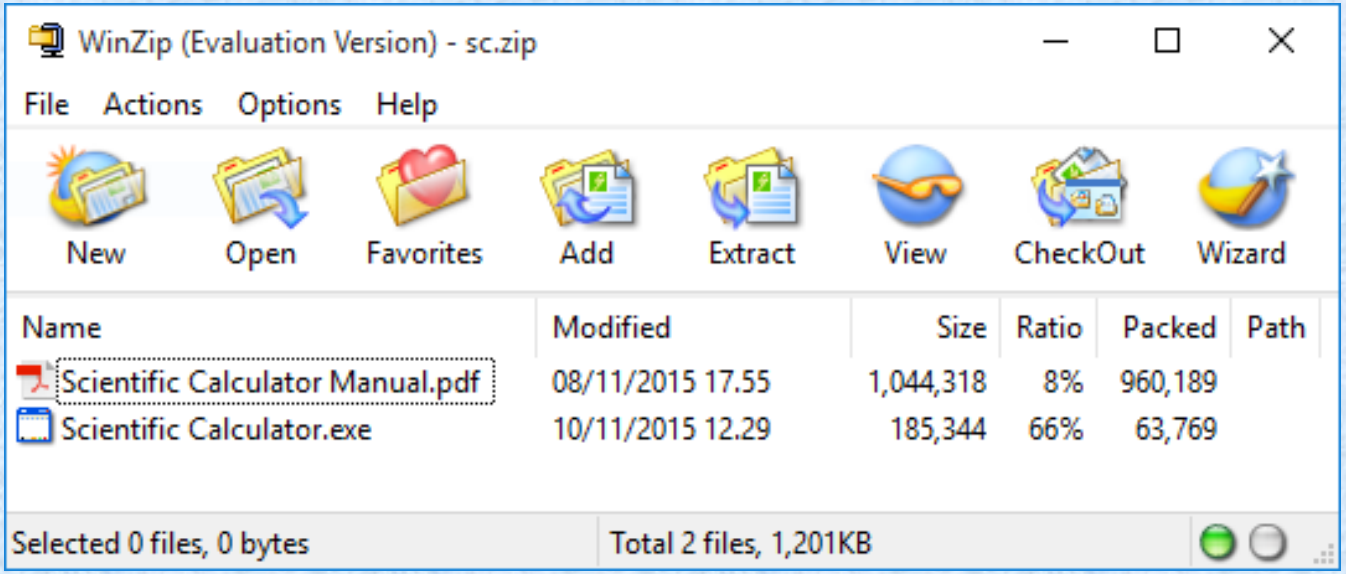
simple to use:

- scientifically sound syntax and rules
- modern user-friendly interface
- blistering fast

versatile :

- complex numbers, vectors, matrices, polynomials
- statistics, linear systems, linear fit, eigenvectors
- personalized sets of constants and functions
- simple scripting language

micvac.dichi.unina.it

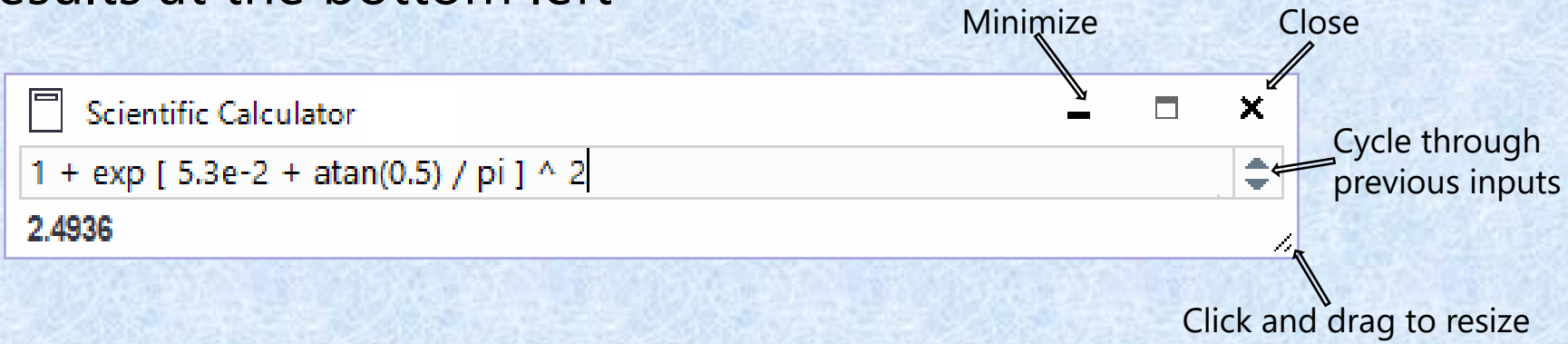


Overview

Università di Napoli Federico II
Dipartimento di Scienze Chimiche

Interface

Basically, **SC** evaluates an input expression and shows the results at the bottom left

