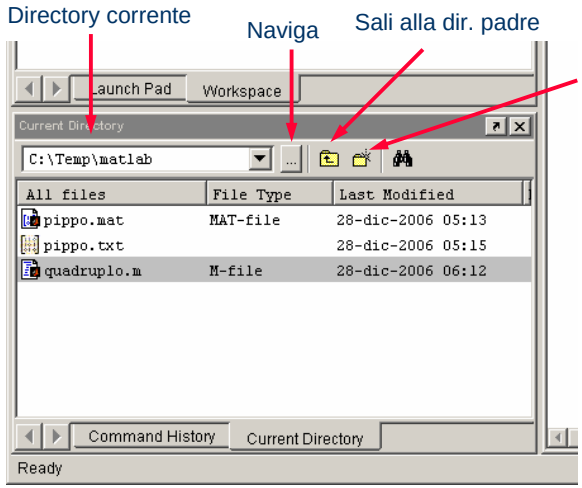
Script – Semplici programmi

- Fino ad ora, i comandi sono stati sempre digitati nella command window:
  - Eseguiti premendo “enter”
  - Modificati utilizzando le frecce o la command history

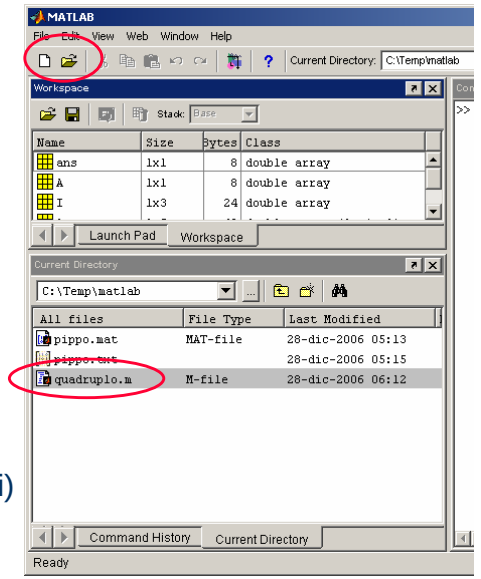
Cosa sono gli Script (m-file)

- Un file contenente comandi MATLAB
  - Può essere ri-eseguito
  - Può essere facilmete modificato
  - Può essere spedito a qualcuno
- I comandi sono eseguiti sequenziualmente
  - Il file viene eseguito digitando il suo nome (senza .m)
  - I risultati appaiono nella command window (se non mettiamo il ;)
- Può essere creato utilizzando un qualsiasi editor di testo
  - Estensione .m
  - Presente nella current directory

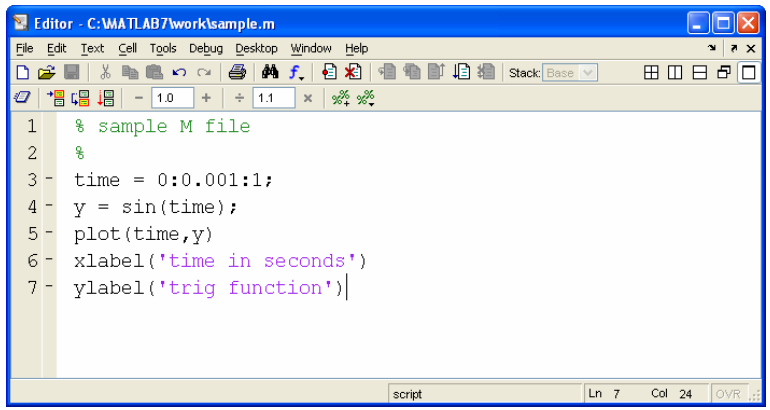


N.B. Possiamo scrivere solo in C:\Temp e sue sottodirectory!!!

