Introduzione a Matlab

Loris Giulivi – Nicolò Folloni

Introduzione al linguaggio Matlab

Struttura di un programma Matlab

- Script: File di testo contenete una sequenza di comandi Matlab. Non ha variabili di input né output, ma viene letto dal compilatore riga per riga, quando eseguito.

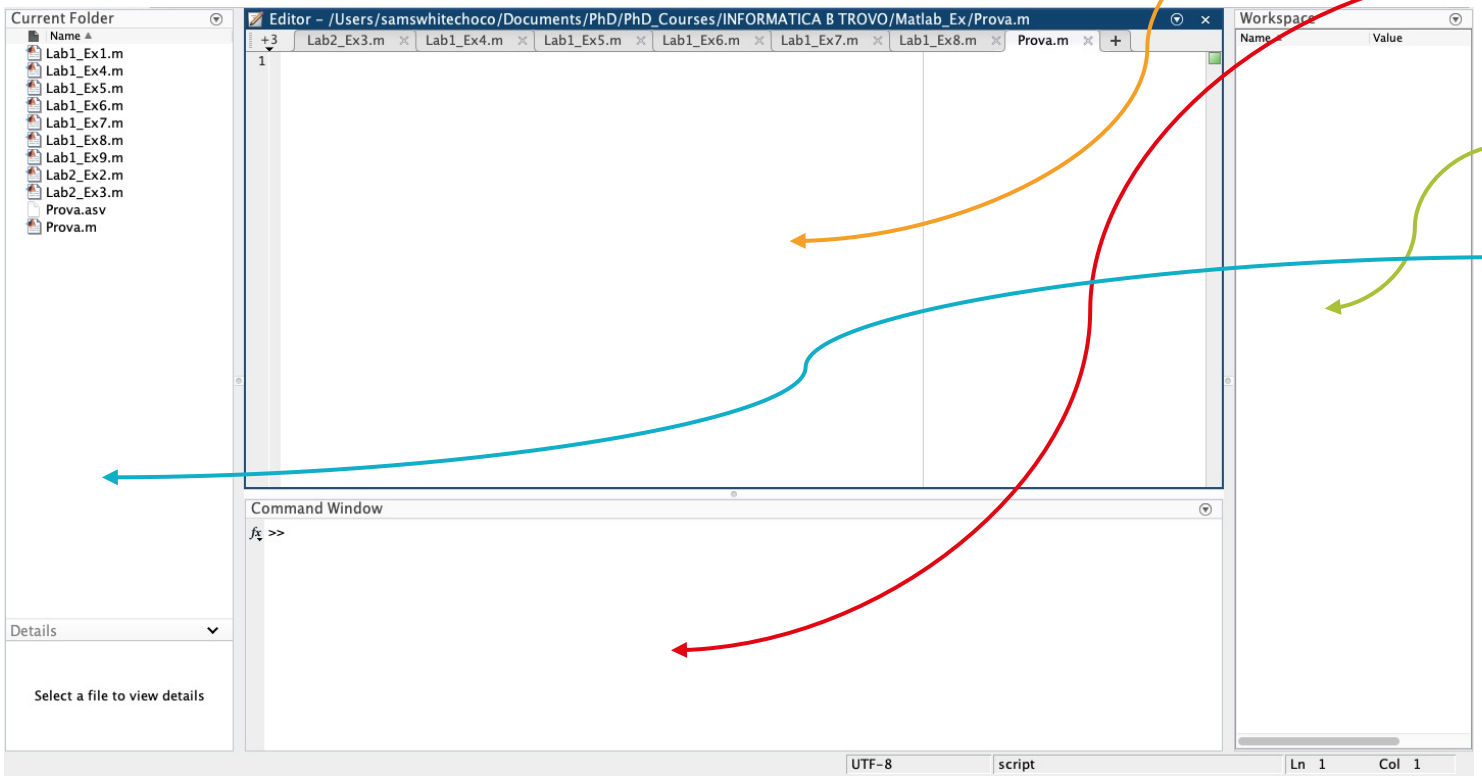
```
% istruzione 1
% istruzione 2
% ...
% istruzione n
```

- Function: Le funzioni sono delle parti di codice che permettono di essere richiamate all'interno di uno script. Sono particolarmente utili per non dover riscrivere più volte la stessa istruzione all'interno del codice.
Matlab dispone di un grande numero di funzioni già implementate.

```
function [outputArg1,outputArg2] = untitled(inputArg1,inputArg2)
%UNTITLED20 Summary of this function goes here
%   Detailed explanation goes here
outputArg1 = inputArg1;
outputArg2 = inputArg2;
end
```

I Primi Passi

Ambiente di Sviluppo



- Editor: Ambiente in cui viene scritto il testo di script e funzioni.
- Command Window: permette di eseguire direttamente una o più linee di codice
- Workspace: Contiene tutte le variabili a cui è stato assegnato un valore.
- Current Folder: Mostra tutti i file presenti nella cartella di lavoro attiva.

