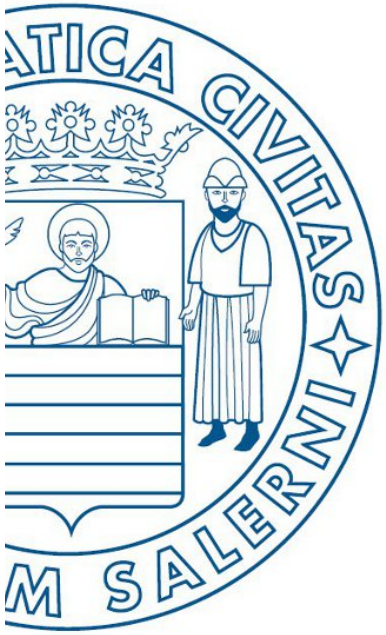


Università di Salerno
Dipartimento di
Ingegneria Industriale

UNIVERSITÀ DEGLI STUDI DI SALERNO

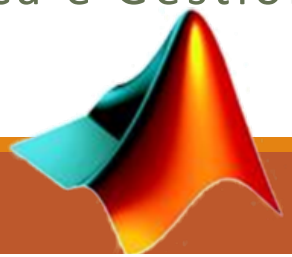
Fondamenti di Informatica

Calcolo Numerico

Prof. Christian Esposito

Corso di Laurea in Ingegneria Meccanica e Gestionale (Classe I)

A.A. 2017/18



MATLAB

OUTLINE

- Equazioni Algebriche Lineari
  - Risoluzione
- Calcolo Numerico
  - Integrazione Numerica
  - Derivazione Numerica
- Calcolo Simbolico
  - Variabili ed espressioni simboliche
  - Derivazione, Integrazione e Limiti
  - MuPAD

Equazioni Algebriche Lineari (1)

Le equazioni algebriche sono equazioni equivalenti ad un polinomio uguagliato a zero. Il grado del polinomio è il grado dell'equazioni, e se si usano polinomi di primo grado si hanno equazioni algebriche lineari:

$$5x - 2y = 13$$

$$7x + 3y = 24$$

Si presentano in molte applicazioni dell'ingegneria, e posso essere risolte "a mano" con carta e penna, con speciali calcolatrici e con MATLAB.

MATLAB offre la maggiore potenza di calcolo e flessibilità.

Equazioni Algebriche Lineari (2)

I sistemi di equazioni algebriche lineari possono essere espressi con una notazione matriciale:

$$5x_1 - 2x_2 = 13$$

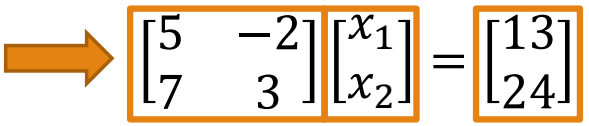
$$7x_1 + 3x_2 = 24$$

Equazioni Algebriche Lineari (2)

I sistemi di equazioni algebriche lineari possono essere espressi con una notazione matriciale:

$5x_1 - 2x_2 = 13$

$7x_1 + 3x_2 = 24$



$$\begin{bmatrix} 5 & -2 \\ 7 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 13 \\ 24 \end{bmatrix}$$

A matrice — x vettore — b vettore

Equazioni Algebriche Lineari (2)

I sistemi di equazioni algebriche lineari possono essere espressi con una notazione matriciale:

$5x_1 - 2x_2 = 13$
$7x_1 + 3x_2 = 24$

$$\rightarrow \begin{bmatrix} 5 & -2 \\ 7 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 13 \\ 24 \end{bmatrix} \rightarrow Ax = b$$

Un sistema di m equazioni in n incognite può essere espressa in forma matriciale, con una matrice dei coefficienti di dimensione m x n, un vettore delle incognite 1 x n e un vettore dei termini noti 1 x n.

La soluzione di ax = b è pari a x = b/a se a ≠ 0. Anche nell'algebra matriciale si ha l'operazione di divisione, da usare per dividere opportunamente b per A.

Equazioni Algebriche Lineari (3)

L'inversa di una matrice è intesa come quella matrice, se esiste, tale che il prodotto matriciale tra le due restituisce la matrice identità:

$$A^{-1}A = AA^{-1} = I.$$

Applicando questa proprietà al sistema di equazioni abbiamo che:

$A^{-1}Ax = A^{-1}b$

Equazioni Algebriche Lineari (3)

L'inversa di una matrice è intesa come quella matrice, se esiste, tale che il prodotto matriciale tra le due restituisce la matrice identità:

$$A^{-1}A = AA^{-1} = I.$$

Applicando questa proprietà al sistema di equazioni abbiamo che:

$A^{-1}Ax = A^{-1}b$

$A^{-1}Ax = Ix = x$

Equazioni Algebriche Lineari (3)

L'inversa di una matrice è intesa come quella matrice, se esiste, tale che il prodotto matriciale tra le due restituisce la matrice identità:

$$A^{-1}A = AA^{-1} = I.$$

Applicando questa proprietà al sistema di equazioni abbiamo che:

$A^{-1}Ax = A^{-1}b \Rightarrow x = A^{-1}b$

$\Downarrow$

$A^{-1}Ax = Ix = x$

Equazioni Algebriche Lineari (3)

L'inversa di una matrice è intesa come quella matrice, se esiste, tale che il prodotto matriciale tra le due restituisce la matrice identità:

$$A^{-1}A = AA^{-1} = I.$$

Applicando questa proprietà al sistema di equazioni abbiamo che:

$A^{-1}Ax = A^{-1}b \rightarrow x = A^{-1}b$

$A^{-1}Ax = Ix = x$

L'operazione di inversione di una matrice è possibile solo se la matrice è

- quadrata - se ha un numero uguale di righe e colonne, detto ordine della matrice;
- non singolare – se il suo determinante è diverso da zero.

Equazioni Algebriche Lineari (4)

Il determinante di una matrice quadrata A è un numero che descrive alcune proprietà algebriche e geometriche della matrice. Il determinante viene calcolato secondo vari approcci, assiomatici o costruttivi, e MATLAB mette a disposizione la funzione det() per il suo calcolo:

```
>> A = [5, -2; 7, 3];
>> det(A)ans =   29.0000
```

Invece per il computo dell'inversa di una matrice è disponibile la funzione inv():

```
>> inv(A)
ans =    0.1034    0.0690   -0.2414    0.1724
```

Equazioni Algebriche Lineari (5)

Pertanto il seguente sistema di equazioni

$$5x_1 - 2x_2 = 13$$

$$7x_1 + 3x_2 = 24$$

può essere risolto in MATLAB come segue:

```
>> A = [5, -2; 7, 3];
>> b = [13; 24];
>> x = inv(A)*b
x =     3     1
```

Equazioni Algebriche Lineari (5)

Pertanto il seguente sistema di equazioni

$$5x_1 - 2x_2 = 13$$

$$7x_1 + 3x_2 = 24$$

può essere risolto in MATLAB come segue:

```
>> A = [5, -2; 7, 3];
>> b = [13; 24];
>> x = inv(A)*b
x =     3     1
```

MATLAB non ha restituito alcun messaggio di errore, quindi ha potuto computare l'inversa della matrice A. Ciò si verifica quando la soluzione del sistema è unica.

Equazioni Algebriche Lineari (6)

Il metodo della matrice inversa ci dice se non esiste una soluzione unica, ovvero quando A è singolare. Il limite di questo metodo è che

- non ci dice se il sistema non ha soluzioni;
- non ci dice se ha un numero infinito di soluzioni;
- si applica solo a sistemi con matrice quadrata.

Per ottenere maggiori informazioni sul sistema è necessario guardare al rango delle sue matrici caratterizzanti.

Il rango di una matrice è l'ordine più alto rispetto al quale esistono matrici quadrate estratte da A con determinante diverso da zero.

Equazioni Algebriche Lineari (7)

In MATLAB possiamo determinare il rango di una matrice con la funzione rank():

```
>> A = [5, -2; 7, 3];
>> rank(A)
ans =     2
```

Un sistema Ax = b ammette soluzioni solo se rank(A) = rank([A b]). Se con r intendiamo il rango di A e con n il numero di incognite e rank(A) = rank([A b]), si ha:

- se r = n, allora il sistema ammette una sola soluzione;
- se r < n, esistono infinite soluzioni.

In un sistema omogeneo (b = 0), rank(A) è sempre pari a rank([A b]), quindi si ha sempre la soluzione banale x = 0. Se rank(A) < n allora si ha una soluzione non nulla.

Equazioni Algebriche Lineari (8)

Esempio:

```
>> A
A =     5    -2     7     3
>> b
b =    13    24
>> rank(A)
ans =     2
>> rank([A b])
ans =     2
>> size(A)
ans =     2     2
```

Equazioni Algebriche Lineari (9)

Quando un sistema è caratterizzato da un elevato numero di incognite e di equazioni, il metodo dell'inversa non è computazionalmente ottimale.

MATLAB dispone del metodo della divisione a sinistra, che si basa sulle riduzioni delle variabili di Gauss. Per impiegare questo metodo bisogna digitare x=A\b.

```
>> A
A =      5     -2      7      3
>> b
b =     13     24
>> x=A\b
x =     3.0000    1.0000
```

Equazioni Algebriche Lineari (10)

Questo metodo si basa sul trasformare il sistema assegnato in un sistema a matrice triangolare superiore, e quindi nel risolvere il sistema equivalente così ottenuto con una backward substitution. Si basa sull'assunto che sistemi di equazioni con la matrice A triangolare sono di immediata risoluzione.

$$\begin{cases} x + y = -2 \\ x + 3y + z = 2 \\ 3x + 7y - z = 5 \end{cases}$$

$$\left(\begin{array}{ccc|c} 1 & 1 & 0 & -2 \\ 1 & 3 & 1 & 2 \\ 3 & 7 & -1 & 5 \end{array}\right)$$

A B

Equazioni Algebriche Lineari (10)

Questo metodo si basa sul trasformare il sistema assegnato in un sistema a matrice triangolare superiore, e quindi nel risolvere il sistema equivalente così ottenuto con una backward substitution. Si basa sull'assunto che sistemi di equazioni con la matrice A triangolare sono di immediata risoluzione.

$$\begin{cases} x + y = -2 \\ x + 3y + z = 2 \\ 3x + 7y - z = 5 \end{cases}$$

$$\left(\begin{array}{ccc|c} 1 & 1 & 0 & -2 \\ 1 & 3 & 1 & 2 \\ 3 & 7 & -1 & 5 \end{array}\right) \rightarrow \left(\begin{array}{ccc|c} 1 & 1 & 0 & -2 \\ 0 & 2 & 1 & 4 \\ & 7 & -1 & 5 \end{array}\right)$$

A B

Sottraggo alla seconda riga la prima riga.

Equazioni Algebriche Lineari (10)

Questo metodo si basa sul trasformare il sistema assegnato in un sistema a matrice triangolare superiore, e quindi nel risolvere il sistema equivalente così ottenuto con una backward substitution. Si basa sull'assunto che sistemi di equazioni con la matrice A triangolare sono di immediata risoluzione.

$$\begin{cases} x + y = -2 \\ x + 3y + z = 2 \\ 3x + 7y - z = 5 \end{cases}$$

$$\left(\begin{array}{ccc|c} 1 & 1 & 0 & -2 \\ 1 & 3 & 1 & 2 \\ 3 & 7 & -1 & 5 \end{array}\right) \rightarrow \left(\begin{array}{ccc|c} 1 & 1 & 0 & -2 \\ 0 & 2 & 1 & 4 \\ 3 & 7 & -1 & \end{array}\right) \rightarrow \left(\begin{array}{ccc|c} 1 & 1 & 0 & -2 \\ 0 & 2 & 1 & 4 \\ 0 & 4 & -1 & 11 \end{array}\right)$$

A B

Sottraggo la terza riga con la seconda moltiplicata per 3.

Equazioni Algebriche Lineari (10)

Questo metodo si basa sul trasformare il sistema assegnato in un sistema a matrice triangolare superiore, e quindi nel risolvere il sistema equivalente così ottenuto con una backward substitution. Si basa sull'assunto che sistemi di equazioni con la matrice A triangolare sono di immediata risoluzione.

$$\begin{cases} x + y = -2 \\ x + 3y + z = 2 \\ 3x + 7y - z = 5 \end{cases}$$

$$\left(\begin{array}{ccc|c} 1 & 1 & 0 & -2 \\ 1 & 3 & 1 & 2 \\ 3 & 7 & -1 & 5 \end{array}\right) \rightarrow \left(\begin{array}{ccc|c} 1 & 1 & 0 & -2 \\ 0 & 2 & 1 & 4 \\ 3 & 7 & -1 & 5 \end{array}\right) \rightarrow \left(\begin{array}{ccc|c} 1 & 1 & 0 & -2 \\ 0 & 2 & 1 & 4 \\ 0 & 4 & -1 & 11 \end{array}\right) \rightarrow \left(\begin{array}{ccc|c} 1 & 1 & 0 & -2 \\ 0 & 1 & 1/2 & 2 \\ 0 & 4 & -1 & 11 \end{array}\right)$$

A B

Divido la seconda riga per due.

