



UNIVERSITÀ  
DEGLI STUDI  
DI PADOVA

## Primi script in MATLAB

Laboratorio di Calcolo Numerico

11 Marzo 2019

# Introduzione alla programmazione

- **Obiettivo:** scrivere programmi e sviluppare algoritmi in MATLAB
- Si utilizzano gli m-file, ovvero file di testo salvati con estensione .m
- I file di testo .m contengono le sequenze di comando da far eseguire al calcolatore
- Esistono due tipologie di programmi in MATLAB:
  - ① **SCRIPT:** lista di comandi/istruzioni tra cui funzioni predefinite e/o create dall'utente
  - ② **FUNCTION:** lista di comandi/istruzioni con variabili di input/output
- Gli **SCRIPT** operano sulle variabili contenute in memoria
- Le variabili della **FUNCTION** sono locali e non restano in memoria dopo l'esecuzione della stessa

## SCRIPT - Esempio

- Scrivere uno script in MATLAB per la stampa della frase 'hello world'
- Avviare MATLAB, aprire un nuovo file (Home -> New) e salvarlo come *hw.m*
- Scrivere il seguente testo nel file *hw.m*:

---

```
1 % Script per la stampa della stringa 'hello world'  
2 fprintf('hello world \n')
```

---

- Salvare il file
- Eseguire lo script da linea di comando:  
    >> filename + INVIO

## Esercizio 1

Scrivere un programma con uno script che valuti la funzione

$$y = 4x^3 - 3x^2 - 3$$

in un punto  $x$ . Il punto scelto è  $x = 3$ . La soluzione sarà  $y = 78$ .

Utilizzare la sintassi del comando `fprintf` per la visualizzazione del risultato dell'operazione. Stampare le variabili con un formato adeguato.

**NOTA:** Salvare lo script come *eval1.m*

## Esercizio 2

Scrivere un programma con uno script che valuti la stessa funzione

$y = 4x^3 - 3x^2 - 3$  in un punto letto da file di input. HINT: usare i comandi `save` e `load` Salvare questo script come *eval2.m*.

## SCRIPT 1

---

---

```
1 close all
2 clear all
3 % Questo script valuta la funzione
4 %  $y=4x^3-3x^2-3$  in  $x=3$ 
5 x = 3.0;
6 y = 4.0*x^3-3.0*x^2-3.0;
7 fprintf('La soluzione y vale %1.2f \n', y)
```

---

## Il ciclo for

Il ciclo `for` serve per ripetere le stesse istruzioni un certo numero di volte. La sintassi è la seguente:

---

---

```
1 ...
2 for contatore = minimo : passo : massimo
3     istruzioni
4 end
5 ...
```

### Esempio

---

---

```
1 n = 10;
2 for i = 1:n
3     fprintf('Il valore di i e'' %d \n', i )
4 end
```

---

## Il ciclo `while`

Il ciclo `while` serve per ripetere le stesse istruzioni fintantoché è vera una determinata condizione. La sintassi è la seguente:

---

```
1 ...
2 while espressione logica
3     istruzioni
4 end
5 ...
```

---

### Esempio

---

```
1 n = 0;
2 a = 6;
3 while n < exp(a)
4     n = n + 1;
5 end
6 fprintf('Il valore finale di n e'' %d \n', n)
```

---

# Operatori logici in MATLAB

Maggiore	>
Maggiore o uguale	>=
Minore	<
Minore o uguale	<=
Uguale	==
Diverso	~=

$a > b$

Vero se 'a' maggiore di 'b'

Falso se 'a' minore o uguale a 'b'

AND	&&	Congiunzione di più predicati
OR		Disgiunzione di più predicati
NOT	~	Negazione di un predicato o di una proposizione