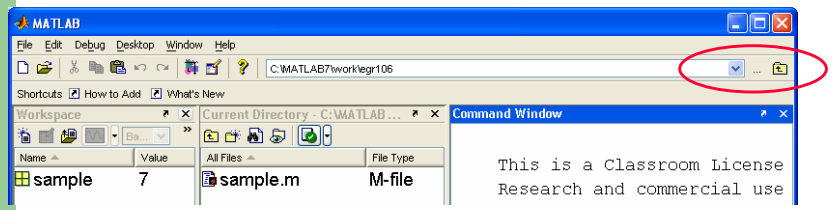
- Matlab include un Editor a Colori :
  - Si può creare un nuovo file o modificarne uno aprendolo (apri o doppio click sul nome del file)
  - Colori usati per aiutare nella creazione del file (comandi, tipi, errori)



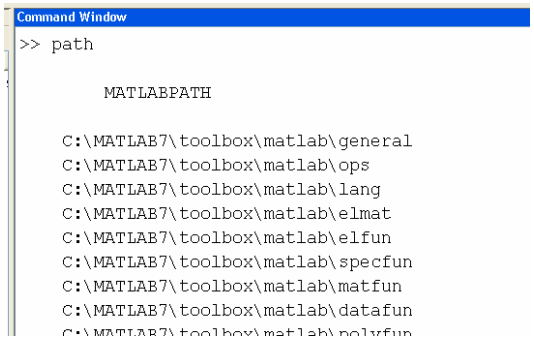
- Solito menu Windows
- Numeri di linea
- "run" oppure F5
- Possibilità di debug
- Linea di commenti
- Notare l'uso del ;
- Uso dei colori
- Commenti con %



- Quando scriviamo un nome nella command window, MATLAB cerca:
  1. Una variabile in workspace
  2. Uno script nella directory corrente
     - Possiamo spostarci utilizzando la barra indirizzo
     - o la sottofinestra "current directory"
  3. Una funzione predefinita



4. Nel percorso di ricerca "search path"



- Morale – usate nomi unici!!!

- Dove prendono i dati?
  - Da vettori definiti nello script
  - Usando il comando "input":
    - Numerico:

      x = input(' how many? ')
    - Stringa:

      x = input(' name? ', 's')

- Come metto fuori i risultati?
  - Con il nome della varibile da visualizzare
  - Usando il comando “disp”:
    - Un vettore esistente (solo UN vettore, ma possiamo usare [ ] !!)

      disp(x) o disp([x y])
    - Messaggio di testo

      disp(' The task is done ')

- Esempio:
- Notare che disp accorcia il risultato (non mette il nome della variabile e le righe bianche!)

```
>> x = rand(3,2);
>> x

x =

    0.9501    0.4860
    0.2311    0.8913
    0.6068    0.7621

>> disp(x)
    0.9501    0.4860
    0.2311    0.8913
    0.6068    0.7621
```


Output formattato

- Con il comando disp visualizziamo solo la matrice o il valore
- Se vogliamo un output più elegante usiamo il comanfo fprintf, esempio:

```
>> nome='bob';eta=23;
>> fprintf(1,'%s ha %g anni\n',nome,eta)
bob ha 23 anni
```


Argomenti di fprintf

`fprintf(1,'%s ha %g anni\n',nome,eta)`

- Il primo argomento è la destinazione dell’ouput (1 vuol dire a schermo)
- Il secondo è la stringa di formattazione, compresa tra apici. Qua metteremo “**%s**” dove inseriamo una variabile stringa, “**%g**” dove inseriamo un numero.
- Poi vanno messi i valori da inserire nella stringa (se ci sono). In ordine!!!
- Nella stringa di formattazione possiamo usare qualche carattere speciale:

\n = a capo \t = tabulazione %% = carattere %

fprintf su file

- Dobbiamo aprire il file prima di poterci scrivere (**`fid=fopen('out.txt','w')`**) se il file esiste lo svuota, se non esiste lo crea!
- Poi scriviamo (con **`fid`** come primo argomento):
  `fprintf(fid,'%s ha %g anni\n',nome,eta)`
- Alla fine chiudiamo:
  `fclose(fid)`

Attenzione ai vettori di celle

```
>> Nomi ={ 'Tim' , 'Tom', 'Jim'};
>> fprintf(1,'secondo nome: %s',Nomi(2))
??? Error using ==> fprintf
Function 'fprintf' is not defined for values of class 'cell'.

>> fprintf(1,'secondo nome: %s',char(Nomi(2)))
secondo nome: Tom
>>
```

Per stampare correttamente una stringa da un vettore di celle, dobbiamo usare la funzione char

COMMENTI

- Il simbolo di commento può essere messo in qualsiasi punto della linea. MATLAB ignorerà tutto quello che viene scritto alla destra del simbolo %. Per esempio:

```
>>% This is a comment.
>>x = 2+3 % So is this.
x =
   5
```

Notate che la parte a sinistra di % viene eseguita, per calcolare x.