Equazioni Algebriche Lineari (10)

Questo metodo si basa sul trasformare il sistema assegnato in un sistema a matrice triangolare superiore, e quindi nel risolvere il sistema equivalente così ottenuto con una backward substitution. Si basa sull'assunto che sistemi di equazioni con la matrice A triangolare sono di immediata risoluzione.

$$\begin{cases} x + y = -2 \\ x + 3y + z = 2 \\ 3x + 7y - z = 5 \end{cases}$$

$$\left(\begin{array}{ccc|c} 1 & 1 & 0 & -2 \\ 1 & 3 & 1 & 2 \\ 3 & 7 & -1 & 5 \end{array}\right) \rightarrow \left(\begin{array}{ccc|c} 1 & 1 & 0 & -2 \\ 0 & 2 & 1 & 4 \\ 3 & 7 & -1 & 5 \end{array}\right) \rightarrow \left(\begin{array}{ccc|c} 1 & 1 & 0 & -2 \\ 0 & 2 & 1 & 4 \\ 0 & 4 & -1 & 11 \end{array}\right) \rightarrow \left(\begin{array}{ccc|c} 1 & 1 & 0 & -2 \\ 0 & 1 & 1/2 & 2 \\ 0 & 4 & -1 & 11 \end{array}\right)$$

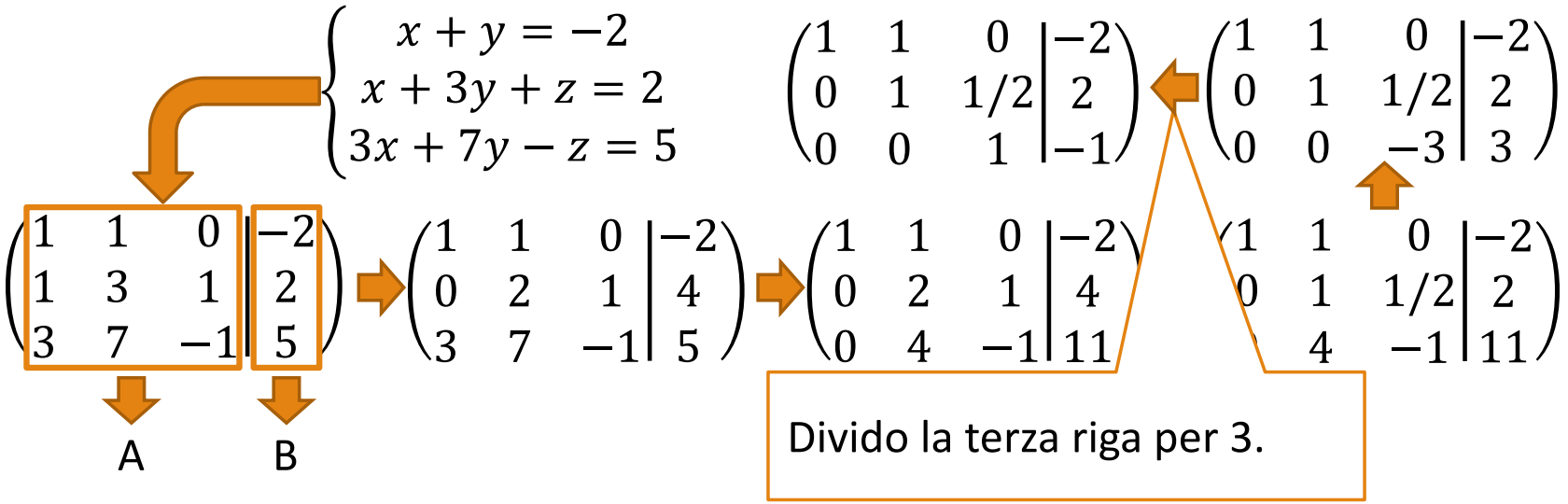
$$\left(\begin{array}{ccc|c} 1 & 1 & 0 & -2 \\ 0 & 1 & 1/2 & 2 \\ 0 & 0 & -3 & 3 \end{array}\right)$$

A B

Sottraggo la terza riga con la seconda moltiplicata per 4.

Equazioni Algebriche Lineari (10)

Questo metodo si basa sul trasformare il sistema assegnato in un sistema a matrice triangolare superiore, e quindi nel risolvere il sistema equivalente così ottenuto con una backward substitution. Si basa sull'assunto che sistemi di equazioni con la matrice A triangolare sono di immediata risoluzione.



Equazioni Algebriche Lineari (10)

Questo metodo si basa sul trasformare il sistema assegnato in un sistema a matrice triangolare superiore, e quindi nel risolvere il sistema equivalente così ottenuto con una backward substitution. Si basa sull'assunto che sistemi di equazioni con la matrice A triangolare sono di immediata risoluzione.

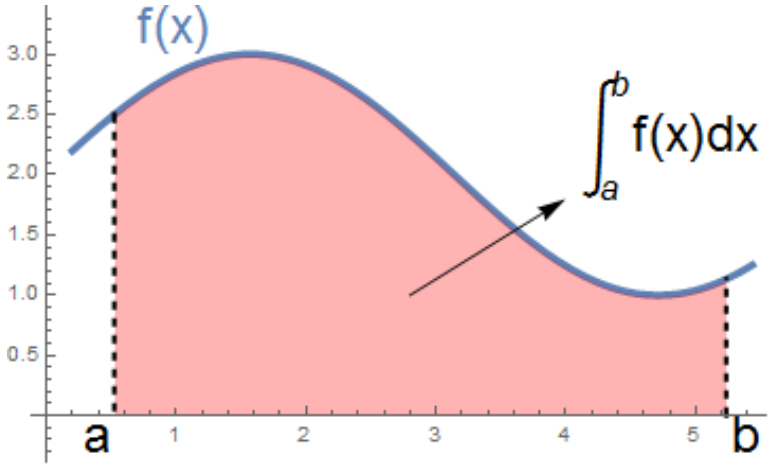
$$\begin{cases} x = -9/2 \\ y = 5/2 \\ z = -1 \end{cases} \leftarrow \begin{cases} x + y = -2 \\ y + z/2 = 2 \\ z = -1 \end{cases} \leftarrow \left(\begin{array}{ccc|c} 1 & 1 & 0 & -2 \\ 0 & 1 & 1/2 & 2 \\ 0 & 0 & 1 & -1 \end{array}\right) \leftarrow \left(\begin{array}{ccc|c} 1 & 1 & 0 & -2 \\ 0 & 1 & 1/2 & 2 \\ 0 & 0 & -3 & 3 \end{array}\right)$$

$$\left(\begin{array}{ccc|c} 1 & 1 & 0 & -2 \\ 1 & 3 & 1 & 2 \\ 3 & 7 & -1 & 5 \end{array}\right) \rightarrow \left(\begin{array}{ccc|c} 1 & 1 & 0 & -2 \\ 0 & 2 & 1 & 4 \\ 3 & 7 & -1 & 5 \end{array}\right) \rightarrow \left(\begin{array}{ccc|c} 1 & 1 & 0 & -2 \\ 0 & 2 & 1 & 4 \\ 0 & 4 & -1 & 11 \end{array}\right) \rightarrow \left(\begin{array}{ccc|c} 1 & 1 & 0 & -2 \\ 0 & 1 & 1/2 & 2 \\ 0 & 4 & -1 & 11 \end{array}\right)$$

A B

Integrazione Numerica (1)

L'integrale di una funzione f(x) per a ≤ x ≤ b viene interpretato come l'area compresa fra la curva formata dalla funzione f(x), l'asse x e le due rette x = a e x =b.



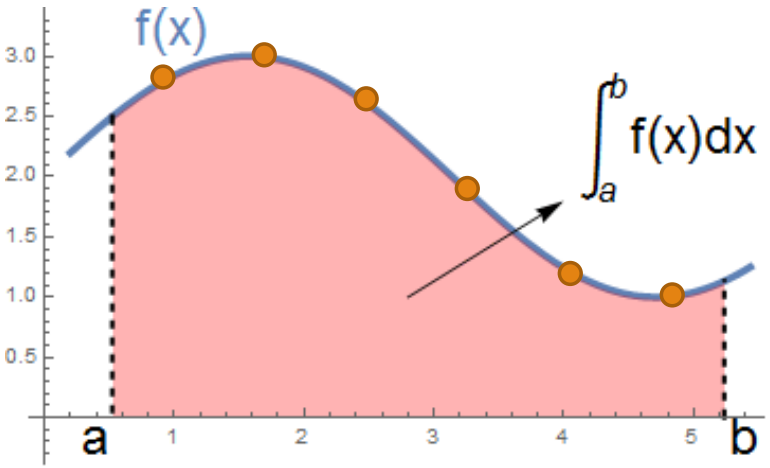
$$A = \int_a^b f(x)dx$$

Un integrale è detto definito se dispone di limiti di integrazione, mentre quando questi limiti sono assenti lo si definisce indefinito.

MATLAB dispone di funzioni per il calcolo di integrali definiti secondo l'approccio trapezoidale e la regola di Simpson.

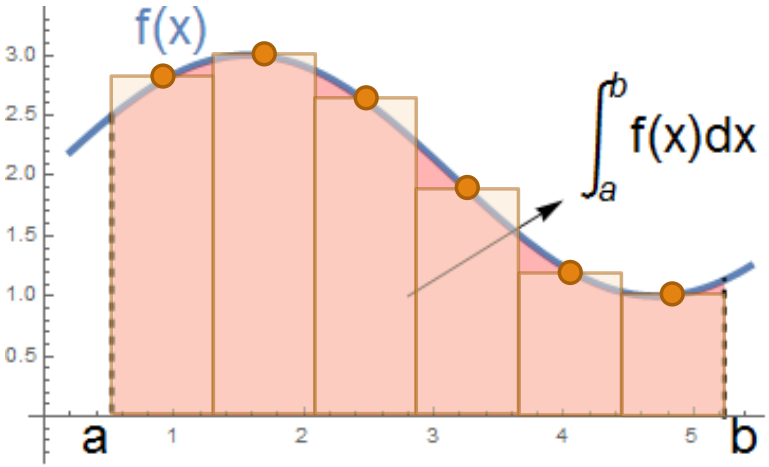
Integrazione Numerica (2)

L'approccio rettangolare alla risoluzione numerica di un integrale definito consiste nell'identificare una serie di punti y_i sulla curva di f(x)…



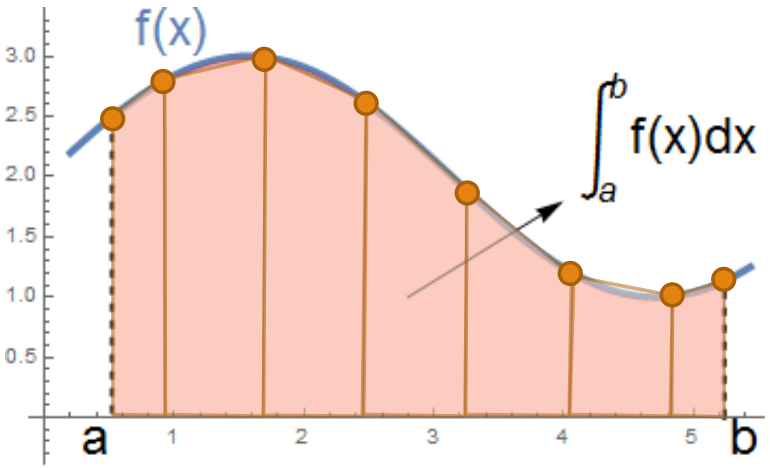
Integrazione Numerica (2)

L'approccio rettangolare alla risoluzione numerica di un integrale definito consiste nell'identificare una serie di punti y_i sulla curva di f(x), e di costruire una serie di rettangoli di ampiezza a (pari per tutti o meno) ed altezza $f(y_i)$. La somma delle aree di questi rettangoli fornisce un'approssimazione dell'area dell'integrale.



Integrazione Numerica (2)

L'approccio rettangolare alla risoluzione numerica di un integrale definito consiste nell'identificare una serie di punti y_i sulla curva di f(x), e di costruire una serie di rettangoli di ampiezza a (pari per tutti o meno) ed altezza $f(y_i)$. La somma delle aree di questi rettangoli fornisce un'approssimazione dell'area dell'integrale.



In alternativa è possibile ricorrere al metodo trapezoidale, dove vengono costruiti dei trapezi tra punti adiacenti della funzione f(x) e si approssima l'integrale con la somma delle aree dei trapezi.

Integrazione Numerica (3)

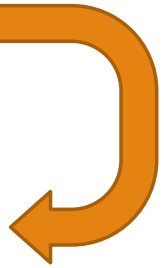
MATLAB mette a disposizione l'integrazione trapezoidale con la funzione trapz(), che richiede in input gli array delle coordinate x e y dei punti lungo la curva disegnata dalla funzione f(x) di cui si intende calcolare l'integrale definito. Dove il primo e l'ultimo punto appartengono agli intervalli di integrazione.

$$A = \int_0^{\pi} \sin x \, dx$$

Integrazione Numerica (3)

MATLAB mette a disposizione l'integrazione trapezoidale con la funzione trapz(), che richiede in input gli array delle coordinate x e y dei punti lungo la curva disegnata dalla funzione f(x) di cui si intende calcolare l'integrale definito. Dove il primo e l'ultimo punto appartengono agli intervalli di integrazione.

$$A = \int_0^{\pi} \sin x \, dx$$



Risoluzione in MATLAB

```
>> x = linspace(0, pi, 10);
>> y = sin(x);
>> A = trapz(x, y)
A =    1.9797
```

Integrazione Numerica (3)

MATLAB mette a disposizione l'integrazione trapezoidale con la funzione trapz(), che richiede in input gli array delle coordinate x e y dei punti lungo la curva disegnata dalla funzione f(x) di cui si intende calcolare l'integrale definito. Dove il primo e l'ultimo punto appartengono agli intervalli di integrazione.

$$A = \int_0^{\pi} \sin x \, dx$$

Risoluzione in MATLAB

```
>> x = linspace(0, pi, 10);
>> y = sin(x);
>> A = trapz(x, y)
A =    1.9797
```

La risoluzione esatta di questo integrale è A = 2, MATLAB commette un errore pari a 0,0203, ovvero 1%.

