**Francesca Mazzia**
**Dipartimento Interuniversitario di Matematica**
**Università di Bari**

## SCILAB: lezione introduttiva

SCILAB: PSE sviluppato all'INRIA e reperibile gratuitamente dal sito:
http://www-rocq.inria.fr/scilab/.
È dotato di un particolare linguaggio di programmazione di tipo interpretativo che consente di sviluppare le proprie applicazione.
Ogni istruzione viene interpretata ed eseguita immediatamente.
Entrati in ambiente SCILAB, compare il prompt:

```
-->
```

il che significa che il calcolatore è pronto a ricevere le istruzioni SCILAB ed ad eseguirle. Se vogliamo eseguire la somma 3+2 basta scrivere:

```
--> 3+2
```

e premere il tasto invio. Otteniamo immediatamente la risposta:

```
ans =
       5
-->
```

e il calcolatore è di nuovo pronto a ricevere un nuovo comando. Il risultato dell'operazione eseguita è memorizzato nella *variabile* ans. Le variabili nei linguaggi di programmazione sono dei contenitori di valori. Se vogliamo assegnare il risultato dell'operazione alla variabile di nome var basta scrivere:

```
-->var=3+2
 var  =

    5.
```

Per evitare la visualizzazione del risultato dell'istruzione eseguita si scrive il simbolo ; alla fine della riga, per esempio:

```
-->var=3+2;
```

Questa istruzione si chiama *istruzione di assegnazione*. Possiamo assegnare a una variabile il valore di una espressione più complicata. Per far questo dobbiamo sapere a quali simboli corrispondono le principali operazioni aritmetico-logiche. In SCILAB questi sono:

| | |
|---|---|
| $+$ | addizione |
| $-$ | sottrazione |
| $*$ | moltiplicazione |
| $/$ | divisione |
| ^ | elevamento a potenza |
| & | and logico |
| \| | or logico |
| ~ | not logico |
| $==$ | uguale |
| ~$=$ | diverso |
| $<$ | minore |
| $<=$ | minore uguale |
| $>$ | maggiore |
| $>=$ | maggiore uguale |

Le espressioni aritmetiche e logiche seguono le classiche regole di precedenza, che è possibile modificare utilizzando le parentesi tonde.

Per avere informazioni su tutte le operazioni che possono essere eseguite in SCILAB si può digitare sul pulsante `help`. Compare lo Scilab Help Panelin cui compare l'elenco di tutte le informazioni sullo Scilab divise per capitoli.

Eseguimo anche i seguenti comandi:

```
-->help "*"
-->help "=="
```

Possiamo ora eseguire:

```
-->var = 5*3/(3+2)^2
 var  =

    0.6
```

È anche possibile calcolare i valori delle principali funzioni. Ad esempio sono predefinite le funzioni in una variabile:

| | |
|---|---|
| abs(x) | valore assoluto di x, |
| log(x) | logaritmo naturale, |
| exp(x) | esponenziale con base e, |
| sin(x) | seno, |
| cos(x) | coseno, |
| tan(x) | tangente. |

Anche per queste funzioni è possibile usare il comando help. Si provi ad eseguire:

```
--> help cos
--> help tan
--> help log
```

È possibile quindi eseguire espressioni del tipo:

```
-->var = (5* 3/(3+cos(5)))^log(2) * 100
 var  =

    286.61038
```

Se eseguiamo il comando

```
-->format("e")
```

e rieseguimo l'operazione

```
-->var = (5* 3/(3+cos(5)))^log(2) * 100
 var  =

    2.866E+02
```

La variabile var = 2.866E+02 è ora una variabile reale espressa in notazione esponenziale, essa è equivalente ad $2.866 \cdot 10^2$.

Oltre alle variabili, sono di particolare utilità anche le *costanti*. Le costanti sono dei contenitori di valori che non cambiano nel corso dell'esecuzione di un algoritmo. In Scilab vi sono delle costanti predefinite, come, ad esempio, la costante `%pi` che contiene il valore di $\pi$:

```
-->%pi
 %pi  =

    3.142E+00
```

Le variabili possono essere scalari oppure vettori o matrici. I vettori e le matrici sono variabili che contengono un certo numero di componenti. Ad esempio dopo l'istruzione:

```
--> A = [ 3, 5, 6]
 A  =

!   3.000E+00    5.000E+00    6.000E+00 !
```

la variabile A sarà un vettore di tre elementi con A(1)=3, A(2)=5, A(3)=6. Gli elementi di un vettore si indirizzano specificando l'indice tra parentesi tonde. Se vogliamo sapere la somma degli elementi di A possiamo eseguire:

```
--> A(1) + A(2) + A(3)
 ans  =

    1.400E+01
ans =
     14
```

Dopo l'istruzione:

```
>>  A = [  3, 5, 7
           8, 9, 10
           2, 3, 5 ];
```

la variabile A sarà una matrice con tre righe e tre colonne con elementi A(1,1)=3, A(1,2)=5, ..., A(2,1)=8, ..., A(3,3)=5. Ogni elemento di una matrice viene individuato da una coppia di indici, il primo indice è detto indice di riga, il secondo indice di colonna.

Tutte le variabili già definite, dette variabili globali, possono essere elencate con il comando:

```
>> who
```

oppure:

```
>> whos()
```

Lo Scilab consente anche di memorizzare una sequenza di istruzioni in un file, aggiungendo alle funzioni Scilab predefinite funzioni definite dall'utente.

