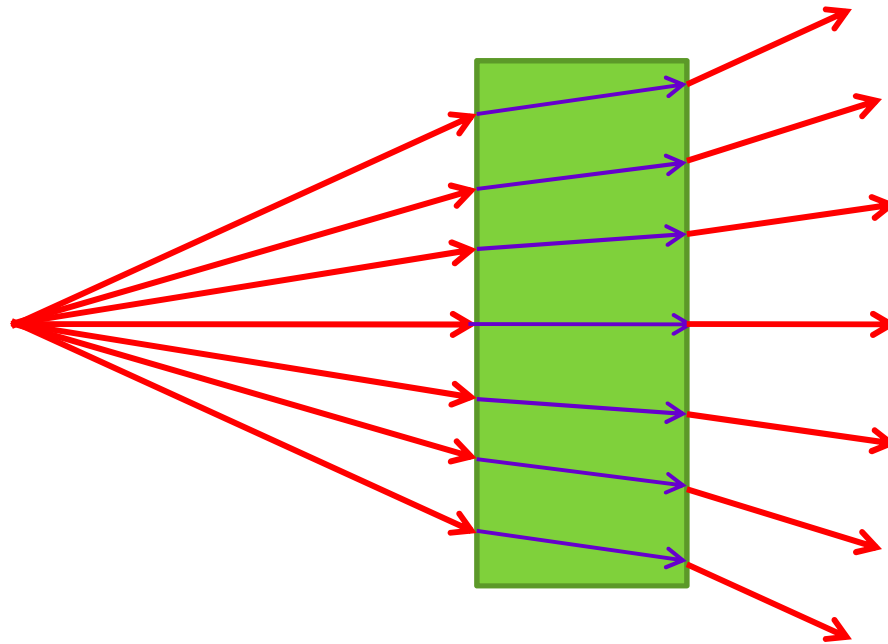


SciLab

Comandos básicos
Al 30/05/19

Ejercicio: Considere una placa de vidrio ortorrómbica (de caras rectangulares paralelas) sobre la que inciden un haz de 10 rayos, divergentes, provenientes desde un punto objeto localizado a $D=10\text{cm}$ de la cara. El vidrio tiene un índice de refracción $n=1.5$ y los rayos inciden formando los siguientes ángulos con la normal a la cara de incidencia: -15° , -10° , -5° , 0° , 5° , 10° , 15° . Realice un gráfico don los rayos incidentes y refractados pro la placa si ésta tiene un espesor $E=5\text{cm}$.



input

```
name=input("What is your name?","string")
```

prompt for user input

Syntax

```
x = input(message [, "string"])
```

Arguments

message

character string

"string"

the character string "string" (may be abbreviated to "s")

x

real number (or character string if "string" is in the syntax)

```
function menor_a()  
    x=input("ingrese un numero real x=")  
    x_max=input("ingrese el extremo mayor x_max=")  
  
    while x<x_max  
        disp(x)  
        x=x+1;  
    end  
endfunction
```

Para ingresar valores de variables a través del teclado

<< x_matrix

Scilab Help >> GUI > x_mdialog

x_mdialog

Dialog for interactive vector/matrix input.

Syntax

```
result=x_mdialog(title,labels,default_inputs_vector)
result=x_mdialog(title,labelsv,labelsh,default_input_matrix)
```

Arguments

title
column vector of strings, dialog general comment

labels
n column vector of strings, labels(i) is the label of the ith required value

default_input_vector
n column vector of strings, default_input_vector(i) is the initial value of the ith required value

labelsv
n vector of strings, labelsv(i) is the label of the ith line of the required matrix

labelsh
m vector of strings, labelsh(j) is the label of the jth column of the required matrix

default_input_matrix
n x m matrix of strings, default_input_matrix(i,j) is the initial value of the (i,j) element of then required matrix

result
n x m matrix of string if returned with "Ok" button or [] if returned with "Cancel" button

Description

Opens a dialog for interactive vector/matrix input.

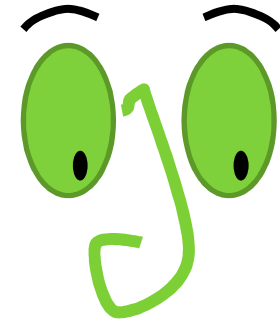
En algunas situaciones es conveniente que las variables tomen valores definidos por el Usuario en forma interactiva, esto es, que el usuario ingrese el conjunto de valores a través Del teclado y que el programa las interprete en forma adecuada.