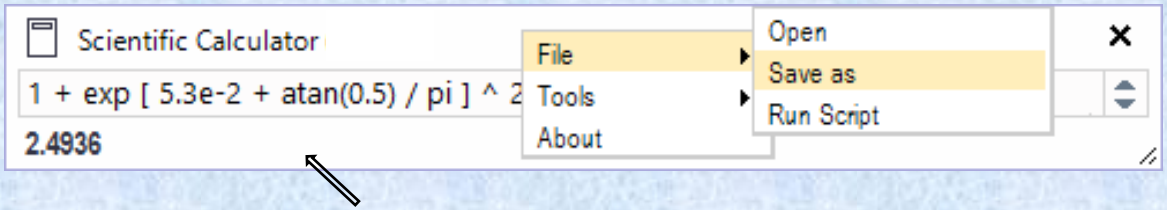


11.50 + 30.45 + 110 + 7.25	**159.2**
ln(0.1+2i) – sin(pi / 3) ^ (1-0.3i)	**-0.1708 + 1.4835i**
(1, 2, 3) + (2, 3, 4)	**(3, 5, 7)**
[(1, 2, 3) , (2, 3, 4)] * [(1, 2),(1, 3),(1, 4)]	**((6, 20) , (9, 29))**
pol(1, 0, 2, 3, 1) – pol(1, 1, 1)	**pol (0, -1, 1, 3, 1)**

- The input line can be any length (scrolling left when needed); blanks , " [ ] " and " { } " can be used to improve readability; characters not allowed are not accepted
- The input line can be edited as usual (copy/paste is allowed) ; pressing ESC clears the line
- The black/red color of the input line indicates that the required calculation is feasible/unfeasible; clicking Enter shows either the results or the nature of the input error

(1 , 2 , 3) + + (1 , 2 , 3)	**Missing expression: ++**
(1 , 2 , 3) ^ pol(1 , 2)	**Not implemented**

- When closing, **SC** stores an image of the current status, automatically loaded when the program is restarted
- Clicking right the background shows a menu allowing to save the current status in a .ssc file selected by the user or to open a previously saved file



Click the lower bar to erase menus

Università di Napoli Federico II
Dipartimento di Scienze Chimiche

Constants and functions

Intrinsic constants:

$i = (-1)^{1/2}$; e = exp(1) ; pi = acos(-1)

Intrinsic functions: all normally available base functions

ln, log, exp; sin, cos, tan; asin, acos, atan; sinh, cosh, tanh;
deg, rad, abs; fact

Specific functions:

- list, ils, ilp (lists)
- vedp, vecp (vectors)
- transp, diag (matrices)

Special functions:

- round
- stat
- lfit
- solve
- val
- eigenv

Multiple expressions

- The input line may contain multiple expressions separated by " ; "
- Values are passed to subsequent expressions in the same line as w1, w2, ...

ln(7) + 1 / exp(2) ; w1 ^ 2 ; w1 + w2	**6.4128**
(1,2,3) ; (-1,-2,-3) ; w1 + w2	**(0,0,0)**
pol(1,2,3) ; pol(1,-2) ; w1 * w2	**pol (1, 0, -1, -6)**

- Calculate $x^3 + x^2y + xy^2 + y^3$ for various x and y

1 ; 3; w1^3 + (w1^2 * w2) + (w1 * w2 ^ 2) + w2^3	**40**

- For a chemical reaction, $K_{eq}(300\ K)=0.1$ and $K_{eq}(500\ K)=100$; evaluate $K_{eq}(373\ K)$

300;0.1;500;100 ; (w3*ln(w4)-w1*ln(w2))/(w3-w1) ; -w1*ln(w2)+w1*w5 ; exp(w5-w6/373)

2.9365

$\Delta S^0/R$ $\Delta H^0/R$

Direct naming

x =1 ; y = 3 ; x^3 + x^2*y + x*y^2 + y^3	**40**

- Names can be any length; as usual, they may contain letters and numbers and must start with a letter

g =9.8 ; initialspeed = 10 ; maxheight = initialspeed ^2 /(2 * g)	**5.102**
htc14=5730 ; t = 3000 ; percent = 100 * exp(-t * ln(2) / htc14)	**69.565**
a = (1,2,3) ; b = ((1,2) , (3,4) , (5,6)) ; c = 2 * a * b	**(44 , 56)**

- Names can be reassigned; at any stage, their value corresponds to the value of the last assigment

a0 = ln(2) ; a0 = 1 + a0 ; a0 = a0 / 100	**0.01693**

User functions

- Users can define their own functions

```
mic ( x , y , z ) = x*y + x*z + y*z ; a = 1 + ln(3) ; mic( -1 , a , 3 )
```

1.1972

- x, y, … are dummy arguments, used only as placeholders; other names would be equally acceptable
- dummy arguments do not specify the nature of actual arguments

```
f1 ( x ) = x + x ^ 2 ; f1 ( pol ( 1, 2, 3 ) )
```

pol (2 , 6, 13 , 12 , 9)

- definitions may also contain non-dummy names, to be defined before the function is executed

```
mic ( x , y , z ) =( x*y + x*z + y*z ) / q ; a = ln(2) ; q = 10 ; mic( -1 , a , 2 )
```