Pillole di Matlab

- clear: elimina tutte le variabili presenti nel workspace
- clc: rimuove il testo già eseguito presente nella command window
- help *nomefunzione*: fornisce informazioni sull'utilizzo della funzione richiesta

In Matlab le variabili non sono tipizzate. Non è quindi necessario indicare il loro tipo quando vengono definite

Per commentare una linea di codice, si usa il comando %
Per commentare più linee di codice, è necessario racchiuderlo tra i comandi %{ e %}

```
a = 1;
b = 'ciao';
c = [1, 2, 3];

% commento
%{
commento 1
commento 2
commento 3
%}
```

La presenza del punto e virgola ; impedisce che la linea di codice corrente venga stampata a video. Non è necessario alla riuscita della compilazione, ma il suo uso è fortemente consigliato.

Le funzioni più comuni

Stampare a schermo - **disp**

La funzione **disp** ci permette di stampare a schermo e comunicare con l'utente.

```
a = 1;
txt = ['Il numero è: ', num2str(a)];
disp(txt)
```

Ricevere input dall'utente - **input**

La funzione **input** ci permette di ricevere in input una variabile dall'utente e salvarla in memoria

```
a = input('Inserisci un numero: ');
b = input('Inserisci un carattere: ', 's');
```

Chiamare una funzione

Per chiamare una funzione, la sintassi è:

```
[output1, output2] = nomefunzione(input1, input2);
```

Le funzioni più comuni (parte2)

zeros(m, n): crea una matrice di zeri di dimensione mxn
ones(m, n): crea una matrice di uni di dimensione mxn
eye(n): crea la matrice identità di dimensione nxn
rand(m, n): crea una matrice random con numeri compresi tra 0 e 1 di dimensione mxn
randi(p,m,n): crea una matrice random con numeri compresi tra 1 e p di dimensione mxn
length(x): restituisce la dimensione del vettore x
size(A): restituisce le dimesnioni della matrice A
max(v): restituisce il massimo del vettore v e il suo indice
min(v): restituisce il minimo del vettore v e il suo indice
mean(v): restituisce la media del vettore v
mod(n,p): restituisce il resto della divisione intera tra n e p
find(p): restituisce gli indici degli elementi che soddisfano p

Per avere informazioni complete su una funzione, usare il comando *help*

```
>> help max
 max    Maximum elements of an array.
    M = max(X) is the largest element in the vector X. If X is a matrix, M
    is a row vector containing the maximum element from each column. For
    N-D arrays, max(X) operates along the first non-singleton dimension.

    When X is complex, the maximum is computed using the magnitude
    max(ABS(X)). In the case of equal magnitude elements the phase angle
    max(ANGLE(X)) is used.

    [M,I] = max(X) also returns the indices into operating dimension
    corresponding to the maximum values. If X contains more than one
    element with the maximum value, then the index of the first one
    is returned.

    C = max(X,Y) returns an array with the largest elements taken from X or
    Y. X and Y must have compatible sizes. In the simplest cases, they can
    be the same size or one can be a scalar. Two inputs have compatible
    sizes if, for every dimension, the dimension sizes of the inputs are
    either the same or one of them is 1.

    M = max(X,[],'all') returns the largest element of X.

    M = max(X,[],DIM) or [M,I] = max(X,[],DIM) operates along the
    dimension DIM.

    M = max(X,[],VECDIM) operates on the dimensions specified in the vector
    VECDIM. For example, max(X,[],[1 2]) operates on the elements contained
    in the first and second dimensions of X.
```

If, for e while

In Matlab, i costrutti if, for e while devono essere terminate dalla keyword **end**

Costrutto **if**

```
if %condizione
        %istruzione 1
        %istruzione 2
        %...
elseif
        %istruzione 3
        %istruzione 4
        %...
else
        %istruzione 5
        %istruzione 6
        %...
end
```

L'*elseif* ha il significato di un if annidato

Costrutto **for**

```
for ii = %vettore
        %istruzione 1
        %istruzione 2
end
```

Le iterazioni del for sono effettuate su un vettore

Costrutto **while**

```
while %condizione
        %istruzione 1
        %istruzione 2
        %...
end
```

Non esiste il costrutto *do while*, ma si può aggirare il problema

Operazioni Logiche

Così come in C, anche in Matlab *vero* significa diverso da zero.

```
a = 0;
b = 1;

a && b %and
a & b %and
a || b %or
a | b %or
~a %negazione

a == b %uguale
a ~= b %diverso
a > b %maggiore
a < b %minore
```

Più condizioni logiche si possono concatenare insieme attraverso l'uso delle parentesi:

```
((a == b) && (a < c)) || (a ~= 0)
```

Condizione 1

Condizione 2

Vettori e Matrici

In Matlab, tutto è trattato come una matrice.
Un vettore è quindi visto come una matrice di dimensione 1x1.