Integrazione Numerica (3)

MATLAB mette a disposizione l'integrazione trapezoidale con la funzione trapz(), che richiede in input gli array delle coordinate x e y dei punti lungo la curva disegnata dalla funzione f(x) di cui si intende calcolare l'integrale definito. Dove il primo e l'ultimo punto appartengono agli intervalli di integrazione.

$$A = \int_0^{\pi} \sin x \, dx$$

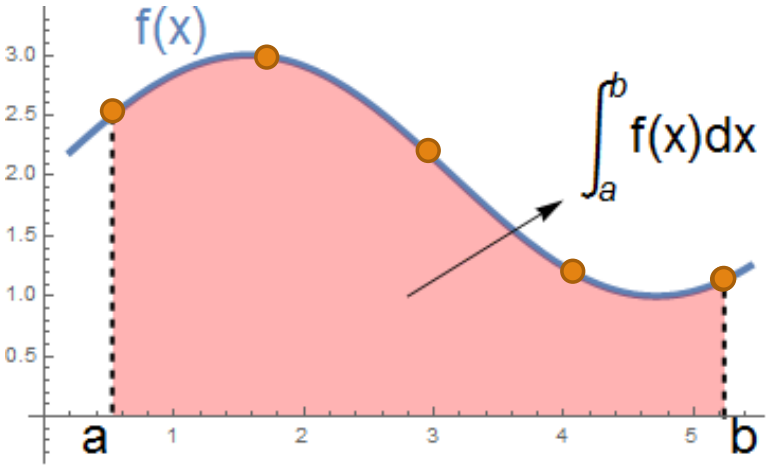
Risoluzione in MATLAB

```
>> x = linspace(0, pi, 100);
>> y = sin(x);
>> A = trapz(x, y)
A =    1.9998
```

Con un numero maggiori di punti commettiamo un errore pari a 0,0002, ovvero 0,01%.

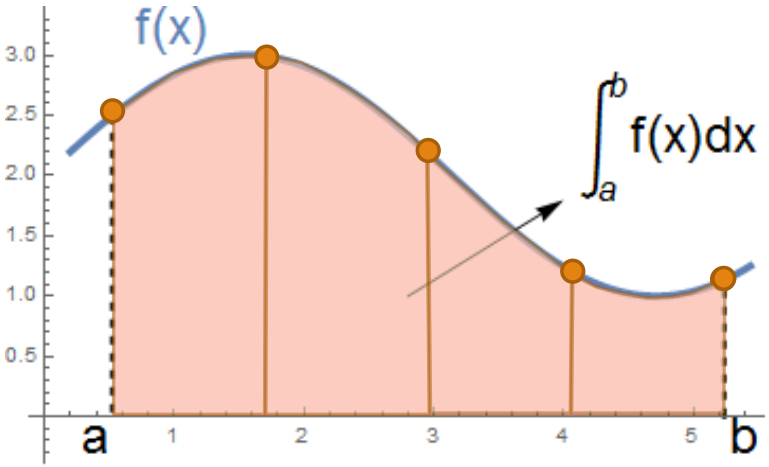
Integrazione Numerica (4)

Il difetto dell'uso della funzione trapz() è che richiede una serie di punti della funzione di cui bisogna computare l'integrale definito. Un approccio alternativo è offerto dalla quadratura, che usa la regola di Cavalieri-Simpson per approssimare l'integrale della funzione f(x) con quello della successione di funzioni quadratiche tra punti adiacenti.



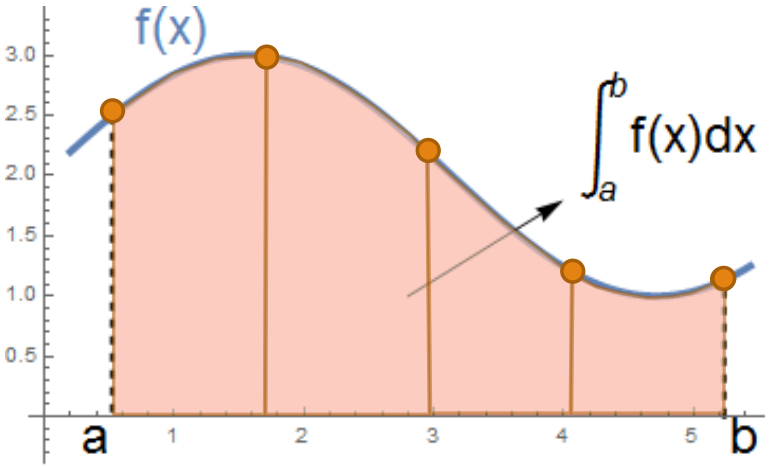
Integrazione Numerica (4)

Il difetto dell'uso della funzione trapz() è che richiede una serie di punti della funzione di cui bisogna computare l'integrale definito. Un approccio alternativo è offerto dalla quadratura, che usa la regola di Cavalieri-Simpson per approssimare l'integrale della funzione f(x) con quello della successione di funzioni quadratiche tra punti adiacenti.



Integrazione Numerica (4)

Il difetto dell'uso della funzione trapz() è che richiede una serie di punti della funzione di cui bisogna computare l'integrale definito. Un approccio alternativo è offerto dalla quadratura, che usa la regola di Cavalieri-Simpson per approssimare l'integrale della funzione f(x) con quello della successione di funzioni quadratiche tra punti adiacenti.



MATLAB mette a disposizione questo metodo con la funzione quad() che richiede in ingresso la funzione da integrare e i due estremi di integrazione. La funzione è passata per mezzo di un handle di funzione.

Integrazione Numerica (5)

Una variabile MATLAB può anche contenere un valore di tipo funzione, ed in questo caso prende il nome di handle.

L'handle può essere applicato a opportuni argomenti per ottenere una invocazione della funzione.

Esempio di un handle di una funzione esistente:

```
>> seno = @sin
>> seno(pi/2)
ans = 1
```

Esempio di un handle di una funzione definita ex-novo:

```
>> sq=@(x)x^2
>> sq(8)
ans =  64
```

Integrazione Numerica (6)

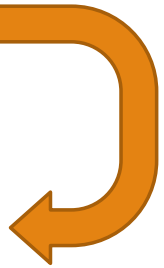
Ipotizziamo di voler calcolare l'integrale dell'esempio precedente:

$$A = \int_0^{\pi} \sin x \, dx$$

Integrazione Numerica (6)

Ipotizziamo di voler calcolare l'integrale dell'esempio precedente:

$$A = \int_0^{\pi} \sin x \, dx$$



Risoluzione con la quadratura

```
>> seno = @sin;
>> A = quad(seno, 0, pi)
A =    2.0000
```

In alternativa, è possibile usare il nome della funzione, passandola come stringa:

```
>> A = quad('sin', 0, pi)
A =    2.0000
```

Integrazione Numerica (7)

Consideriamo un esempio più complesso:

$$A = \int_0^{\sqrt{2\pi}} \cos x^2 x \, dx$$

Integrazione Numerica (7)

Consideriamo un esempio più complesso:

$$A = \int_0^{\sqrt{2\pi}} \cos x^2 x \, dx$$

Risoluzione con la quadratura

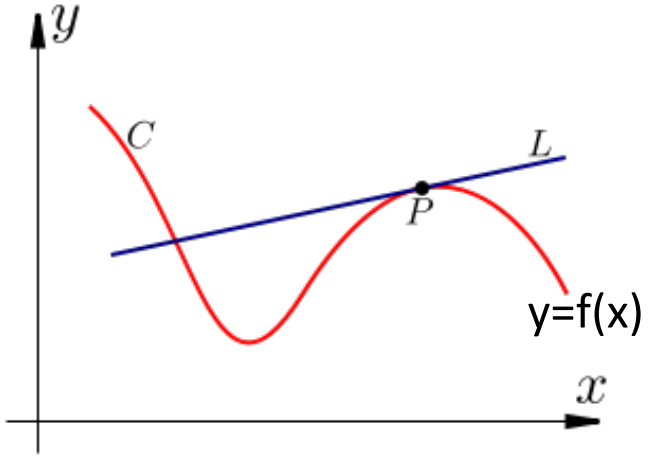
```
>> funzione = @(x)cos(x.^2);
>> A = quad(funzione, 0, sqrt(2*pi))
A =    0.6119
```

In alternativa, posso compattare le due righe in un solo comando:

```
>> A = quad(@(x)cos(x.^2), 0, sqrt(2*pi))
A =    0.6119
```

Derivazione Numerica (1)

La derivata di una funzione in un determinato punto è interpretabile graficamente come la pendenza della tangente al grafico della funzione in quel punto. Più formalmente risulta pari al limite del rapporto incrementale della funzione nel punto al tendere dell'incremento a zero:

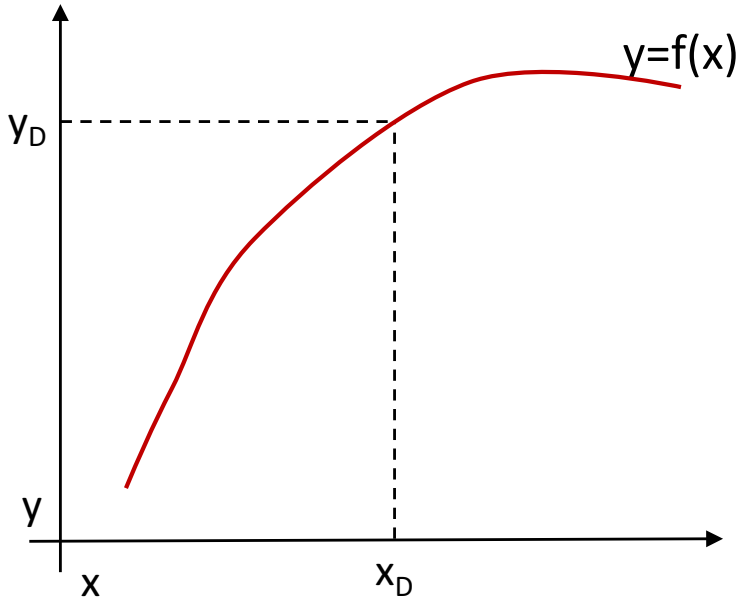


$$\frac{dy}{dx} = \lim_{\Delta x \to 0} \frac{\Delta y}{\Delta x}$$

Come è possibile determinare la derivata di una funzione f(x) in un determinato punto x_D?

Derivazione Numerica (2)

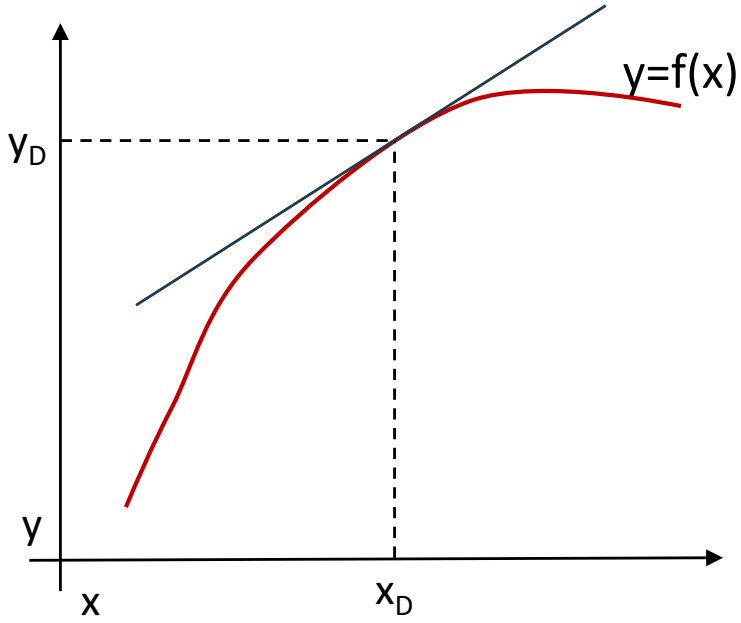
Consideriamo il grafo della funzione nel piano cartesiano.



La derivata di f(x) nel punto x_D è data dalla pendenza della retta che passa per il punto (x_D, y_D), dove y_D è l'uscita della funzione con input pari a x_D.

Derivazione Numerica (2)

Consideriamo il grafo della funzione nel piano cartesiano.

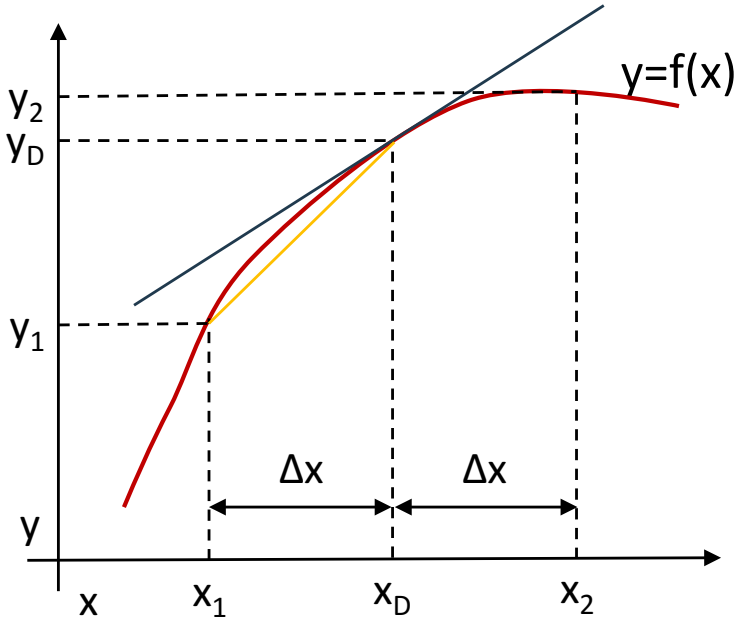


La derivata di f(x) nel punto x_D è data dalla pendenza della retta che passa per il punto (x_D, y_D), dove y_D è l'uscita della funzione con input pari a x_D.

Non disponendo di un secondo punto su questa retta, non possiamo stimarne la pendenza.

