Università degli Studi di Ferrara

Corso di Laurea in Chimica - A.A. 2018 – 2019

Programmazione
Lezione 11 – Controllo di Flusso in MATLAB

Docente: Lorenzo Caruso – lorenzo.caruso@unife.it

Nelle lezioni precedenti

- Script: scrivere codice in MATLAB
- Funzioni in MATLAB
- Debug: concetti e comandi utili per risolvere problemi nel codice

In questa lezione

- Operatori relazionali e logici
- Costrutto if
- Ciclo for
- Ciclo while

Operatori relazionali

Espressione	Significato
a < b	minore
a > b	maggiore
a <= b	minore od uguale
a >= b	maggiore od uguale
a == b	uguale
a ~= b	diverso

Operatori relazionali

Vengono utilizzati quando si desidera eseguire un **confronto**
Si osservi che:

- a e b possono essere due variabili oppure una variabile ed una costante
- Se si confrontano uno scalare ed una variabile con indice (array), lo scalare si “ espande” fino alla dimensione della variabile

La risposta è una matrice dello stesso ordine della matrice che si confronta con elementi uguali a

- 1 se il confronto risulta VERO
- 0 se il confronto risulta FALSO

E’ importante non confondere l’operazione di confronto “a è uguale a b?” che presenta la sintassi “a == b” con l’**operatore di assegnazione** “a=b” (copia in a il valore contenuto in b).

Operatori realizionali: esempio

Proviamo ad eseguire le seguenti istruzioni nella command window:

```
>> x=5;
>> x>=[2 -1 3;8 0 6]
```

Il risultato che otterremo sarà una matrice così formata:

```
ans =
1 1 1
0 1 0
```

Operatori logici

Espressione	Significato
&	AND logico
|	OR logico
~	NOT logico

Operatori logici: esempi

```
>> (3>4) & 1
ans =
   logical
   0

>> (3<4) & 1
ans =
   logical
   1
```

Precedenze tra operatori

Abbiamo dunque visto 3 tipologie di operatori:

- Operatori aritmetici
- Operatori relazionali (di confronto)
- Operatori logici

Nelle istruzioni in cui tali operatori sono presenti, vengono rispettate le seguenti precedenze:

La programmazione strutturata in Matlab

MATLAB può essere considerato un linguaggio di programmazione alla stregua del Fortran o del C in quanto permette l'utilizzo delle strutture sintattiche tradizionali e l'uso appropriato di funzioni per codificare in modo semplice e flessibile gli algoritmi classici del Calcolo Numerico.
Ricordiamo che MATLAB è un linguaggio interpretato, quindi non utilizza compilatori:

- dal punto di vista dell'efficienza, della velocità e della portabilità può risultare meno competitivo rispetto ad altri, per esempio il C.

La programmazione strutturata in Matlab

Ricordiamo:
La **sequenza**: specifica l'ordine in cui le istruzioni si susseguono
La **selezione**: permette di scegliere fra due alternative la sequenza di esecuzione
L'**iterazione**: permette di ripetere più volte la stessa sequenza fino al verificarsi di una condizione

Le strutture di controllo devono essere pensate come schemi di composizione:

- Una sequenza può contenere iterazioni e selezioni
- Una iterazione può contenere a sua volta sequenze e selezioni
- Una selezione può contenere a sua volta sequenze e iterazioni

Il costrutto if

if
se si verifica la proposizione indicata subito dopo, si eseguono tutte le istruzioni fino a trovare l'istruzione di **end** o l'istruzione **else**

else
si eseguono le istruzioni fino al primo end successivo se la proposizione indicata nell'if **non** risulta verificata.

Il costrutto if

La struttura descritta si sintetizza nel seguente schema :

if <una o più espressioni di confronto>
↑
istruzioni
↓
else
↑
istruzioni
↓
end

Il costrutto if: esempio

Si vuole calcolare il valore assoluto del numero reale x .
Questo si realizza con la sequenza:

```
if x >= 0
assoluto=x;
else
assoluto=-x;
end
assoluto
```

Il costrutto for

In MATLAB la sintassi del costrutto **for** è la seguente:

for k=n1:n3:n2

Dove

- n1: primo valore assunto dalla variabile k
- n2: massimo valore che può assumere k
- n3: incremento che ha k ad ogni esecuzione del ciclo (facoltativo)

La variabile k assume così i valori: n1, n1+n3, n1+2*n3, […] finché non viene superato il valore n2 .

