



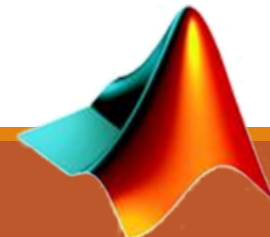
UNIVERSITÀ DEGLI STUDI DI SALERNO

Fondamenti di Informatica

Cenni di Debugging in MATLAB

Prof. Arcangelo Castiglione

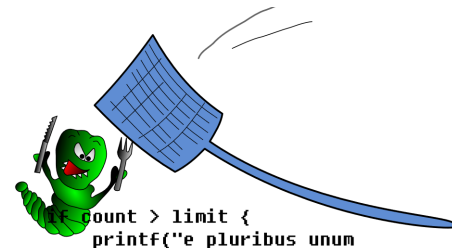
A.A. 2016/17



MATLAB®

Debugging – 1/7

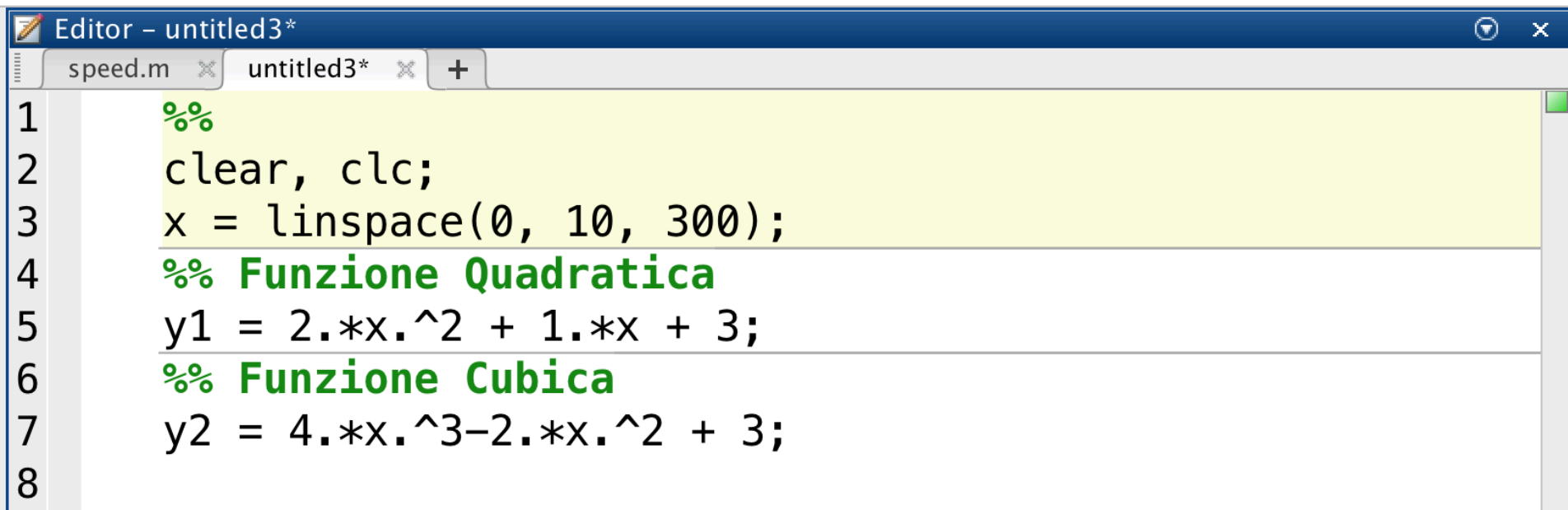
- Il **debugging** (o semplicemente debug) è l'attività che permette al programmatore di individuare la porzione di programma (codice sorgente) in cui è contenuto un errore o **bug** (bug letteralmente significa **insetto**), al fine di correggerlo
- MATLAB fornisce un *Debugger*
 - Strumento per supportare il programmatore durante la fase di debugging (ovvero per l'individuazione di errori/bug)



Debugging – 2/7

- MATLAB permette di isolare e valutare singolarmente *sezioni (celle)* di codice, al fine di restringere la ricerca di eventuali bug soltanto a tali sezioni
 - Questa tecnica viene detta modalità cella
- Una *sezione (o cella)* di codice valutata dal debugger inizia con %%
 - **N.B.**: Da non confondere con i **commenti**, che iniziano con %

Debugging – 3/7

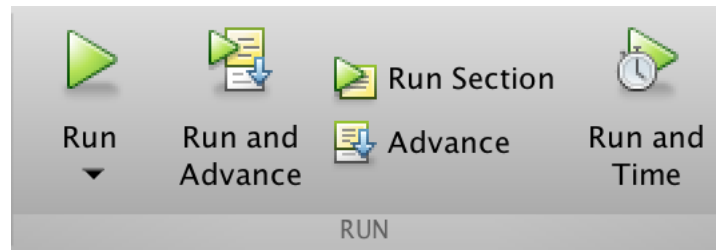


```
Editor - untitled3*
speed.m x untitled3* x +
1 %%
2 clear, clc;
3 x = linspace(0, 10, 300);
4 %% Funzione Quadratica
5 y1 = 2.*x.^2 + 1.*x + 3;
6 %% Funzione Cubica
7 y2 = 4.*x.^3-2.*x.^2 + 3;
8
```

- Questa porzione di codice è composta da **tre sezioni**
- MATLAB separa visivamente ogni sezione mediante linee orizzontali