Tale problema può essere ovviato usando dei punti vicini, fissando una distanza dal punto x_D pari a Δx.

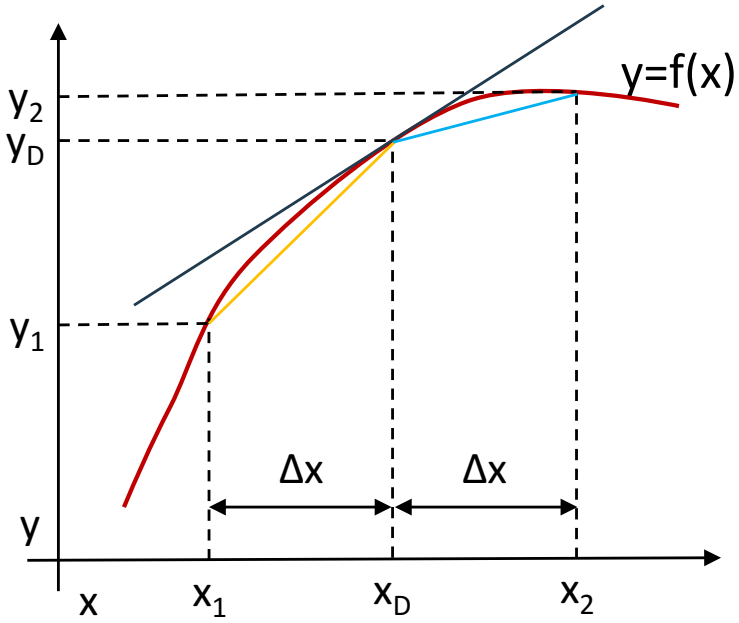
Derivazione Numerica (3)



Metodo delle differenze all'indietro: si consideri la retta tra i punti (x_1, y_1) e (x_D, y_D), la cui pendenza è data da:

$$m_A = \frac{y_D - y_1}{\Delta x}$$

Derivazione Numerica (3)



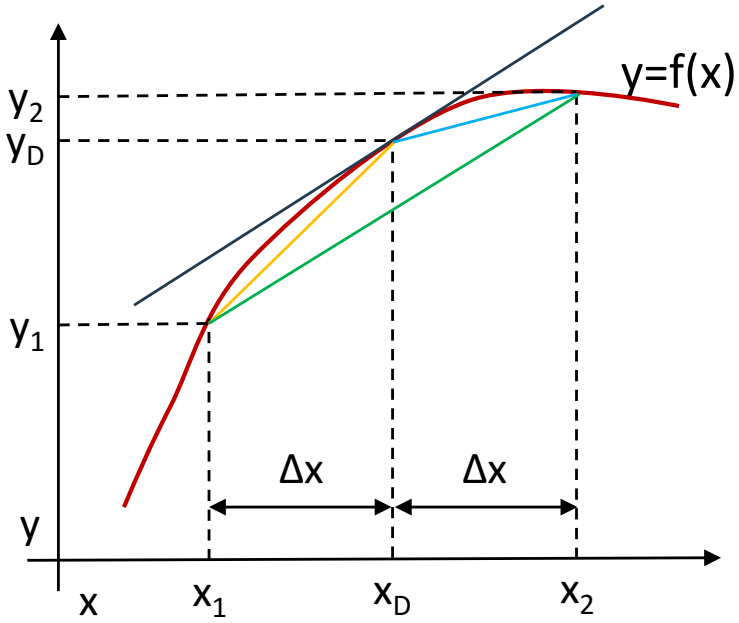
Metodo delle differenze all'indietro: si consideri la retta tra i punti (x_1, y_1) e (x_D, y_D), la cui pendenza è data da:

$$m_A = \frac{y_D - y_1}{\Delta x}$$

Metodo delle differenze in avanti: si usi la retta tra (x_D, y_D) e (x_2, y_2), la cui pendenza è:

$$m_B = \frac{y_2 - y_D}{\Delta x}$$

Derivazione Numerica (3)



Metodo delle differenze all'indietro: si consideri la retta tra i punti (x_1, y_1) e (x_D, y_D), la cui pendenza è data da:

$$m_A = \frac{y_D - y_1}{\Delta x}$$

Metodo delle differenze in avanti: si usi la retta tra (x_D, y_D) e (x_2, y_2), la cui pendenza è:

$$m_B = \frac{y_2 - y_D}{\Delta x}$$

Metodo delle differenze centrali: si assuma la retta tra (x_1, y_1) e (x_2, y_2), la cui pendenza è:

$$m_C = \frac{m_A + m_B}{2} = \frac{y_2 - y_1}{2\Delta x}$$

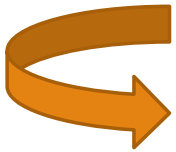
Derivazione Numerica (4)

MATLAB mette a disposizione la funzione diff() per la stima della derivata di una funzione. Tale funzione prende in ingresso un vettore di punti e restituisce la serie delle differenze tra punti adiacenti dell'input.

$$x = [x(1), x(2), \ldots, x(n)]$$

Derivazione Numerica (4)

MATLAB mette a disposizione la funzione diff() per la stima della derivata di una funzione. Tale funzione prende in ingresso un vettore di punti e restituisce la serie delle differenze tra punti adiacenti dell'input.

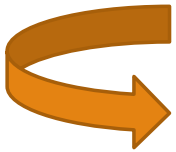


x = [x(1), x(2), …, x(n)]

diff(x) = [x(2) – x(1), x(3) – x(2), … x(n) – x(n-1)]

Derivazione Numerica (4)

MATLAB mette a disposizione la funzione diff() per la stima della derivata di una funzione. Tale funzione prende in ingresso un vettore di punti e restituisce la serie delle differenze tra punti adiacenti dell'input.



x = [x(1), x(2), …, x(n)]

diff(x) = [x(2) – x(1), x(3) – x(2), … x(n) – x(n-1)]

Utilizziamo diff() per la stima della derivata di $f(x) = \sqrt{x}$ nel punto x = 4.

La risoluzione esatta ci dice che $f'(x) = \frac{1}{2\sqrt{x}}$

Pertanto f'(4) = 0,25.

Derivazione Numerica (5)

In MATLAB la derivata di una funzione può essere stimata con diff(y)/diff(x), dove x e y sono le coordinate di n punti della funzione in esame per i tre metodi:

```
>> x1 = 2:.1:4; x2 = 4:.1:6; x3 = [x1 x2];
>> y1 = sqrt(x1); y2 = sqrt(x2); y3 = [y1 y2];
```

Derivazione Numerica (5)

In MATLAB la derivata di una funzione può essere stimata con diff(y)/diff(x), dove x e y sono le coordinate di n punti della funzione in esame per i tre metodi:

```
>> x1 = 2:.1:4; x2 = 4:.1:6; x3 = [x1 x2];
>> y1 = sqrt(x1); y2 = sqrt(x2); y3 = [y1 y2];
```

Definisco i tre intervalli di punti (x,y)

Derivazione Numerica (5)

In MATLAB la derivata di una funzione può essere stimata con diff(y)/diff(x), dove x e y sono le coordinate di n punti della funzione in esame per i tre metodi:

```
>> x1 = 2:.1:4; x2 = 4:.1:6; x3 = [x1 x2];
>> y1 = sqrt(x1); y2 = sqrt(x2); y3 = [y1 y2];

>> diff1 = diff(y1)/diff(x1)
diff1 =    0.2929
```

Derivazione Numerica (5)

In MATLAB la derivata di una funzione può essere stimata con diff(y)/diff(x), dove x e y sono le coordinate di n punti della funzione in esame per i tre metodi:

```
>> x1 = 2:.1:4; x2 = 4:.1:6; x3 = [x1 x2];
>> y1 = sqrt(x1); y2 = sqrt(x2); y3 = [y1 y2];

>> diff1 = diff(y1)/diff(x1)
diff1 =    0.2929
```

Metodo delle differenze all'indietro – Errore pari a 0.0429.

Derivazione Numerica (5)

In MATLAB la derivata di una funzione può essere stimata con diff(y)/diff(x), dove x e y sono le coordinate di n punti della funzione in esame per i tre metodi:

```
>> x1 = 2:.1:4; x2 = 4:.1:6; x3 = [x1 x2];
>> y1 = sqrt(x1); y2 = sqrt(x2); y3 = [y1 y2];

>> diff1 = diff(y1)/diff(x1)
diff1 =    0.2929
>> diff2 = diff(y2)/diff(x2)
diff2 =    0.2247
```

Derivazione Numerica (5)

In MATLAB la derivata di una funzione può essere stimata con diff(y)/diff(x), dove x e y sono le coordinate di n punti della funzione in esame per i tre metodi:

```
>> x1 = 2:.1:4; x2 = 4:.1:6; x3 = [x1 x2];
>> y1 = sqrt(x1); y2 = sqrt(x2); y3 = [y1 y2];

>> diff1 = diff(y1)/diff(x1)
diff1 =    0.2929
>> diff2 = diff(y2)/diff(x2)
diff2 =    0.2247
```

Metodo delle differenze in avanti – Errore pari a 0.0253.

Derivazione Numerica (5)

In MATLAB la derivata di una funzione può essere stimata con diff(y)/diff(x), dove x e y sono le coordinate di n punti della funzione in esame per i tre metodi:

```
>> x1 = 2:.1:4; x2 = 4:.1:6; x3 = [x1 x2];
>> y1 = sqrt(x1); y2 = sqrt(x2); y3 = [y1 y2];

>> diff1 = diff(y1)/diff(x1)
diff1 =    0.2929
>> diff2 = diff(y2)/diff(x2)
diff2 =    0.2247
>> diff3 = diff(y3)/diff(x3)
diff3 =    0.2588
```

Derivazione Numerica (5)

In MATLAB la derivata di una funzione può essere stimata con diff(y)/diff(x), dove x e y sono le coordinate di n punti della funzione in esame per i tre metodi:

```
>> x1 = 2:.1:4; x2 = 4:.1:6; x3 = [x1 x2];
>> y1 = sqrt(x1); y2 = sqrt(x2); y3 = [y1 y2];

>> diff1 = diff(y1)/diff(x1)
diff1 =    0.2929
>> diff2 = diff(y2)/diff(x2)
diff2 =    0.2247
>> diff3 = diff(y3)/
diff3 =    0.2588
```

Metodo delle differenze centrali – Errore commesso pari a 0.0088.

Calcolo Simbolico (1)

Per calcolo simbolico si intende la manipolazione di variabili e la risoluzione di espressioni ed equazioni in cui sono trattate variabili di tipo simbolico, senza considerarne un ipotetico valore numero.

Grazie al Symbolic Math Toolbox, Matlab ci fornisce un set di istruzioni per il calcolo simbolico, che si basa sull'uso di variabili con sym e syms, che verranno impiegate per la definizione di un' espressione simbolica.

```
>> syms x y z
```

oppure

```
>> x = sym('x');
```

Calcolo Simbolico (1)

Per calcolo simbolico si intende la manipolazione di variabili e la risoluzione di espressioni ed equazioni in cui sono trattate variabili di tipo simbolico, senza considerarne un ipotetico valore numero.

Grazie al Symbolic Math Toolbox, Matlab ci fornisce un set di istruzioni per il calcolo simbolico, che si basa sull'uso di variabili con sym e syms, che verranno impiegate per la definizione di un' espressione simbolica.

```
>> syms x y z
```

Crea nel workspace le variabili simboliche x, y e z.

oppure

```
>> x = sym('x');
```

Calcolo Simbolico (1)

Per calcolo simbolico si intende la manipolazione di variabili e la risoluzione di espressioni ed equazioni in cui sono trattate variabili di tipo simbolico, senza considerarne un ipotetico valore numero.

Grazie al Symbolic Math Toolbox, Matlab ci fornisce un set di istruzioni per il calcolo simbolico, che si basa sull'uso di variabili con sym e syms, che verranno impiegate per la definizione di un' espressione simbolica.

```
>> syms x y z
```

oppure

```
>> x = sym('x');
```

Metodo alternativo per creare una variabile simbolica x.

Calcolo Simbolico (2)

Una volta definite una serie di variabili simboliche, possiamo definire una espressione in termini funzionali. Ipotizziamo di voler rappresentare la seguente funzione:

$$f(x,y) = \frac{1}{e^{\sin(x^2+y^2)}}$$

```
>> syms x y
>> f = 1/exp(sin(x^2+y^2))
```

A questo punto possiamo effettuare le tipiche operazioni di calcolo simbolico, come derivate, integrali e limiti.