A differenza di C, gli indici in Matlab partono da 1.

```
v = [1, 2, 3, 4, 5];
w = 1:5;
z = v';

v(1) = 4;
v(end) = 2;
v(1:end) = 1;
I = [2,4];
v(I) = 3;
```

La dichiarazione di un vettore riga si effettua così. Le virgole sono facoltative

La keyword *end* corrisponde all'indice dell'ultimo elemento del vettore considerato

```
A = [1, 2; 3, 4];
B = A';
A(1,2) = 3;
A(:, 1) = [0; 0];

C = A + B;
D = A*B;
E = A.*B;
F = A.^2;
```

Per considerare tutte le righe/colonne di una matrice, al posto di *1:end* basta scrivere *:*

Il prodotto tra matrici è indicato con *, mentre il prodotto elemento per elemento con .*

Esercizio 1

ES1: Scrivere un programma che chiede in input N numeri interi con N stabilito dall'utente, e che li stampa in ordine crescente, ne stampa la media e stampa il numero di valori distinti inseriti in input

Esercizio 1 - Soluzione

```
%%
clc
clear

prompt = 'Inserisci il valore di N: ';
N = input(prompt);

x = zeros(N,1);
for i = 1:N
    prompt = ['Inserisci il numero in posizione ', num2str(i), ': '];
    x(i) = input(prompt);
end
```

```
%%
% Soluzione Matlab
y_matlab = sort(x, 'ascend');

% Soluzione C
y_c = zeros(N,1);
x_sort = x;
for i = 1:N
    max_val = -inf;
    for j = 1:N-i+1
        x_val = x_sort(j);
        if x_val > max_val
            y_c(i) = x_val;
            max_val = x_val;
            j_max = j;
        end
    end
    x_sort(j_max) = [];
end
```

```
%%
% Soluzione Matlab
media_matlab = mean(x);

% Soluzione C
sum = 0;
for i = 1:N
    sum = sum + x(i);
end
media = sum/N;
```

```
%%
% Soluzione Matlab
dist_val_mat = length(unique(x));

% Soluzione C
dist_val = N;
for i = 1:N
    for j = 1:N
        if (i ~= j && x(i) == x(j) && x(i) ~= inf)
            x(j) = inf;
        end
    end
end

count = 0;
for i = 1:N
    if x(i) ~= inf
       count = count + 1;
    end
end

txt = ['Il numero di valori distinti è: ', num2str(count)];
disp(txt)
```

Esercizio 2

ES2: Scrivere un programma che richiesta in input all'utente la sua data di nascita (nella forma ddmmyyyy) stabilisce in che stagione è nato.

Esercizio 2 - Soluzione

```
birth = input('Inserisci la tua data di nascita: ', 's');
day = str2double(birth(1:2));
month = str2double(birth(3:4));
year = str2double(birth(5:end));


primavera = (month == 3 && day >= 21)||(month == 4)||(month == 5)||(month == 6 && day <=20);
estate = (month == 6 && day >= 21)||(month == 7)||(month == 8)||(month == 9 && day <=22);
autunno = (month == 9 && day >= 23)||(month == 10)||(month == 11)||(month == 12 && day <=21);
inverno = (month == 12 && day >= 22)||(month == 1)||(month == 2)||(month == 3 && day <=20);

if primavera
    disp('Sei nato in primavera!')
elseif estate
    disp('Sei nato in estate!')
elseif autunno
    disp('Sei nato in autunno!')
elseif inverno
    disp('Sei nato in inverno!')
end
```

Esercizio 3

ES3: Scrivere un programma che chiesta in input una parola stabilisce se questa contiene più vocali o consonanti. Se il numero di vocali e consonanti è uguale il programma continua a chiedere parole finchè non si verifica una disuguaglianza.

