Calcolo numerico e **programmazione**
Introduzione a Scilab

Tullio Facchinetti
<tullio.facchinetti@unipv.it>

16 aprile 2012

14:06

http://robot.unipv.it/toolleeo

MatLab®: MATrix LABoratory

da it.wikipedia.org:

"MatLab® (abbreviazione di Matrix Laboratory) è un ambiente per il calcolo numerico e un linguaggio di programmazione (interpretato) creato dalla MathWorks. MatLab® consente facili manipolazioni di matrici, visualizzazione di funzioni e dati, implementazione di algoritmi, consente la creazione di interfacce utente e si interfaccia con altri programmi. [...]"

Alternative a MatLab®

esistono software liberi, non del tutto compatibili ma ugualmente validi, tra i quali:

- Octave: http://www.gnu.org/software/octave/
- Scilab: http://www.scilab.org/
- RlabPlus: http://rlabplus.sourceforge.net/

Esecuzione di comandi

i comandi possono essere impartiti:

- direttamente da console
  - la linea di comando è rapida ma consente l'immissione di un comando per volta
- attraverso un file di script con estensione `.sce`
  - lo script consiste in una sequenza di comandi da eseguire e viene richiamato da console

uno script non è altro che un file di testo
può essere scritto e modificato con un qualunque editor di testo (meglio se in grado di evidenziare la sintassi)

L'editor

l'editor è un programma utile per scrivere gli script

- richiamabile da console col comando `editor()`
- gestisce la sintassi di Scilab colorando opportunamente i termini (syntax highlighting) e indentando i comandi
- gestione avanzata delle finestre

Esecuzione di comandi

i comandi contenuti in uno script possono essere eseguiti in vari modi:

- richiamando la funzione `exec` dalla console
- attraverso opportune voci di menu dall'editor

dall'editor è possibile eseguire lo script dal menu `Execute` in varie modalità

- `...file with no echo`: esegue il file completo senza output a video che non sia esplicitamente invocato
- `...file with echo`: esegue il file completo visualizzando anche le istruzioni che vanno ad essere eseguite
- `...until the caret with echo`: esegue fino al punto nel quale si trova il cursore, con echo

Le variabili

in Scilab sostanzialmente tutto è rappresentato in forma di matrice

in Scilab le variabili possono essere:

- matrici bidimensionali
- vettori, ovvero matrici di dimensione $1 \times N$
- scalari, ovvero matrici 1×1

Le variabili

non occorre dichiarare una variabile: vengono create al primo utilizzo e la dimensione può variare

i tipi di dato supportati nativamente dall'ambiente sono:

- carattere
- intero con o senza segno
- numero in virgola mobile
- numeri complessi
- valore logico (booleano)
- stringhe (sequenze di caratteri)

Nomi delle variabili

- il nome di una variabile (identificatore) può essere lungo a piacere
- solo i primi 24 caratteri sono effettivamente usati per identificare una variabile
- i seguenti caratteri sono ammessi: ['a', ..., 'z'], ['A', ..., 'Z'], ['0', ..., '9'], %, _, #, !, $, ?

il linguaggio di SciLab è case-sensitive, cioè distingue tra lettere maiuscole e minuscole

Creazione e accesso a variabili

l'operatore = serve per assegnare un valore ad una variabile

```
-->A = 5
 A  =
    5.
-->a = 10
 a  =
    10.
-->b = a * A
 b  =
    50.
```

Visualizzazione dei risultati

di norma tutti i risultati di un calcolo vengono visualizzati

```
-->a = 44 / 3
 a  =
    14.666667
```

concludendo una istruzione con il ; (punto e virgola) il risultato non viene visualizzato

```
-->a = 44 / 3;
-->
```

La funzione disp

la funzione `disp()` serve per visualizzare un oggetto a video

```
-->a = 44 / 3;
-->disp(a);
    14.666667
```