Calcolo Simbolico (3)

- Per la derivazione si usa la funzione diff() passando come ingresso la funzione simbolica, la variabile simbolica di derivazione e l'ordine di derivazione.

```
>> syms x y
>> f = 1/exp(sin(x^2+y^2))
f = exp(-sin(x^2 + y^2))
>> df = diff(f, x, 1)
df = -2*x*exp(-sin(x^2 + y^2))*cos(x^2 + y^2)
```

Calcolo Simbolico (4)

- Per il calcolo dell'integrale indefinito della funzione simbolica f(x,y) si impiega la funzione MATLAB int che richiede come input la funzione da integrare e la variabile di integrazione.

```
>> syms a b
>> s = sin(a * b);
>> int(s, b)
ans = -cos(a*b)/a
```

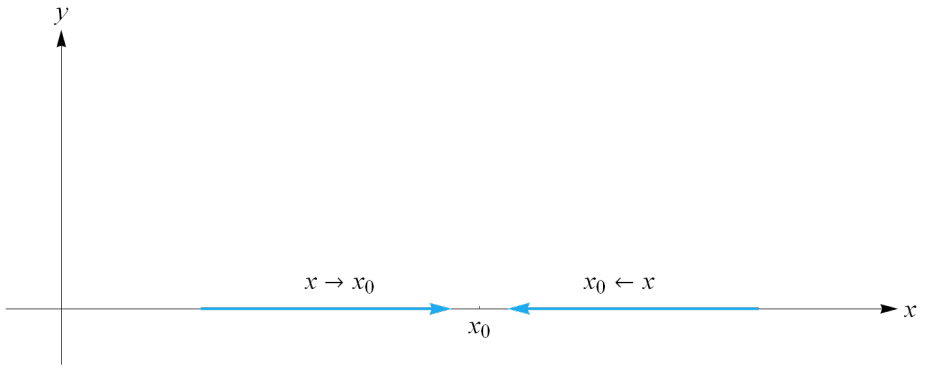
- È possibile fornire anche degli estremi di integrazione.

```
>> int(s, b, 0 ,2 * pi/a)
ans = 0
```

Calcolo Simbolico (5)

- Per il calcolo del limite di una funzione simbolica è possibile usare la funzione MATLAB limit(), che richiede in input la funzione, la variabile di cui calcolare il limite, il valore a cui il limite deve tendere e il verso del limite, se è da destra si passa 'right', mentre se è da sinistra si passa 'left'.

$$\begin{cases} x \to x_0^-, & \text{da sinistra} \\ x \to x_0^+, & \text{da destra} \end{cases}$$



Calcolo Simbolico (6)

$$\lim_{x \to 3^-} \frac{1-2x}{9-x^2}$$

```
>> syms x
>> f = (1- 2*x)/(9 - x^2);
>> limit(f, x, 3, 'left')
ans = -Inf
```

$$\lim_{x \to 3^-} \frac{1-2x}{9-x^2} = \frac{-5}{0^+} = -\infty$$

Calcolo Simbolico (7)

$$\lim_{x \to 2^+} \frac{7 - x}{x - 2}$$

```
>> syms x
>> f = (7 - x)/(x - 2);
>> limit(f, x, 2, 'right')
 ans = Inf
```

$$\lim_{x \to 2^+} \frac{7 - x}{x - 2} = \frac{5}{0^+} = +\infty$$

Valutare Espressioni Simboliche

In MATLAB è possibile valutare espressioni simboliche con la funzione subs, nel modo seguente:

```
>> syms x
>> y = x^2
y = x^2
```

Definiamo un'espressione simbolica e un valore da usare per la sua valorizzazione.

Valutare Espressioni Simboliche

In MATLAB è possibile valutare espressioni simboliche con la funzione subs, nel modo seguente:

```
>> syms x
>> y = x^2
y = x^2
>> y(2)
Index exceeds matrix dimensions.Error in
sym/subsref (line 805)                R_tilde
= builtin('subsref',L_tilde,Idx);
```

Se proviamo a valorizzare l'espressione con la sintassi di una chiamata funzionale otteniamo un errore.

Valutare Espressioni Simboliche

In MATLAB è possibile valutare espressioni simboliche con la funzione subs, nel modo seguente:

```
>> syms x
>> y = x^2
y = x^2
>> y(2)
Index exceeds matrix dimensions.Error in
sym/subsref (l
= builtin('sub
>> subs(y, 2)
ans = 4
```

La soluzione corretta è quella di impiegare subs() dopo aver valorizzato la variabile simbolica. Tale funzione sostituisce le variabili simboliche con i valori numerici passati come secondo input. Se tale sostituzione non produce uno scalare, ma un'espressione, utilizzare vpa() con input l'espressione da risolvere.

Valutare Espressioni Simboliche

In MATLAB è possibile valutare espressioni simboliche con la funzione subs, nel modo seguente:

```
>> y = sqrt(1-x)
y = (1 - x)^(1/2)
>> z = subs(y, 4)
z = 3^(1/2)*1i
>> vpa(z, 3)
ans = 1.73i
```

Studio di una funzione (1/8)

Una volta che si è definita una funzione per mezzo di un handle, è possibile studiarne le caratteristiche matematiche. Il primo aspetto è quello di studiare gli zeri di una funzione, ovvero dei punti x in cui f(x) = 0.

Nel caso di un polinomio si usa roots:

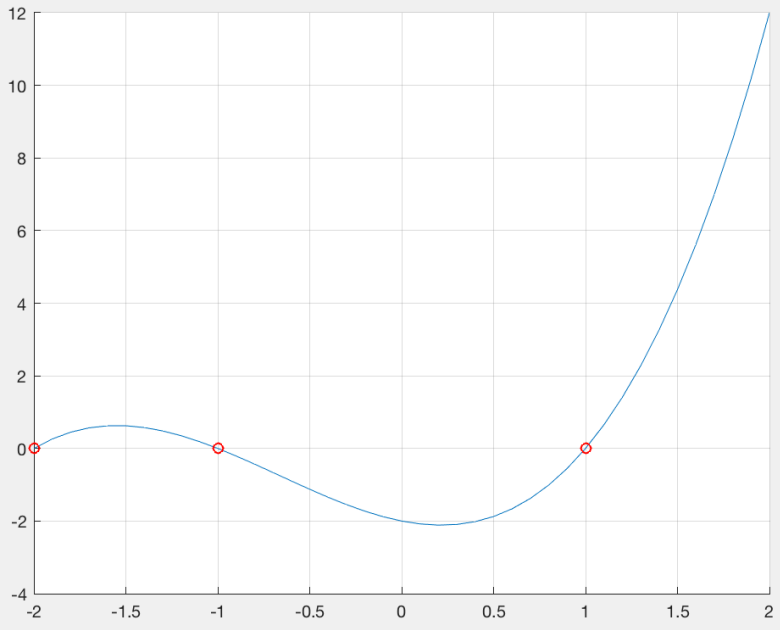
```
>> p = [1 2 -1 -2];
>> r = roots(p);
>> x = -2:0.1:2;
>> y = polyval(p,x);
>> hold on
>> plot(x,y)
>> plot(r,0,'or')
>> grid on
```

Studio di una funzione (1/8)

Una volta che si è definita una funzione per mezzo di un handle, è possibile studiarne le caratteristiche matematiche. Il primo aspetto è quello di studiare gli zeri di una funzione, ovvero dei punti x in cui f(x) = 0.

Nel caso di un polinomio si usa roots:

```
>> p = [1 2 -1 -2];
>> r = roots(p);
>> x = -2:0.1:2;
>> y = polyval(p,x);
>> hold on
>> plot(x,y)
>> plot(r,0,'or')
>> grid on
```



Studio di una funzione (2/8)

Non è così per funzioni più complesse.

Se indichiamo con un handle di nome fun, la funzione di cui vogliamo calcolare gli zeri.

```
>> fun = @(x) exp(x)-2;
```

Il comando fzero offerto da Matlab consente di calcolare uno zero di una funzione nelle vicinanze di un punto x0. Questa funzione si basa su un algoritmo numerico che sfrutta il metodo di bisezione.

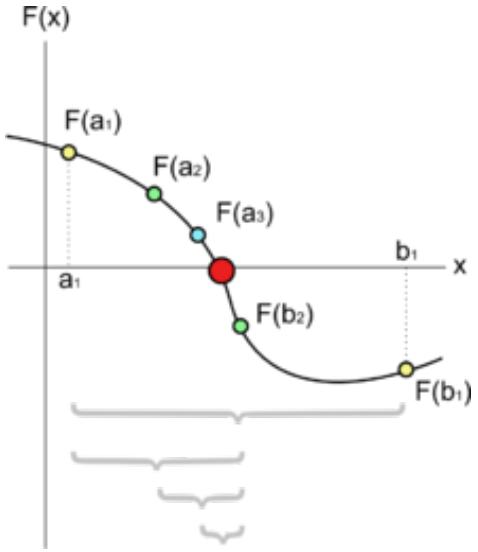
Risulta conveniente individuare prima (ad es., sul grafico della funzione) l'intervallo in cui lo zero potrebbe trovarsi, e poi utilizzare il comando fzero. Il comando non funziona se si omette di specificare il punto x0 o un intervallo.

Studio di una funzione (3/8)

Se indichiamo con un handle di nome fun, la funzione di cui vogliamo calcolare gli zeri.

```
>> fun = @(x) (x-2).*(x.^2+.5);
```

Il comando fzero calcola uno zero o radice di una funzione in un intervallo o nelle vicinanze di un punto x0.

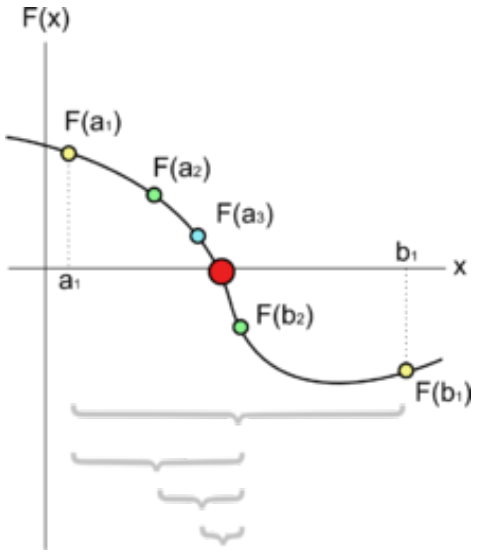


Studio di una funzione (3/8)

Se indichiamo con un handle di nome fun, la funzione di cui vogliamo calcolare gli zeri.

```
>> fun = @(x) (x-2).*(x.^2+.5);
```

Il comando fzero calcola uno zero o radice di una funzione in un intervallo o nelle vicinanze di un punto x0.



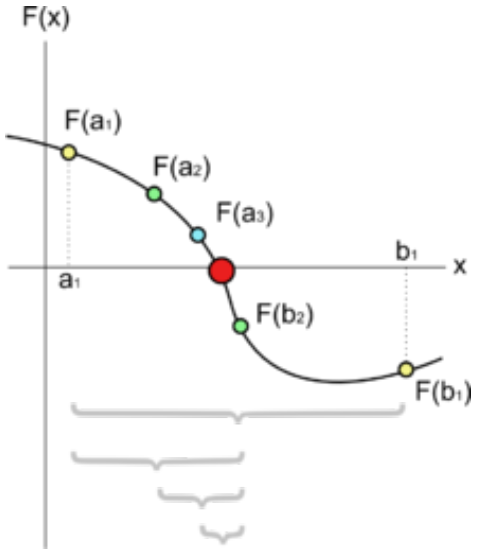
Secondo il Teorema degli zeri per le funzioni continue, l'intervallo deve essere scelto tale che il valore assunto dalla funzione nei due estremi deve avere segno diverso:
f(a)<0< f(b) o f(b)<0< f(a).

Studio di una funzione (3/8)

Se indichiamo con un handle di nome fun, la funzione di cui vogliamo calcolare gli zeri.

```
>> fun = @(x) (x-2).*(x.^2+.5);
```

Il comando fzero calcola uno zero o radice di una funzione in un intervallo o nelle vicinanze di un punto x0.



Questa funzione si basa sul metodo di bisezione.

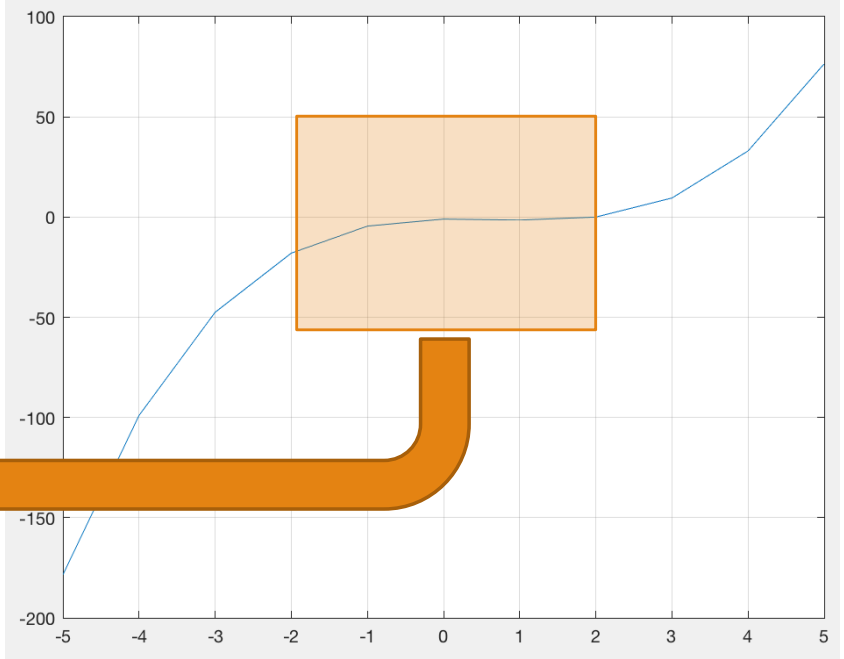
Si calcola il valore della funzione nel punto medio. Se risulta che la funzione in quel punto ritorna 0, allora si è trovata la radice. Altrimenti, tra i due intervalli alla sinistra e la destra del punto medio si sceglie quello ai cui estremi la funzione assume valori di segno opposto. Si ripete il procedimento di dimezzamento.