-0.13069

- user functions of user functions are allowed

```
f1(x) = x + x ^ 2 ; ff(a,b) = f1(a) + ln( f1(b) ) ; ff( 1 , 3)
```

4.4849

- Angles of a 3-D vector with the cartesian axes

```
vers (x) = x / val(x) ; angles(y) = deg(acos(vers(y))) ; angles( (1 , 0, -1 ))
```

(45 , 90 , 135)

when x is a vector, vers(x) is the corresponding versor; when y is a vector, angles(y) is the 1D-array containing the angles with the cartesian axes in degrees

angles(y) is a user function of the previously defined user function vers(x)

- Third Kepler's Law: orbital period *T* as a function of the average distance *d* from the Sun (*d*, millions of kilometers ; *T*, days)

```
T(d)=2*pi*[(d^3)/(G*M)]^0.5*3.66e8;  G=6.67e-11;  M=1.989e30 ; T((57.9,108.2,149.6))
```

Mercury, Venus, Earth: **(87.963 , 224.71, 365.32)**

$T = 2\pi(d^3/GM)^{1/2}$

d = average distance from the Sun; *G* = universal gravitational constant ; *M* = mass of the Sun ; 3.66e8 = conversion factor

Internal precision and Output precision

- Calculations are performed (and the results stored) with 16 digits
- The number of digits **shown** (5 by default) can be changed from 1 to 15 using the round() command

```
a = ln(7) + 1 / exp(2) ; a + a ^ 2
```

6.4128

```
a = ln(7) + 1 / exp(2) ; a + a ^ 2 ; round(13)
```

6.412827981728

- A round() command can be placed anywhere and remains in effect until a new round() command is issued

- Real rounding of single objects : the round function

```
round(13)  ; round( (pi , pi^2 ) ,7 )
```

(3.141593 , 9.869604)

- the input object (a) is not modified, but the resulting object is actually rounded to 7 digits

Università di Napoli Federico II
Dipartimento di Scienze Chimiche

Lists

Definitions

The operational object of **SC** is the List, defined by:

A structured set of numbers and strings with a name and a literal qualifier specifying how the set will be handled

SC operators (+, -, *, … ln, cos, …) and user functions take as input one or more such lists and create a new output list according to the rules established for the given qualifier(s)

List 1 , List 2 , … —Operator / Qualifiers→ New List

```
x = ( 1 , 2 + 3i ) ; x + ( ( 4 , 5 , 6 ) , ( 1 , 0 , 2i ) , ( 1 , 2 , -1 ) )
```

Name: x
Set: 1, 2 , 1, 2+3i
Qualifier: blank

rows — columns — a 1D-list

Name: internal name
Set: 3, 3, 4 , 5 , 6 , 1 , 0 , 2i , 1 , 2 , -1
Qualifier: blank

a 2D-list

Not implemented

1D-lists

1D-lists are handled on a item-by-item basis

(1, 2, 3) + (1, 0, -1)	**(2, 2, 2)**	(1, 2, 3) ^ (1, 0, -1)	**(1, 1, 0.33333)**
(1, 2, 3) / 2	**(0.5, 1, 1.5)**	(1, 2, 3) + 2	**(3, 4, 5)**
ln (1 , 2 , 3 , i-1)	**(0, 0.69315, 1.0986, 0.3466+2.3562i)**		

1D-list expressions

x=(1, 2, 3, 4) ; 5*x^2 + 2*(ln(x))^3	**(5, 20.666, 47.652, 85.328)**
a=(0, 1, 2) ; b=(6,7,8); c=a^2-b^2	**(-36, -48, -60)**
atan(1, 2, 3) ^ 2 -1	**(-0.38315, 0.22578, 0.56012)**

- Generating 1D-lists : the function list()

list(4)	**(0, 0, 0, 0)**	list(5, 3)	**(3, 3, 3, 3, 3)**
list(6, 1, 2)	**(1, 3, 5, 7, 9, 11)**	list(80, 1, -1, 1, -1)	**(1, -1, 1, -1, 1, -1 , ...)**

# of columns, first member, increment

#of columns, repeating pattern

- Extending and joining 1D-lists

((1, 2, 3), 4, 5)	**(1, 2, 3, 4, 5)**	((1, 2, 3) , (4 , 5))	**(1, 2, 3, 4, 5)**	l_1 <> l_2

- Referencing and setting 1D-lists **a = (1, 2, 3, 4)**

a(2) * a(3)	**6**	a(2) = 5	**(1 , 5 , 3, 4)**
a(2, 4)	**(2 , 3, 4)**	a(2, 4) = (5, 6, 7)	**(1 , 5 , 6, 7)**

- 1D-list specific functions: ils() and ilp()

ils (1 / fact(list(10, 0, 1)))	**2.7183**	internal list sum ($\Sigma_{n=0}^{9} 1/n!$)
ilp ((list(5, 2, 2) ^ 0.5))	**61.968**	internal list product ($\Pi_{n=2}^{10} n^{1/2}$; n even)