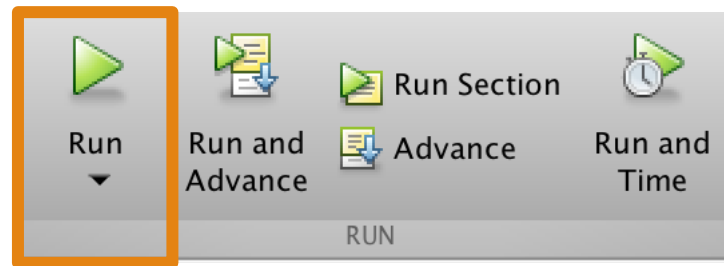
Debugging – 4/7

- È possibile eseguire la porzione di codice
 - Interamente
 - Sezione per sezione
 - Solo una determinata sezione



Debugging – 4/7

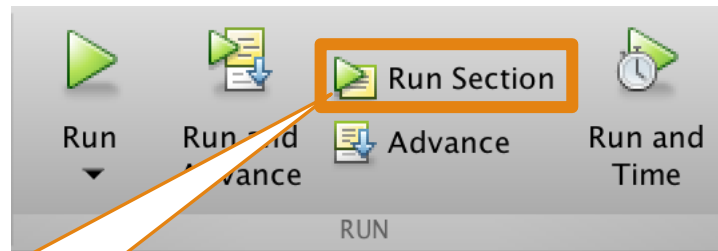
- È possibile eseguire la porzione di codice
 - Interamente
 - Sezione per sezione
 - Solo una determinata sezione



Esegue l'intera porzione di codice

Debugging – 4/7

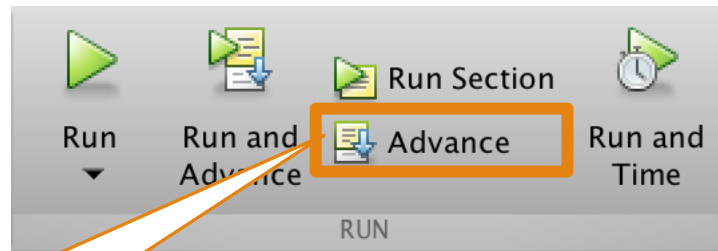
- È possibile eseguire la porzione di codice
 - Interamente
 - Sezione per sezione
 - Solo una determinata sezione



Esegue solo la sezione selezionata, o corrente

Debugging – 4/7

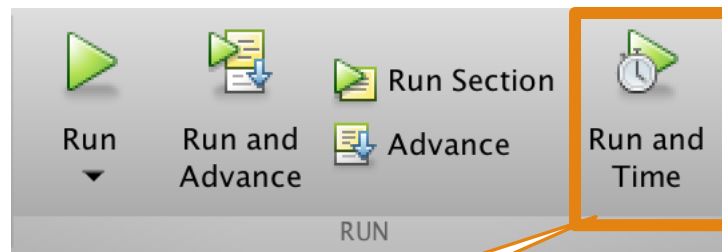
- È possibile eseguire la porzione di codice
 - Interamente
 - Sezione per sezione
 - Solo una determinata sezione



Sposta la selezione alla prossima sezione

Debugging – 4/7

- È possibile eseguire la porzione di codice
 - Interamente
 - Sezione per sezione
 - Solo una determinata sezione



Esegue tutto il codice e misura il relativo tempo di esecuzione

Debugging – 5/7

- La **modalità debug** è un altro meccanismo per supportare il debugging di codice MATLAB
 - Consiste nel collocare dei *breakpoints* (letteralmente *punti di rottura*) all'interno di tale codice
- Un *breakpoint* è un punto (riga di codice) in cui l'esecuzione del programma si interrompe temporaneamente (una sorta di pausa)
 - In tal modo è possibile esaminare i valori correnti (ed intermedi) delle variabili, semplificando l'individuazione di eventuali bug

Debugging – 6/7

L'editor di MATLAB fornisce un apposito menù per la gestione dei *breakpoints* all'interno di un dato programma

The screenshot shows the MATLAB editor interface. The top toolbar includes the 'Breakpoints' menu, which is highlighted with an orange box. The 'Breakpoints' menu is also highlighted with an orange box and contains the following options:

- Clear All**: Clear all breakpoints in all files
- Set/Clear**: Set or clear breakpoint on current line
- Enable/Disable**: Enable or disable breakpoint on current line
- Set Condition**: Set or modify conditional breakpoint

Below these options is a section titled 'ERROR HANDLING' with the following options:

- Stop on Errors**: dbstop if error
- Stop on Warnings**: dbstop if warning
- More Error and Warning Handling Options...**

The editor window shows a script named 'speed.m' with the following code:

```
1 %  
2 clear  
3 x = 1  
4 % Funz  
5 y1 = 2  
6 % Funz  
7 y2 = 4  
8
```

Debugging – 6/7

The screenshot shows the MATLAB Editor interface. The 'Breakpoints' menu is open, displaying several options. The 'Set/Clear' option is highlighted with an orange border. The menu items are:

- Clear All: Clear all breakpoints in all files
- Set/Clear**: Set or clear breakpoint on current line
- Enable/Disable: Enable or disable breakpoint on current line
- Set Condition: Set or modify conditional breakpoint
- ERROR HANDLING
- Stop on Errors: dbstop if error
- Stop on Warnings: dbstop if warning
- More Error and Warning Handling Options...

The background shows a script editor with the following code:

```
1 %  
2 clear  
3 x = 1  
4 % Funz  
5 y1 = 2  
6 % Funz  
7 y2 = 4  
8
```

- Inserisce o rimuove un *breakpoint* alla riga corrente
 - Ciascuna riga è selezionata mediante il cursore del mouse

Debugging – 6/7

The screenshot shows the MATLAB IDE interface. At the top, there is a toolbar with icons for 'Breakpoints', 'Run', 'Run and Advance', 'Run Section', and 'Run and Time'. Below the toolbar is the 'EDITOR' section, which contains a code editor window titled 'Editor - speed.m'. The code editor shows the following code:

```
1 %  
2 clear  
3 x = 1  
4 % Funz  
5 y1 = 2  
6 % Funz  
7 y2 = 4  
8
```

The 'Breakpoints' menu is open, showing the following options:

- Clear All: Clear all breakpoints in all files
- Set/Clear: Set or clear breakpoint on current line
- Enable/Disable: Enable or disable breakpoint on current line
- Set Condition: Set or modify conditional breakpoint

Below these options is the 'ERROR HANDLING' section, which includes:

- Stop on Errors: dbstop if error
- Stop on Warnings: dbstop if warning
- More Error and Warning Handling Options...