La variabile ans

quando si esegue un calcolo senza assegnare esplicitamente un valore ad una variabile, il valore calcolato viene assegnato alla variabile **ans**

```
-->log(8)
 ans  =
    2.0794415
-->risultato = exp(ans)
 risultato  =
    8.
```

la variabile **ans** può essere utilizzata come qualsiasi altra variabile

Operatori aritmetici

```
+       somma
-       sottrazione
*       moltiplicazione
.*      moltiplicazione per elementi
./      divisione per elementi
'       trasposizione
/       divisione a destra, i.e. x/y = x y^-1
\       divisione a sinistra, i.e. x\y = x^-1 y
^       elevamento a potenza (x^y)
**      elevamento a potenza (come ^)
```

Altri operatori

```
;       nasconde il risultato
:       genera una sequenza di valori
..      continua l'istruzione alla linea successiva
//      commento
%       introduce valori speciali
```

Esempio

un commento seguito da una espressione articolata su più linee di testo

```
-->// questo e‘ un commento
-->x = 1 ..
-->+2 ..
-->+3 ..
-->+4
 x  =
    10.
```

Valori speciali

```
%pi      pi-greco
%e       numero di Eulero (e)
%i       unita' immaginaria
```

verifica della identità di Eulero ($e^{i\pi} + 1 = 0$)

```
-->%e ^ (%i * %pi) + 1
 ans  =
     1.225D-16i
```

il risultato non è esattamente 0 poichè i numeri sono rappresentati in virgola mobile, quindi con precisione limitata

Funzioni numeriche avanzate

verifichiamo che $\sin^2(x) + \cos^2(x) = 1$

```
-->x=10
 x  =
    10.
-->a = sin(x)
 a  =
  - 0.5440211
-->b = cos(x)
 b  =
  - 0.8390715
-->c = a^2 + b^2
 c  =
    1.
```

Funzioni numeriche avanzate: trigonometria

```
acos   acosd   acosh  acoshm  acosm  acot   acotd   acoth
acsc   acscd   acsch  asec    asecd  asech  asin    asind
asinh  asinhm  asinm  atan    atand  atanh  atanhm  atanm
cos    cosd    cosh   coshm   cosm   cotd   cotg    coth
cothm  csc     cscd   csch    sec    secd   sech    sin
sinc   sind    sinh   sinhm   sinm   tan    tand    tanh
tanhm  tanm
```

Funzioni numeriche avanzate: varie

```
exp   expm log  log10  log1p   log2 logm  max
maxi  min  mini modulo pmodulo sign signm sqrt
sqrtm
```

La funzione `help`

è possibile ottenere informazioni dettagliate su qualsiasi aspetto del linguaggio utilizzando la funzione `help` da console

```
-->help ans
```

```
-->help log
```

viene aperta una finestra contenente le informazioni riguardanti la voce richiesta

Operatori di confronto

```
>       maggiore
<       minore
>=      maggiore o uguale
<=      minore o uguale
~=      diverso
<>      diverso
==      uguaglianza
=       assegnazione
```

Variabili e costanti booleane

servono per memorizzare un valore vero/falso

le costanti booleane sono le seguenti:

```
%t      vero (True)
%f      falso (False)
```

es:

```
-->10>20
 ans  =
  F
```

Operatori logici

```
&       AND logico
|       OR logico
~       negazione logica (NOT)
```

es.

```
-->~((%t | %f) & %f)
 ans  =
  T
```

Numeri complessi

```
real   parte reale
imag   parte immaginaria
imult  moltiplicazione per i
isreal ritorna true se la variabile ha parte
       immaginaria nulla
```

Creazione di matrici

sono caratterizzate da numero di righe, numero di colonne e tipo dei singoli elementi
(intero, virgola mobile, numero complesso)

- le matrici sono specificate elencandone gli elementi tra parentesi quadre
- se due elementi sono separati da uno spazio o da una virgola, allora si considera che appartengano alla stessa riga
- quando tra due elementi si mette il carattere ; Matlab passa alla riga successiva