Il valore n3, se omesso, si assume uguale a 1.

Il costrutto for: esempio

Supponiamo di voler calcolare il valore di:

$$\sum_{k=1}^{n} a_k \, , n = 10, a_k = k, k = 1,2, \ldots, 10.$$

In MATLAB possiamo implementare il calcolo richiesto con la seguente sequenza di istruzioni:

```
clear;clc;
n=10;
a=[1 2 3 4 5 6 7 8 9 10];
somma=0;
for k=1:n
   somma=somma+a(k);
end
somma
```

Cicli annidati: esempio

Supponiamo di voler costruire la matrice di Hilbert H , di ordine 10. Considerato che i suoi elementi sono

$$h_{ij} = \frac{1}{i + j - 1}$$

per i, j = 1, 2, . . . , 10

questo si realizza con il seguente codice:

```
clear; clc;
n=10;
for i=1:n
   for j=1:n
       H(i,j)=1/(i+j-1);
   end
end
H
```

Il costrutto while

- Ricordiamo: l'istruzione **while** si utilizza se si prevede di eseguire un insieme di istruzioni finché non risultano verificate una o più condizioni.
- Supponiamo di voler ripetere un certo blocco di istruzioni finché non vengano verificate contemporaneamente le proposizioni prop1 e prop2.

- Per realizzare questo algoritmo si utilizza la seguente sequenza:

  - while prop1 & prop2
  - % blocco di istruzioni
  - end

Il costrutto while: esempio

Si vuole determinare la precisione di macchina (eps)

```
clear;clc;
precma=1;
while precma+1>1
    precma=precma/2;
end
precma=precma*2
```

L’ultima istruzione riporta il valore della variabile ad eguagliare eps in quanto l’ultima divisione è già stata eseguita.

L’istruzione break

- Un ciclo FOR o WHILE può essere interrotto in qualunque momento inserendo l’istruzione break
- Quando tale comando è eseguito, MATLAB salta direttamente all’istruzione end , con cui termina il ciclo

➔Non è possibile inserirsi all’interno di un ciclo senza passare necessariamente da una istruzione di *for* o di *while*

Esercizio 1

Implementare in uno script l’algoritmo precedentemente visto per il calcolo del valore assoluto di un numero con inserimento del numero da parte dell’utente

Esercizio 1b

Trasformare il precedente script in una function, testare la function richiamandola da command window

Esercizio 2

Implementare in uno script l'algoritmo per il calcolo di:

$$\sum_{k=1}^{n} a_k \, , n = 10, a_k = k, k = 1,2, \ldots, 10.$$

Esercizio 2b

Trasformare il precedente algoritmo in una function che realizzi la sommatoria generica di qualunque vettore le venga passato in ingresso

Realizzare uno script che inizializzi il vettore a piacere e richiami la funzione, visualizzi poi il risultato

Si confronti il proprio risultato con il risultato della funzione predefinita di MATLAB sum()

Esercizio 3

Realizzare uno script con che trovi il massimo di un vettore:

- Inizializzare un vettore di n elementi (a piacere) con numeri interi
- Implementare l’algoritmo di ricerca del massimo

Soluzione esercizio 3

```
clear;clc;
v=[2 -3 5 7]; n=length(v);
massimo=v(1);
for j=2:n
   if v(j)>massimo
      massimo=v(j);
   end
end
massimo
```

Grazie per l’attenzione

Riferimenti

Il corso di programmazione per il primo anno della Laurea Triennale in Matematica nasce con l'intento di unire ai principi di programmazione una conoscenza basilare di uno degli strumenti software più diffusi nell'ambito matematico: Matlab.