- Abilita (o disabilita) il *breakpoint* collocato alla riga corrente
 - Quando un *breakpoint* è disabilitato, esso è ignorato durante l'esecuzione del programma

Debugging – 6/7

The screenshot shows the MATLAB IDE interface. At the top, there is a toolbar with various icons for editing and debugging. Below the toolbar, the 'Breakpoints' menu is open, displaying several options: 'Clear All', 'Set/Clear', 'Enable/Disable', 'Set Condition', and 'ERROR HANDLING'. The 'Set Condition' option is highlighted with an orange box. In the background, a code editor window is visible, showing a script named 'speed.m' with the following code:

```
1 %  
2 clear  
3 x = 1  
4 % Funz  
5 y1 = 2  
6 % Funz  
7 y2 = 4  
8
```

- Un *breakpoint* può essere considerato (valutato) solo al verificarsi di una determinata condizione (preimpostata), in base ai valori correnti delle variabili
 - La condizione è valutata prima dell'esecuzione dell'istruzione definita alla riga dove è stato inserito il *breakpoint*

Debugging – 6/7

The screenshot shows the MATLAB IDE interface. The top toolbar includes the 'Breakpoints' button, which is currently selected. The 'Breakpoints' menu is open, displaying the following options:

- Clear All**: Clear all breakpoints in all files
- Set/Clear**: Set or clear breakpoint on current line
- Enable/Disable**: Enable or disable breakpoint on current line
- Set Condition**: Set or modify conditional breakpoint

Below these options, the 'ERROR HANDLING' section is highlighted with an orange box. It contains the following options:

- Stop on Errors**: dbstop if error
- Stop on Warnings**: dbstop if warning
- More Error and Warning Handling Options...**

The background shows the MATLAB Editor window with a script named 'speed.m' open. The script content is as follows:

```
1 %  
2 clear  
3 x = 1  
4 % Funz  
5 y1 = 2  
6 % Funz  
7 y2 = 4  
8
```

MATLAB fornisce diverse strategie per la gestione di eventuali errori di programmazione, che possono presentarsi durante l'esecuzione di un programma

Debugging – 6/7

The screenshot shows the MATLAB IDE interface. At the top, there is a toolbar with various icons for editing and debugging. Below the toolbar, the 'Breakpoints' menu is open, displaying several options. The 'Clear All' option is highlighted with an orange border. The menu items are:

- Clear All**: Clear all breakpoints in all files
- Set/Clear**: Set or clear breakpoint on current line
- Enable/Disable**: Enable or disable breakpoint on current line
- Set Condition**: Set or modify conditional breakpoint

Below these options, there is a section for 'ERROR HANDLING' with the following options:

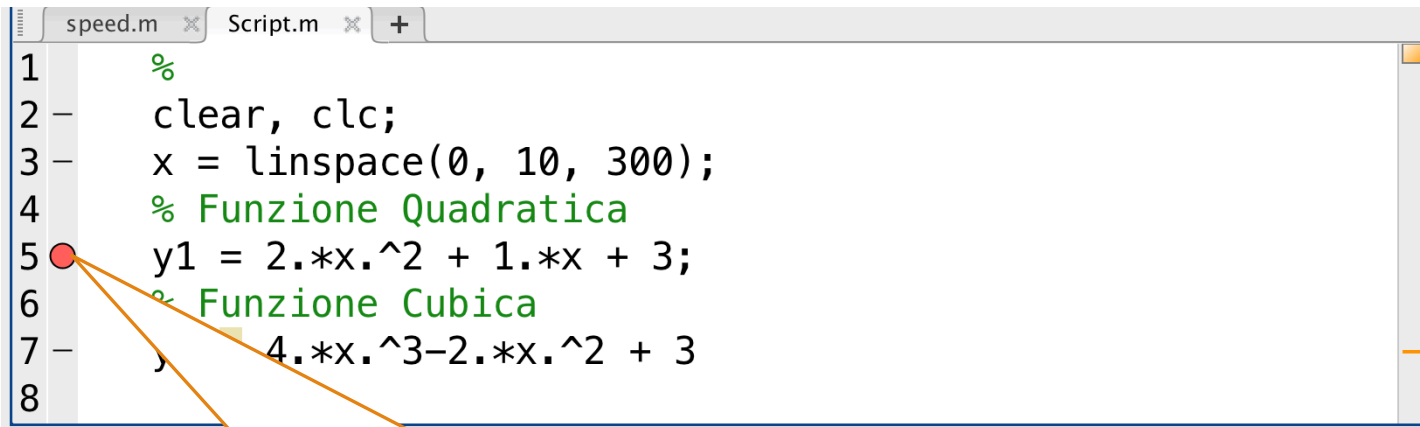
- Stop on Errors**: dbstop if error
- Stop on Warnings**: dbstop if warning
- More Error and Warning Handling Options...**

In the background, the MATLAB Editor window is visible, showing a script named 'speed.m' with the following code:

```
1 %  
2 clear  
3 x = 1;  
4 % Funz  
5 y1 = 2;  
6 % Funz  
7 y2 = 4;  
8
```

Cancella tutti i *breakpoints* nel programma

Debugging – 7/7



The image shows a MATLAB editor window with two tabs: 'speed.m' and 'Script.m'. The script content is as follows:

