Programming in Scilab - Sections 1, 2 and 3

João L. R. Neto

July 6, 2020

Contents

List of Figures

List of Tables

1 Introduction

The Scilab is open-source software for scientific computing. It includes hundreds of predefined mathematical functions and a high-level programming language, allowing access to advanced data structures and graphical functions in 2 and 3 dimensions. It has many features such as control, simulation, optimization, signal processing, and Xcos, a model and simulator of hybrid dynamic systems that are provided with the platform.

1.1 Download and installation

Figure 1: 2020-05-28_13-39-17

https://www.scilab.org/

2 Getting started with Scilab

2.1 Scilab interface

2.1.1 The basic interface activated when starting Scilab

Figure 2

2.1.2 The basic interface and the script editor activated from the activation of the highlighted icon

Figure 3

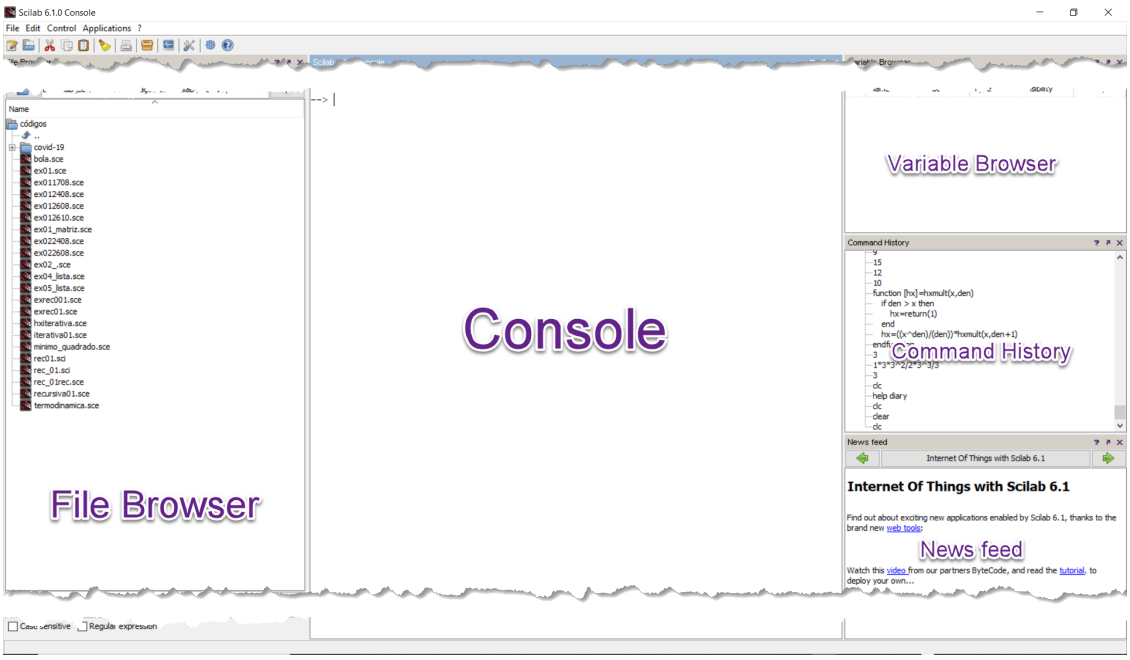

Figure 2: 2020-05-26_17-34-04

2.1.3 Components of the basic interface menu items (click in the console area)