Per la parte introduttiva di MATLAB:
L. Pareschi, G. Dimarco “Introduzione a MATLAB”, corso di Laboratorio di Calcolo Numerico 2006

Università degli Studi di Ferrara

Corso di Laurea in Chimica - A.A. 2018 - 2019

Programmazione
Lezione 13 – Input Output

Docente: Lorenzo Caruso – lorenzo.caruso@unife.it

Nelle lezioni precedenti

- Operatori relazionali e logici
- Costrutto if
- Ciclo for
- Ciclo while

In questa lezione

- Costrutto elseif
- Costrutto switch
- Funzioni per Input Output
- Input Output su e da file

Costrutto elseif

if Condizione1
blocco di istruzioni
elseif Condizione2
blocco di istruzioni
...
else
blocco di istruzioni
end

Il primo blocco di istruzioni sarà eseguito solo se la Condizione1 risulta essere verificata, il secondo solo se la Condizione1 risulta essere **falsa** e la Condizione2 **vera** e così via.
Il blocco di istruzioni che segue **else** sarà eseguito soltanto se nessuna delle precedenti condizioni risulta essere vera.

Il costrutto switch

Il costrutto **switch** risulta particolarmente utile qualora si debbano eseguire numerose scelte di tipo esclusivo (se una è verificata le altre sono false)

switch Espressione
case Valore1
blocco di istruzioni
case Valore2
blocco di istruzioni
...
otherwise
blocco di istruzioni
end

I blocchi di istruzioni sono eseguiti solo se l'Espressione assume il corrispondente Valore.
L'ultimo blocco di istruzioni, indicato dalla keyword otherwise, sarà eseguito solo nel caso in cui Espressione non abbia assunto nessuno dei precedenti valori

Input/Output: disp

L'istruzione **disp** consente di visualizzare una stringa di testo sullo schermo del calcolatore.
La sintassi della funzione disp è la seguente

disp(stringa di caratteri)

dove stringa di caratteri rappresenta un array di stringhe di tipo vettore o matrice, dove ogni stringa è racchiusa tra apici.
Ad esempio potremo scrivere:
>> disp('Oggi e'' lunedi''')
oggi e' lunedi'

Osserviamo che per utilizzare i simboli di apostrofo/accento all'interno della stringa si deve utilizzare il simbolo '' come delimitatore per evitare conflitti con il segnale di inizio e fine della stringa.

Input/Output: disp

```
>> disp(['Gennaio ','Febbraio ','Marzo'])
Gennaio Febbraio Marzo
```

In questo caso l'argomento è un vettore riga di stringhe, che vengono così concatenate.
Se vogliamo costruire vettori colonna è importante ricordarsi che le stringhe devono avere tutte la stessa dimensione

```
>> disp(['Gennaio ';'Febbraio';'Marzo   '])
Gennaio
Febbraio
Marzo
```

cosa che può essere facilmente ottenuta inserendo un opportuno numero di spazi.

Input/Output: valori numerici

In molte circostanze si ha inoltre la necessità di rappresentare valori numerici in uscita (con una certa formattazione).
Vediamo due differenti possibilità, la prima tramite il seguente esempio:

```
>> x=10;
>> stringa = ['x = ', num2str(x)];
>> disp(stringa)
x = 10
```

In questo caso abbiamo utilizzato la funzione num2str che consente di convertire una variabile numerica in una stringa.
Abbiamo poi **concatenato** i due elementi stringa per ottenere un output più elaborato

Input/Output: valori numerici

Un modo più versatile di costruire stringhe di caratteri per l'output di testo con formato è fornito dalle funzioni fprintf e sprintf
La sintassi della prima è del tipo

fprintf(Fid, Formato, Variabili)

Dove

- Formato è una stringa di testo che tramite l'uso di caratteri speciali indica il tipo di formato dell'output
- Variabili è una lista opzionale di variabili separate da una virgola e che hanno un corrispondente all'interno della stringa Formato
- Fid è un identificatore opzionale del file al quale l'output è inviato

Input/Output: valori numerici

La funzione sprintf ha invece la sintassi

Stringa = sprintf(Formato, Variabili)

dove in questo caso l'output viene indirizzato su una stringa di testo.