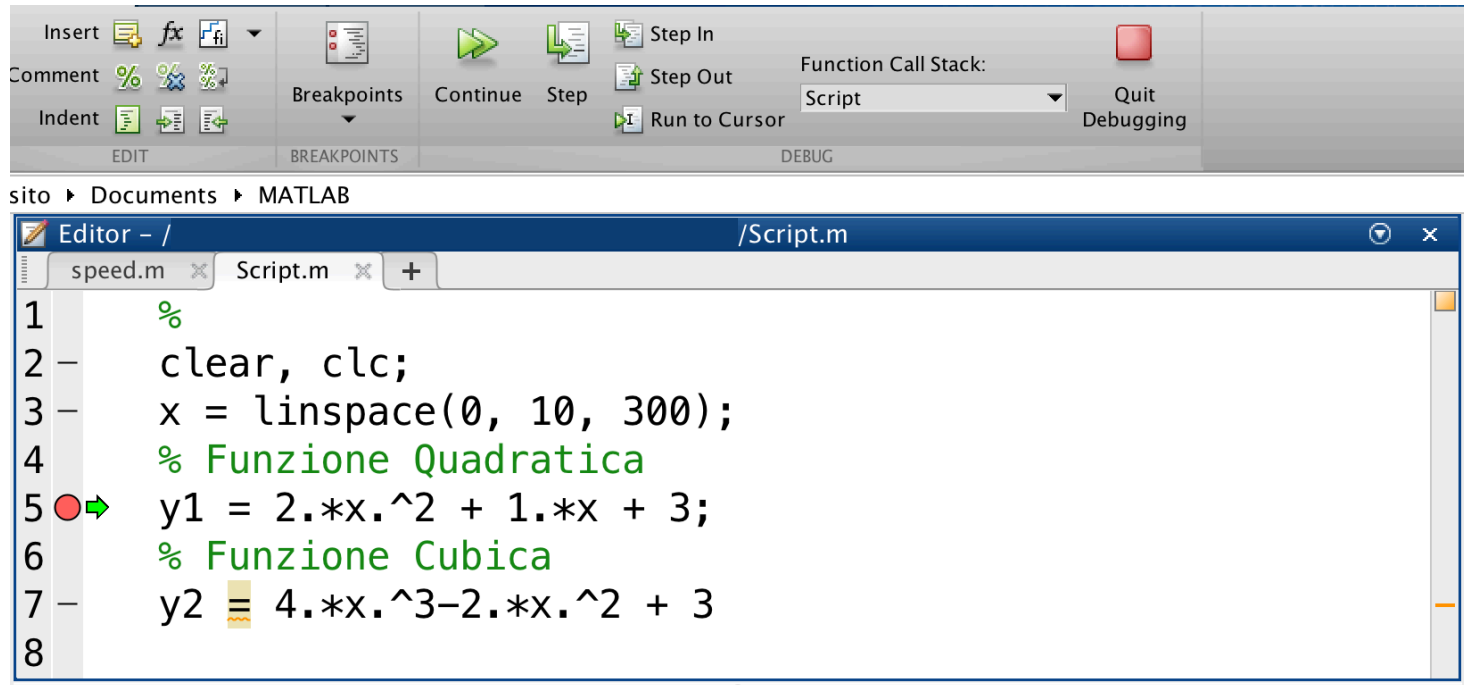
```
1 %  
2 clear, clc;  
3 x = linspace(0, 10, 300);  
4 % Funzione Quadratica  
5 y1 = 2.*x.^2 + 1.*x + 3;  
6 % Funzione Cubica  
7 y2 = 4.*x.^3 - 2.*x.^2 + 3;  
8
```

A red dot is placed on the left margin of line 5, indicating a breakpoint. An orange arrow points from this red dot to a text box below the editor.

Quando è inserito un *breakpoint* in una determinata riga, l'editor lo segnala con un **pallino rosso**

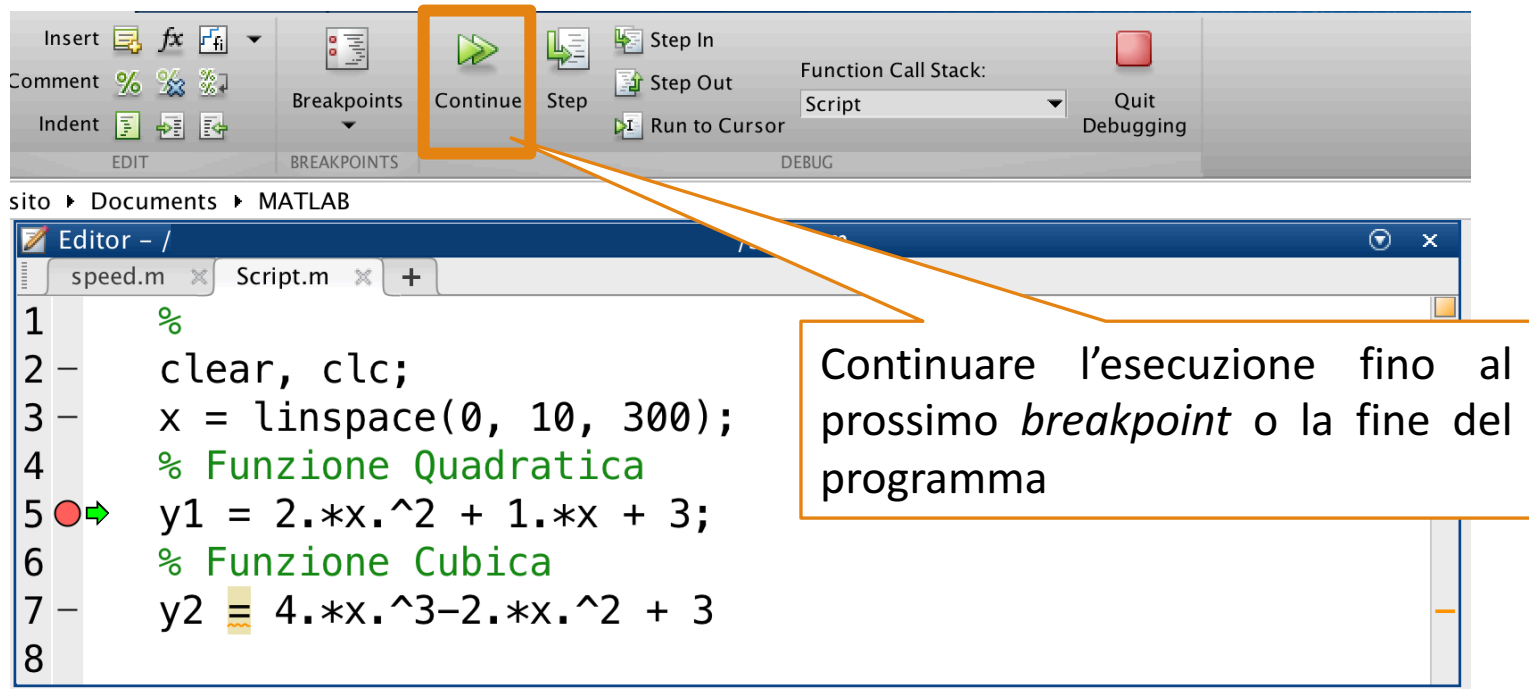
- Se scegliamo di eseguire la suddetta porzione di codice, noteremo che al punto del *breakpoint* si avrà un'interruzione temporanea del programma