Se pueden utilizar las funciones:

“input”....

“x_mdialog()” en conjunto con “evstr()”

Ejemplo con el comando “x_mdialog”...



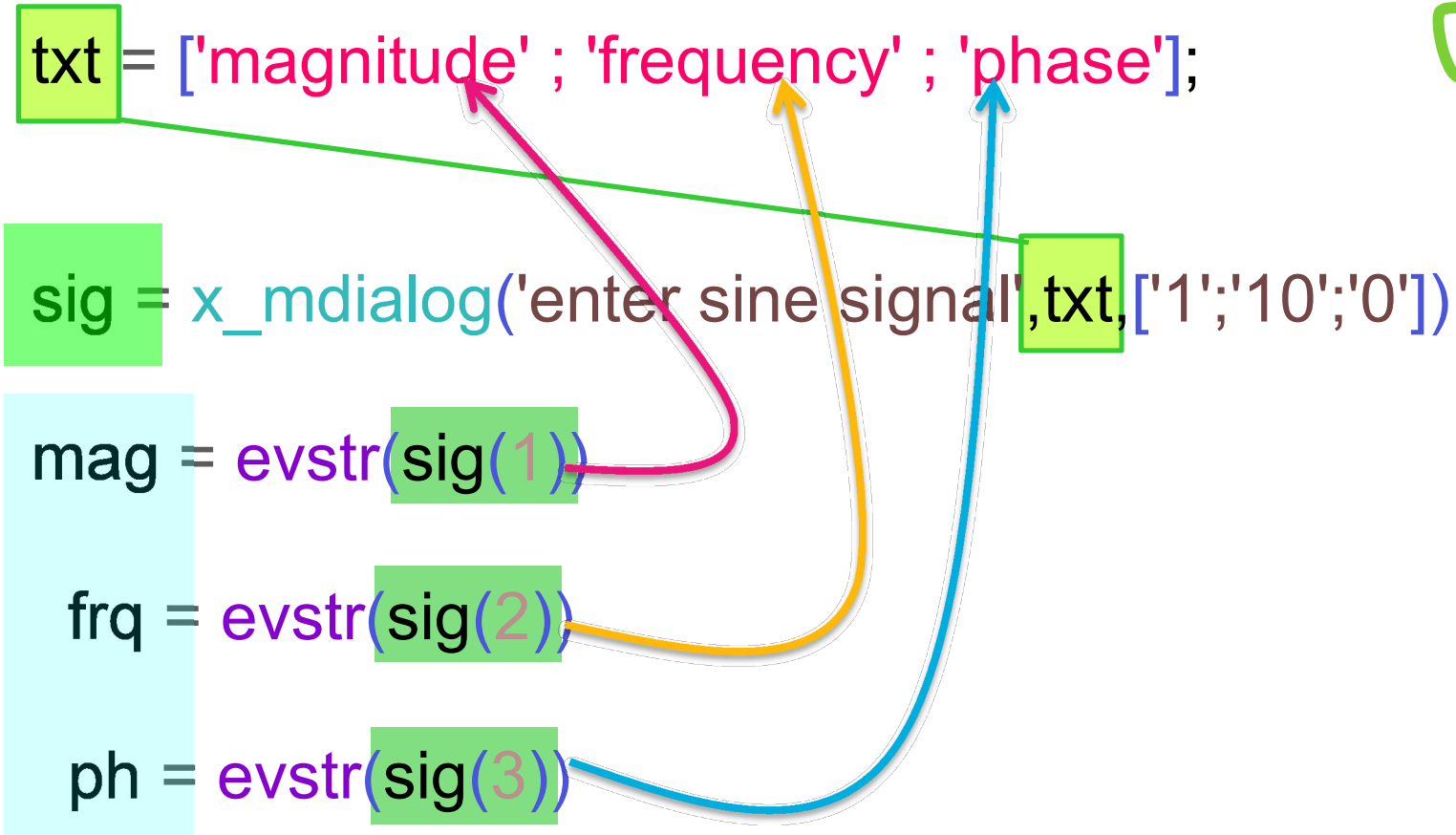
```
txt = ['magnitude' ; 'frequency' ; 'phase'];
```

```
sig = x_mdialog('enter sine signal',txt,['1';'10';'0'])
```

```
mag = evstr(sig(1))
```

```
frq = evstr(sig(2))
```

```
ph = evstr(sig(3))
```



