ESERCITAZIONI
MATLAB – Come iniziare

Indicizzazione di array

Ogni variabile di MATLAB® è un array che può contenere molti numeri. Per accedere a elementi selezionati di un array, utilizzare l'indicizzazione.

Ad esempio, si consideri la matrice 4x4 A:

```
A = [1 2 3 4; 5 6 7 8; 9 10 11 12; 13 14 15 16]
```

```
A = 4×4

     1     2     3     4
     5     6     7     8
     9    10    11    12
    13    14    15    16
```

Vi sono due modi per fare riferimento a un particolare elemento di un array. Il metodo più diffuso consiste nello specificare pedici di righe e colonne, come

```
A(4,2)
```

```
ans = 14
```

Meno diffuso, ma a volte utile, è utilizzare un singolo pedice che percorre verso il basso tutte le colonne nell'ordine:

```
A(8)
```

```
ans = 14
```

L'utilizzo di un singolo pedice per fare riferimento a un particolare elemento in un array prende il nome di *indicizzazione lineare*.

ESERCITAZIONI
MATLAB – Come iniziare

Indicizzazione di array

Se si prova a fare riferimento a elementi esterni a un array, sul lato destro di una dichiarazione di assegnazione, MATLAB genera un errore.

```
test = A(4,5)
```

```
Index in position 2 exceeds array bounds (must not exceed 4).
```

Sul lato sinistro di una dichiarazione di assegnazione, invece, è possibile specificare elementi esterni alle dimensioni correnti. Le dimensioni vengono aumentate per adattarle alla nuova immissione.

```
A(4,5) = 17
```

```
A = 4×5

     1     2     3     4     0
     5     6     7     8     0
     9    10    11    12     0
    13    14    15    16    17
```

Per fare riferimento a più elementi di un array utilizzare l'operatore virgola, che consente di specificare un range di forma `start:end`. Ad esempio, elencare gli elementi nelle prime tre righe e nella seconda colonna di `A`:

```
A(1:3,2)
```

```
ans = 3×1

     2
     6
    10
```

ESERCITAZIONI
MATLAB – Come iniziare

Indicizzazione di array

La sola virgola, senza valori iniziali e finali, specifica tutti gli elementi in quella determinata dimensione. Ad esempio, selezionare tutte le colonne nella terza riga di `A`:

```
A(3,:)
```

```
ans = 1×5

     9    10    11    12     0
```

L'operatore virgola consente anche di creare un vettore di valori di spaziatura uniforme utilizzando la forma più generale `start:step:end`.

```
B = 0:10:100
```

```
B = 1×11

     0    10    20    30    40    50    60    70    80    90   100
```

Se si omette il passaggio intermedio, come in `start:end`, MATLAB utilizza il valore incrementale predefinito `1`.

ESERCITAZIONI
MATLAB – Come iniziare

Variabili del workspace

Il *workspace* contiene variabili create dall'utente o importate in MATLAB® da file di dati o da altri programmi. Ad esempio, le seguenti dichiarazioni creano le variabili `A` e `B` nel workspace.

```
A = magic(4);
B = rand(3,5,2);
```

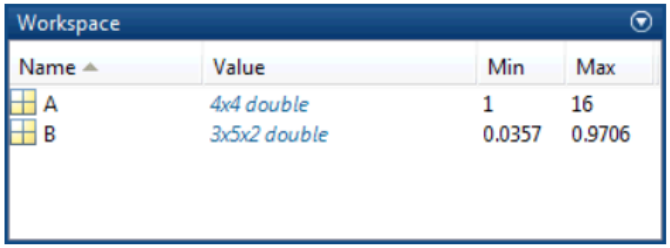
È possibile visualizzare il contenuto del workspace con `whos`.

```
whos
```

```
  Name      Size             Bytes  Class     Attributes

  A         4x4                128  double
  B         3x5x2              240  double
```

Le variabili appaiono anche nel pannello del workspace sul desktop.



ESERCITAZIONI
MATLAB – Come iniziare

Variabili del workspace

Le variabili del workspace non vengono conservate quando si esce da MATLAB. Salvare i dati per un uso successivo con il comando `save`,

```
save myfile.mat
```

Il salvataggio consente di conservare il workspace nella cartella di lavoro corrente, in un file compresso denominato file MAT, con estensione `.mat`.

Per eliminare tutte le variabili dal workspace utilizzare il comando `clear`.

Per ripristinare i dati da un file MAT nel workspace utilizzare il comando `load`.

```
load myfile.mat
```

ESERCITAZIONI
MATLAB – Come iniziare

Testo e caratteri

Testo in array di stringhe

Quando si lavora con il testo, racchiudere le sequenze di caratteri tra virgolette doppie. È possibile assegnare del testo a una variabile.

```
t = "Hello, world";
```

Se il testo include delle virgolette doppie, usare due virgolette doppie all'interno della definizione.

```
q = "Something ""quoted"" and something else."
```

```
q =

    "Something "quoted" and something else."
```

`t` e `q` sono array, come tutte le variabili di MATLAB®. La loro *classe* o tipo di dati è `string`.

```
whos t
```

```
  Name        Size            Bytes  Class     Attributes
  t           1x1               174  string
```