Debugging – 7/7



- Quando l'esecuzione del programma si interrompe su un *breakpoint*
- MATLAB fornisce al programmatore diverse azioni da poter svolgere

Debugging – 7/7

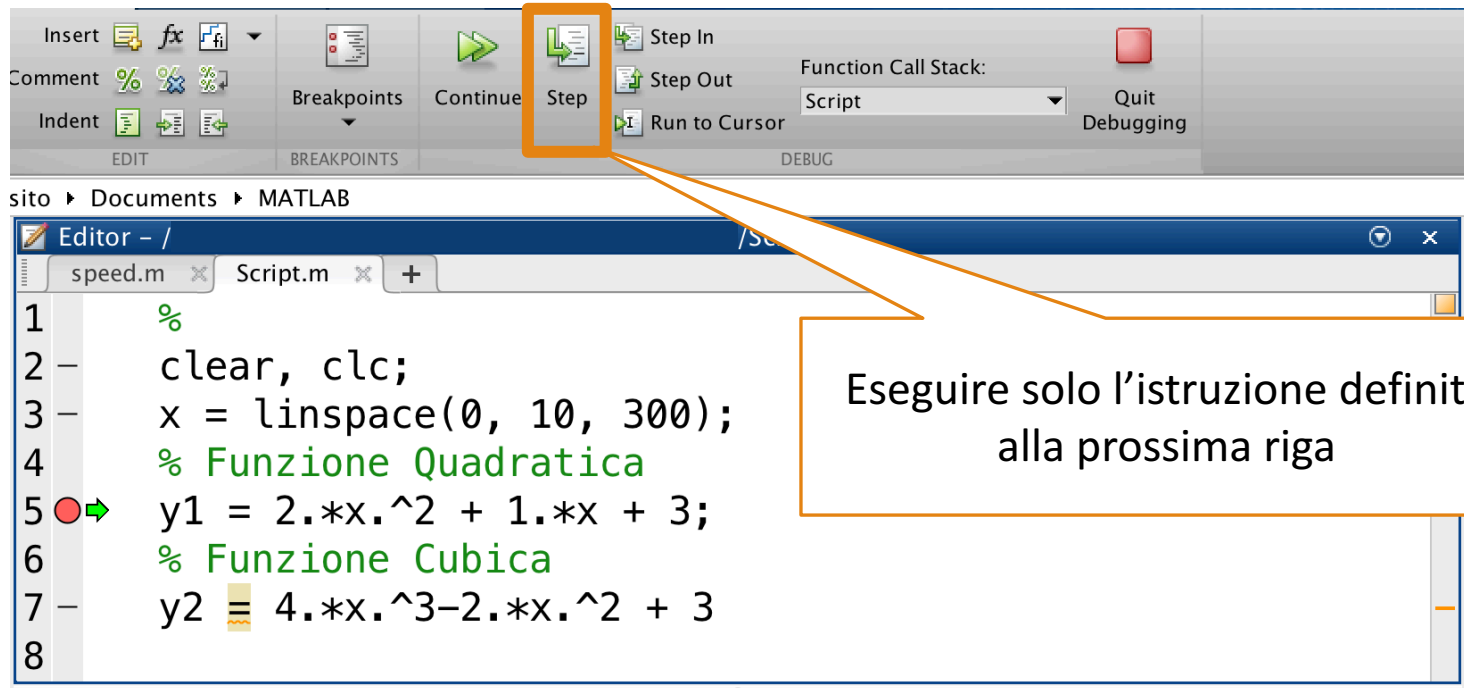


The screenshot shows the MATLAB debugging toolbar with the 'Continue' button (a green play icon) highlighted with an orange box. A callout box points to this button with the text: "Continuare l'esecuzione fino al prossimo *breakpoint* o la fine del programma". Below the toolbar, the MATLAB Editor window shows a script named 'Script.m' with the following code:

```
1 %  
2 - clear, clc;  
3 - x = linspace(0, 10, 300);  
4 % Funzione Quadratica  
5 ● → y1 = 2.*x.^2 + 1.*x + 3;  
6 % Funzione Cubica  
7 - y2 = 4.*x.^3 - 2.*x.^2 + 3  
8
```