Ficheros de escritura o script

- Sucesión de instrucciones de SCILAB que se ejecutan en modo comando tecleando el nombre del archivo.
- Por convenio, tienen extensión .sce .
- Son útiles como apoyo de los archivos de función. Pueden actuar como ficheros de datos que necesite algún fichero de función más importante, para pequeñas tareas o como programa principal de varios archivos de función.
- Para la ejecución, se usa
 - `-- > exec("filename") //` (Repite todas las instrucciones en pantalla)
 - `-- > exec('filename',-1) //` (No repite las instrucciones en pantalla)

Comandos de información: **who, whos, what, why, pwd, workspace**

Eliminación de variables: **clear**

Almacenamiento: **save, load, diary, mopen, mclose**

```
-- > a = 8; b = 9,5;  
-- > save('misdatos.dat', a, b)
```

```
-- > help save
```

```
-- > clear a  
-- > clear b  
-- > load('misdatos.dat')  
-- > a
```

```
a =  
8
```

```
-- > help load
```

Para guardar datos en un archivo de texto

Usamos el comando “write”

Para cargar datos desde un archivo de texto

Usamos el comando “read”

En Scilab 5.4.X y

```
--> x=eye(3,3);
```

```
--> write('val.dat', x)
```

Variable (matriz, vector, etc)

```
--> clear x;
```


Formato de los datos guardados

```
--> read('val.dat', [format])
```



```
--> x=rand(8,6)           // X es una matriz de 8x6
```

```
x =
```



```
0.6561381 0.2659857 0.2367841 0.824862 0.7263671 0.4952356
0.2445539 0.9709819 0.7015344 0.5798843 0.9009498 0.4194877
0.5283124 0.8875248 0.1202527 0.2791808 0.3948993 0.8626222
0.8468926 0.2066753 0.8287412 0.9545111 0.565518 0.285751
0.7876622 0.8525161 0.3161073 0.9071155 0.706149 0.2512136
0.1262083 0.6744698 0.5305191 0.3360149 0.6787831 0.3389102
0.7883861 0.9152874 0.5715175 0.1175613 0.4132936 0.3921976
0.3453042 0.028486 0.0478015 0.9253724 0.1402291 0.4681552
```

```
--> write('pru-write.dat',x)
```

```
--> clear x
```

```
--> read('pru-write.dat',3,3) ←
```

// lee los datos correspondientes a una matriz de 3x3

```
ans =
```

```
0.6561381 0.2659857 0.2367841
0.824862 0.7263671 0.4952356
0.2445539 0.9709819 0.7015344
```

```
--> read('pru-write.dat',8,6)
```

ans =

0.6561381	0.2659857	0.2367841	0.824862	0.7263671	0.4952356
0.2445539	0.9709819	0.7015344	0.5798843	0.9009498	0.4194877
0.5283124	0.8875248	0.1202527	0.2791808	0.3948993	0.8626222
0.8468926	0.2066753	0.8287412	0.9545111	0.565518	0.285751
0.7876622	0.8525161	0.3161073	0.9071155	0.706149	0.2512136
0.1262083	0.6744698	0.5305191	0.3360149	0.6787831	0.3389102
0.7883861	0.9152874	0.5715175	0.1175613	0.4132936	0.3921976
0.3453042	0.028486	0.0478015	0.9253724	0.1402291	0.4681552

Ejemplo con el comando “input”...

```
a= input("ingrese la coordenada (xo) del punto objeto: ", )
```

```
b=input("ingrese la coordenada (yo) del punto objeto:..", )
```

```
dlo=input("distancia Lente-Objeto:..", )
```

```
s=input("ingrese el ángulo del haz incidente con el eje x:..", )
```

Sucesivamente las variables definidas como “a”, “b”, “dlo” y “s” tomarán los valores que el usuario va ingresando por teclado. El programa se detiene cada vez hasta que el usuario ingresa un valor por teclado.

4.5.1 Overview of the scope of variables

In this section, we present the scope of variables and how this can interact with the behavior of functions.