1D-lists are handled as vectors in vector-specific functions

vedp((1, 2, 3) , (2, 3, 4))	**20**	vector dot (scalar) product
vecp((1, 2, 3) , (2, 3, 4))	**(-1, 2, -1)**	vector cross (vector) product

- Percent of ^{14}C remaining at various times

```
htc14=5730 ; t = 3000 ; perc = 100 * exp( -t * ln(2) / htc14  )
```

69.565

```
htc14=5730 ; t = ( 1000, 3000, 6000, 10000)  ; perc = 100 * exp( -t * ln(2) / htc14  )
```

(88.606 , 69.565 , 48.393 , 28.829)

- The angle between two vectors

```
a = (1, 1, 0) ; b = (0, 1, 1) ; deg{ acos [ vedp(a,b) / ( val(a) * val(b) ) ] }
```

60

- Sum of the 5-th powers of integers from 6 to 250

```
ils( list( 245, 6, 1 ) ^ 5 ) ; round(14)
```

41180013011200

- First 7 terms of the series expansion of cos(0.5)

$\cos(x) = 1 - x^2/2! + x^4/4! - \ldots$

```
x = 0.5 ; n = 7 ; a = list(n,0,2) ; list(n,1,-1,1,-1) * [ x ^ a / fact(a) ]
```

(1 , -0.125 , 0.0026042 , -2.1701e-05 , 9.6881e-08 , -2.6911e-10 , 5.0969e-13)

Polynomials

When their qualifier is "p", 1D-lists are handled as polynomials

pol(1, 2, 3) + pol(1, 1)	**pol(2, 3, 3)**

$(1+2x+3x^2) + (1+x) = (2 + 3x + 3x^2)$

pol(1, 2) * pol(2, 2, 3)	**pol(2, 7, 9, 6)**

$(1+2x) * (2+2x+3x^2) = (2 + 7x + 9x^2 + 6x^3)$

pol(1, 1) ^ 7	**pol(1, 7, 21, 35, 35, 21, 7, 1)**

pol(1, 5, 10, 10, 5, 1) / pol(1, 2, 1)	**pol(1, 3, 3, 1)**

$(1+5x+10x^2 + 10x^3 + 5x^4 + x^5) / (1+2x +x^2) = (1 + 3x + 3x^2 + x^3)$

pol(1, 1) ^ 5 / pol(2, 1)	**((1, 2, 4, 3, 1) , (-1, 0, 0, 0, 0))**
pol(1, 1) ^ 5 / pol(1, 2, 5, 3, 1)	**((2, 1, 0, 0) , (-1, 0, -2, -1))**

remainder

2D-lists

2D-lists are always handled as matrices

[(1, 2, 3) , (4, 5, 6)] * [(0, 1) , (2, 2) , (1, -1)]	**((7, 2) , (16, 8))**	conformable
[(1 , 1-2i) , (0 , 1)] ^ 4	**((1, 4-8i) , (0, 1))**	square
3 / [(1, 2) , (4, 5)]	**((-5, 2) , (4 , -1))**	square and invertible

in 1D-list/matrix operations, 1D-lists are handled as row/column vectors

[(1 , 2) , (3 , 4) , (5 , 6)] * (1 , 1)	**(3, 7, 11)**
(1, 1, 1) * [(1 , 2) , (3 , 4) , (5 , 6)]	**(9, 12)**

- Generating matrices : list() and diag()

(list(4) , list(4,1))	**((0, 0, 0, 0) , (1, 1, 1, 1))**
(list(3, 1, 2) , list(3, 1, 2, 1) , list(3, 2i))	**((1, 3, 5) , (1, 2, 1) , (2i, 2i, 2i)**
diag(0.1, 2+i, 3)	**((0.1, 0, 0) , (0, 2 + i, 0) , (0, 0, 3))**

a = ((1, 2) , (4, 5) , (0, 3))
b = (8, 9)

- Extending and joining matrices

(a , b)	**((1, 2) , (4, 5) , (0, 3) , (8, 9))**
(a , a)	**((1, 2) , (4, 5) , (0, 3) , (1, 2) , (4, 5) , (0, 3)))**

- Referencing and setting matrices

a (3 , 2)	**3**	a (3 , 2) = 7	**((1, 2) , (4, 5) , (0, 7))**
a (2)	**(4 , 5)**	a (2) = b	**((1, 2) , (8, 9) , (0, 3))**
a(0 , 2)	**(2 , 5 , 3)**	a(0 , 2) = (6, 7, 8)	**((1, 6) , (4, 7) , (0, 8))**
a(1 , 1 , 2 , 2)		**((1, 2) , (4, 5))**	
a(1 , 1 , 2 , 2) = ((0, 0) , (1, 1))		**((0, 0) , (1, 1) , (0, 3))**	

- Matrix-specific functions: transp() and diag()

transp(a)	**((1, 4, 0) , (2, 5, 3))**	transpose
diag(1, 2, 3)	**((1, 0, 0) , (0, 2, 0) , (0, 0, 3))**	
m = ((1, 2, 3) , (2, 0, 2) , (4, -1, 7)) ; diag(m)	**(1, 0, 7)**	diagonal