ESERCITAZIONI
MATLAB – Come iniziare

Testo e caratteri

Testo in array di stringhe

Per aggiungere testo alla fine di una stringa, utilizzare l'operatore +.

```
f = 71;
c = (f-32)/1.8;
tempText = "Temperature is " + c + "C"
```

```
tempText =
"Temperature is 21.6667C"
```

Come accade con gli array numerici, gli array di stringhe possono presentare elementi multipli. Utilizzare la funzione `strlength` per trovare la lunghezza di ciascuna stringa all'interno di un array.

```
A = ["a","bb","ccc"; "dddd","eeeeee","fffffff"]
```

```
A =
  2×3 string array
    "a"       "bb"        "ccc"
    "dddd"    "eeeeee"    "fffffff"
```

```
strlength(A)
```

```
ans =

     1     2     3
     4     6     7
```

ESERCITAZIONI
MATLAB – Come iniziare

Testo e caratteri

Dati negli array di caratteri

A volte i caratteri rappresentano dati che non corrispondono a testo, come una sequenza di DNA. È possibile memorizzare questo tipo di dati in un array di caratteri, che presenta il tipo di dati `char`. Per gli array di caratteri si utilizzano gli apici.

```
seq = 'GCTAGAATCC';
whos seq
```

```
  Name      Size            Bytes  Class    Attributes
  seq       1x10               20  char
```

In questo caso, ciascun elemento dell'array contiene un carattere singolo.

```
seq(4)
```

```
ans =
    'A'
```

Concatenare gli array di caratteri con parentesi quadre, come si farebbe con gli array numerici.

```
seq2 = [seq 'ATTAGAAACC']
```

```
seq2 =
    'GCTAGAATCCATTAGAAACC'
```

Gli array di caratteri sono diffusi nei programmi scritti prima dell'introduzione delle virgolette doppie per la creazione di stringhe nella R2017a. Tutte le funzioni di MATLAB che accettano i dati `string` accettano anche i dati `char` e viceversa.

ESERCITAZIONI
MATLAB – Come iniziare

Richiamo delle funzioni

MATLAB® offre un ampio numero di funzioni per l'esecuzione dei compiti di calcolo. Le funzioni sono equivalenti alle *subroutine* o ai *metodi* in altri linguaggi di programmazione.

Per richiamare una funzione, come `max`, racchiuderne gli argomenti di input tra parentesi:

```
A = [1 3 5];
max(A)
```

```
ans = 5
```

Se vi sono più argomenti di input, separarli con delle virgole:

```
B = [3 6 9];
union(A,B)
```

```
ans = 1×5

     1     3     5     6     9
```

ESERCITAZIONI
MATLAB – Come iniziare

Richiamo delle funzioni

Per restituire l'output da una funzione assegnarla a una variabile:

```
maxA = max(A)
```

```
maxA = 5
```

Quando vi sono più argomenti di output, racchiuderli in parentesi quadre:

```
[minA,maxA] = bounds(A)
```

```
minA = 1

maxA = 5
```

Racchiudere qualsiasi input di testo tra virgolette:

```
disp("hello world")
```

```
hello world
```

Per richiamare una funzione che non richiede input e non restituisce output, digitarne solo il nome:

```
clc
```

La funzione `clc` svuota la finestra di comando.

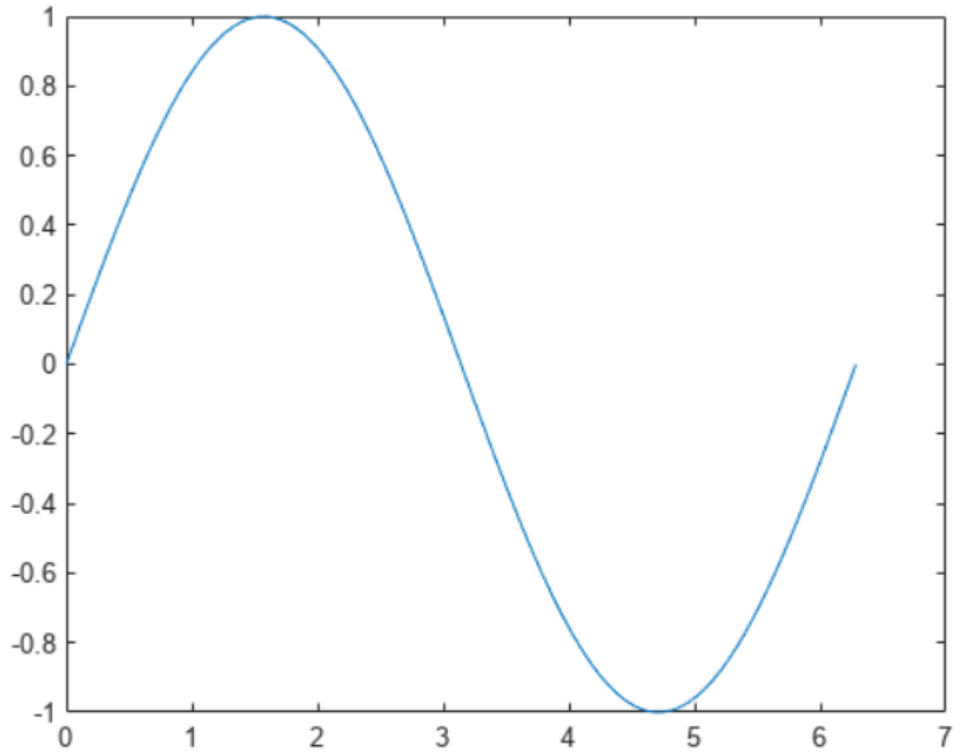
ESERCITAZIONI
MATLAB – Come iniziare

Grafici bidimensionali e tridimensionali

Grafici lineari

Per creare grafici lineari bidimensionali utilizzare la funzione `plot`. Ad esempio, tracciare la funzione seno su un vettore di valori spaziati linearmente da 0 a 2π:

```
x = linspace(0,2*pi);
y = sin(x);
plot(x,y)
```