Assume that a variable, for example `a`, is defined in a script and assume that the same variable is used in a function called directly or indirectly. What happens in this case depends on the first statement reading or writing the variable `a` in the body of the function.

- If the variable `a` is first read, then the value of the variable at the higher level is used.
- If the variable `a` is first written, then the local variable is changed (but the higher level variable `a` is not changed).

This general framework is now presented in several examples.

In the following script, we present a case where the variable is read first. The following function `f` calls the function `g` and evaluates an expression depending on the input argument `x` and the variable `a`.

```
function y = f ( x )
    y = g ( x )
endfunction
function y = g ( x )
    y = a(1) + a(2)*x + a(3)*x^2
endfunction
```

Level #0
`a = [1 2 3];`

Level #1
`function y = f (x)
y = g (x)`

Level #2
`function y = g (x)
y = a(1) + a(2)*x + a(3)*x^2`

Using variable "a"...
Searching for variable "a" at level #2... none.
Searching for variable "a" at level #1... none.
Searching for variable "a" at level #0: "a" found!

```
-->a = [1 2 3];  
-->x = 2;  
-->y = f ( x )  
y =  
    17.  
-->a  
a =  
    1.    2.    3.
```

En el siguiente Script se hace un uso más claro de la variable "a".
En este caso la función "g" es redefinida para incluir en los argumento de entrada a la variable "a".

```
function y = gfixed ( x , a )  
    y = a(1) + a(2)*x + a(3)*x^2  
endfunction
```

La función "f" también debe ser redefinida a fin de proveerle a la función "gfixed" el argumento que necesita...

```
function y = ffixed ( x , a )  
    y = gfixed ( x , a )  
endfunction
```

```
-->a = [1 2 3];  
-->x = 2;  
-->y = ffixed ( x , a )  
y =  
    17.
```

Ahora presentemos el caso en el que la variable "a" es definida primero. En lo que sigue la función "f2" "llama" a la función "g2" y evalúa la expresión en base al argumento de entrada "x", y la variable "a".

```
function y = f2 ( x )  
    y = g2 ( x )  
endfunction  
function y = g2 ( x )  
    a = [4 5 6]  
    y = a(1) + a(2)*x + a(3)*x^2  
endfunction
```

Note que la variable "a" es definida antes de ser usada

```
-->a = [1 2 3];  
-->x = 2;  
-->y = f2 ( x )  
y =  
    38.  
-->a
```

Notemos que el valor de la función "f2" es 38, y no "17" como en el caso anterior.

Esto confirma que la variable "local" a = [4 5 6], definida en el nivel #2, fue usada, y no la variable a = [1 2 3], definida en el nivel #0.

Además, observemos que la variable "a" definida en el nivel #0 no se modificó luego de "llamar" a "f2", su valores permanecen invariables a = [1 2 3].

Quando se debe seleccionar entre seguir un conjunto de instrucciones u otro dependiendo del resultado obtenido en otra sección el programa se puede usar la función "case" ...

case

keyword used in statement "select"

Description

Keyword used in statement `select ... case ... end`

Keyword `case` points out one variant of a variable value (it is the variable value which is selected from a range of values).

Use it in the following way:

```
select expr0
  case expr1 then
    instructions 1
  case expr2 then
    instructions 2
    ...
  case exprn then
    instructions n
  [else instructions]
end
```


Ejemplo con “case” “select”....

En este programa se sigue una instrucción u otra dependiendo del valor de la variable “A”

```
1
2 A = 10
3 select A
4 case 1 then
5     disp('hola')
6 case 2 then
7     disp('chau')
8 else
9     disp(':)')
10 end|
```

El comando “**select**” captura el valor de “**A**”
En tanto, “**case**” compara el valor
“capturado” por “**select**” con el asignado a
cada “**caso**” - se siguen las instrucciones
correspondientes al caso cuyo “**valor**
asignado” coincide con “**select**”

Case_function.sce 

```
1 A = 'a'
2 select A
3 case 'a' then
4     disp('hola')
5 case 'b' then
6     disp('chau')
7 else
8     disp(':)')
9 end
10
```

Scilab 6.0.1 Console

```
--> exec('C:\Users\Claudia\Documents\A_Opticas\Opt_Computac\scilab\Case_function.sce')
```

```
hola
```

```
-->
```