Università di Napoli Federico II
Dipartimento di Scienze Chimiche

Special functions

Special functions

- val (number / list)

val(-pi)	**3.1416**	absolute value
val(i+1)	**1.4142**	complex magnitude = $(c \cdot c^*)^{1/2}$
val(1, -2, 3, -3)	**4.7958**	1D-list norm = $(\Sigma x_i^2)^{1/2}$
val((1,-2, 3) , (1, 2, 3), (1, 1, 1))	**-8**	matrix determinant

- val (polynomial , a=number / 1D-list): polynomial value for x=a

p = pol (2, 1, 2) ; val(p , 3)	**23**
a = (2, pi, 5, 5+i) ; val(pol(2, 1, 2) , a)	**(12, 24.881, 57, 55+21i)**

Warning: The result is rounded such that absolute values of real or imaginary terms < 10^{-10} are set to 0

- solve (2D-list, conformable 1D-list): linear systems

$$\begin{aligned} x + y + z &= 0 \\ x + 3y + 2z &= 2 \\ 2x + y - z &= 3 \end{aligned}$$

```
solve [ ((1, 1, 1), (1, 3, 2), (2, 1, -1)) , (0, 2, 3) ]
```

(-0.2, 1.8, -1.6)

```
a = (( 1, 2, 1) , ( 2, 4, 1-3i), ( 0, 3, 2) ) ; b = solve ( a, ( 1, 1, 1) )
```

(0.36667 – 0.1i , 0.26667 + 0.2i , 0.1 – 0.3i)

```
a = (( 1, 2, 1) , ( 2, 4, 1-3i), ( 0, 3, 2) ) ; b = solve ( a, ( 1, 1, 1) ) ; a*b
```

(1, 1, 1)

- When the linear system has infinite solutions, a simple one is given

```
a = (( 1, 2, 1) , ( 2, 4, 2), ( 3, 6, 3) ) ; solve ( a, ( 1, 2, 3) )
```

(2, -1, 1)

- solve (polynomial) : polynomial roots

```
solve( pol (2, 1, 2, 1 ) )
```

(-2, -i, i)

- Roundoff errors accumulate in ill-conditioned cases, increasingly with increasing number of closely spaced roots
- The solving routine is unable to distinguish between very closely spaced roots

- stat (1D-list): statistics of list members

```
y = ( 2, 2.02, 1.97, 2.02, 1.99, 2.02, 1.98, 1.8, 2.2) ; c=stat(y)
```

(2 , 0.033871 , 0.10161 , 0.057778 , -0.0012709 , 0.33676)

a = average

σ_a

σ mean absolute deviation

skewness

kurtosis

deviations from the Gaussian distribution

skewness = $(1/N) \Sigma [(x_i - a) / \sigma]^3$; > 0 , positive tail ; < 0 , negative tail

kurtosis = $(1/N) \Sigma [(x_i - a) / \sigma]^4 - 3$; > 0 , peaked ; < 0 , flat

- lfit (1D-list , 1D-list , optional 1D-list): linear fitting

x values

y values

σ_y (when known): else $\sigma_y = 0.01 * y$

```
lfit ( (1, 2, 3, 4, 5) , (11, 12, 13, 14, 15) , (0.1 , 0.2 , 0.3 , 0.4 , 05 ))
```

(10, 1, 0.17161 , 0.10226, 0 , 1)

a = intercept

b = slope

σ_a

σ_b

correlation coefficient

$\chi_r^2 = \Sigma [(y_i - a - bx_i) / \sigma_i]^2/(N-3)$

$\chi_r^2 >> 1$ (low probability to observe the input data)
$\chi_r^2 \approx 1$ (the input data can be explained by the model)
$\chi_r^2 << 1$ (overfitting : the input data are too good to be true)

Looney's law: $y = \text{atan}(a+b*\ln(x))$, i.e. $\tan(y)=a + b*\ln(x)$

```
x= (1, 2, 3, 4, 5); y= (-0.79  , 0.37 , 0.87 , 1.06 , 1.15); lfit(ln(x), tan(y))
```

(-1.0064, 2.0091, 0.0080238, 0.010496, 1.0901, 0.99998)

- R. Boyle's original *P*-*V* data (*P* inches Hg, *V* arbitrary units)

```
P = (29, 35.3, 47, 70.7, 93 ) ;  V = (48, 40, 30, 20, 15 ) ; f=lfit( P , V )
```

(52.139 , -0.41272 , 0.37495 , 0.0048473 , 337.13 , -0.95919)

```
P = (29, 35.3, 47, 70.7, 93 ) ;  V = (48, 40, 30, 20, 15 ) ; f=lfit( ln(P) , V )
```

(131,19 , -25.798 , 1.2502 , 0.29296 , 84.81 , -0.99052)

```
P = (29, 35.3, 47, 70.7, 93 ) ;  V = (48, 40, 30, 20, 15 ) ; f=lfit( 1 /P , V )
```

(-0.027354 , 1406 , 0.26343 , 15.797 , 1.0725 , 0.99985)

```
P = (29, 35.3, 47, 70.7, 93 ) ;  V = (48, 40, 30, 20, 15 ) ; f=lfit( 1 /P , V ) ; f(1) + f(2) / P
```

(48.455 , 39.802 , 29.887 , 19.859 , 15.091)

- PV = 1406±16 a.u.
- Setting σ_V=(0.5, 0.5, 0.5, 0.5, 0.5), PV = 1394±25 a.u.