- Quando l'esecuzione del programma si interrompe su un breakpoint
- MATLAB fornisce al programmatore diverse azioni da poter svolgere

Debugging – 7/7



The screenshot shows the MATLAB IDE interface. The top toolbar contains several debugging buttons: 'Continue' (green play icon), 'Step' (green play icon with a red dot), 'Step In' (green play icon with a red dot and a downward arrow), 'Step Out' (green play icon with a red dot and an upward arrow), 'Run to Cursor' (green play icon with a red dot and a cursor), and 'Quit Debugging' (red stop icon). The 'Step' button is highlighted with an orange box. A callout box with an orange border points to the 'Step' button and contains the text: "Eseguire solo l'istruzione definita alla prossima riga". Below the toolbar, the MATLAB Editor window is open, showing a script named 'Script.m'. The script contains the following code:

```
1 %  
2 clear, clc;  
3 x = linspace(0, 10, 300);  
4 % Funzione Quadratica  
5 y1 = 2.*x.^2 + 1.*x + 3;  
6 % Funzione Cubica  
7 y2 = 4.*x.^3 - 2.*x.^2 + 3;  
8
```

The cursor is positioned at the beginning of line 5, and a red dot with a green arrow points to the start of the line, indicating that the 'Step' operation is about to be performed.

- Quando l'esecuzione del programma si interrompe su un *breakpoint*
- MATLAB fornisce al programmatore diverse azioni da poter svolgere

Debugging – 7/7

The screenshot shows the MATLAB IDE interface. The top toolbar contains several debugging icons, with the 'Step In' icon (a green arrow pointing into a document) highlighted by an orange box. Below the toolbar, the 'Function Call Stack' panel is visible. The main editor window shows a script named 'Script.m' with the following code:

```
1 %  
2 clear, clc;  
3 x = linspace(0, 10,  
4 % Funzione Quadrati  
5 y1 = 2.*x.^2 + 1.*x  
6 % Funzione Cubica  
7 y2 = 4.*x.^3 - 2.*x.^2 + 3  
8
```

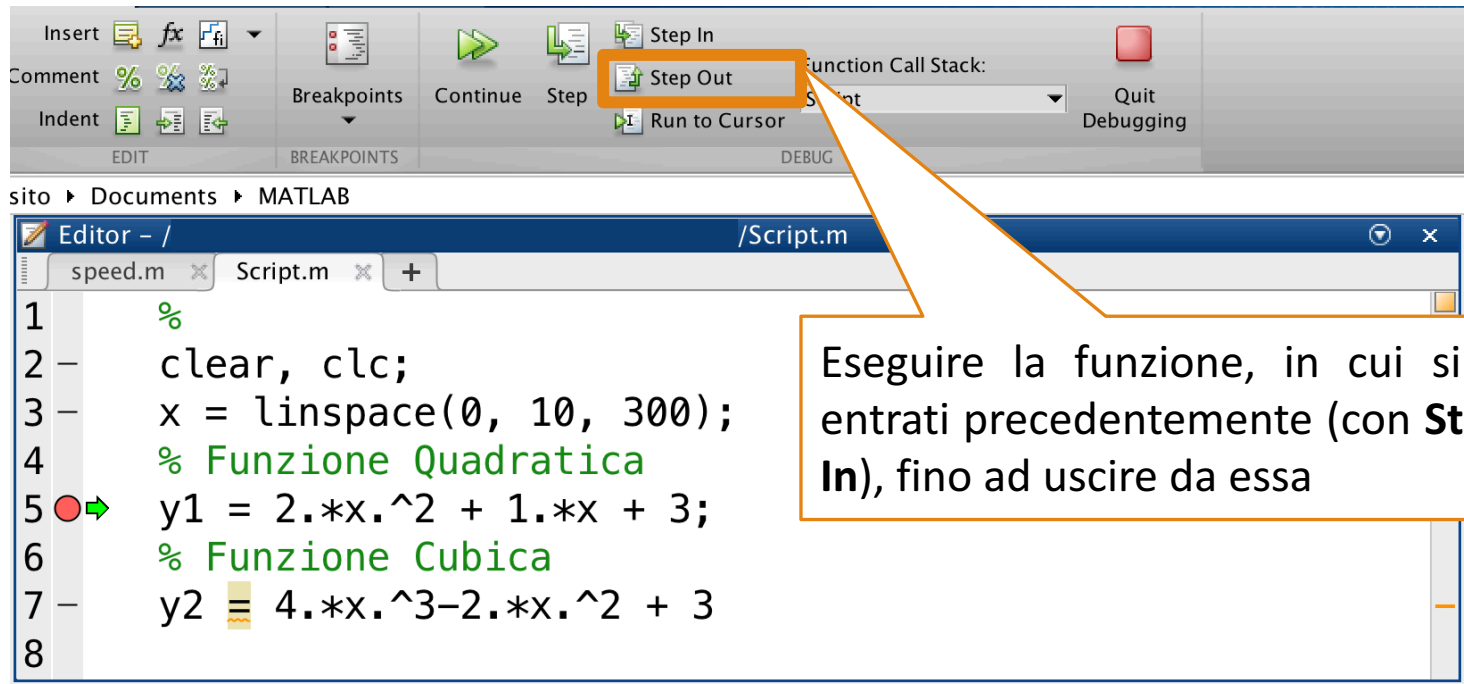
A red circle and a green arrow point to line 5. An orange callout box points to the 'Step In' icon and contains the following text:

Eseguire l'istruzione alla riga successiva

- Permette di entrare all'interno del codice sorgente dell'eventuale funzione invocata

