

Introducción a SCILAB

Juan Manuel Rabasedas

18/08/2017

Scilab es un software matemático, con un lenguaje de programación de alto nivel, para cálculo científico, interactivo de libre uso y disponible en múltiples sistemas operativos.

Desarrollado por INRIA (Institut National de Recherche en Informatique et Automatique) y la ENPC (École Nationale des Ponts et Chaussées) desde 1990.

<http://www.scilab.org/>

<http://wiki.scilab.org/Tutorials> archives

- **Comentarios**

-->// Las barras se usan para comentarios como este

- **Constantes:** para declarar una constante se escribe el nombre y se iguala al valor que va a tener.

-->val = 5

val =

5.

SCILAB distingue entre mayúsculas y minúsculas.

Operaciones y Funciones

- Suma

--> $4 + val$

$ans =$

9.

- Resta

--> $ans - 4$

$ans =$

5.

- Producto

--> $A = 3 * 3$

$A =$

9.

- División

--> $val = A/3$

$val =$

3.

- Potencia

--> $p = val * * 2$ --> $p = val^2$

$val =$

9.

Operaciones y Funciones

- Potencia

--> $p = val ** 2$ --> $p = val^2$

$val =$

9.

- Raíz Cuadrada

--> $p = \text{sqrt}(b)$

$val =$

3.

Operaciones y Funciones

- Potencia

--> $p = val ** 2$ --> $p = val^2$

val =

9.

- Raíz Cuadrada

--> $p = \text{sqrt}(b)$

val =

3.

- Función e^x

--> $\text{exp}(1)$

ans =

2.7182818

Operaciones y Funciones

- Potencia

--> $p = val ** 2$ --> $p = val^2$

val =

9.

- Raíz Cuadrada

--> $p = \text{sqrt}(b)$

val =

3.

- Función e^x

--> $\text{exp}(1)$

ans =

2.7182818

- Función Logaritmo: \log , \log_{10} y \log_2

--> $\log(2.7182818)$

ans =

1.

- Seno: *sin*
-->*sin(%pi)*
ans =
1.225E - 16
- Arcoseno: *asin*
- Coseno: *cos*
- Arcocoseno: *acos*
- Tangente: *tan*
- Arcotangente: *atan*
- Cotangente: *Cotg*

Constantes Predefinidas

- π

--> `%pi`

`%pi = 3.1415927`

Constantes Predefinidas

- π

--> `%pi`

`%pi` = 3.1415927

- e

--> `%e`

`%e` = 2.7182818

Constantes Predefinidas

- π

--> `%pi`

`%pi` = 3.1415927

- e

--> `%e`

`%e` = 2.7182818

- ϵ es el menor valor que cumple $1 + \epsilon > 1$.

--> `%eps`

`%eps` = $2.220E - 16$

Constantes Predefinidas

- π

--> `%pi`

`%pi` = 3.1415927

- e

--> `%e`

`%e` = 2.7182818

- ϵ es el menor valor que cumple $1 + \epsilon > 1$.

--> `%eps`

`%eps` = $2.220E - 16$

--> `%eps + 1 > 1`

`ans` = `T`

- ∞

--> `%inf`

`%inf` = `Inf`

--> `1/ %inf`

`ans` = 0

Constantes Predefinidas

- *True* y *False* Constantes Booleanas

--> %t

%t =

T

--> %f

%f =

F

Constantes Predefinidas

- *True* y *False* Constantes Booleanas

--> %t

%t =

T

--> %f

%f =

F

- *Not* (\sim), *And* (&) y *Or* (|): Operadores Booleanos.

--> %t | %t

ans =

T

Constantes Predefinidas

- *True* y *False* Constantes Booleanas

--> %t

%t =

T

--> %f

%f =

F

- *Not* (~), *And* (&) y *Or* (|): Operadores Booleanos.