Creazione di matrici

```
-->a = [1 2 4; 3 6 7]
 a  =
    1.    2.    4.
    3.    6.    7.
```

oppure, in modo leggermente più chiaro:

```
-->a = [1 2 4
-->3 6 7]
 a  =
    1.    2.    4.
    3.    6.    7.
```

Accesso ai valori di una matrice

un elemento viene referenziato indicandone esplicitamente la posizione (numero di riga e numero di colonna)

```
-->a(2,3)
 ans  =
    7.
```

ma attenzione a non violare i limiti per gli indici:

```
-->a(0,3)
        !--error 21
Invalid index.
-->a(1,6)
        !--error 21
Invalid index.
```

Creazione di matrici particolari

SciLab fornisce funzioni per la creazione di matrici di particolare utilità o di uso comune:

```
eye         matrice indentita‘
linspace    vettore linearmente spaziato
ones        matrice di uno
zeros       matrice di zero
testmatrix  matrici speciali
rand        matrice di numeri casuali (valori tra 0 e 1)
grand       matrice di numeri casuali
            (possibilita‘ di scegliere la distribuzione)
```

Esempi

identità 3×3:

```
-->eye(3,3)
 ans  =
    1.    0.    0.
    0.    1.    0.
    0.    0.    1.
```

vettore di 5 elementi equispaziati tra 0 e 10 (estremi compresi):

```
-->linspace(0, 10, 5)
 ans  =
    0.    2.5    5.    7.5    10.
```

Esempi

matrice 2×4 di tutti valori pari a 5:

```
-->5 * ones(2, 4)
 ans  =
    5.    5.    5.    5.
    5.    5.    5.    5.
```

un vettore 1×7 di valori nulli:

```
-->zeros(1,7)
 ans  =
    0.    0.    0.    0.    0.    0.    0.
```

Esempi

matrice 3×4 di numeri causali compresi in $[a, b]$

```
-->a = 10
 a  =
    10.
-->b = 50
 b  =
    50.
-->a+(b-a)*rand(3,4)
 ans  =
    24.465444    29.305888    30.061366    35.302979
    21.689067    23.286876    27.47435     26.207816
    32.656995    33.740379    20.772499    46.738831
```

Dimensioni di una matrice

la funzione `size` ritorna il numero di righe, colonne o elementi di una matrice

```
-->A = ones(4,5);
-->size(A)
 ans  =
    4.    5.
```

Dimensioni di una matrice

dimensioni di righe, colonne, o numero di elementi:

```
-->size(A, "r")
 ans  =
    4.
-->size(A, "c")
 ans  =
    5.
-->size(A, "*")
 ans  =
    20.
```

Dimensioni di una matrice

le dimensioni possono anche essere ottenute in variabili distinte

```
-->A = ones(4,5);
-->[nr, nc] = size(A)
 nc  =
    5.
 nr  =
    4.
-->nr * nc
 ans  =
    20.
```

Matrice vuota

- è la matrice creata con [ ]
- è utile per "eliminare" il contenuto di una variabile
- permette di liberare la memoria occupata da una variabile

```
-->A=ones(1000,1000);
-->size(A)
 ans  =
    1000.    1000.
-->A=[]
 A  =
     []
-->size(A)
 ans  =
    0.    0.
```

L'operatore ':'

serve per generare un vettore di elementi da un valore minimo ad un massimo in step predefiniti

```
-->vettore = 5:10
 vettore  =
    5.    6.    7.    8.    9.    10.
```

in step di 3 unità:

```
-->vettore = 5:3:15
 vettore  =
    5.    8.    11.    14.
```

L'operatore ':'

il valore dello step può essere negativo:

```
-->vettore = 50:-4:30
 vettore  =
    50.    46.    42.    38.    34.    30.
```

ma attenzione ai limiti dell'intervallo:

```
-->vettore = 30:-4:50
 vettore  =
     []
```