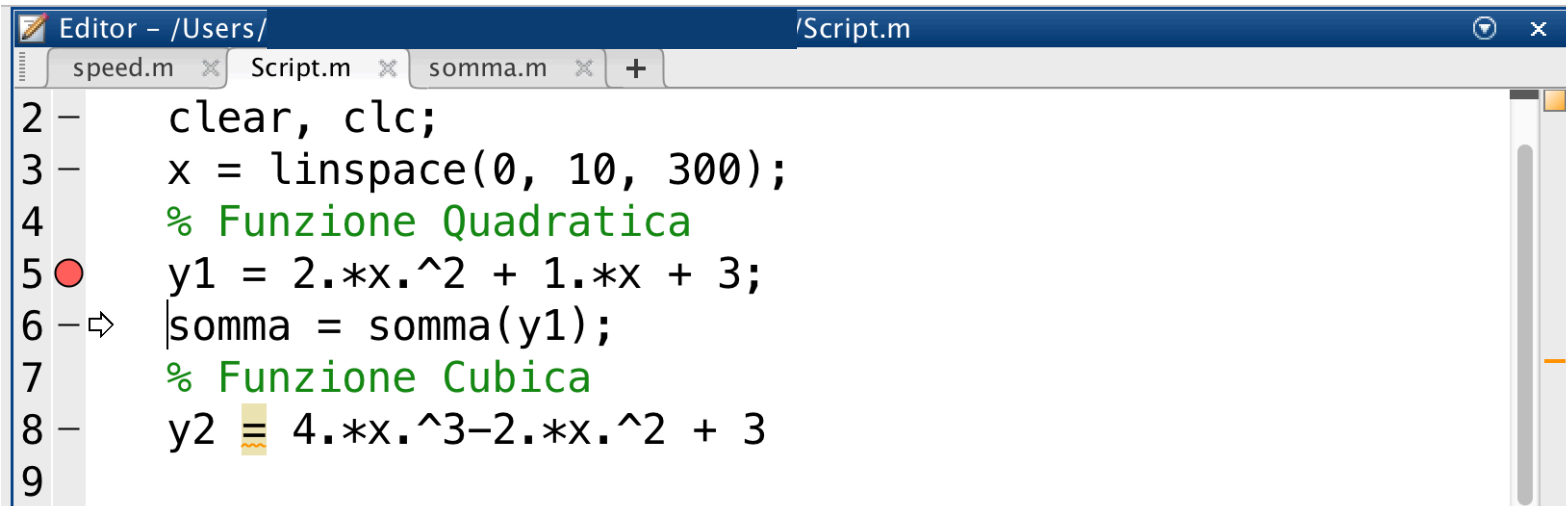
- Quando l'esecuzione del programma si interrompe su un *breakpoint*
- MATLAB fornisce al programmatore diverse azioni da poter svolgere

Debugging – 7/7



- Quando l'esecuzione del programma si interrompe su un *breakpoint*
- MATLAB fornisce al programmatore diverse azioni da poter svolgere

Debugging – 7/7



The image shows a MATLAB Editor window titled "Editor - /Users/ Script.m". The window contains a script with the following code:

```
2 - clear, clc;
3 - x = linspace(0, 10, 300);
4 - % Funzione Quadratica
5 ● y1 = 2.*x.^2 + 1.*x + 3;
6 -> somma = somma(y1);
7 - % Funzione Cubica
8 - y2 = 4.*x.^3 - 2.*x.^2 + 3
9
```

A red circle breakpoint is set on line 5. A mouse cursor is pointing to line 6. The script includes comments in Italian: "% Funzione Quadratica" and "% Funzione Cubica".

Debugging – 7/7

The image shows two MATLAB Editor windows. The top window, titled 'Editor - /Users/ /Script.m', contains the following code:

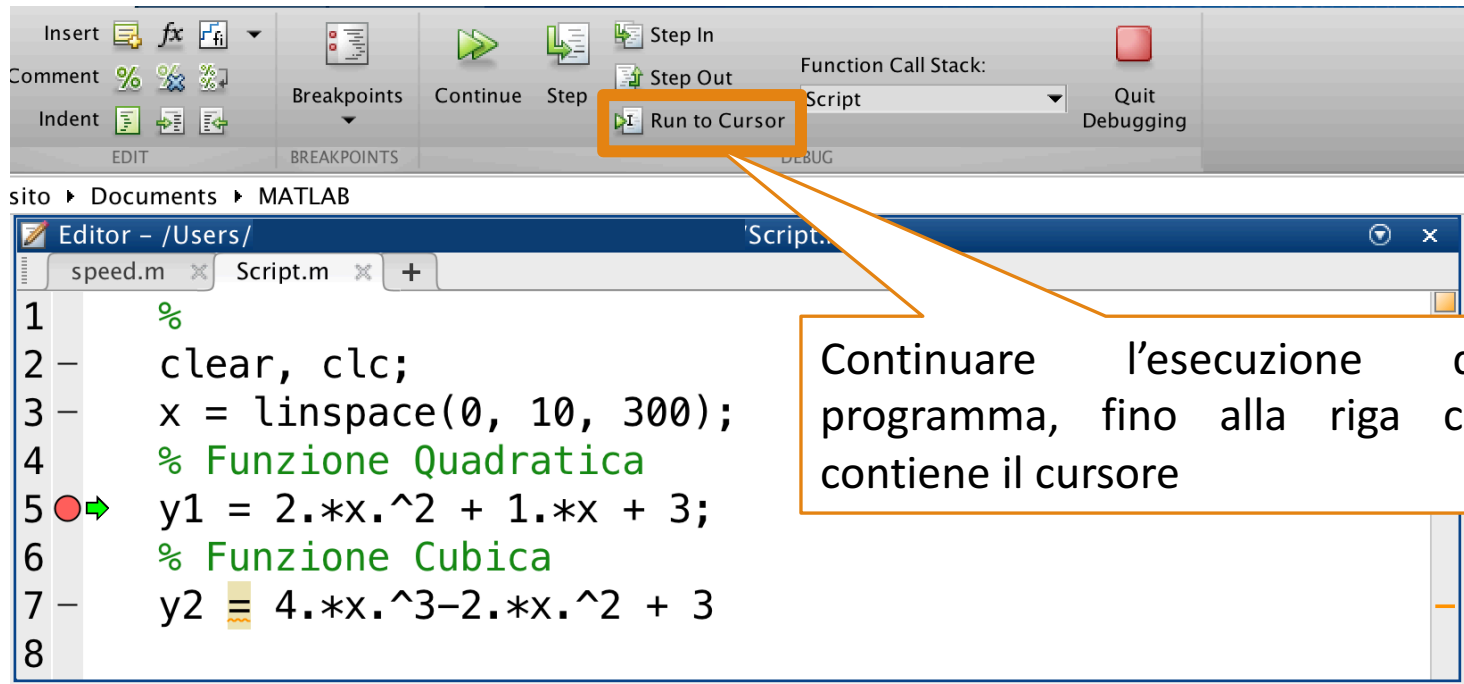
```
2 - clear, clc;  
3 - x = linspace(0, 10, 300);  
4 - % Funzione Quadratica  
5 - y1 = 2.*x.^2 + 1.*x + 3;  
6 - somma = somma(y1);  
7 - % Funzione Cubica
```