ESERCITAZIONI
MATLAB – Come iniziare

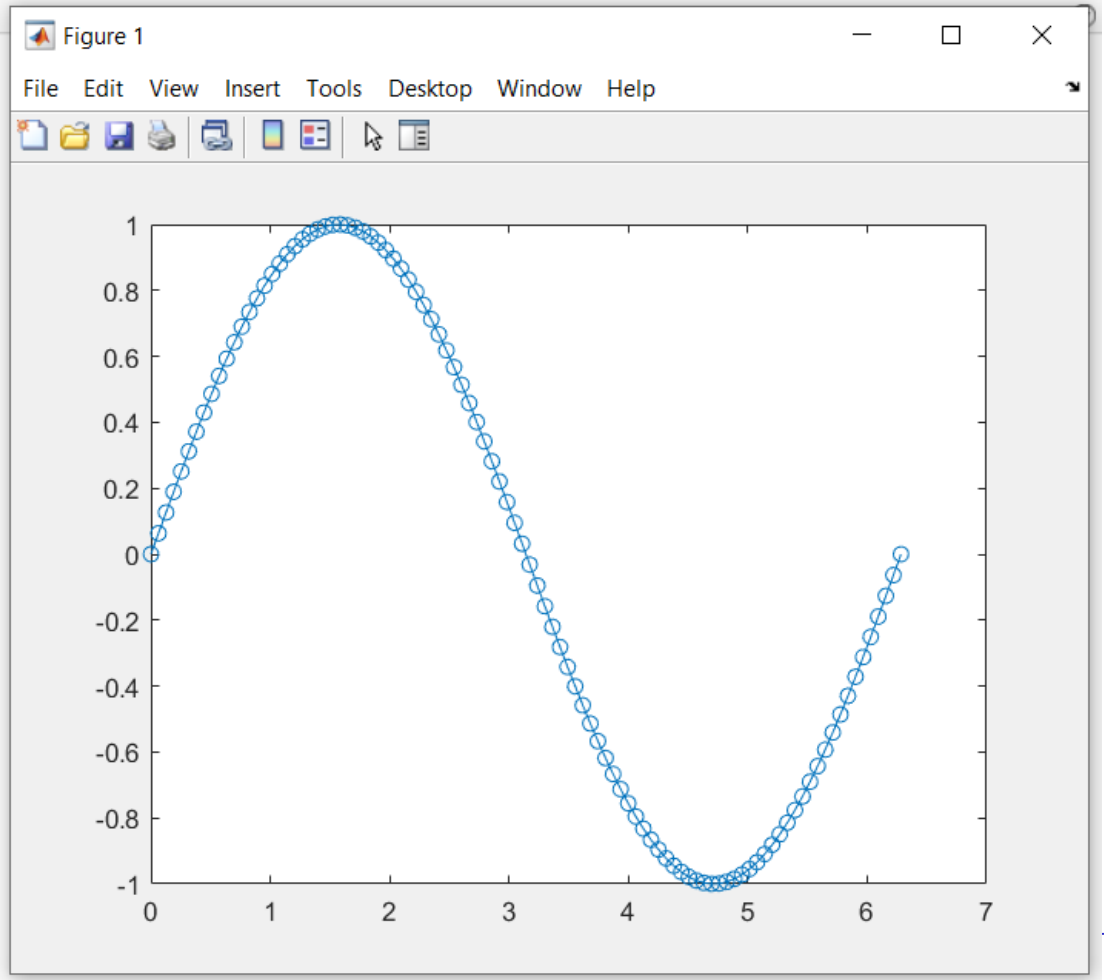
Grafici bidimensionali e tridimensionali

Grafici lineari

Per creare grafici lineari bidimensionali utilizzare la funzione `plot`. Ad esempio, tracciare la funzione seno su un vettore di valori spaziati linearmente da 0 a 2π:

Command Window

```
>> x=linspace(0,2*pi);
y=sin(x);
plot(x,y,'o-')
>>
```



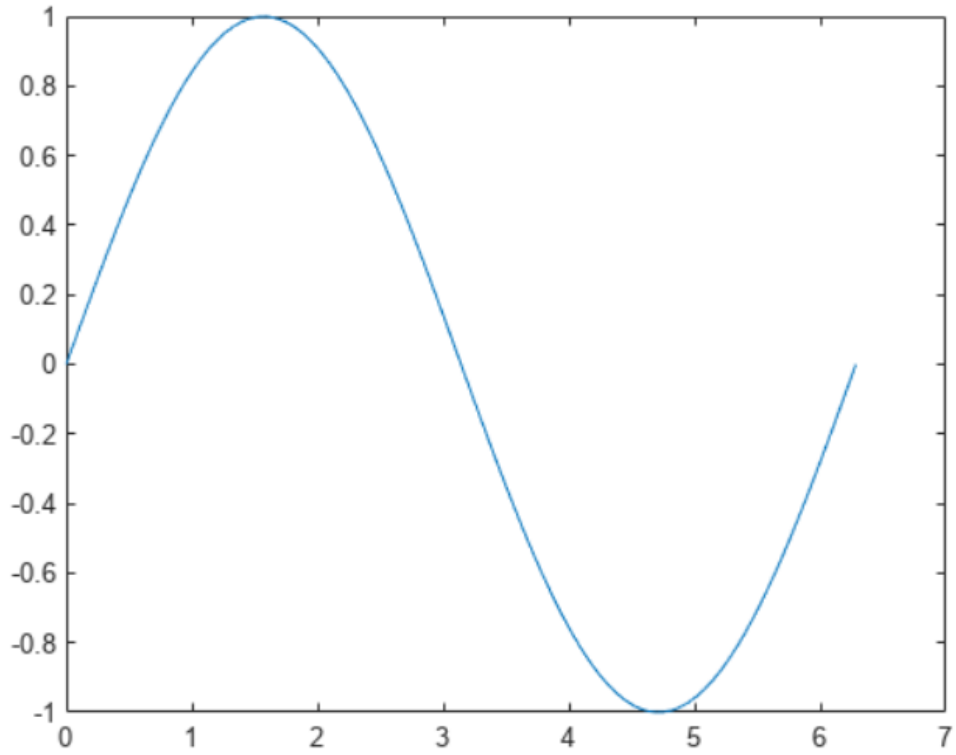
ESERCITAZIONI
MATLAB – Come iniziare

Grafici bidimensionali e tridimensionali

Grafici lineari

Per creare grafici lineari bidimensionali utilizzare la funzione `plot`. Ad esempio, tracciare la funzione seno su un vettore di valori spaziati linearmente da 0 a 2π:

```
x = linspace(0,2*pi);
y = sin(x);
plot(x,y)
```



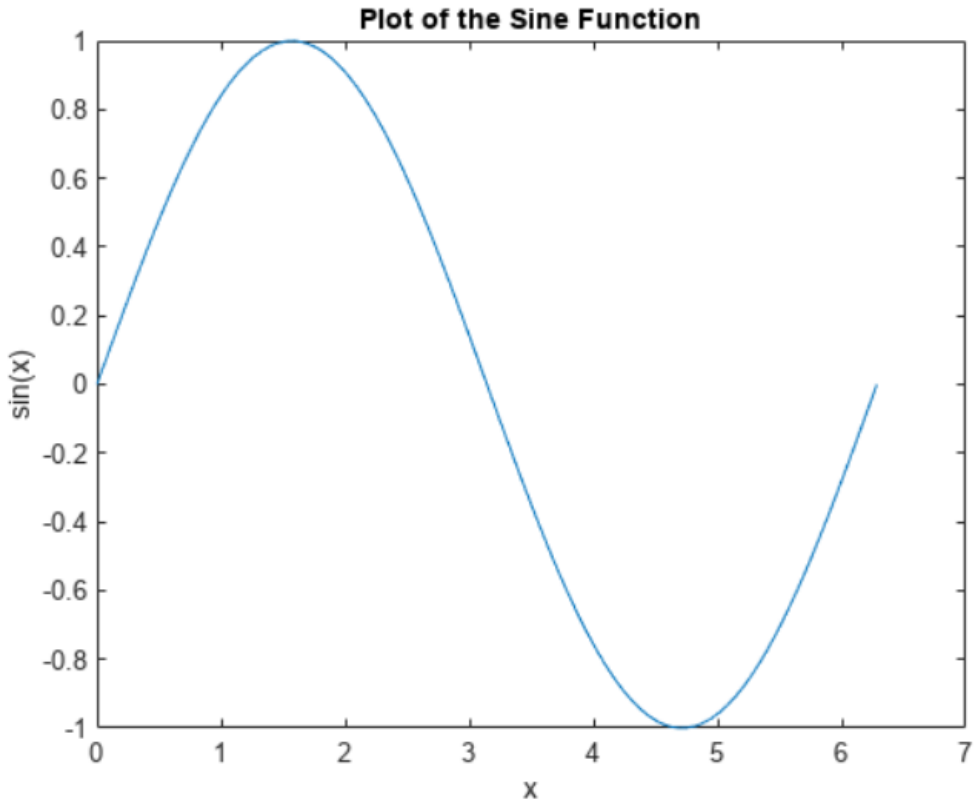
ESERCITAZIONI
MATLAB – Come iniziare

Grafici bidimensionali e tridimensionali

Grafici lineari

È possibile impostare etichette per gli assi e aggiungere un titolo.

```
xlabel("x")
ylabel("sin(x)")
title("Plot of the Sine Function")
```