Differenza tra sprintf e fprintf

```
>> x=10;y=5.5;

>> fprintf('x = %d e y= %f\n',x,y);
x = 10 e y= 5.500000

>> stringa=sprintf('x = %d e y= %f\n',x,y);
>> disp(stringa)
x = 10 e y= 5.500000
```

Input/Output: formattatori

La stringa di Formato contiene codici che specificano il tipo di variabile che deve essere convertita in stringa e rappresentata sullo schermo del calcolatore o su file.

- il formato **%d** serve a visualizzare un numero intero
- il formato **%f** è utilizzato per i numeri reali

Input/Output: formattatori

Formato	Significato
%s	Formato stringa
%d	Formato senza parte frazionaria (intero)
%f	Formato numero deciamale
%e	Formato in notazione scientifica
%g	Formato in forma compatta usando %f o %e
\n	Inserisce un carattere di ritorno a capo
\t	Inserisce un carattere di tabulazione

Input/Output: formattatori - Esempio

Valore	%6.3f	%6.3e	%6d
2	2.000	2.000w+00	2
0.02	0.020	2.000e-02	2.000000e-02
200	200.000	2.000e+02	200
sqrt(2)	1.414	1.414e+00	1.414214e+00
sqrt(0.02)	0.141	1.414e-01	1.414214e-01

Input/Output: Utilizzo vettoriale

La principale differenza tra le funzioni MATLAB fprintf e sprintf è data dalla possibilità **di utilizzarle in modo vettoriale**

Abbiamo già visto in precedenza come costruire una semplice tabella di valori per le funzioni seno e coseno. Proviamo a costruire un nuovo script che realizzi la stessa tabella di valori ma visualizzandola in modo più efficace con l'utilizzo **vettoriale** delle funzioni appena viste

Input/Output: Utilizzo vettoriale

```
% tabcossin.m
% Realizza una tabella di valori di coseno e seno
%
n=input('Inserisci il numero di valori ? : ');
x=linspace(0,pi,n);
c=cos(x);
s=sin(x);
disp('-----------------------------------------');
fprintf('k\t x(k)\t cos(x(k))\t sin(x(k))\n');
disp('-----------------------------------------');
fprintf('%d\t %3.2f\t %6.5f\t %6.5f\n',[1:n;x;c;s]);
```

Input/Output: Utilizzo vettoriale

che produce il seguente output

```
>> tabcossin
>> Inserisci il numero di valori ? : 5
-----------------------------------------
k       x(k)          cos(x(k))          sin(x(k))
-----------------------------------------
1             0.00           1.00000            0.00000
2             0.79           0.70711            0.70711
3             1.57           0.00000            1.00000
4             2.36          -0.70711            0.70711
5             3.14          -1.00000            0.00000
```

Input/Output: Utilizzo vettoriale

Si noti l'uso vettoriale di **fprintf**:
bisogna prestare attenzione nel caso in cui si utilizzi una matrice come variabile in output

- il comando legge la matrice per colonne, applicando ad ogni elemento il corrispondente descrittore di formato
- la visualizzazione avviene naturalmente per righe e questo porta ad una sorta di 'trasposizione' della matrice

Input/Output: la funzione input

Abbiamo già trovato la funzione input in diversi esempi ed esercitazioni, vediamone le caratteristiche
La sintassi della funzione è

Variabile=input(Stringa di caratteri)

La funzione attende un ingresso da tastiera di un'**espressione MATLAB** da assegnare a Variabile.

Il valore assegnato potrà quindi essere di tipo scalare, vettoriale, oppure matrice **utilizzando la sintassi standard di MATLAB.**

Input/Output su file: il comando diary

Se volessimo salvare su file una copia della tabella generata con l'algoritmo appena visto, il modo più semplice consiste nell'utilizzo del comando **diary**
Il comando trasferisce su file tutto ciò che compare nella finestra di comando di MATLAB.
La sintassi del comando è:

diary Nomefile

il comando attiverà la copia su file di testo Nomefile di tutto ciò che comparirà sulla **command window** fino a quando non verrà dato il comando

diary off

Il quale andrà ad interrompere il salvataggio.