A red circle is next to line 5, and a green arrow points to line 6. A 'Step' button is located between the two windows. The bottom window, titled 'Editor - /Users/ /somma.m', contains the following function definition:

```
1 - function somma = somma( x )  
2 -     l = length(x);  
3 -     somma = 0;  
4 -     for i = 1:l  
5 -         somma = somma+x(i);  
6 -     end  
7 - end  
8 -
```

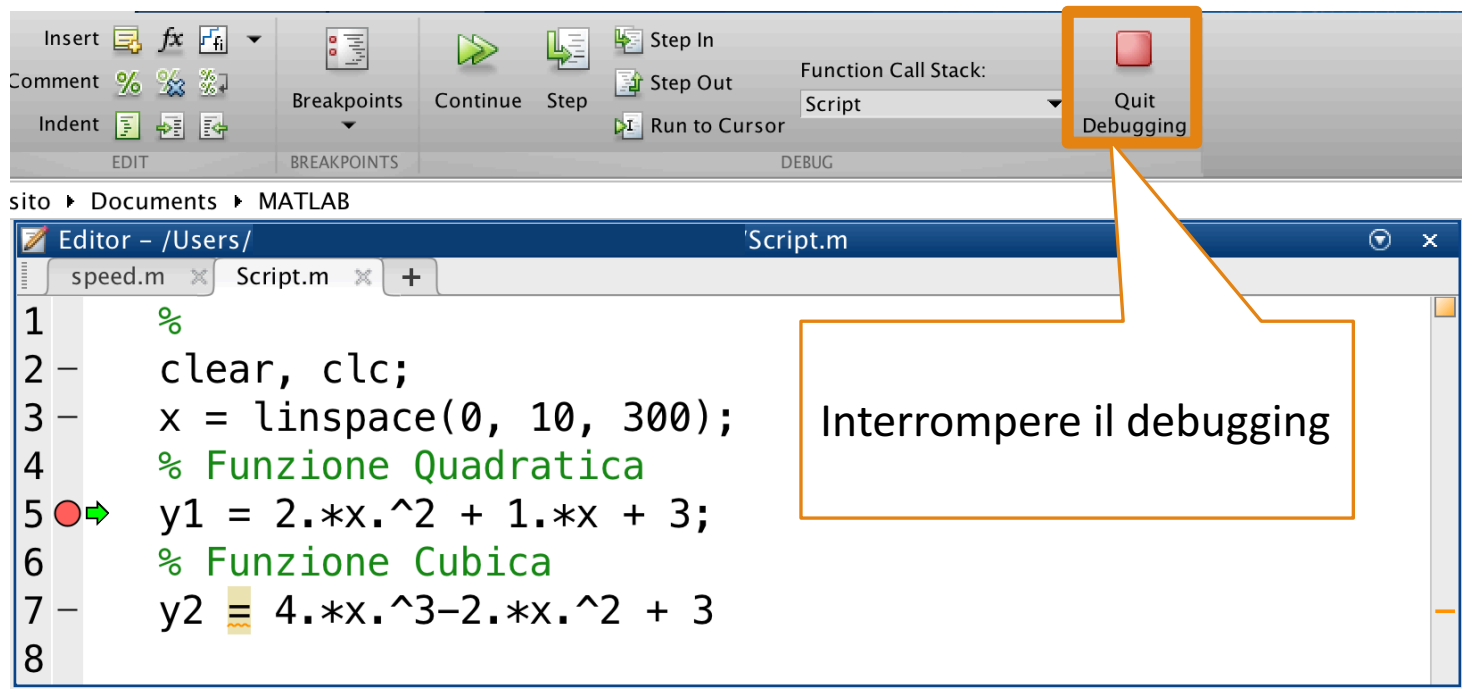
A green arrow points to line 3. Two orange arrows indicate the flow of execution: one from the 'Step' button to line 6 in the top window, and another from line 3 in the bottom window to the 'Step' button.

Debugging – 7/7



- Quando l'esecuzione del programma si interrompe su un *breakpoint*
- MATLAB fornisce al programmatore diverse azioni da poter svolgere

Debugging – 7/7



- Quando l'esecuzione del programma si interrompe su un *breakpoint*
- MATLAB fornisce al programmatore diverse azioni da poter svolgere

Riferimenti

- Capitolo 4
 - Paragrafo 8 [**Debugging dei programmi di MATLAB**]