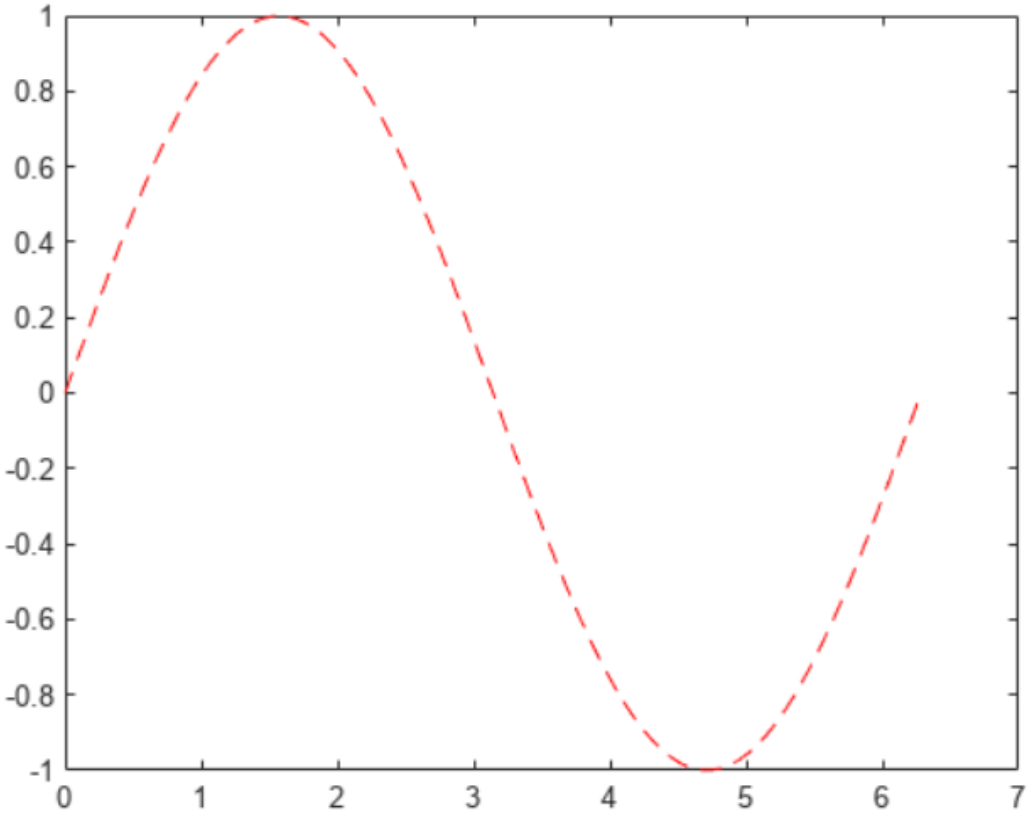
ESERCITAZIONI
MATLAB – Come iniziare

Grafici bidimensionali e tridimensionali

Grafici lineari

Aggiungendo un terzo argomento di input alla funzione `plot` è possibile tracciare le stesse variabili utilizzando una linea rossa tratteggiata.

```
plot(x,y,"r--")
```



ESERCITAZIONI
MATLAB – Come iniziare

Grafici bidimensionali e tridimensionali

Grafici lineari

"`r--`" è una *specifica della linea*. Ciascuna specifica può comprendere caratteristiche per colore, stile e marcatore della linea. Un marcatore è un simbolo riportato per ogni punto di dati tracciato, come `+`, `o` o `*`. Ad esempio, "`g:*`" richiede una linea verde punteggiata con i marcatori *.

Si noti che i titoli e le etichette definiti per il primo grafico non si trovano più nella finestra della figura corrente. Come impostazione predefinita, MATLAB® elimina la figura a ogni richiamo della funzione di plottaggio, resetta gli assi e altri elementi per preparare il nuovo grafico.

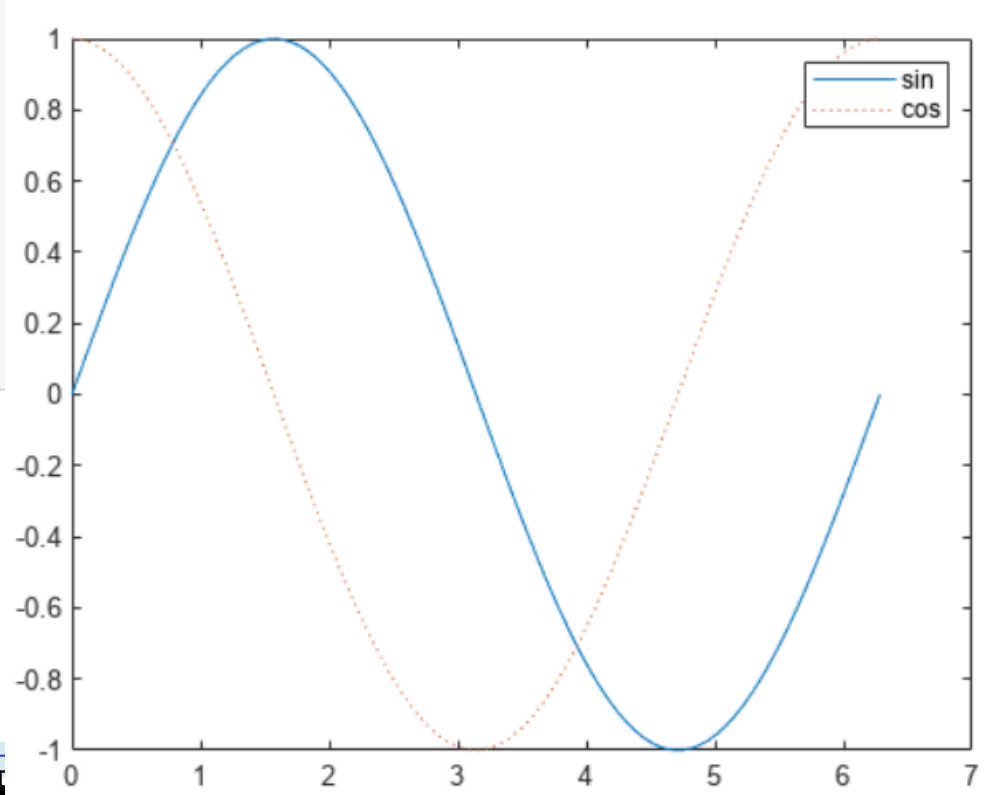
Per aggiungere grafici alla figura esistente utilizzare `hold on`. Fino a quando non si usa `hold off` o si chiude la finestra, tutti i grafici appaiono nella finestra della figura corrente.

```
x = linspace(0,2*pi);
y = sin(x);
plot(x,y)

hold on

y2 = cos(x);
plot(x,y2,":")
legend("sin","cos")

hold off
```