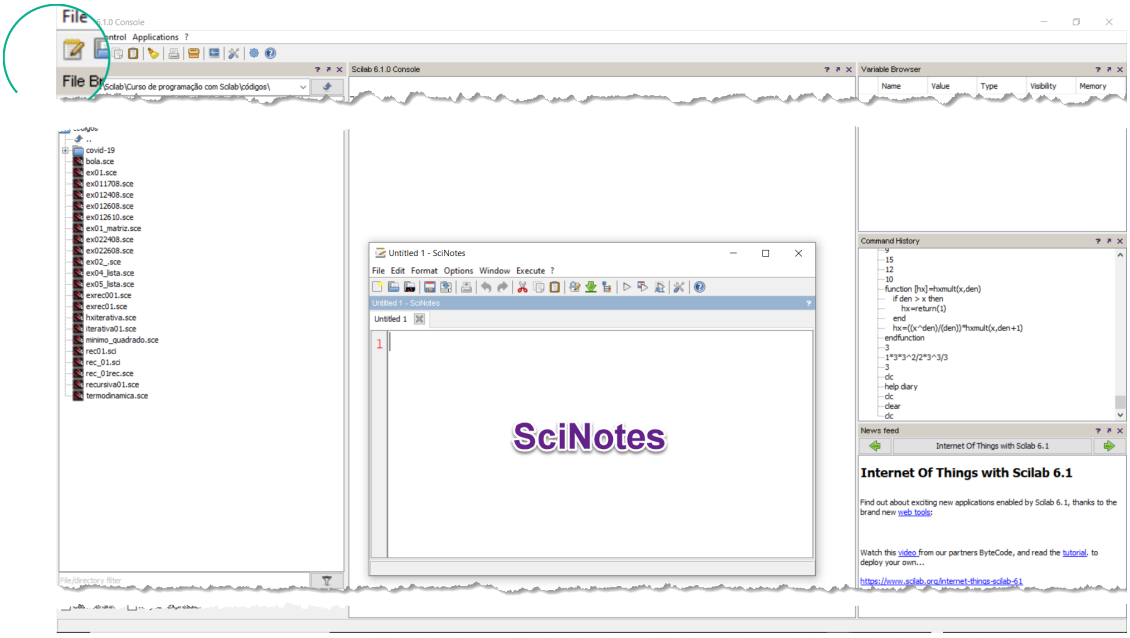

Figure 3: 2020-05-26_17-40-00

2.1.4 File

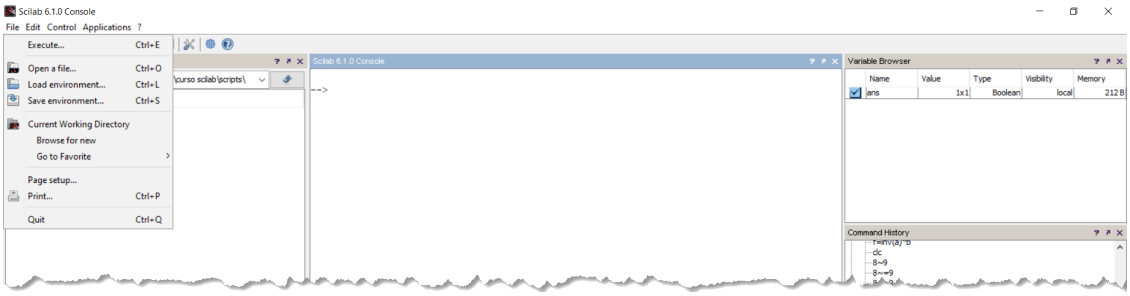

Figure 4: 2020-05-27_15-18-35

- Run or Ctrl + E: Run script files
- Open a File or Ctrl + O: Load script files
- Load environment or Ctrl + L: Load binary files (of variables) saved with save
- Save environment or Crtl + S: Saves a binary file containing variables
- Current Working Directory: Change working directory
- Page setup or Ctrl + P: Print scripts
- Quit or Crtl + Q: Close the section and exit the Scilab environment

2.1.5 Edit

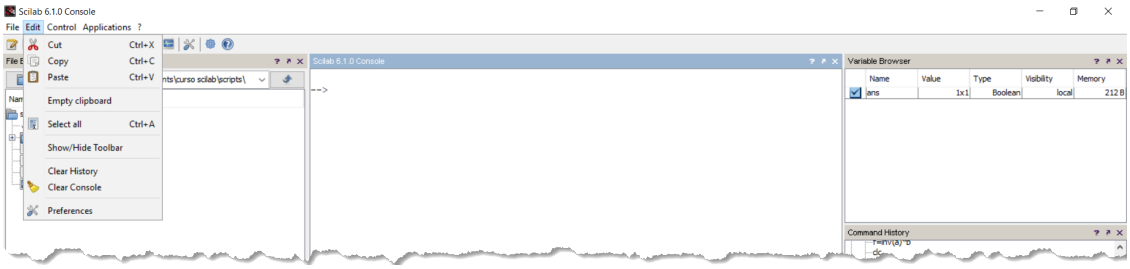

Figure 5: 2020-05-27_15-43-34

- Cut or Ctrl + X: 'Cut' a text
- Copy or Ctrl + C: Copy selected text to the clipboard
- Paste or Ctrl + V: 'Paste' what was copied
- Empty clipboard: 'Clean' the clipboard
- Select all or Ctrl + A: Select all the current text in the environment
- Clear History: 'Clear' the history area
- Clear Console: 'Clear' the console area
- Preferences: Customize the Scilab environment