Input/Output su file - esempio

Il seguente script costruisce la tabella visualizzata in precedenza in un file di testo chiamato tabella.txt

```
% salvatabella.m
% Esempio di uso della funzione diary
%
diary tabella.txt
tabcossin
diary off
```

E’ importante notare che il comando diary, registrando la command window, andrà a salvare tutta la sessione di lavoro.
Per un maggiore controllo è possibile utilizzare il comando **save**

Input/Output su file: il comando save

Il comando save consente di registrare alcune variabili presenti nella memoria di MATLAB su file secondo la sintassi

save Nomefile ElencoVariabili Formato

dove Formato è un parametro opzionale.
Se tale parametro è omesso il file è salvato in **formato binario** e qualora il file non abbia estensione a questo è aggiunta l'estensione ".mat"
Il formato **-ascii** consente di salvare file in modalità testo

Input/Output su file: il comando save - esempio

```
% salvavalori.m
% Esempio di output su file con save
%
n=input('Inserisci il numero di valori: ');
x=linspace(0,pi,n);
c=cos(x);
s=sin(x);
z=1:n;
save tabella.dat z x c s -ascii
```

Input/Output su file: il comando load

Possiamo caricare il file precedentemente creato con il comando save utilizzando il comando **load** per leggere il file.
Sintassi:

load Nomefile Formato

Se vogliamo leggere un file di testo che non ha estensione allora l'uso del formato **-ascii** è obbligatorio anche in lettura, altrimenti è opzionale.
Utilizzando questo comando MATLAB crea una matrice contenente i valori precedentemente registrati ed a questa assegna il nome del file in lettura.

Nell'esempio precedente l'istruzione **load tabella.dat** legge i valori presenti nel file tabella.dat e li assegna ad una matrice chiamata tabella

Input/Output su file: il comando load - esempio

```
% visualizzavalori.m
% Esempio di input da file con load
%
load tabella.dat
A=tabella;
disp('----------------------------------------');
fprintf('k\t x(k)\t cos(x(k))\t sin(x(k))\n');
disp('----------------------------------------');
fprintf('%d\t %3.2f\t %6.5f\t %6.5f\n',A);
```

Input/Output su file: il comando print

In MATLAB è possibile salvare i grafici prodotti su file utilizzando diversi formati grafici.
La sintassi base del comando è la seguente

print Opzioni Nomefile

E consente di salvare il contenuto della attuale finestra grafica sul file Nomefile utilizzando un particolare formato grafico specificato in Opzioni

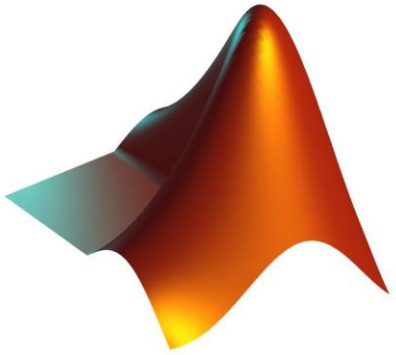
Input/Output su file: il comando print

Opzione	Formato corrispondente
-dps	PostScript bianco e nero
-deps	Encapsulated PostScript bianco e nero
-dpsc	PostScript a colori
-depsc	Encapsulated PostScript a colori
-dgif	Immagine GIF
-dbmp	Immagine Bitmap
-djpeg	Immagine JPEG compressa

Input/Output su file: il comando print - esempio

Il seguente script realizza la figura tridimensionale che MATLAB utilizza come logo, tramite il comando logo, e la salva in formato jpeg nel file mlogo.jpg

```
% salvalogo.m
% Esempio di output grafico su file
logo
print -djpeg mlogo.jpg
```



Esercizio

- Realizzare uno script che calcoli la tabella di valori di seno e coseno e la salvi su file
- Realizzare uno script che carichi il file precedentemente salvato e realizzi il grafico sovrapposto delle funzioni, si salvi poi il grafico in formato jpeg

Grazie per l’attenzione

Riferimenti

Il corso di programmazione per il primo anno della Laurea Triennale in Matematica nasce con l'intento di unire ai principi di programmazione una conoscenza basilare di uno degli strumenti software più diffusi nell'ambito matematico: Matlab.

Per la parte introduttiva di MATLAB:
L. Pareschi, G. Dimarco “Introduzione a MATLAB”, corso di Laboratorio di Calcolo Numerico 2006