- eigenv(matrix): eigenvalues and eigenvectors (2*2 or 3*3 square matrices)

```
a = (( 1, 0, 1) , ( 0, 1, 1), ( 1, 1, 0) ) ; eigenv ( a )
```

((-1, 1, 2) , (1 , 1 , -2) , (-1 , 1, 0) , (1 , 1 , 1))

eigenvalues

eigenvectors

$\mathbf{a} * \mathbf{x}_i = \lambda_i * \mathbf{x}_i$; $\mathbf{a}*\mathbf{x}_i - \lambda_i*\mathbf{x}_i = \mathbf{0}$

```
a = (( 1, 0, 1) , ( 0, 1, 1), ( 1, 1, 0) ) ; b = eigenv ( a ) ; a*b(2)-b(1,1)*b(2)
```

(0 , 0 , 0)

matrix diagonalization

```
b = eigenv [((3, 2, 4) , (2, 0, 2), (4, 2, 3))] ; d = transp( b(2), b(3), b(4) ) ; d*diag(b(1))* (1/d)
```

((3, 2, 4) , (2, 0, 2), (4, 2, 3))

Università di Napoli Federico II
Dipartimento di Scienze Chimiche

Files, libraries and scripts

Storing lines

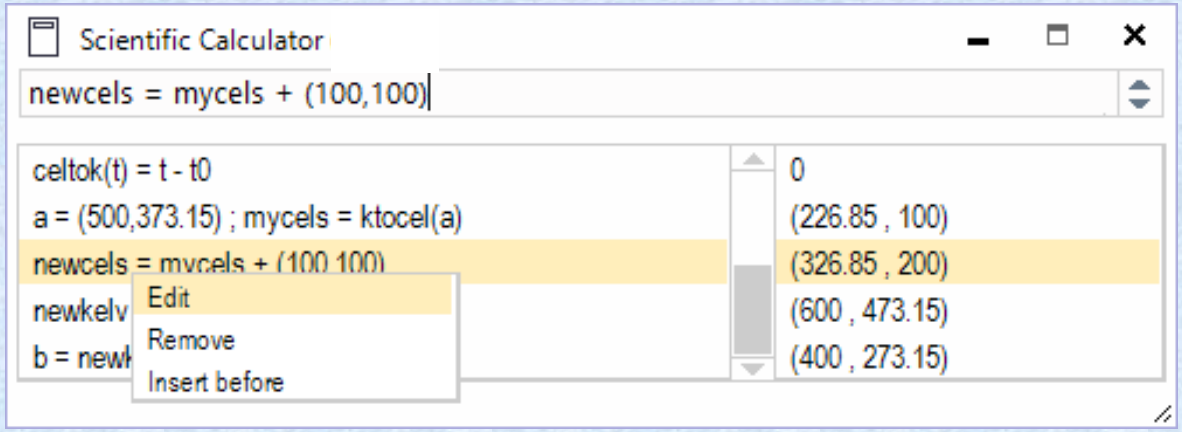
- Within a line, sublines are evaluated using a volatile memory, cleared after each line evaluation; however, names and functions defined in a line can be stored

- Stored lines are added to a persisting memory, and their content is available anywhere until the memory is erased by clicking Erase Memory (Tools menu)
- When the memory is not empty, the icon changes; clicking the icon's center square shows/hides the memory window

- The right side shows 0 or the last evaluated result of each line

- The memory window length is practically unlimited; with more than 5 lines, a vertical bar allows scrolling

- Clicking left in the left part of the window selects an item; when an item is selected, clicking right shows a menu allowing to edit/remove the selected item or to insert a blank line before

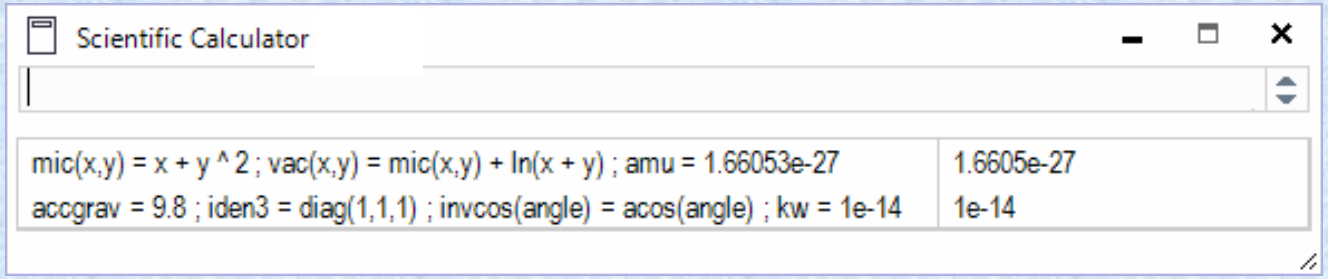


- When Edit is clicked, the textbox turns yellow; pressing Enter, the stored line is modified and all stored lines are recalculated