2.1.6 Control

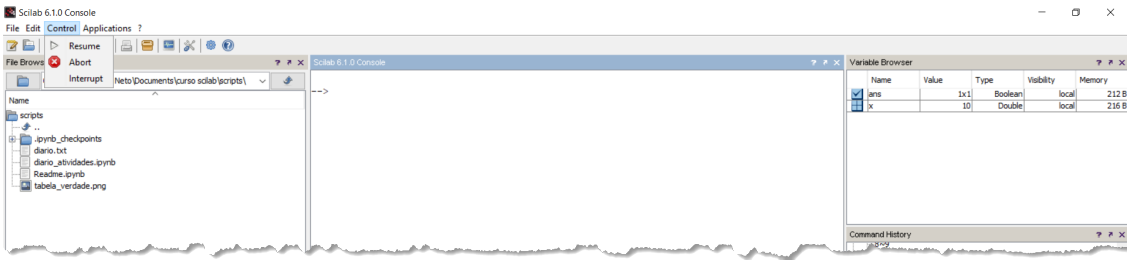

Figure 6: 2020-05-27_16-05-18

- Resume: The execution of the execution of an instruction continues after a pause or due to a stop
- Abort: Stop the execution of a process
- Interrupt: Stop a process, equivalent to Ctrl + C

2.1.7 Applications

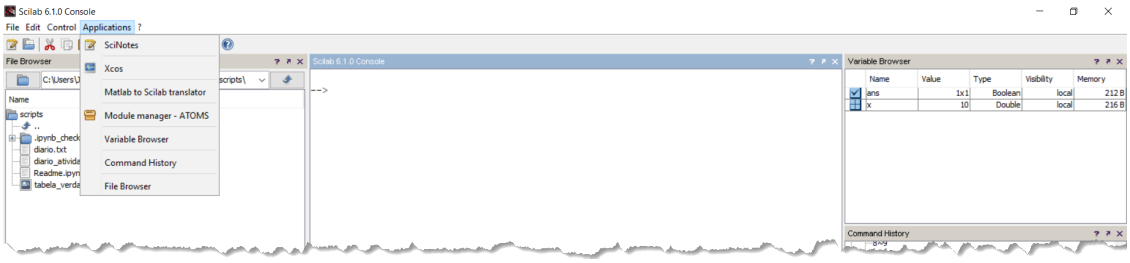

Figure 7: 2020-05-27_16-17-05

- SciNotes: Load the script editor (text editor)
- Xcos: Load the modeler and simulator of hybrid dynamic systems, allowing to create block diagram and graphical interfaces
- Matlab to Scilab translator: Code conversion options from Matlab to Scilab
- Variable Browser: View the variable browser
- Command History: View the command history
- File Browser: View the file and folder browser

2.1.8 Help (?)

Figure 8: 2020-05-27_16-46-02

- Scilab Help or F1: Scilab resources reference
- Scilab Dmonstrations: Application demonstrations with Scilab
- News feed: News about Scilab
- Link: Links about Scilab
- About Scilab or Shift + F1: About Scilab

2.2 Start a diary session

[1]:
```
x=diary('diary.txt')
```

```
 x  =

   1.
```

[2]:
```
a=10
b=20
c=a+b
```

```
 a  =

   10.

 b  =

   20.

 c  =

   30.
```

Note: Check the creation of the file 'diary.txt' in the folder where you are working