--> %t | %t

ans =

T

- significa "not a number", al operar esta constante con cualquier valor resulta **Nan**

--> %nan + %pi

ans =

Nan

- --> $vec_fila = [3 \ 4 \ 5]$

- --> $vec_fila = [3 \ 4 \ 5]$ $vec_fila = [3, 4, 5]$

- --> $\text{vec_fila} = [3 \ 4 \ 5]$ $\text{vec_fila} = [3, 4, 5]$
 $\text{vec_fila} =$
3. 4. 5.

- --> $vec_fila = [3 \ 4 \ 5]$ $vec_fila = [3, 4, 5]$
 $vec_fila =$
3. 4. 5.
- --> $intervalo = 5 : 2 : 10$ Inicio:Paso:Fin

- --> $vec_fila = [3 \ 4 \ 5]$ $vec_fila = [3, 4, 5]$
 $vec_fila =$
3. 4. 5.
- --> $intervalo = 5 : 2 : 10$ Inicio:Paso:Fin
 $intervalo =$
5. 7. 9.

- -->`vec_fila = [3 4 5]` `vec_fila = [3, 4, 5]`
`vec_fila =`
3. 4. 5.
- -->`intervalo = 5 : 2 : 10` Inicio:Paso:Fin
`intervalo =`
5. 7. 9.
- `linspace(inicio,fin,cantidad)` genera un vector linealmente espaciado
-->`linspace(5,10,3)` Genera 3 valores entre el 5 y el 10

- `-->vec_fila = [3 4 5]` `vec_fila = [3, 4, 5]`
`vec_fila =`
3. 4. 5.
- `-->intervalo = 5 : 2 : 10` Inicio:Paso:Fin
`intervalo =`
5. 7. 9.
- `linspace(inicio,fin,cantidad)` genera un vector linealmente espaciado
`-->linspace(5,10,3)` Genera 3 valores entre el 5 y el 10
5. 7.5 10.

- `-->vec_fila = [3 4 5]` `vec_fila = [3,4,5]`
`vec_fila =`
3. 4. 5.
- `-->intervalo = 5 : 2 : 10` Inicio:Paso:Fin
`intervalo =`
5. 7. 9.
- `linspace(inicio,fin,cantidad)` genera un vector linealmente espaciado
`-->linspace(5,10,3)` Genera 3 valores entre el 5 y el 10
5. 7.5 10.
- `-->transpuesta = intervalo'`
`transpuesta =`
5.
7.
9.

- `-->vec_fila = [3 4 5]` `vec_fila = [3, 4, 5]`
`vec_fila =`
3. 4. 5.
- `-->intervalo = 5 : 2 : 10` Inicio:Paso:Fin
`intervalo =`
5. 7. 9.
- `linspace(inicio,fin,cantidad)` genera un vector linealmente espaciado
`-->linspace(5,10,3)` Genera 3 valores entre el 5 y el 10
5. 7.5 10.
- `-->transpuesta = intervalo'`
`transpuesta =`
5.
7.
9.

Vectores

- --> `vec_columna = [3; 4; 5]`
`vec_columna =`
3.
4.
5.
- --> `vec_largo = 0 : 1 : 1000`
`vec_largo =`
0. 1. 2. 3. ...

- `-->vec_columna = [3; 4; 5]`
`vec_columna =`
3.
4.
5.
- `-->vec_largo = 0 : 1 : 1000`
`vec_largo =`
0. 1. 2. 3. ...
`-->vec_largo = 0 : 1 : 1000 ;`
(;) Evita que se imprima el vector.

- `-->vec_columna = [3; 4; 5]`
`vec_columna =`
3.
4.
5.
- `-->vec_largo = 0 : 1 : 1000`
`vec_largo =`
0. 1. 2. 3. ...
`-->vec_largo = 0 : 1 : 1000 ;`
(;) Evita que se imprima el vector.
- `-->vec_largo(1)` valor del vector en la posición 1

- `-->vec_columna = [3; 4; 5]`
`vec_columna =`
3.
4.
5.
- `-->vec_largo = 0 : 1 : 1000`
`vec_largo =`
0. 1. 2. 3. ...
`-->vec_largo = 0 : 1 : 1000 ;`
(;) Evita que se imprima el vector.
- `-->vec_largo(1)` valor del vector en la posición 1
`ans =`
0.

