Scilab Quick Summary

Literal

Floating point:	2.5	-3.75	1e6 (=10^6)	3.5e10	-2.7e-5
Integer:	2	10	-5		
Special:	%i (square root of -1)		%pi (π)	%e (base of natural log)	

Operator

=	Assignment
[ ]	designates matrix beginning and end
( )	parenthesis, brackets subscripts, brackets arguments of functions
‘ ‘	delimiters for character string
, or space	separates elements in row of an array
,	separates arguments in function call
; or return	separates rows of an array, end of statement
...	continues command on next line
//	designates following characters as comments
:	designate a range of values in row
*	multiplication
/	right division
\	left division (a\b = b/a = $a^{-1}b$), Solve matrix equation (A\b -> $A^{-1}b$)
^	exponentiation
.*	“elementwise” multiplication
./	“elementwise” right division
.\	“elementwise” left division
.^	“elementwise” exponentiation
+	Addition
-	Subtraction and negation
'	Conjugate Transpose (Hermitian) of a matrix
.'	Nonconjugated transpose of a matrix
(:)	single column matrix transformation
= =	“is equal to” logical comparator
~=	“is not equal to” logical comparator
<	“is less than” logical comparator
<=	“ is less than or equal to” logical comparator
>	“ is greater than” logical comparator
>=	“ is greater than or equal to” logical comparator
~	logical “not”
&	logical “and”
|	logical “or”

Variables

Recommend use only characters (lower case, upper case), number, underscore (_)
Example: x X y Y num (Note that x is different from X.)
Default data type is double, unless specified otherwise.
Also, scilab supports complex values as well.

Function

A function in scilab can return one or more values (can be scalar, vector, or matrix). There are a lot of built-in functions, such as mathematics (sin, cos, etc.), graphics (plot, etc.), etc., and users can define their own functions (in the form of .sci, .sce file) as well.

Useful commands

help Lists topics on which help is available

help Bob Lists documentation on the command Bob if it exists

Statements

Use assignment operator: *variable = expression*

An expression specifies an operation which returns a value. A basic expression consists of operator(s) and operand(s), which can be literal, variable or both.

x = 3.5 y = -5 z = x+%i*y w = sin(x) u = exp(y)

v = 2*cos(x) t = -2/(1+x)

Furthermore, an expression can be a function call as well.

Function call structure

One return value : *variable = function_name(par1, par2, …)*

Example : y = sin(%pi/2)

Multiple return values : [*var1, var2, …*] = *function_name(par1, par2, …)*

Example: a = [3 2 1 4 7 5 0 6]; [y,m] = max(a);

Note that when using *variable*(*s*) as input parameter(s), those variables must have values, i.e., after some assignment.

Vectors & Matrices

Vectors can be regarded as matrices which have either one row or one column.

Assignment, calculation Example

```
x = [1 2 3];      // row vector
y = [1; 2; 3];    // column vector
A = [1 2 3; 4 5 6; 7 8 9] % 3-by-3 matrix
B = [1 2 3; 4 5 6]; % 2-by-3 matrix
A2 = [1 0 1; 0 1 1; 1 1 0];
C = A + A2; // 3-by-3 matrix
D = A*A2 ;
B2 = B*B' ; // 2-by-2 matrix
B3 = B'*B ; // 3-by-3 matrix
z = A*y ; // column vector
z2 = A*x' ; // column vector
```

Note that the dimensions of operands must agree, e.g., A*B is invalid.

Sequence

```
k = 1:5 // generates row vector [1 2 3 4 5]
m = 1:2:10 // generates row vector [1 3 5 7 9]
x = 0:.1:%pi // generates points between [0,pi]
```

Note: sequences are useful in iterations.

Plot

plot is used to plot 2-dimensional graph.

Example

```
x = 0:.1:2*%pi;
plot(x, sin(x));
```

Note: use command *help plot* for more details.

Control Statements

if-Statement Structure:

```
if a<0 then              // general structure of if-statement
    x = 1                  //calculation when if true
end                    // closing statement of if statement
```

logical comparators					
<	>	<=	>=	= =	~=

if-elseif-else Statment Structure:

```
if  a<0 then          //  general structure of if-statement
  x=1                   // calculation for if true
elseif a>0 then       // elseif general statement
  x=2                   // calculation for elseif true
else                  //  else statement
  x=3                   // calculation when no if or elseif is true
end                   // closing statement of if statement
```

logical operators		
~	NOT	
&	AND	
|	OR	
xor(A,B)	Exclusive OR	

while Loop Structure:

```
while k>0         // enter loop when while statement is true
    k=k-1           // calculation when while is true
end //while       // closing statement of while statement
```

for Loop Structure:

```
for k=1:n          //  enter for loop, where  k =first:increment:last
    x=x+5            //  calculation
    if (x>100)       // secondary condition to exit loop
      break            // send program to statement following end//for
    end //if         //closing statement of if
end //for          // closing statement of for
```

switch-case-otherwise Statement Structure:

```
switch flag          //  set switch flag  where flag is a number (usually integer)
  case value1        // use this case if flag = = value1
      x=x+1            // calculations for case 1
 case value2         // use this case if flag = = value2
      x = x+2          // calculations for case 2
  case value3        // use this case if flag = = value3
      x = x+3          // calculations for case 3
  otherwise          // use this case if flag ~= any value#
      x = 0            // calculations for case otherwise
end //case           // closing statement for switch
```

Other loop and if related commands:

break	breaks out of inner most for or while loop, goes to line following end%loop
return	returns control to calling program

Other Useful Commands:

what	List scilab functions in current directory
dir	Lists all files and folders in current directory
ls	Lists all files and folders in current directory.
pwd	Display name of current directory
cd *folder*	Changes current directory to the one named *folder*
quit OR exit	Exits scilab

Predefined Variables:

ans	Holds the result of the last unnamed expression or calculation
%inf	Value of infinity
%nan	Value of 0 / 0
%pi	Value of 3.14159.....
%i	Value of $\sqrt{-1}$

.sci, .sce file

For convenience and efficiency, users can define their own scripts and functions, which in scilab are in the forms of .sce and .sci files, respectively. A script is simply a set of scilab commands represented in one file to make it more convenient to execute, while a function is user-defined function which requires specification of both argument(s) and return value(s). Conventionally, .sce files represent scripts, while .sci files include a collection of user-defined functions (Not mandatory!). When executing a script or loading functions, use *exec* command followed by a filename.

Script Code Example

```
// test1.sce
m = 1:100;
n = m.^2;
y = cos(n*%pi/1e4);
plot(m, y);
```

```
In scilab
--> exec test1.sce // OR exec('test1.sce')
```

Function Code Example1

```
// test2.sci
function y = test2(m) // one input argument, one return value
x = modulo(m,2);
n = x == 0;
y = x-n;
endfunction
```

```
In scilab
--> exec test2.sci
--> test2(5)
```

Function Code Example2 (MyTaylorExp.sci)

```
function y = MyTaylorExp(x, n)
// x: input value, n : number of terms
y = 1+x; m = 1; xx = x;
for k = 2:n
    xx = xx * x;
    m = m * k ;
    y = y + xx/m;
end
endfunction
```

```
In scilab
--> exec MyTaylorExp.sci
--> z = MyTaylorExp(1,5)
--> w = MyTaylorExp(1,10)
```

NOTE functions must begin with "`function`" and end with "`endfunction`".