ESERCITAZIONI
MATLAB – Come iniziare

Grafici bidimensionali e tridimensionali

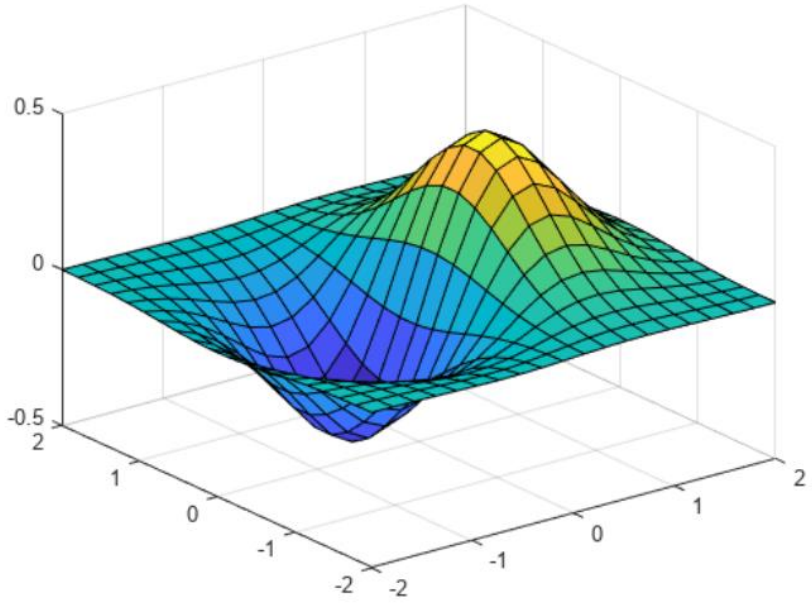
Grafici tridimensionali

I grafici tridimensionali solitamente visualizzano una superficie definita da una funzione in due variabili, $z = f(x, y)$. Ad esempio, calcolare $z = xe^{-x^2-y^2}$ dati i vettori riga e colonna `x` e `y` con 20 punti ciascuno nell'intervallo [-2,2].

```
x = linspace(-2,2,20);
y = x';
z = x .* exp(-x.^2 - y.^2);
```

Creare quindi un grafico a superficie.

```
surf(x,y,z)
```



Sia la funzione `surf`, sia la funzione dello stesso tipo `mesh` visualizzano superfici tridimensionali. `surf` visualizza a colori sia le linee di connessione che le superfici. `mesh` produce superfici wireframe in cui risultano colorate solo le linee di collegamento.

ESERCITAZIONI
MATLAB – Come iniziare

Grafici bidimensionali e tridimensionali

Grafici multipli

È possibile visualizzare più grafici in diverse parti della stessa finestra utilizzando `tiledlayout` o `subplot`.

La funzione `tiledlayout` è stata introdotta nella R2019b e fornisce un maggiore controllo sulle etichette e sulla spaziatura rispetto a `subplot`. Ad esempio, creare un layout 2x2 in una finestra della figura. Quindi, richiamare `nexttile` ogni volta che si desidera che un grafico appaia nella regione successiva.