I nomi dei file Scilab devono avere l'estensione .sci, cioè devono essere del tipo *nomefile.sci*. Essi possono essere scritti utilizzando un qualsiasi editore di testi.

Essi definiscono una nuova funzione Scilab, accettano dei dati di input e restituiscono dei dati di output come risultato della loro elaborazione. L'insieme dei dati di input ed autput può essere vuoto. La prima istruzione in una function deve essere:

$$\text{function } [\ o_1, o_2, \ldots, o_n] = \text{nomefunction}(i_1, i_2, \ldots, i_m)$$

dove $o_1, o_2, \ldots, o_n$ sono le variabili di output e $i_1, i_2, \ldots, i_m$ sono le variabili di input.

Per essere usata una nuova funzione deve essere caricata nell'ambiente con l'istruzione `getf(‘‘nomefunction.sci’’)`

Per esempio possiamo scrivere il file `f1.sci`

```
function y=f1(x)
  y=exp(x)-(2*x).^2;
```

al prompt possiamo caricare la nuova funzione nell'ambiente

```
-->getf("f1.sci")
```

ed eseguirla

```
-->y=f1(10)
 y  =

    2.163E+04
```

Quando abbiamo terminato di lavorare con lo Scilab possiamo uscire con il comando:

```
--> exit
```

Quando si esce dallo Scilsb tutte le variabili definite vanno perdute. È possibile tuttavia salvarne il contenuto in un file mediante il comando:

```
-->   save('ambiente');
```

Se nomefile è seguito da un elenco di variabili vengono memorizzate solo le variabili considerate.

```
-->   save('ambiente',f1,y);
```

```
-->  load('ambiente')
```

oppure

```
-->  load('ambiente','f1')
```

Se i file con cui dobbiamo lavorare si trovano in un'altra directory, diversa da quella corrente possiamo utilizzare il comanda

```
--> chdir(``nome-directory'')
```

per cambiare directory. Per verificare in quale directory ci troviamo il comando è:

```
--> pwd
```

In Scilab possiamo usare anche variabili complesse

```
-->y=sqrt(-1)
 y  =

    1.000E+00i
-->
```

Una costante importante è la costante `%eps` che contiene la precisione di macchina:

```
-->%eps
 %eps  =

    4.441E-16

-->
```

Altre costanti collegate alla rappresentazione dei numeri sono:

```
-->%i
 %i  =

    1.000E+00i

-->%e
 %e  =

    2.718E+00

-->
```

Le costanti `%t, %f` vengono utilizzate nelle espressioni logiche

```
-->%t
 %t  =

  T

-->%f
 %f  =

  F

-->
```

Scilab utilizza l'aritmetica IEEE e permette di rappresentare infinito e NaN, per cambiare il floating point exception mode si usa la function `ieee`

```
-->ieee(0)

-->1/0
    !--error    27
division by zero...


-->0/0
    !--error    27
```

```
division by zero...


-->ieee(1)

-->1/0
Warning :division by zero...

 ans  =

          INF

-->0/0
Warning :division by zero...

 ans  =

          NAN

-->ieee(2)

-->1/0
 ans  =

          INF

-->0/0
 ans  =

          NAN

-->
```

Ci sono alcune function che permettono di verificare se un numero è infinito o se è uguale ad NaN. Eseguima le seguenti operazioni:

```
 -->isinf(1/0)
 ans  =
```

```
  T

-->isnan(1/0)
 ans  =

  F

-->isnan(0/0)
 ans  =

  T

-->
```

Scilab consente di scrivere le function utilizzando la sequenza, la selezione e la ripetizione. Per la selezione si usa le parole chiavi `if then else`. Eseguiamo:

```
SINTASSI

  if expr1 then istruzioni
  elseif expri then istruzioni
    ....
  else  istruzioni
  end


ESEMPIO
     i=2
     for j = 1:3,
        if i == j then
          a(i,j) = 2;
        elseif abs(i-j) == 1 then
          a(i,j) = -1;
        else a(i,j) = 0;
        end,
     end
```

L'istruzione `select` permette di selezionare una parola chiave

```
DESCRIZIONE
   select expr,
     case expr1 then instructioni1,
     case expr2 then instructioni2,
     ...
     case exprn then instructionin,
     [else instructioni],
   end
```

```
ESEMPIO

    n=round(10*rand(1,1))
    select n
    case 0 then
      disp(0)
    case 1 then
      disp(1)
    end
```

Per i cicli di ripetizione possiamo utilizzare l'istruzione `for` e l'istruzione `while`

```
DESCRIZIONE

 for varibile=expressione ,instructione, ,instructione,end

 for varibile=expressione do instructione, ,instructione,end
```

```
ESEMPIO

  n=5;
  for i = 1:n, for j=1:n, a(i,j) = 1/(i+j-1);end;end
  for j = 2:n-1, a(j,j) = j; end; a
```

```
DESCRIZIONE

  while expr ,instructioni,...[,else instructioni], end

  while expr do instructioni,...[,else instructioni], end

  while expr then instructioni,...[,else instructioni], end

ESEMPIO

  e=1; a=1; k=1;
  while norm(a-(a+e),1) > %eps, e=e/2; k=k+1; end
  e,k
```

L'istruzione `break` all'interno di un loop, permette di interromperlo.

```
k=0; while 1==1, k=k+1; if k > 100 then  break,end; end
```