3 Key Scilab items

3.1 Constants

Constants do not change the value during the execution of an algorithm

3.1.1 Predefined Constants

The value of the constant π = 3.1415927

[3]:
```
%pi
```

```
 %pi  =

   3.1415927
```

Base of natural logarithms e = 2.7182818

[4]:
```
%e
```

```
 %e  =

   2.7182818
```

Imaginary unit; square root of -1

[5]:
```
%i
```

```
 %i  =

   0. + i
```

Infinite

[6]:
```
%inf
```

```
 %inf  =

   Inf
```

True logical value

[7]:
```
%t
```

```
ans  =

 T
```

False logical value

[8]:
```
%f
```

```
ans  =

 F
```

not a number

[9]:
```
%nan
```

```
%nan  =

  Nan
```

[10]:
```
%eps
```

```
%eps  =

  2.220D-16
```

Scilab accuracy

[11]:
```
%s
```

```
%s  =

 s
```

Polynomial with a root at zero and variable s

[12]:
```
%z
```

```
%z  =

 z
```

Polynomial with a root of zero and variable z

3.2 Variables

They change the value during the execution of an algorithm. Variables are created dynamically. By assigning (operator =) a value to a valid identifier the variable is created.

Creating a variable represents referencing a space in main memory (RAM).

3.2.1 Valid identifiers

Characters from a ... z and A ... Z. Combinations of letters and numbers as long as it starts with a letter. Combinations with special characters #,!, $, _. Other special characters are not allowed.

[13]:
```
a=10
A=5
a20=100
```

```
a  =

  10.

A  =

  5.

a20  =

  100.
```

[14]:
```
x!=10
```

```
x!  =

  10.
```

[15]:
```
c#3=30
```

```
c#3  =

  30.
```

[16]:
```
v$=0.3
```

```
v$  =

  0.3
```

[17]:
```
a 9=5
```

```
a 9=5
   ^^
Error: syntax error, unexpected =, expecting end of file
```

Note: White space, for example, is not a valid special character in the character combination to create a variable

3.2.2 Data Types

Numeric

[18]:
```
a1=27
a2=4.56
```

```
a1  =

  27.

a2  =

  4.56
```

[19]:
```
a3 = 4+%i
```

```
a3  =

  4. + i
```

String and character

[20]:
```
phrase = "This is a string variable"
```

```
phrase  =

 "This is a string variable"
```

[21]:
```
phrase = 'This is a string variable'
```

```
phrase  =

 "This is a string variable"
```

[22]:
```
character = "B"
```

```
character  =

 "B"
```

[23]:
```
character = 'B'
```

```
character  =

 "B"
```

Logical: T (True) and F (False)

[24]:
```
option = %f
```

```
option  =

 F
```

[25]:
```
option = %t
```

```
option  =

 T
```

Arrays : Assign data to an identifier in square brackets (space or comma is column and semicolon is line)

[26]:
```
matrix1 = [3 2 6 4;7 4 8 3;1,2,3,4]
```

```
matrix1  =

  3.   2.   6.   4.
  7.   4.   8.   3.
  1.   2.   3.   4.
```

[27]:
```
matrix2 = ["One","matrix";"with two rows","two columns"]
```

```
matrix2  =

 "One"            "matrix"
 "with two rows"  "two columns"
```

[28]:
```
matrix3 = [4 5 "string"]
```

```
Undefined operation for the given operands.
check or define function %s_c_c for overloading.
```

Lists

[29]:
```
list1 = list("personal data", ["name"; "address"],[1250.45 45])
```

```
 list1  =


      list1(1)

 "personal data"

      list1(2)

 "name"
 "address"

      list1(3)

  1250.45   45.
```

ans variable When we do not create any identifier, Scilab assigns data to a variable called 'ans', of answer. The 'ans' variable will always contain the content of the last operation performed.

[30]:
```
345
```

```
 ans  =

   345.
```

[31]:
```
"Another example"
```

```
 ans  =

  "Another example"
```

[32]:
```
[4 3 6;6 4 7]
```

```
 ans  =

   4.   3.   6.
   6.   4.   7.
```

[33]:
```
list(["name"],[5 3])
```

```
ans  =

     ans(1)

 "name"

     ans(2)

  5.   3.
```

3.3 Arithmetic expressions

3.3.1 Arithmetic operators

Addition (+)

Subtraction (-)

Multiplication (*)

Division (/) Mumerator / Denominator

Division (\ ) Denominator / Numerator

[34]:
```
x=20
```

```
x  =

  20.
```

[35]:
```
x=x+5
```

```
x  =

  25.
```

[36]:
```
y=10
```

```
y  =
```

```
  10.
```

[37]:
```
z=x-y
```

```
z  =
```

```
  15.
```

[38]:
```
a=5
b=10
```

```
a  =
```

```
  5.
```

```
b  =
```

```
  10.
```

[39]:
```
c=a*b
```

```
c  =
```

```
  50.
```

[40]:
```
d=a/b
```

```
d  =
```

```
  0.5
```

[41]:
```
e=a\b
```

```
e  =

  2.
```

3.3.2 Scalar operations by an array

Note: When creating a matrix - space or comma is column change, and a semicolon is a new line.