- `-->vec_columna = [3; 4; 5]`
`vec_columna =`
3.
4.
5.
- `-->vec_largo = 0 : 1 : 1000`
`vec_largo =`
0. 1. 2. 3. ...
`-->vec_largo = 0 : 1 : 1000 ;`
(;) Evita que se imprima el vector.
- `-->vec_largo(1)` valor del vector en la posición 1
`ans =`
0.
- `-->vec_largo(1) = 55;` Reemplaza el valor del vector en la posición 1

- `-->vec_columna = [3; 4; 5]`
`vec_columna =`
3.
4.
5.
- `-->vec_largo = 0 : 1 : 1000`
`vec_largo =`
0. 1. 2. 3. ...
`-->vec_largo = 0 : 1 : 1000 ;`
(;) Evita que se imprima el vector.
- `-->vec_largo(1)` valor del vector en la posición 1
`ans =`
0.
- `-->vec_largo(1) = 55;` Reemplaza el valor del vector en la posición 1
`-->vec_largo(1) =`
55. 1. 2. 3. ...

- `-->vec_columna = [3; 4; 5]`
`vec_columna =`
3.
4.
5.
- `-->vec_largo = 0 : 1 : 1000`
`vec_largo =`
0. 1. 2. 3. ...
`-->vec_largo = 0 : 1 : 1000 ;`
(;) Evita que se imprima el vector.
- `-->vec_largo(1)` valor del vector en la posición 1
`ans =`
0.
- `-->vec_largo(1) = 55;` Reemplaza el valor del vector en la posición 1
`-->vec_largo(1) =`
55. 1. 2. 3. ...

- --> `length(vec_largo)`
`ans = 1001.`

- --> `length(vec_largo)`
`ans = 1001.`
- --> `size(vec_largo)`
`ans = 1. 1001.`

- `-->length(vec_largo)`
`ans = 1001.`
- `-->size(vec_largo)`
`ans = 1. 1001.`
- `-->sum(vec_largo)`
`ans = 500500`

- `-->length(vec_largo)`
`ans = 1001.`
- `-->size(vec_largo)`
`ans = 1. 1001.`
- `-->sum(vec_largo)`
`ans = 500500`
- `-->prod([1 : 1 : 3])`
`ans = 6`
- Multiplicación por un escalar
`-->n = [4 5 6];`
`-->5 * n`
`ans = 20. 25. 30.`

- Producto escalar

--> $n = [4 \ 8 \ 6]$;

--> $m = [2; 6; 8]$;

--> $n * m$

- Producto escalar

--> $n = [4 \ 8 \ 6];$

--> $m = [2; 6; 8];$

--> $n * m$

$ans = 104.$

- $m * n$

$ans =$

8. 16. 12.

24. 48. 36.

32. 64. 48.

- Producto escalar

--> $n = [4 \ 8 \ 6];$

--> $m = [2; 6; 8];$

--> $n * m$

$ans = 104.$

- $m * n$

$ans =$

8. 16. 12.

24. 48. 36.

32. 64. 48.

- Producto cruz

--> $a = [4 \ 5 \ 9];$

--> $b = [7 \ 1 \ 2];$

--> $a. * b$

- Producto escalar

```
-->n = [4 8 6];  
-->m = [2; 6; 8];  
-->n * m  
ans = 104.
```

- $m * n$

```
ans =  
8. 16. 12.  
24. 48. 36.  
32. 64. 48.
```

- Producto cruz

```
-->a = [4 5 9];  
-->b = [7 1 2];  
-->a. * b  
ans = 28. 5. 18.
```


- --> $matriz = [1 \ 2 \ 3; 4 \ 5 \ 6; 7 \ 8 \ 9]$

$matriz =$

1. 2. 3.