L'operatore ':'

lo step può anche essere minore di 1:

```
-->vettore = 1:0.2:2
 vettore  =
    1.    1.2    1.4    1.6    1.8    2.
```

oppure in generale non intero:

```
-->vettore = 0:%pi:10
 vettore  =
    0.    3.1415927    6.2831853    9.424778
```

Uso di ':' per l'indicizzazione di matrici

```
A          l'intera matrice
A(:,:)     l'intera matrice
A(i:j,k)   gli elementi alle righe da i a j, e colonna k
A(i,j:k)   elementi alla riga i, dalle colonne j a k
A(i,:)     la riga i
A(:,j)     la colonna j
```

Esempi

```
-->A = round(100*rand(5, 5));
-->A(:,:)
 ans  =
    0.     12.    52.    29.    52.
    59.    61.    39.    9.     29.
    31.    68.    24.    62.    65.
    26.    33.    51.    35.    9.
    63.    3.     42.    71.    45.
-->A(2:3,2:4)
 ans  =
    61.    39.    9.
    68.    24.    62.
-->A(5,:)
 ans  =
    63.    3.    42.    71.    45.
```

Variazione dinamica della dimensione di matrici

aggiungere un elemento al di fuori della dimensione corrente della matrice causa il ridimensionamento automatico della matrice stessa

```
-->A=[1 2 3; 4 5 6]
 A  =
    1.    2.    3.
    4.    5.    6.
-->A(4,4) = 100
 A  =
    1.    2.    3.    0.
    4.    5.    6.    0.
    0.    0.    0.    0.
    0.    0.    0.    100.
```

Variazione dinamica della dimensione di matrici

il ridimensionamento automatico può essere sfruttato per "unire" matrici e/o vettori

```
-->A=[1 2 3; 4 5 6]
 A  =
    1.    2.    3.
    4.    5.    6.
-->B=[10 20 30; 40 50 60]
 B  =
    10.    20.    30.
    40.    50.    60.
```

Variazione dinamica della dimensione di matrici

```
-->[A B]
 ans  =
    1.    2.    3.    10.    20.    30.
    4.    5.    6.    40.    50.    60.
-->[A; B]
 ans  =
    1.     2.     3.
    4.     5.     6.
    10.    20.    30.
    40.    50.    60.
```

Esempi: operatori (1)

```
-->A = [10 10 10; 10 10 10];
-->B = [1 2 3; 4 5 6];
-->A + B
 ans  =
    11.    12.    13.
    14.    15.    16.
-->A - B
 ans  =
    9.    8.    7.
    6.    5.    4.
-->A .* B
 ans  =
    10.    20.    30.
    40.    50.    60.
```

Esempi: operatori (2)

```
-->B = [1 2 3; 4 5 6]
 B  =
    1.    2.    3.
    4.    5.    6.
-->B'
 ans  =
    1.    4.
    2.    5.
    3.    6.
```

Esempi: operatori (3)

```
-->A = 3 * ones(2, 3)
 A  =
    3.    3.    3.
    3.    3.    3.
-->B = [1 2 3; 4 5 6]
 B  =
    1.    2.    3.
    4.    5.    6.
-->A' * B
 ans  =
    15.    21.    27.
    15.    21.    27.
    15.    21.    27.
```

Esempi: operatori (4)

```
-->A = 3 * ones(2, 3)
 A  =
    3.    3.    3.
    3.    3.    3.
-->B = [1 2 3; 4 5 6]
 B  =
    1.    2.    3.
    4.    5.    6.
-->A * B'
 ans  =
    18.    45.
    18.    45.
```

Esempi: operatori (5)

```
-->vet = 0:1.5:10
 vet  =
    0.    1.5    3.    4.5    6.    7.5    9.
-->vet > 5
 ans  =
  F F F F T T T
-->vet <= 3
 ans  =
  T T T F F F F
-->vet == 1.5
 ans  =
  F T F F F F F
-->vet <> 1.5
 ans  =
  T F T T T T T
```