[42]:
```
m=[3 4 5 6 7 8]
```

```
m  =

  3.   4.   5.   6.   7.   8.
```

[43]:
```
n=5+m
```

```
n  =

  8.   9.   10.   11.   12.   13.
```

[44]:
```
l=[4;6;2;8]
```

```
l  =

  4.
  6.
  2.
  8.
```

[45]:
```
u=3+1
```

```
u  =

   7.
   9.
   5.
   11.
```

3.3.3 Matrix operations

Addition. The matrices must be the same size (equal number of rows and columns).

[46]:
```
a=[ 4 5 6 1;8 9 0 1]
```

```
a  =

   4.   5.   6.   1.
   8.   9.   0.   1.
```

[47]:
```
b=[9 1 5 3;7 1 0.5 9 ]
```

```
b  =

   9.   1.   5.    3.
   7.   1.   0.5   9.
```

[48]:
```
c=a+b
```

```
c  =

   13.   6.    11.   4.
   15.   10.   0.5   10.
```

[49]:
```
e=[9 4; 9 0; 2 1;8 6]
```

```
e  =

   9.   4.
   9.   0.
```

```
    2.    1.
    8.    6.
```

[50]:
```
f=a+e
```

```
Inconsistent row/column dimensions.
```

Subtraction. The same addition rules.

[51]:
```
x=[5 3 6;6 3 9]
```

```
 x  =

    5.    3.    6.
    6.    3.    9.
```

[52]:
```
y=[7 1 9; 0 3 1]
```

```
 y  =

    7.    1.    9.
    0.    3.    1.
```

[53]:
```
z=x-y
```

```
 z  =

   -2.    2.   -3.
    6.    0.    8.
```

[54]:
```
p=[4 2;5 6;9 3]
```

```
 p  =

    4.    2.
    5.    6.
    9.    3.
```

[55]:
```
w=z-p
```

```
Inconsistent row/column dimensions.
```

Matrix multiplication. Number of columns in one matrix must equal the number of rows in the other matrix.

[56]:
```
a=[4 6 1 4]
```

```
a  =

   4.   6.   1.   4.
```

[57]:
```
b=[5 3 7;6 4 9;1 2 3;6 4 5]
```

```
b  =

   5.   3.   7.
   6.   4.   9.
   1.   2.   3.
   6.   4.   5.
```

[58]:
```
c=a*b
```

```
c  =

   81.   54.   105.
```

[59]:
```
d=[5 5 2;5 6 4]
```

```
d  =

   5.   5.   2.
   5.   6.   4.
```

[60]:
```
e=[2 3 4;8 6 7]
```

```
e  =

   2.   3.   4.
   8.   6.   7.
```

[61]:
```
f=d*e
```

```
Inconsistent row/column dimensions.
```

Point-to-point multiplication (. *). Matrices of the same size.

[62]:
```
x=[1 2 3;5 4 6;8 7 9]
y=[5 4 7;1 2 3;-9 4 0]
```

```
x  =

   1.   2.   3.
   5.   4.   6.
   8.   7.   9.

y  =

   5.   4.   7.
   1.   2.   3.
  -9.   4.   0.
```

[63]:
```
z=x.*y
```

```
z  =

   5.    8.    21.
   5.    8.    18.
  -72.   28.   0.
```

Matrix division. The division operation will be the multiplication of the inverse of one matrix by the other matrix. Notes:

1) In this example we are using the rand () function. Randomly generates values.
2) We also use the inv () function, which calculates the inverse of a matrix.

[64]:
```
x=rand(3,3)
```

```
x  =

  0.2113249   0.3303271   0.8497452
  0.7560439   0.6653811   0.685731
  0.0002211   0.6283918   0.8782165
```

[65]:
```
y=rand(3,1)
```

```
y  =

  0.068374
  0.5608486
  0.6623569
```

[66]:
```
z=x\y
```

```
z  =

 -0.3561912
  1.7908789
 -0.5271342
```

[67]:
```
z=inv(x)*y
```

```
z  =

 -0.3561912
  1.7908789
 -0.5271342
```

Point-to-point division (./ and .\ ). Matrices of the same size.