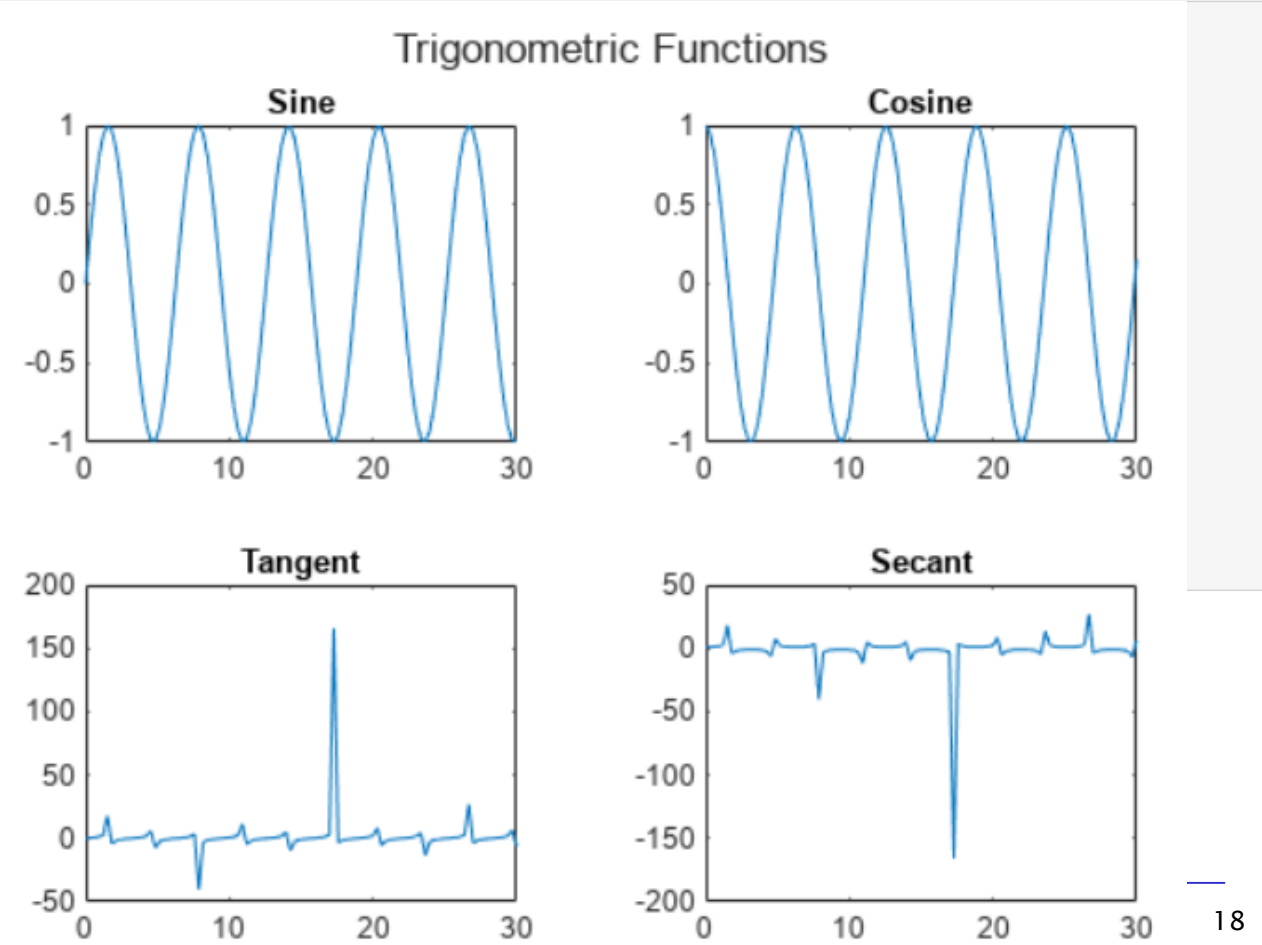
```
t = tiledlayout(2,2);
title(t,"Trigonometric Functions")
x = linspace(0,30);

nexttile
plot(x,sin(x))
title("Sine")

nexttile
plot(x,cos(x))
title("Cosine")

nexttile
plot(x,tan(x))
title("Tangent")

nexttile
plot(x,sec(x))
title("Secant")
```



ESERCITAZIONI
MATLAB – Come iniziare

Grafici bidimensionali e tridimensionali

Grafici multipli

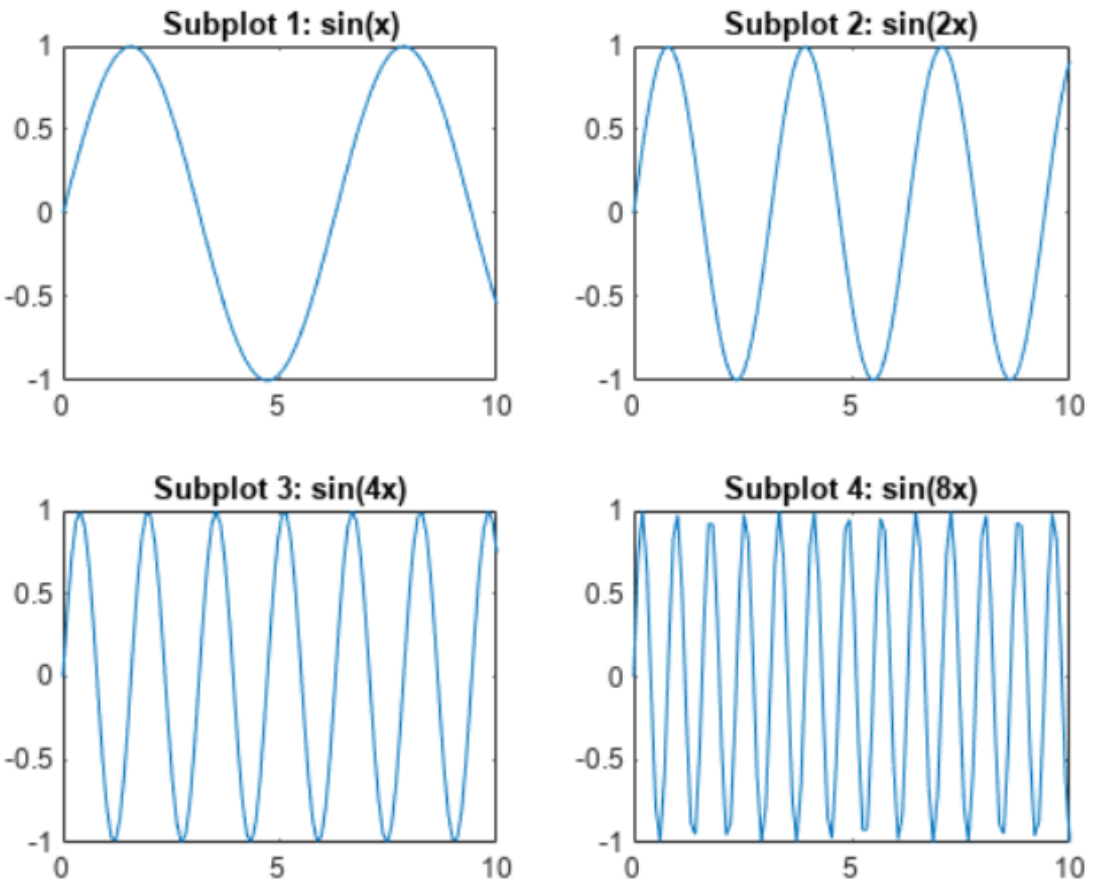
È possibile visualizzare più grafici in diverse parti della stessa finestra utilizzando `tiledlayout` o `subplot`.

```
subplot(2,2,1)
x = linspace(0,10);
y1 = sin(x);
plot(x,y1)
title('Subplot 1: sin(x)')

subplot(2,2,2)
y2 = sin(2*x);
plot(x,y2)
title('Subplot 2: sin(2x)')

subplot(2,2,3)
y3 = sin(4*x);
plot(x,y3)
title('Subplot 3: sin(4x)')

subplot(2,2,4)
y4 = sin(8*x);
plot(x,y4)
title('Subplot 4: sin(8x)')
```