Studio di una funzione (4/8)

Risulta quindi conveniente individuare prima (ad esempio, sul grafico della funzione) l'intervallo in cui lo zero potrebbe trovarsi e poi utilizzare il comando fzero. Il comando non funziona se si omette di specificare il punto x0 o un intervallo:

```
>> y = -5:5;
>> plot(y, fun(y));
>> grid on


>> radice = fzero(
          fun, [-2 2]);
```



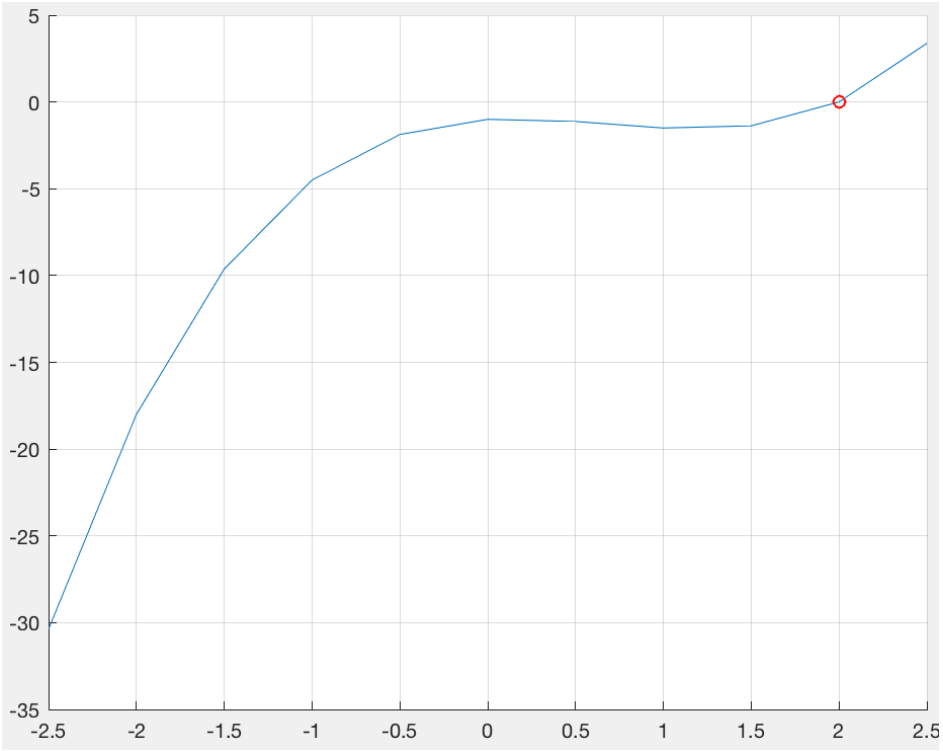
Studio di una funzione (5/8)

Visualizziamo la radice sul grafico della funzione:

```
>> y = -2.5:.5:2.5;
>> hold on
>> plot(y, fun(y))
>> plot(radice,
   fun(radice), 'or')
>> grid on
```

In alternativa:

```
>> radice = fzero(
fun, 0);
radice =     2
```



Studio di una funzione (6/8)

Il comando fzero è anche utile per risolvere equazioni non omogenee. Consideriamo la risoluzione dell'equazione

$$xe^x = 1.$$

```
>> sol = fzero('x.*exp(x)-1' ,-0.5)
sol =    0.5671
```

Inoltre fzero può essere utilizzato per trovare i punti di intersezione tra i grafici di due funzioni. Consideriamo due funzioni ln(x) e e^{-x}; l'equazione $\ln(x)=e^{-x}$, si risolve come $\ln(x)-e^{-x}= 0$.

```
>> int = fzero('log(x)-exp(-x)' ,1.5)
int =    1.3098
```

Studio di una funzione (7/8)

Ipotizziamo di voler determinare i punti di massimo e di minimo di una funzione. Il comando fminbnd (bounded minimization) consente tale calcolo in un determinato intervallo, ed è analoga, anche se non uguale, a fzero.

- fminbnd(fun, a, b), dove a<b sono gli estremi dell'intervallo in cui si vuole minimizzare la funzione e resituisce il punto di minimo.
- [xmin, min] = fminbnd(fun, a, b) restituisce il punto di minimo e il minimo.

Per più punti di minimo, fminbnd si limita al primo.

Il calcolo dei massimi locali può essere effettuato con lo stesso comando, ricordando che max(f) =−min(−f).

Studio di una funzione (8/8)

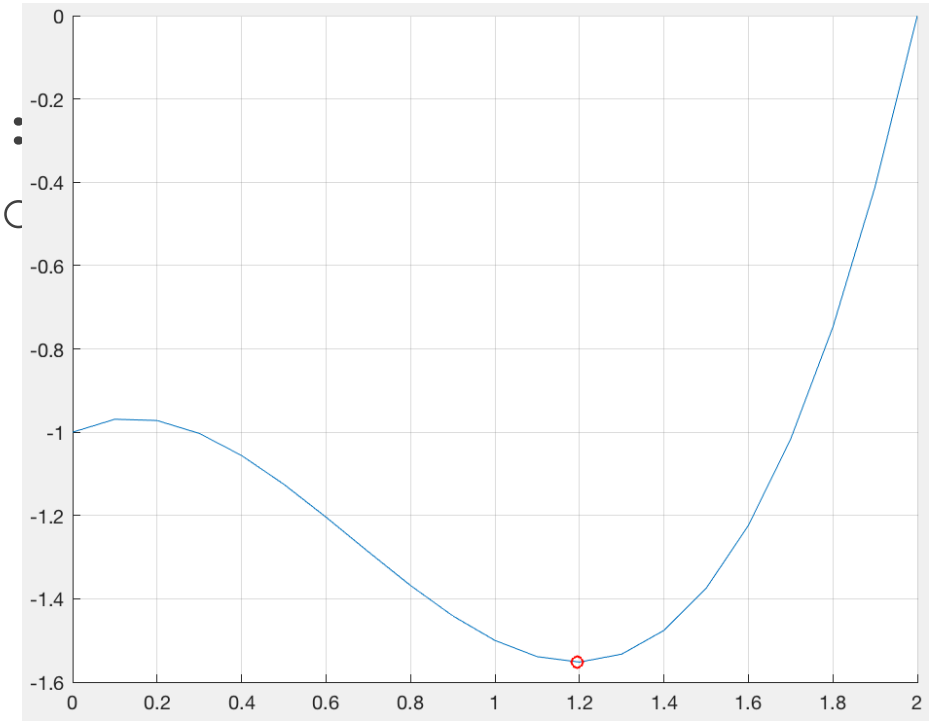
Esempio:

```
>> fun = @(x) (x-2).*(x.^2+.5);
>> [xmin, ymin]=fminbnd(fun, 0, 2);
>> hold on
>> plot(0:.1:2, fun(0:.1:2))
>> plot(xmin, ymin, 'or')
>> grid on
```

Studio di una funzione (8/8)

Esempio:

```
>> fun = @(x) (x-2).*(x.^2+.5);
>> [xmin, ymin]=fminbnd(fun, 0, 2);
>> hold on
>> plot(0:.1:2, fun(0:
>> plot(xmin, ymin, 'c
>> grid on
```



MuPAD (1)

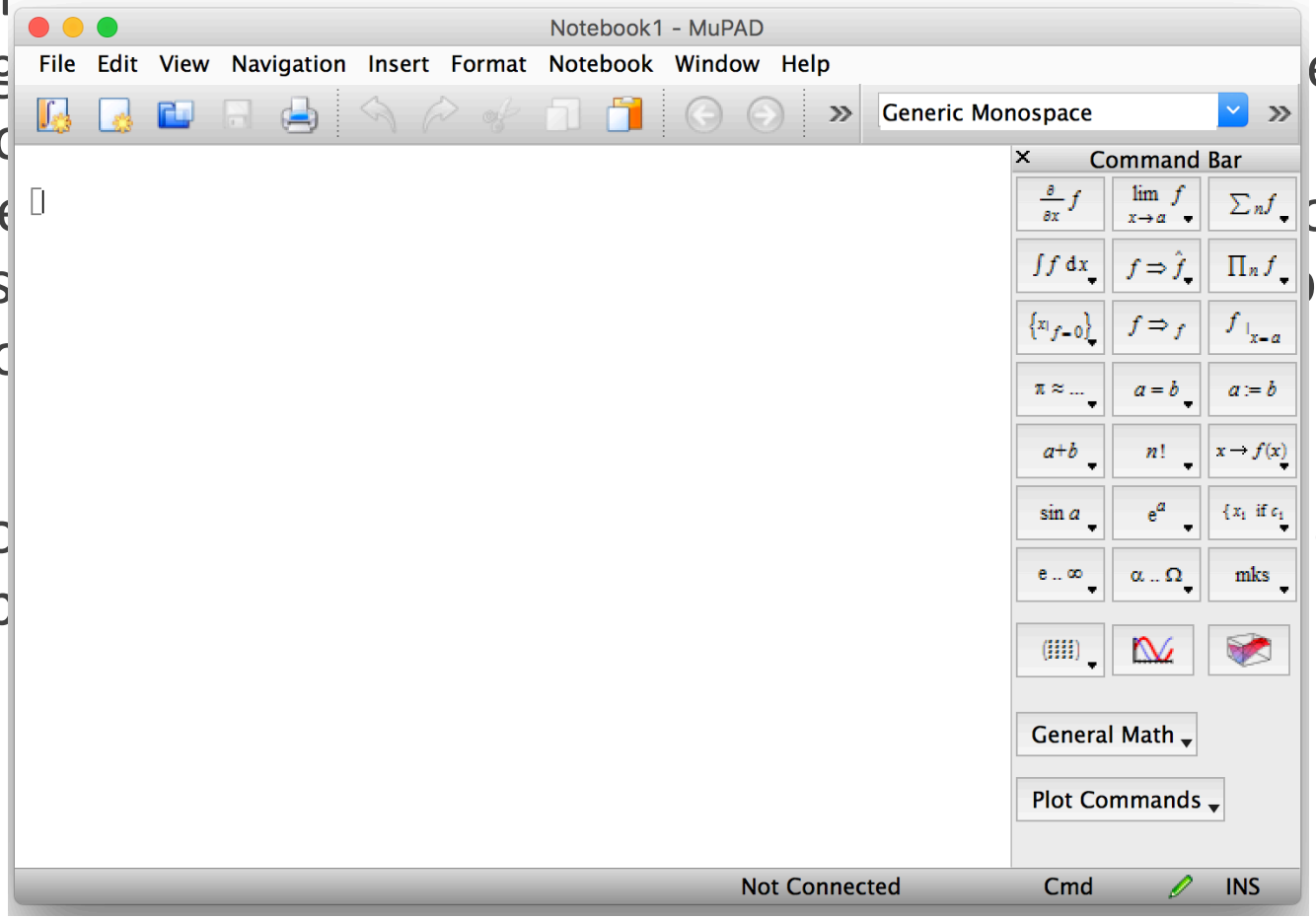
MuPAD è un package di MATLAB che offre un ricco ventaglio di funzionalità per il calcolo simbolico, ed è incluso nel toolbox Symbolic Math. Questo pacchetto offre un'interfaccia con un editore per l'inserimento di espressioni simboli e l'esecuzione di operazioni per il calcolo simbolico.

Per poter invocare questo editor basta digitare mupad sul prompt dei comandi.

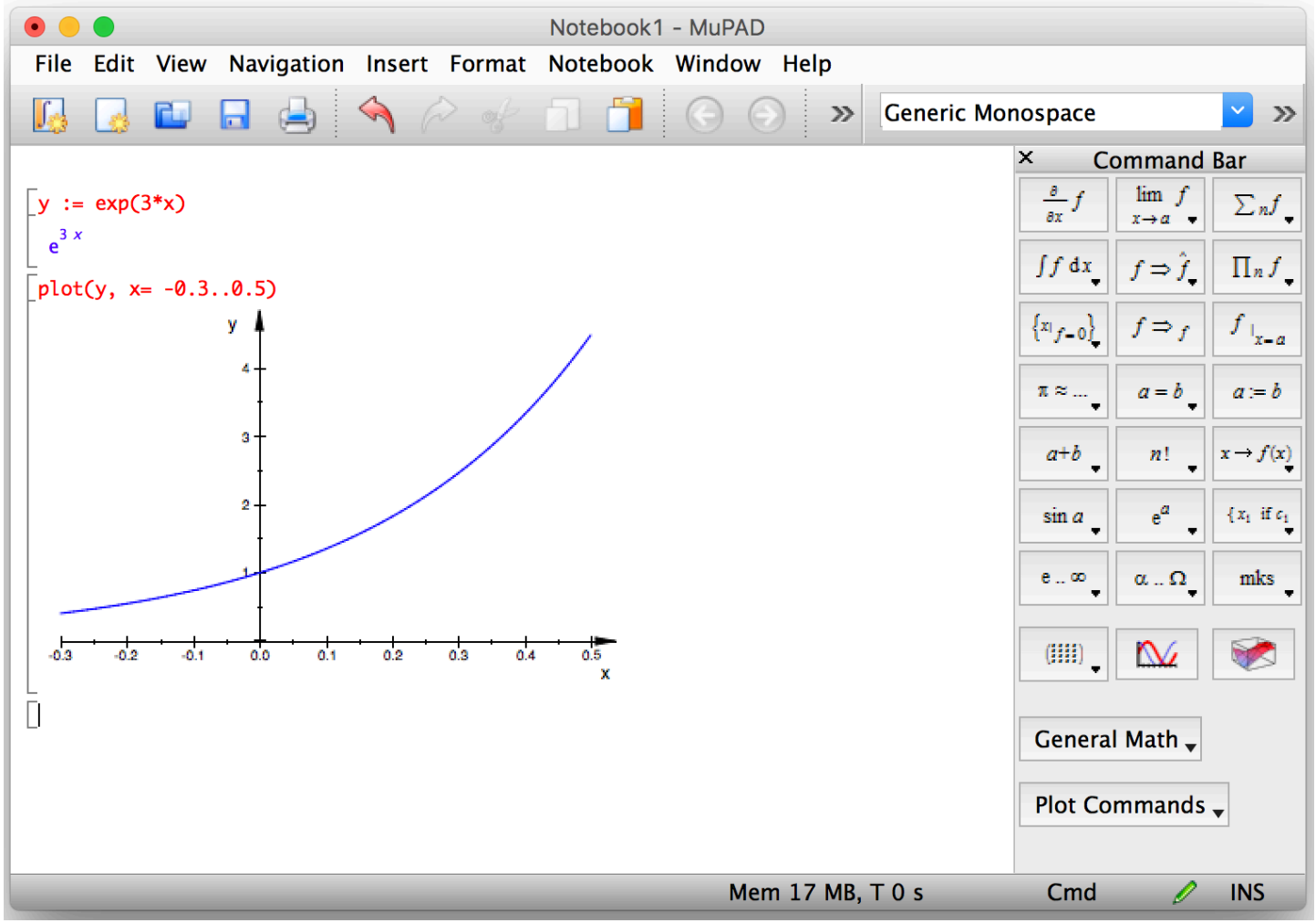
MuPAD (1)