[68]:
```
a=rand(3,3)
b=rand(3,3)
```

```
a  =

   0.7263507   0.2320748   0.8833888
   0.1985144   0.2312237   0.6525135
   0.5442573   0.2164633   0.3076091

b  =

   0.9329616   0.3616361   0.4826472
   0.2146008   0.2922267   0.3321719
   0.312642    0.5664249   0.5935095
```

[69]:
```
c=a./b
```

```
c  =

   0.7785429   0.6417357   1.8302992
   0.9250403   0.7912479   1.964385
   1.7408324   0.3821571   0.5182884
```

[70]:
```
d=a.\b
```

```
d  =

   1.2844507   1.558274    0.5463588
   1.0810339   1.2638265   0.5090652
   0.5744378   2.6167252   1.9294277
```

Power with matrices (^ and .^)

[71]:
```
x=rand(2,3)
```

```
x  =

   0.5015342   0.2693125   0.4051954
   0.4368588   0.6325745   0.9184708
```

[72]:
```
y=x^2
```

```
at line    20 of function %s_pow ( C:\Program
Files\scilab-6.1.0\modules\overloading\macros\%s_pow.sci line 32 )
at line     3 of function %s_p_s ( C:\Program
Files\scilab-6.1.0\modules\overloading\macros\%s_p_s.sci line 15 )

%s_pow: Wrong size for input argument #1: Square matrix expected.
```

[73]:
```
y=x.^2
```

```
 y  =

   0.2515365   0.0725292   0.1641833
   0.1908456   0.4001505   0.8435886
```

[74]:
```
a=rand(3,3)
```

```
 a  =

   0.0437334   0.4148104   0.7783129
   0.4818509   0.2806498   0.211903
   0.2639556   0.1280058   0.1121355
```

Note: With a square matrix it is possible to use the operator (^). Noting that a matrix multiplication will occur.

[75]:
```
b=a^2
```

```
 b  =

   0.4072294   0.2341861   0.2092143
   0.2122373   0.3057659   0.4582631
   0.1028222   0.1597703   0.2451392
```

3.4 Logical expressions

3.4.1 Relational operators

The logical operators, list two objects (constants, variables, expressions) and return False (F) or True (T). They are also called relational operators.

Greater (>)

Minor (<)

Greater than or equal (> =)

Less than or equal (<=)

Equal (==)

Different (~ =) or (<>)

Note: The equality relational operator is (==), different from the assignment operator (=).

[76]:
```
a=5
b=6
```

```
a  =

   5.

b  =

   6.
```

[77]:
```
c=a>b
```

```
c  =

  F
```

[78]:
```
d=a<b
```

```
d  =

  T
```

[79]:
```
e=a>=10
```

```
e  =

 F
```

[80]:
```
g=a~=b
```

```
g  =

 T
```

3.4.2 Logical operators

Tabela 1: A proposition can be a simple relation, an expression. In the Table 1 we consider two propositions any P and Q. These propositions can assume the values True (V) or False (F). In this way, all possible combinations are indicated in the first and second columns of the table.

Table 1: Logical Operators

Proposition P	**Proposition** Q	**P and Q** P && Q	**P or Q** P || Q	**notP** ∼ P	**notQ** ∼ Q
V	V	V	V	F	F
V	F	F	V	F	V
F	V	F	V	V	F
F	F	F	F	V	V

[81]:
```
2>3 && 5<4
```

```
ans  =

 F
```

[82]:
```
2>3 & 5<4
```

```
ans  =

 F
```

[83]:
```
x=10
y=3
```

```
x  =

  10.

y  =

   3.
```

[84]:
```
x>y || x==y
```

```
ans  =

 T
```

[85]:
```
x>y | x==y
```

```
ans  =

 T
```

[86]:
```
~(x>y) || (x==y)
```

```
ans  =

 F
```

References

[1] Mário Leite. *Scilab: Uma abordagem Prática e Didática*. Editora Ciência Moderna, Rio de janeiro, 2009.

[2] https://scilab.org. Consulted on 05/28/2020.