4. 5. 6.

7. 8. 9.

- $--> \text{matriz} = [1 \ 2 \ 3; 4 \ 5 \ 6; 7 \ 8 \ 9]$
 $\text{matriz} =$
1. 2. 3.
4. 5. 6.
7. 8. 9.
- Podemos declarar los vectores con anterioridad
 $--> a = [1 \ 4]; \ b = [2 \ 5]; \ c = [3 \ 6];$
 $\text{matriz} = [a; b; c]$

- $\rightarrow \text{matriz} = [1 \ 2 \ 3; 4 \ 5 \ 6; 7 \ 8 \ 9]$
 $\text{matriz} =$
1. 2. 3.
4. 5. 6.
7. 8. 9.
- Podemos declarar los vectores con anterioridad
 $\rightarrow a = [1 \ 4]; \ b = [2 \ 5]; \ c = [3 \ 6];$
 $\text{matriz} = [a; b; c]$
 $\text{matriz} =$
1. 2. 3.
4. 5. 6.
7. 8. 9.

- $\rightarrow \text{matriz} = [1 \ 2 \ 3; 4 \ 5 \ 6; 7 \ 8 \ 9]$
 $\text{matriz} =$
1. 2. 3.
4. 5. 6.
7. 8. 9.
- Podemos declarar los vectores con anterioridad
 $\rightarrow a = [1 \ 4]; \ b = [2 \ 5]; \ c = [3 \ 6];$
 $\text{matriz} = [a; b; c]$
 $\text{matriz} =$
1. 2. 3.
4. 5. 6.
7. 8. 9.

- --> *matriz*(2,2)
 ans =
 5.

- --> *matriz*(2,2)

ans =

5.

- --> *matriz*(:,2)

ans =

2.

5.

8.

- `-->matriz(2,2)`

`ans =`

5.

- `-->matriz(:,2)`

`ans =`

2.

5.

8.

- `-->matriz(2,:)`

`ans =`

4. 5. 6.

- --> `diag(matriz)` Diagonal principal

`ans =`

1. 5. 9.

- --> `diag(matriz)` Diagonal principal
`ans =`
1. 5. 9.
- --> `diag(matriz, -1)` Infra Diagonal
`ans =`
4. 8.

- `-->diag(matriz)` Diagonal principal
`ans =`
1. 5. 9.
- `-->diag(matriz, -1)` Infra Diagonal
`ans =`
4. 8.
- `-->eye(2,2)` Matriz Identidad
`ans =`
1. 0.
0. 1.

- `-->diag(matriz)` Diagonal principal

ans =

1. 5. 9.

- `-->diag(matriz, -1)` Infra Diagonal

ans =

4. 8.

- `-->eye(2,2)` Matriz Identidad

ans =

1. 0.

0. 1.

- `-->zeros(2,2)` Matriz Nula

ans =

0. 0.

0. 0.

- Matriz inversa
--> $A = \begin{bmatrix} 4 & 5 & 9 \\ 7 & 4 & 3 \\ 8 & 6 & 3 \end{bmatrix}$;
--> $\text{inv}(A)$

- Matriz inversa

--> $A = \begin{bmatrix} 4 & 5 & 9 \\ 7 & 4 & 3 \\ 8 & 6 & 3 \end{bmatrix};$

--> $\text{inv}(A)$

$\text{ans} =$

$-0.0740741 \quad 0.4814815 \quad -0.2592593$

$0.0370370 \quad -0.7407407 \quad 0.6296296$

$0.1234568 \quad 0.1975309 \quad -0.2345679$

- Matriz inversa

--> $A = \begin{bmatrix} 4 & 5 & 9 \\ 7 & 4 & 3 \\ 8 & 6 & 3 \end{bmatrix};$

--> $\text{inv}(A)$

$\text{ans} =$

$-0.0740741 \quad 0.4814815 \quad -0.2592593$

$0.0370370 \quad -0.7407407 \quad 0.6296296$

$0.1234568 \quad 0.1975309 \quad -0.2345679$

- --> $\det(A)$ Determinante de A

$\text{ans} = 81$

- Matriz inversa

--> $A = \begin{bmatrix} 4 & 5 & 9 \\ 7 & 4 & 3 \\ 8 & 6 & 3 \end{bmatrix}$;