MuPAD è un package di MATLAB che offre un ricco
ventag ed è
inclus offre
un'inte o di
espres er il
calcol

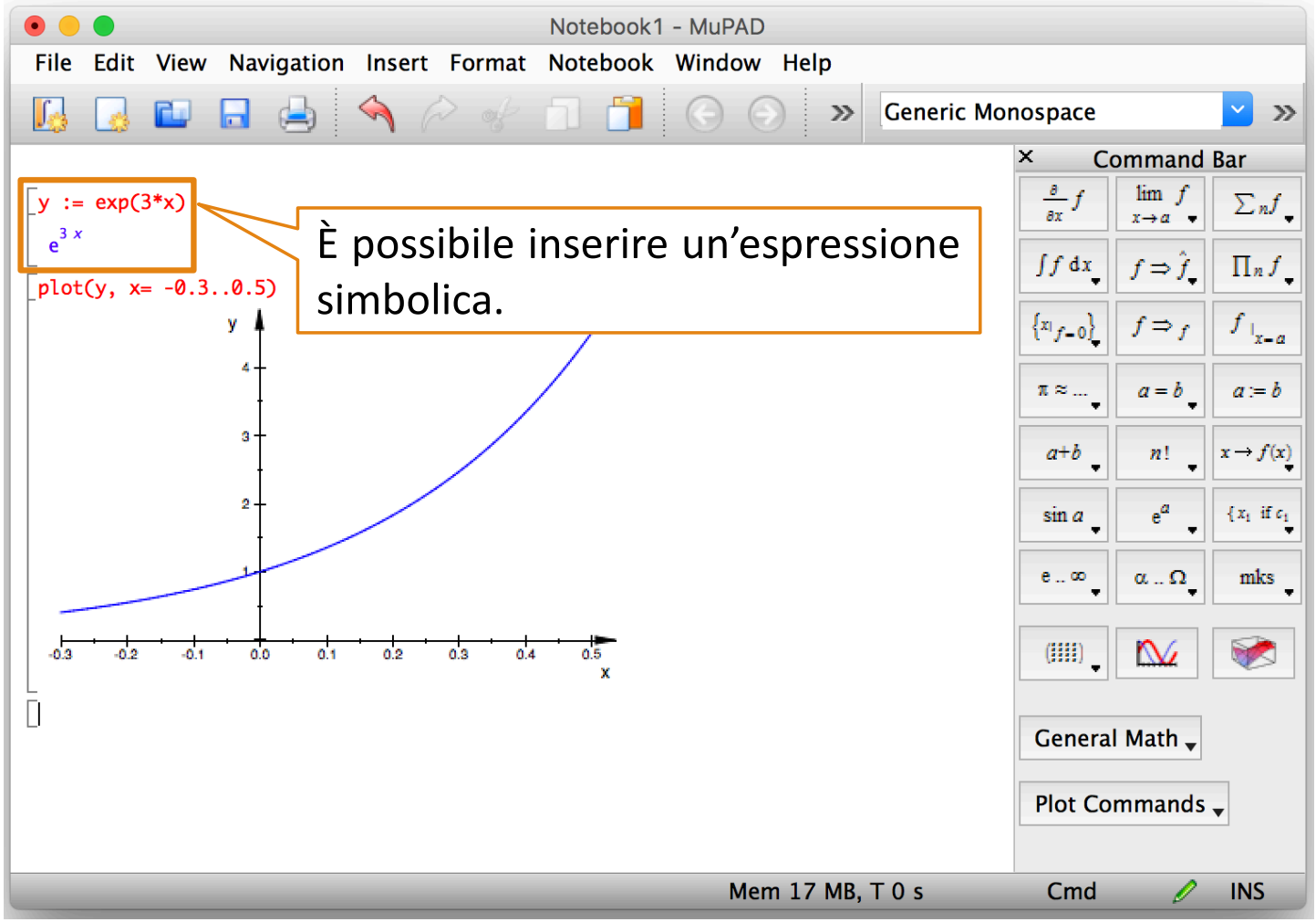
Per pc d sul
promp



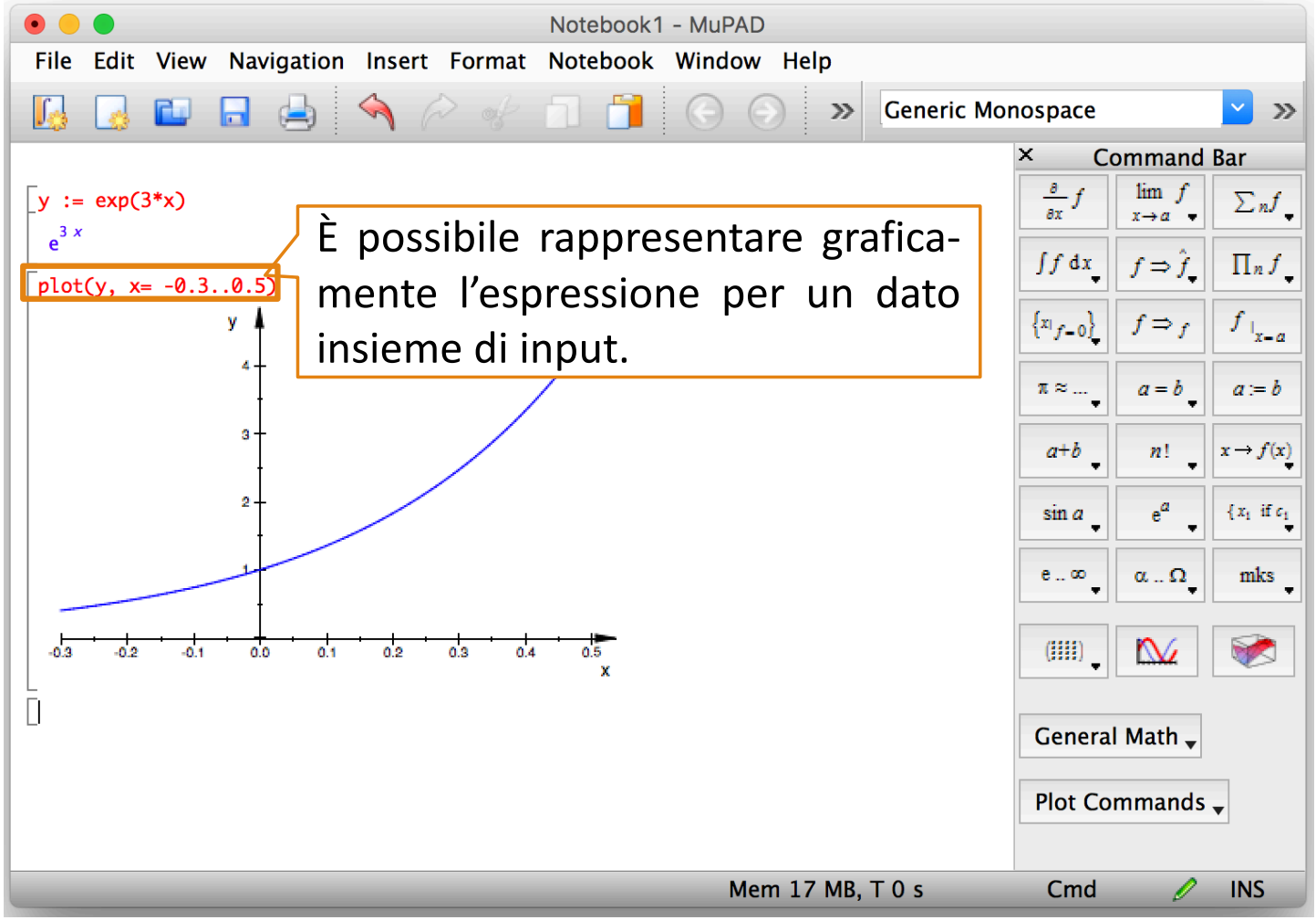
MuPAD (2)


Notebook1 - MuPAD
File Edit View Navigation Insert Format Notebook Window Help
Generic Monospace
Command Bar
y := exp(3*x)
e^(3 x)
plot(y, x= -0.3..0.5)
y
x
-0.3 -0.2 -0.1 0.0 0.1 0.2 0.3 0.4 0.5
1 2 3 4
General Math
Plot Commands
Mem 17 MB, T 0 s
Cmd
INS

MuPAD (2)


Notebook1 - MuPAD
File Edit View Navigation Insert Format Notebook Window Help
Generic Monospace
Command Bar
y := exp(3*x)
e^(3 x)
plot(y, x= -0.3..0.5)
È possibile inserire un'espressione simbolica.
y
x
-0.3 -0.2 -0.1 0.0 0.1 0.2 0.3 0.4 0.5
1 2 3 4
General Math
Plot Commands
Mem 17 MB, T 0 s
Cmd
INS

MuPAD (2)


Notebook1 - MuPAD
File Edit View Navigation Insert Format Notebook Window Help
Generic Monospace
Command Bar
y := exp(3*x)
e^3x
plot(y, x= -0.3..0.5)
È possibile rappresentare graficamente l'espressione per un dato insieme di input.
y
x
-0.3 -0.2 -0.1 0.0 0.1 0.2 0.3 0.4 0.5
1 2 3 4
General Math
Plot Commands
Mem 17 MB, T 0 s
Cmd
INS

MuPAD (3)

```
y := exp(3*x)
```
$$e^{3x}$$
```
y(3)
```
$$(e^{3x})_{(3)}$$
```
delete(y)
y
```
$$y$$
```
y := exp(3*x)
```
$$e^{3x}$$
```
y
```
$$e^{3x}$$

L'editor consente di inserire comandi per il calcolo di espressioni numeriche. Per il calcolo simbolico si ha una differenza tra espressioni e funzioni:

- per assegnare ad un identificativo un'espressione bisogna impiegare l'operatore ':=', mentre per eliminarlo si usa la funzione delete() passando in input l'identificativo da cancellare;
- l'espressione è diversa da una funzione perché non è possibile la sua valutazione per un dato valore delle sue variabili simboliche;

MuPAD (4)

```
f:= x -> x^2+3*x+7
```

$$x \to x^2 + 3x + 7$$

```
f(x)
```

$$x^2 + 3x + 7$$

```
f(y)
```

$$3e^{3x} + e^{6x} + 7$$

```
f(2.5)
```

$$20.75$$

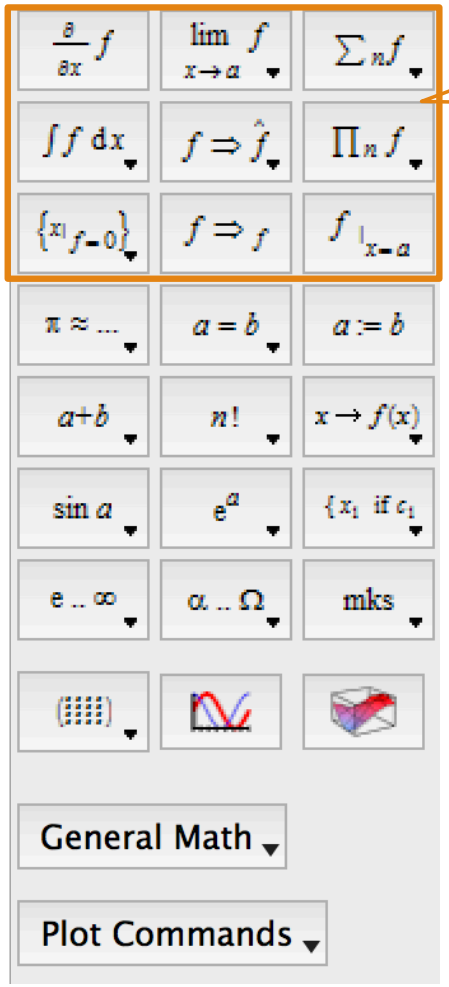
- è possibile definire una funzione nel modo seguente:

  <nome> := <incognita> -> <espressione>

- è possibile valutare una funzione per un dato valore delle sue incognite, oppure combinare più funzioni ed espressioni.

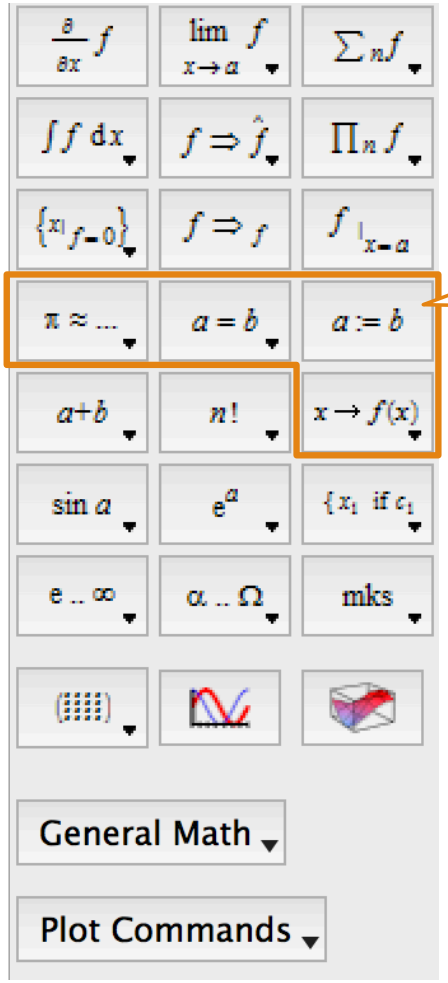
MuPAD dispone di vari comandi a supporto dell'analisi di espressioni e funzioni. Tali comandi possono essere invocati da riga di comando, oppure possono essere scelti per mezzo dell'interfaccia.