Quando si usano i file di *script*, tenere presente:

1. Il nome del file deve cominciare con una lettera e può contenere cifre e il carattere *underscore*, fino a 31 caratteri.

2. Non dare lo stesso nome al file di *script* e a una variabile.

3. Non chiamare uno *script* con lo stesso nome di un comando o funzione MATLAB. Per verificare se esiste già qualcosa con un nome utilizzare la funzione `exist`.

Debugging (ricerca errori) degli Script

Gli errori di programmazione di solito rientrano in una di queste categorie:

1. **Errori di sintassi**, come omettere una parentesi, una virgola, o digitare un comando in modo errato. MATLAB di solito rileva gli errori più comuni, e visualizza un messaggio che descrive l'errore e la sua posizione.
2. Errori dovuti ad una procedura matematica sbagliata, chiamati anche **errori di *runtime***. Spesso dipende dai valori di input, un tipico esempio è la divisione per zero.

Per trovare gli errori, provate questa procedura:

1. Eseguite il programma con una semplice versione del problema, che possa anche essere verificata "a mano".
2. Visualizzare tutti i risultati intermedi del proccedimento, togliendo i "punto e virgola" alla fine delle istruzioni.
3. Utilizzare gli strumenti forniti dall' Editor/Debugger.

Per una programmazione ordinata:

1. *Sezione dei commenti:*

a. Il nome del programma e le parole chiave, nella prima riga.

b. La data di creazione e i nomi degli autori nella seconda riga.

c. La definizione dei nomi delle variabili per ogni variabile di input e di output. Specificare anche le variabili uste nei calcoli e *le unità di misura per tutte le variabili di input e di output!*

d. Il nome di ogni funzione creata dall'utente che viene usata nel programma.

(continua …)

Per una programmazione ordinata (segue)

2. *Sezione di Input* : Inserire i dati in input e/o le funzioni di input e commenti per chiarezza.

3. *Sezione di calcolo*

4. *Sezione di output* Questa sezione contieme le funzioni per visualizzare i risultati del programma..

Esempio di un file di Script

Problema:

La velocità di un oggetto lasciato cadere da fermo è data come funzione del tempo t da $v = gt$.

Disegnare v come funzione di t per $0 \le t \le t_f$ dove t_f è l'istante finale, inserito dall'utente.

(continua …)

Esempio di un file di script (segue)

```
% Program falling_speed.m:
% Plots speed of a falling object.
% Created on March 1, 2004 by W. Palm
%
% Input Variable:
% tf = final time (in seconds)
%
% Output Variables:
% t = array of times at which speed is
% computed (in seconds)
% v = array of speeds (meters/second)
%
```

(continua …)

Esempio di un file di script (segue)

```
% Parameter Value:
g = 9.81; % Acceleration in SI units
%
% Input section:
tf = input('Enter final time in seconds:');
```

(continua …)

Esempio di un file di script (segue)

```
% Calculation section:
dt = tf/500;
% Create an array of 501 time values.
t = [0:dt:tf];
% Compute speed values.
v = g*t;
%
% Output section:
plot(t,v),xlabel('t(s)'),ylabel('v m/s)')
```

La sezione commenti iniziale…

….È FONDAMENTALE!!! Perché…

- Il comando lookfor cerca nella descrizione iniziale (prima riga di commento)!
- Il comando help visualizza tutta la sezione dei commenti all'inizio dello script!

Altri esempi di Scripts

Editor - C:\MATLAB7\work\egr106\example1.m*

```
1  % script example #1
2  % computes areas of triangles
3  %
4  bases = [ 1.2, 3.3, 6.7 ];
5  heights = [ 3.6, 2.1, 4.5 ];
6  areas = 0.5 * bases * heights
```


Editor - C:\MATLAB7\work\egr106\example2.m

```
1  % script example #2
2  % computes areas of triangles
3  % ask for parameters
4  %
5  base = input('how long is the base? ');
6  height = input('how high is the triangle? ');
7  area = 0.5 * base * height;
8  disp('The area is ')
9  disp(area)
```

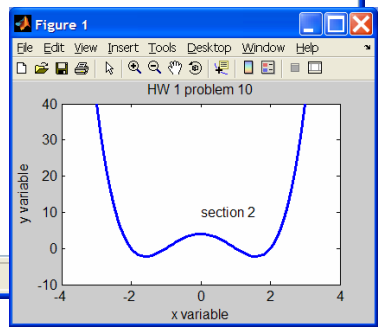
Command Window

```
>> example2
how long is the base? 3
how high is the triangle? 4
The area is
     6

>>
```