ESERCITAZIONI
MATLAB – Come iniziare

Grafici bidimensionali e tridimensionali

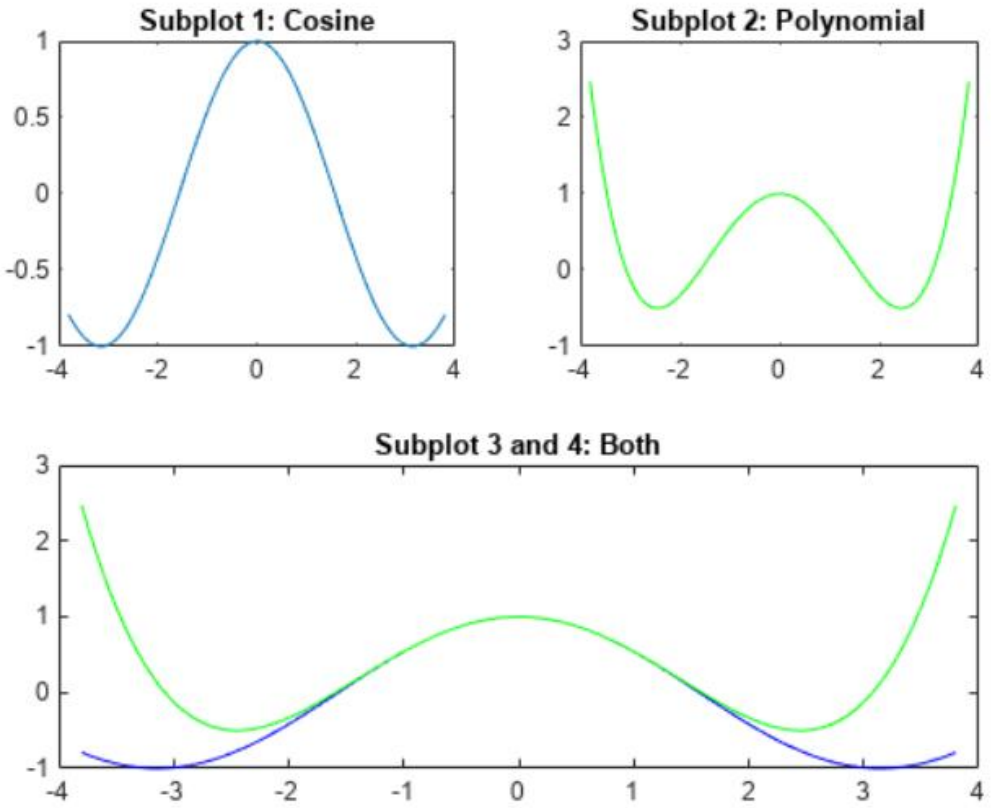
Grafici multipli

È possibile visualizzare più grafici in diverse parti della stessa finestra utilizzando `tiledlayout` o `subplot`.

```
subplot(2,2,1);
x = linspace(-3.8,3.8);
y_cos = cos(x);
plot(x,y_cos);
title('Subplot 1: Cosine')

subplot(2,2,2);
y_poly = 1 - x.^2./2 + x.^4./24;
plot(x,y_poly,'g');
title('Subplot 2: Polynomial')

subplot(2,2,[3,4]);
plot(x,y_cos,'b',x,y_poly,'g');
title('Subplot 3 and 4: Both')
```



Riferimenti Bibliografici

[1] https://it.mathworks.com