- Saving to a .ssc file with File, Save as creates an image of the current status, including the persisting memory

Libraries

When the starting directory contains .ssc files with name ending deflib.ssc, the first (in ascending alphanumeric order) is automatically loaded



vacatello deflib.ssc

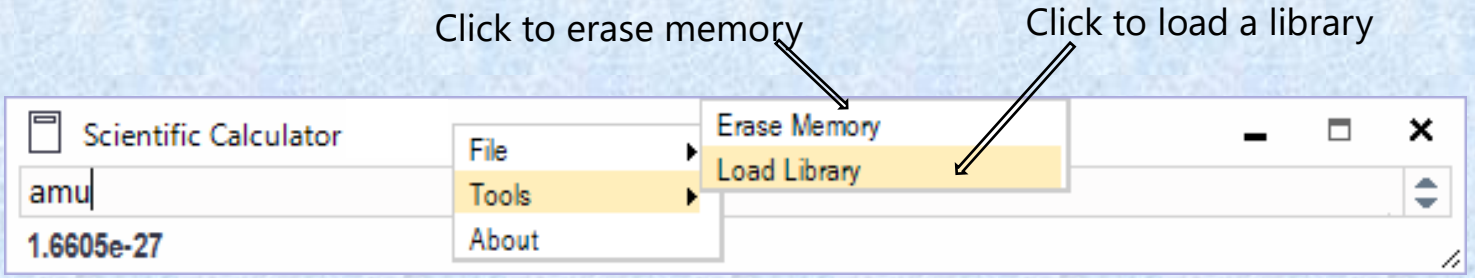
- The loaded library is user-transparent, but all entities and functions defined in the memory window when the file was saved are immediately available at program start

12.011 * amu * 1000	**1.9945e-23**	vac(1 , 3)	**11.386**

- Default library constants and functions are handled as intrinsics

kw = 3	**Reserved name: kw**
vac(a,b,c) = a+b*c	**Redefining library function: vac**

- Several libraries can be simultaneously loaded; conflicting items of additional libraries are handled on a First In, First Out basis

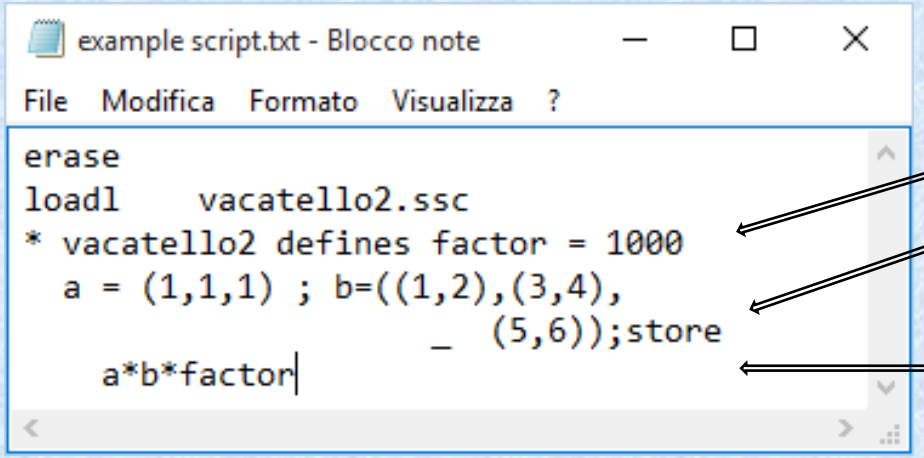


- Libraries can be opened, edited and saved like any other .ssc file

Users can define their own personalized sets of constants and functions

Scripts

Scripts are simple text files, prepared with any text editor



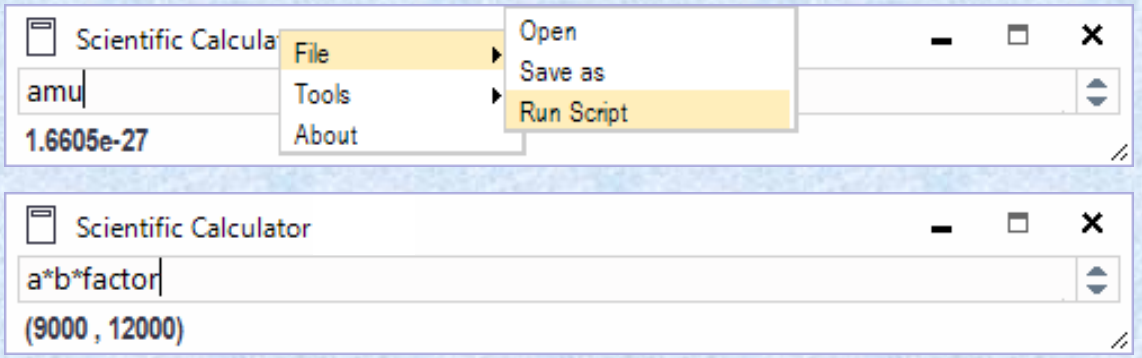
Lines starting with * are comments

Lines starting with underscore (_) are continuation lines

Spaces and blank lines can be freely inserted to improve readability

Commands: erase (Erase Memory) loadl (Load Library)