Esercizio 3 - Soluzione

```
vocali = ['a', 'e', 'i', 'o', 'u'];
flag = 0;

while flag == 0
    txt = input('Inserisci una parola: ', 's');
    N = length(txt);
    v_counter = 0;


    for i = 1:N
        if ismember(txt(i), vocali)
            v_counter = v_counter + 1;
        end
    end

    c_counter = N - v_counter;

    if v_counter > c_counter
        disp('Ci sono più vocali')
        flag = 1;
    elseif c_counter > v_counter
        disp('Ci sono più consonanti')
        flag = 1;
    else
        disp('Ci sono lo stesso numero di vocali e consonanti')
    end
end
```

Esercizio 4

ES4: Produrre una matrice di dimensione NxN tale che in posizione (i,j) è contenuta la probabilità che se estraggo con probabilità uniforme un numero naturale tra 0 e i, questo sia maggiore o uguale di j.

Esercizio 4 - Soluzione

```
N = input('Inserire dimensione N: ');
M = zeros(N,N);

for i = 1:N
    for j = 1:i
        if i >= j
            M(i,j) = (i+1-j)/(i+1);
        end
    end
end

disp(M)
```

Esercizio 5

ES5: Data la matrice A di dimensione NxM, rappresentante il punteggio ottenuto da ogni concorrente (N) in un determinato gioco (M), e sapendo che il punteggio totale è dato dalla somma dei punteggi nei singoli giochi, stabilire:

1) quale concorrente ha ottenuto il punteggio totale più alto
2) quale concorrente ha ottenuto il punteggio totale più basso
3) in quale gioco si è ottenuto, in media, il punteggio più alto
4) in quale gioco si è ottenuto, in media, il punteggio più basso
5) riordinare la matrice A in ordine decrescente per righe, in base al punteggio ottenuto da ciascun giocatore

Esercizio 5 - Soluzione

```
%% 1-2
P = sum(A,2);
[~, pmax] = max(P);
[~, pmin] = min(P);

%% 3-4
G = sum(A,1);
[~, gmax] = max(G);
[~, gmin] = min(G);

%% 5
[~, I] = sort(P, 'descend');
A_sorted = A(I, :);
```

Esercizio 6

ES6: Data una matrice rappresentante una scacchiera 8x8, chiedere all'utente di inserire in input la posizione di un cavallo e di una torre. Stabilire se la torre può mangiare il cavallo in una sola mossa (e viceversa).

Esercizio 6 - Soluzione

```
S = zeros(8);

cavallo = zeros(2,1);
for i = 1:2
    cavallo(i) = input(['Inserisci la posizione in coordinata ', num2str(i), ' del cavallo: ']);
end
S(cavallo(1), cavallo(2)) = 1;

torre = zeros(2,1);
for i = 1:2
    torre(i) = input(['Inserisci la posizione in coordinata ', num2str(i), ' della torre: ']);
end
S(torre(1), torre(2)) = -1;
```

```
%% La torre può mangiare il cavallo?
if sum(S(torre(1), :)) == 0 || sum(S(:,torre(2))) == 0
    disp('La torre ha mangiato il cavallo')
else
    disp('La torre non può mangiare il cavallo')
end
```

```
%% Il cavallo può mangiare la torre?
indices = [-2, -1, 1, 2];
flag = 0;
for i=indices
    for j=indices
        if j~=i && cavallo(1) + i > 0 && cavallo(1) + i < 9 && cavallo(2) + j > 0 && cavallo(2) + j < 9
            pos = S(cavallo(1) + i, cavallo(2) + j);
            if pos == -1
                flag=1;
            end
        end
    end
end

if flag == 0
    disp('Il cavallo non può mangiare la torre')
else
    disp('Il cavallo ha mangiato la torre')
end
```

Esercizio 7

ES7: Generare una matrice Nx2 rappresentante un percorso a pannelli di vetro sospesi nel vuoto. Ad ogni passo è possibile scegliere se posizionarsi sul pannello di destra o di sinistra, e per ciascuna iterazione, una delle due posizioni non reggerà il peso del concorrente, mentre l'altra sì. Implementare uno script che simula il percorso, richiedendo in input all'utente quale direzione prendere a ciascun passo.

Esercizio 7 - Soluzione

```
N = 15;
P = zeros(N,2);
i = randi(2,N,1);
P(i==1,1) = 1;
P(i==2,2) = 1;

vivo = 1;
passo = 1;
disp('Scrivi d per andare a destra, s per andare a sinistra.')

while vivo == 1
    dir = input('Quale pannello scegli? ', 's');
    if dir == 'd'
        pos = 2;
    else
        pos = 1;
    end

    if P(passo, pos) == 1 && passo < N
        vivo = 0;
        disp('Fine del gioco...')
    elseif P(passo, pos) == 0 && passo < N
        txt = ['Bene! Mancano ', num2str(N-passo), ' pannelli.'];
        disp(txt)
    else
        disp('Complimenti! Hai vinto!')
        vivo = 2;
    end

    passo = passo + 1;

end
```