--> $inv(A)$

$ans =$

-0.0740741 0.4814815 - 0.2592593

0.0370370 - 0.7407407 0.6296296

0.1234568 0.1975309 - 0.2345679

- --> $det(A)$ Determinante de A

$ans = 81$

- --> $rank(A)$; Rango de A

$ans = 3$.

- Matriz inversa

--> $A = \begin{bmatrix} 4 & 5 & 9 \\ 7 & 4 & 3 \\ 8 & 6 & 3 \end{bmatrix}$;

--> $inv(A)$

$ans =$

-0.0740741 0.4814815 - 0.2592593

0.0370370 - 0.7407407 0.6296296

0.1234568 0.1975309 - 0.2345679

- --> $det(A)$ Determinante de A

$ans = 81$

- --> $rank(A)$; Rango de A

$ans = 3$.

- --> $Poly_Coef = poly([1 \ -5 \ 2], 'x', 'c')$
'c' indica que se genera por coeficientes

- --> $Poly_Coef = poly([1 \ -5 \ 2], 'x', 'c')$
'c' indica que se genera por coeficientes
 $Poly_Coef = 1 - 5x + 2x^2$

- --> $Poly_Coef = poly([1 \ -5 \ 2], 'x', 'c')$
'c' indica que se genera por coeficientes
 $Poly_Coef = 1 - 5x + 2x^2$
- --> $Poly_Raices = poly([1 \ -1], 'x', 'r')$
'r' indica que se genera por sus raíces

- --> $Poly_Coef = poly([1 \ -5 \ 2], 'x', 'c')$
'c' indica que se genera por coeficientes
 $Poly_Coef = 1 - 5x + 2x^2$
- --> $Poly_Raices = poly([1 \ -1], 'x', 'r')$
'r' indica que se genera por sus raíces
 $Poly_Raices = -1 + x^2$

- --> $Poly_Coef = poly([1 \ -5 \ 2], 'x', 'c')$
'c' indica que se genera por coeficientes
 $Poly_Coef = 1 - 5x + 2x^2$
- --> $Poly_Raices = poly([1 \ -1], 'x', 'r')$
'r' indica que se genera por sus raíces
 $Poly_Raices = -1 + x^2$
- --> $x = poly(0, 'x')$
 $x = x$

- --> $Poly_Coef = poly([1 \ -5 \ 2], 'x', 'c')$
'c' indica que se genera por coeficientes
 $Poly_Coef = 1 - 5x + 2x^2$
- --> $Poly_Raices = poly([1 \ -1], 'x', 'r')$
'r' indica que se genera por sus raíces
 $Poly_Raices = -1 + x^2$
- --> $x = poly(0, 'x')$
 $x = x$
--> $x^3 + x^2 + x + 1$
 $ans =$
 $1 + x^2 + x^3$

- --> $Poly_Coef = poly([1 \ -5 \ 2], 'x', 'c')$
'c' indica que se genera por coeficientes
 $Poly_Coef = 1 - 5x + 2x^2$
- --> $Poly_Raices = poly([1 \ -1], 'x', 'r')$
'r' indica que se genera por sus raíces
 $Poly_Raices = -1 + x^2$
- --> $x = poly(0, 'x')$
 $x = x$
--> $x^3 + x^2 + x + 1$
 $ans =$
 $1 + x^2 + x^3$
- --> $horner(Poly_Raices, 3)$
 $ans =$
8.