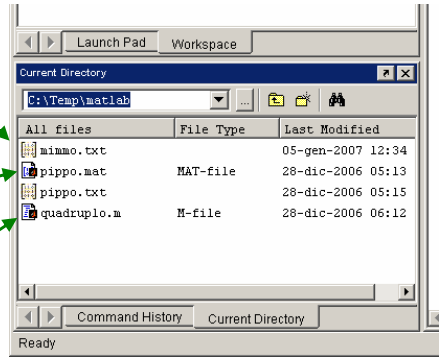

Editor - C:\Program Files\MATLAB71\work\homework1.m

```
1  % HW 1 problem 10
2  xv = -3:0.1:3;
3  yv = xv.^4 - 5*xv.^2 + 4;
4  plot(xv,yv)
5  xlabel('x variable')
6  ylabel('y variable')
7  title('HW 1 problem 10')
8  text(0,10,'section 2')
```



I file usati da Matlab

- Tipi:
  - ascii = file di testo
  - .mat = file proprietari di Matlab (varibili multiple)
  - .m = script

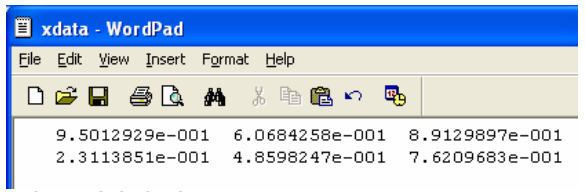


- Salvare i dati:
  - save filename
  - save filename array1 array2
  - save filename –ascii x

```
x =

    0.9501    0.6068    0.8913
    0.2311    0.4860    0.7621

>> save xdata -ascii
```



- Leggere (caricare) i dati:
  - load filename
  - load filename array1 array2
  - load filename –ascii

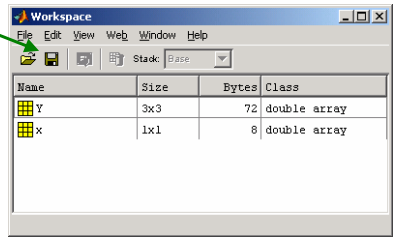
Attenzione! Sovrascrive senza avviso!!!

• Load e Save (senza l'opzione ascii) salvano tutto il workspace (o solo una sua parte)

• Viene prodotto un file .mat che contiene tutte le variabili che abbiamo deciso di salvare

• Possiamo salvare il workspace anche con l'apposita icona nella finestra workspace

• Per aprire un workspace salvato (pippo.mat) possiamo fare direttamente doppio click dalla current directory (o file > open)



Importare file da fogli di calcolo

ALcuni fogli di calcolo salvano i dati nel formato `.wk1`. In MATLAB si può usare il comando `M = wk1read('filename')` per importare questi dati nella matrice `M`.

Il comando `A = xlsread('filename')` importa i file di Microsoft Excel `filename.xls` nella matrice `A`.
Il comando `[A, B] = xlsread('filename')` importa i dati numerici nella matrice `A` e tutti i dati di testo nella matrice di celle `B`.

Import Wizard

Per importare dati da file di testo (ASCII), bisogna conoscere in che modo sono formattati i dati.

Per esempio, molti file di testo utilizzano in formato fisso per righe e colonne.

(continua …)

Import Wizard (segue)

Per i file di testo bisogna conoscere:

- Quanti dati ci sono in ogni riga?
- Questi dati sono numerici, stringhe o entrambi?
- Le righe o le colonne hanno un'intestazione?
- Qual è il carattere utilizzato come *delimiter*, cioè il carattere utilizzato per separare i vari dati all' interno di una riga? (Chiamato anche *separatore di colonna*)

(continua …)

Import Wizard (segue)

Si possono importare molti formati di dati da file di testo, o dagli appunti. Quando si utilizza l'import wizard per creare una variabile nel workspace di MATLAB, questo sovrascrive un'eventuale variable con lo stesso nome *senza avvisi!!!!*

L'Import Wizard presenta una serie di finestre di dialogo nelle quali vengono richiesti:

1. Il nome del file da importare,
2. Il *delimiter* utilizzato
3. Le variabili da importare.