$(a > -2) \ \&\& \ (a <= 2)$

Vero se  $a \in [-2, 2]$

Falso se  $a < -2$  o  $a > 2$

# Operatori logici in MATLAB

- Congiungere (AND) due predicati VERI  $\rightarrow$  proposizione VERA
- Congiungere (AND) due predicati FALSI  $\rightarrow$  proposizione FALSA
- Congiungere (AND) un predicato VERO e uno FALSO  $\rightarrow$  proposizione FALSA
- Disgiungere (OR) due predicati VERI  $\rightarrow$  proposizione VERA
- Disgiungere (OR) due predicati FALSI  $\rightarrow$  proposizione FALSA
- Disgiungere (OR) un predicato VERO e uno FALSO  $\rightarrow$  proposizione VERA
- Negare (NOT) un predicato FALSO  $\rightarrow$  predicato VERO
- Negare (NOT) un predicato VERO  $\rightarrow$  predicato FALSO

## Il costrutto if

Il costrutto `if` permette di verificare se un'espressione logica risulti vera o falsa. Se l'espressione è vera il programma esegue le istruzioni 1, altrimenti vengono eseguite le istruzioni 2. La sintassi è la seguente:

---

---

```
1 ...
2 if (espressione logica)
3     {blocco istruzioni 1}
4 else
5     {blocco istruzioni 2}
6 end
7 ...
```

---

### Esempio

---

---

```
1 % Trovare il massimo tra a e b (variabili scalari)
2 a = 6;
3 b = 2;
4 if (a < b)
5     maxval = b;
6 else
7     maxval = a;
8 end
```

---

### Esercizio 3

Scrivere un programma mediante uno script che valuti la funzione  $y = 4x^3 - 3x^2 - 3$  in  $n + 1$  punti equispaziati nell'intervallo

$I = [x_{min}, x_{max}]$ :  $x_0 = x_{min}, x_1 = x_0 + h, \dots, x_n = x_{max}$ , con  $h = (x_{max} - x_{min})/n$ .

Considerare, per esempio,  $x_{min} = -4$ ,  $x_{max} = 6$  e  $n = 11$ .

Utilizzare un ciclo `for`.

## SCRIPT 3

---

---

```
1 close all
2 clear all
3 % Questo script valuta la funzione  $y=4x^3-3x^2-3$  in n+1 punti
4 xmin = -4.0;
5 xmax = 6.0;
6 n = 11;
7 h = (xmax-xmin)/n;
8 for i = 1:(n+1)
9     xi = xmin + h*(i-1);
10    y = 4.0*xi^3-3.0*xi^2-3.0;
11    fprintf('La soluzione y vale %1.2f in x = %1.2f \n', y, xi)
12 end
```

---

## Esercizio 4

Scrivere un programma mediante uno script che calcoli le radici reali  $x_1, x_2$  di una equazione di secondo grado  $ax^2 + bx + c = 0$ . I coefficienti  $a, b, c$  sono acquisiti dalla Command Window.

La sintassi per invocare il comando radice quadrata è:  $y = \mathbf{sqrt}(x)$ .  
Utilizzare il costrutto *if* per controllare che il discriminante (positivo, negativo, nullo).

## SCRIPT 4

Sostituire i puntini e scrivere le istruzioni per i vari casi *if*.

```
1 clear all
2 close all
3 % Calcolo delle radici reali di un' equazione di secondo grado
4 %
5 delta = b^2 - 4.0*a*c;      % Calcolo il delta
6 % Costrutto logico
7 if (delta > 0)              % Esistono due soluzioni
8     ...
9 else
10     if (delta == 0)         % Soluzioni coincidenti
11         ...
12     else
13         ...                % Non ci sono soluzioni
14     end
15 end
```

## Esercizi

- 1 Implementare il calcolo del fattoriale di un numero  $n$ :

$$n! = n \cdot (n - 1) \cdot \dots \cdot 1$$

HINT: utilizzare il ciclo `while`

- 2 Implementare in uno script un programma che, dati due numeri interi  $a$  e  $b$ , controlla se il primo è multiplo dell'altro. HINT: utilizzare la funzione di MATLAB `mod(a,b)` che restituisce il resto della divisione tra interi
- 3 L'utente inserisce un anno ed il calcolatore verifica se è bisestile. Un anno è bisestile quando è divisibile per 4 ma non per 100 oppure se è divisibile per 400.