Running scripts:



- A simple (5 lines) script for a complex calculation with matrices and derivatives: the conformational entropy of polymethylene chains of n methylene groups (R units) at temperature T (K) per mol of CH_2 groups as given by Flory's RIS model

peflory.txt - Blocco note

File Modifica Formato Visualizza ?

```
* define s(T), o(T) and the 3*3 matrix m(T) at temperature T (K)
 s(T) = exp( - 250 / T) ; o(T) = exp( - 1000 / T) ; store
 m(T) = [ (1, s(T), s(T)) , (1, s(T), s(T)*o(T)) , (1, s(T)*o(T), s(T)) ] ; store

* define the conformational partition function Z(T,n) for a chain of n
* methylene groups as the sum of the first row elements of m(T)^(n-3)
 Z(T,n) = ils( ( m(T)^(n-3) )(1) ) ; store

* define the conformational entropy (R units) per mol of methylene
* groups as (1/n)*(ln(Z) + d(lnZ)/d(ln(T)))
 Ent(T,n) = (1/n) * { ln(Z(T,n)) + [ ln(Z(T+1,n))-ln(Z(T,n)) ]
        _  / [ ln(T+1) - ln(T) ] } ; store
```

Scientific Calculator

```
ent(400,500)
0.88161
```

Università di Napoli Federico II
Dipartimento di Scienze Chimiche

A note about files, libraries and scripts

- When a .ssc file is opened with File, Open, the status is restored exactly as it was when the file was saved

- When a .ssc file is loaded with Load library, all constants and function defined in the library are placed in the permanent memory and are immediately available, but they are not added to the memory window. Hence, a .ssc file saved after loading a library does not include the memory window of the original library. The file can be then opened with File, Open, but the definitions of the original library are lost when it is loaded as a library

- The same is true for scripts, since constants and functions defined in a script are placed in the permanent memory, but are not added to the memory window

Operation Tables : +,-

+,-	Number	1D-list	Pol	Matrix
Number				NI
1D-list		C	NI	NI
Pol		NI		NI
Matrix	NI	NI	NI	C

C: conformable 1D-list/matrix

Relevant Examples:

Number + Pol: 1- pol(2,3,4) = pol(-1,-3,-4)

Pol + Pol: pol(1,2) + pol(3,4,5) = pol(4,6,5)

1D-list + 1D-list: (1,2,3) + (4,5,6) = (5,7,9)

Matrix + Matrix: ((1,2),(3,4)) + ((5,6),(7,8)) = ((6,8),(10,12))

Operation Tables : *

*	Number	1D-list	Pol	Matrix
Number				
1D-list			NI	C
Pol		NI		NI
Matrix		C	NI	C

C: conformable 1D-list/matrix

Relevant Examples:

Pol*Pol: pol(1,2) * pol(3,4,5) = pol(3,10,13,10)

Matrix/1D-list * Matrix/1D-list:

(1,2,3) * (4,5,6) = (4,10,18)

((1,2,3),(3,4,5)) * ((1,1),(2,2),(3,3)) = ((14,14),(26,26))

Operation Tables : /

/	Number	1D-list	Pol	Matrix
Number			NI	I
1D-list		NI	NI	I
Pol		NI	R	NI
Matrix		NI	NI	I

I: invertible square matrix
R: check for reminder

Relevant Examples:

Pol / Pol: pol(1,3,3,1) / pol(1,1) = pol(1,2,1)

pol(1,2,3) / pol(4,5) = ((-0.08,0.6) , (1.32,0))

i.e. $(1+2x+3x^2)/(4+5x) = -0.08+0.6x$ with remainder 1.32

1 / ((1,2),(3,4)) = ((-2,1),(1.5,-0.5)) inverse matrix

((1,2),(3,4)) / ((1,2),(2,2)) = ((1,0),(1,1))

Operation Tables : ^

^	Number	1D-list	Pol	Matrix
Number			NI	NI
1D-list			NI	NI
Pol	I>0	NI	NI	NI
Matrix	I,S	NI	NI	NI

I>0: positive integer power

I,S: integer power of square matrix (invertible for negative power)

Relevant Examples:

pi ^ (1,2,3) = (3.1416, 9.8696, 31.006)

(1,2,3) ^ 3 = (1, 8, 27)

pol(2,1) ^ 4 = pol(16, 32, 24, 8, 1)

((1,2),(1,1)) ^ (-2) = ((3,-4),(-2,3))