- --> `Poly_Coef = poly([1 - 5 2], 'x', 'c')`
'c' indica que se genera por coeficientes
 $Poly_Coef = 1 - 5x + 2x^2$
- --> `Poly_Raices = poly([1 - 1], 'x', 'r')`
'r' indica que se genera por sus raíces
 $Poly_Raices = -1 + x^2$
- --> `x = poly(0, 'x')`
 $x = x$
--> $x^3 + x^2 + x + 1$
`ans =`
 $1 + x^2 + x^3$
- --> `horner(Poly_Raices, 3)`
`ans =`
8.
- --> `derivat(Poly_Raices)`
`ans =`
 $2x$

- Raices de un polinomio

-->*roots(Poly_Raices)*

ans =

1. - 1.

- Raices de un polinomio
-->*roots(Poly_Raices)*
ans =
1. - 1.
- División de polinomios con *pdiv*
-->*x = poly(0,'x')*;
-->*p1 = (1 + x^2) * (1 - x)*;
-->*p2 = 1 - x*;
-->*[r, q] = pdiv(p1, p2)*

- Raíces de un polinomio
--> `roots(Poly_Raices)`
`ans =`
1. -1.
- División de polinomios con `pdiv`
--> `x = poly(0,'x');`
--> `p1 = (1 + x^2) * (1 - x);`
--> `p2 = 1 - x;`
--> `[r, q] = pdiv(p1, p2)`
`q =`
 $1 + x^2$

- Raíces de un polinomio
--> `roots(Poly_Raices)`
`ans =`
1. -1.
- División de polinomios con `pdiv`
--> `x = poly(0,'x');`
--> `p1 = (1 + x^2) * (1 - x);`
--> `p2 = 1 - x;`
--> `[r, q] = pdiv(p1, p2)`
`q =`
 $1 + x^2$
`r =`
0.

- Raíces de un polinomio
--> `roots(Poly_Raices)`
`ans =`
1. -1.
- División de polinomios con `pdiv`
--> `x = poly(0,'x');`
--> `p1 = (1 + x^2) * (1 - x);`
--> `p2 = 1 - x;`
--> `[r, q] = pdiv(p1, p2)`
`q =`
 $1 + x^2$
`r =`
0.

- --> *Deff('x = suma3(a, b, c)', 'x = a + b + c')*
--> *suma(4, 5, 1)*
ans = 10.

Definición de funciones

- `-->Deff('x = suma3(a, b, c)', 'x = a + b + c')`
`-->suma(4, 5, 1)`
`ans = 10.`
- `-->deff('x = Derivada(p, val)', 'x = horner(derivat(p), val)')`
`-->derivada(Poly_Raices, 4)`
`ans = 8.`

Definición de funciones

- `-->Deff('x = suma3(a, b, c)', 'x = a + b + c')`
`-->suma(4, 5, 1)`
`ans = 10.`
- `-->deff('x = Derivada(p, val)', 'x = horner(derivat(p), val)')`
`-->derivada(Poly_Raices, 4)`
`ans = 8.`
- `-->deff('x = factorial(val)', 'x = prod(1 : 1 : val)')`

- Si no recordamos como usar un comando de scilab podemos escribir *help* seguido del nombre del comando
-->*help format*

- Si no recordamos como usar un comando de scilab podemos escribir *help* seguido del nombre del comando
-->*help format*
- Si queremos encontrar una lista de comando relacionados a una palabra podemos escribir *apropos* y la palabra a buscar.
Ejemplo:
-->*apropos logarithm*

- Si no recordamos como usar un comando de scilab podemos escribir *help* seguido del nombre del comando
-->*help format*
- Si queremos encontrar una lista de comando relacionados a una palabra podemos escribir *apropos* y la palabra a buscar.
Ejemplo:
-->*apropos logarithm*

Ejercicios:

- 1 Explica para que se usa el comando *format*. Realiza un ejemplo de uso.
- 2 ¿Tiene Scilab una función para convertir números a base 16? Calcular el número 61453 en base 16.