MuPAD (5)



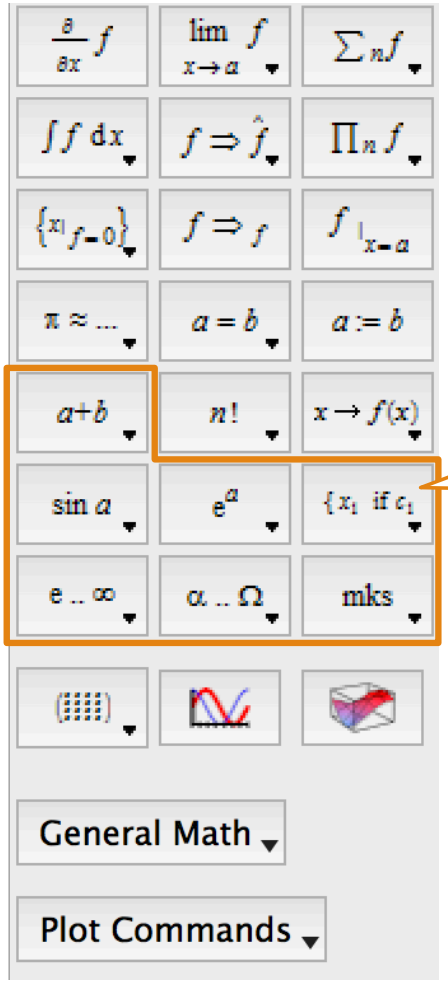
Operazioni di calcolo simboli come derivate, limiti, integrali etc.

MuPAD (5)



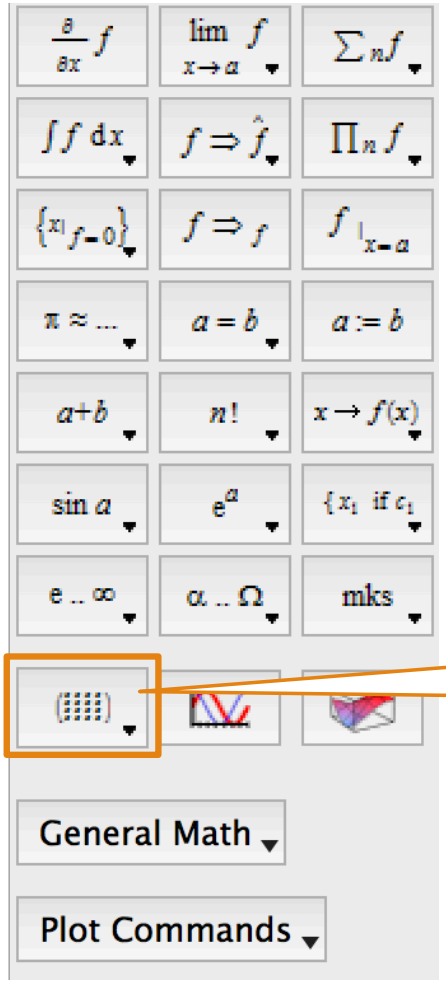
Approssimazioni, assegnazioni e dichiarazioni di espressioni e funzioni.

MuPAD (5)



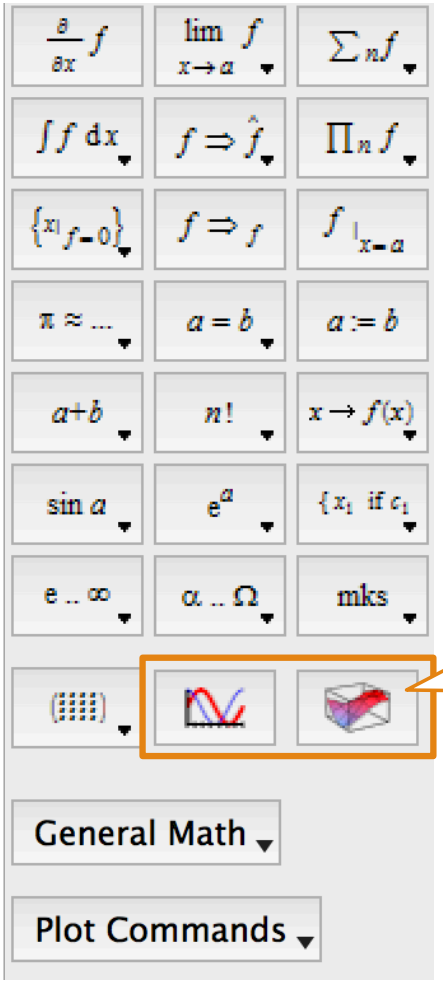
Principali funzioni ed operazioni built-in di MATLAB, simboli ed unità di misura fisiche.

MuPAD (5)



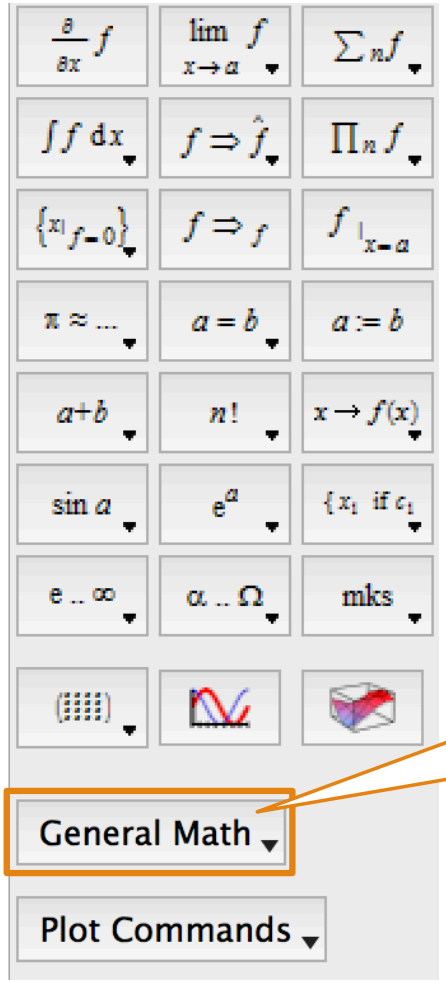
Definizioni di matrici.

MuPAD (5)



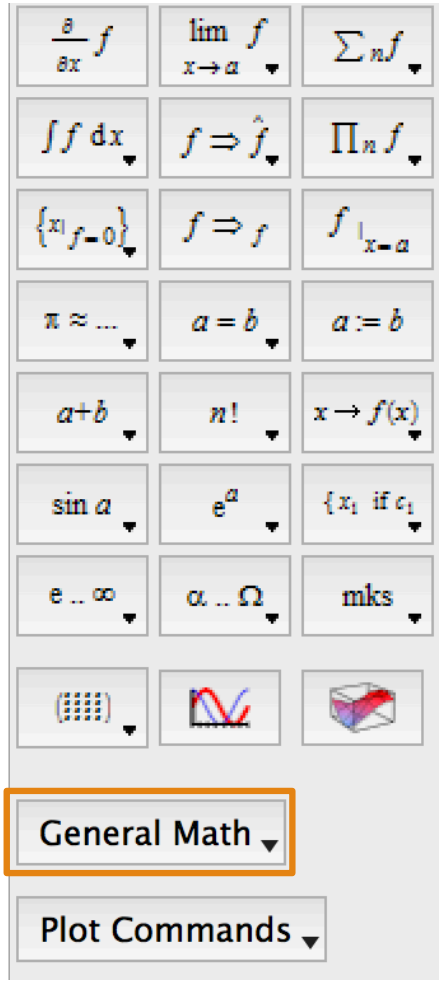
Funzioni per i diagrammi 2D e 3D di espressioni e funzioni.

MuPAD (5)



Principali operazioni da invocare su funzioni ed espressioni.

MuPAD (5)



- La funzione expand() espande una espressione p utilizzando una serie di regole:

```
p1 := (x-5)^2+(y-3)^2
```

$$(x-5)^2+(y-3)^2$$

```
expand(p1)
```

$$x^2-10\,x+y^2-6\,y+34$$

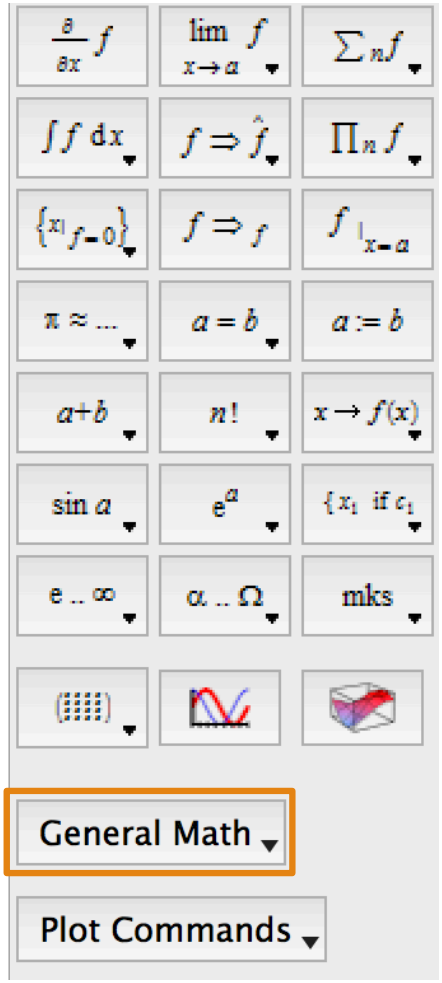
```
p1 := sin(x+y)
```

$$\sin(x+y)$$

```
expand(p1)
```

$$\cos(x)\sin(y)+\cos(y)\sin(x)$$

MuPAD (5)



- La funzione expand() espande una espressione p utilizzando una serie di regole;
- La funzione factor() fattorizza un'espressione data in input:

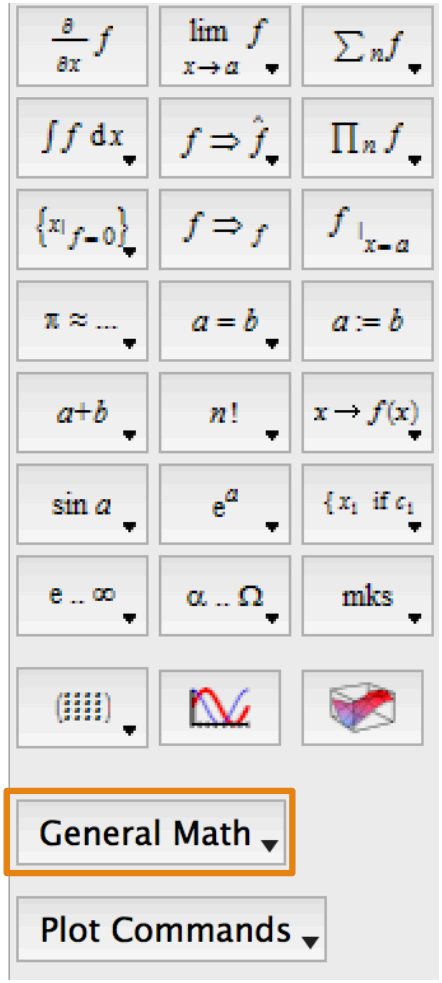
```
p1 := x^2-1
```

$$x^2 - 1$$

```
factor(p1)
```

$$(x - 1)\,(x + 1)$$

MuPAD (5)



- La funzione expand() espande una espressione p utilizzando una serie di regole;
- La funzione factor() fattorizza un'espressione data in input;
- La funzione normal() trova un denominatore comune per le espressioni razionali:

```
p1 := x/(1+x)-3/(1-x)
```

$$\frac{x}{x+1}+\frac{3}{x-1}$$

```
normal(p1)
```

$$\frac{x^2+2x+3}{(x-1)(x+1)}$$

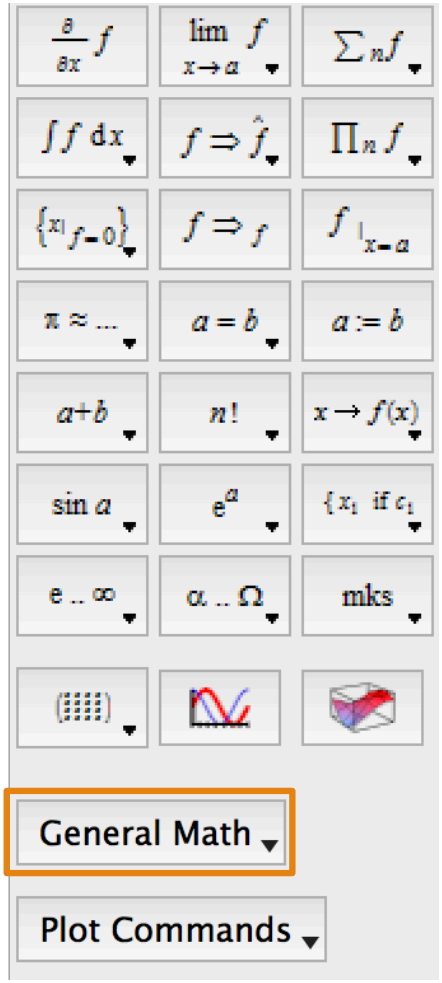
```
p1 := x^2/(x + y) - y^2/(x+y)
```

$$\frac{x^2}{x+y}-\frac{y^2}{x+y}$$

```
normal(p1)
```

$$x-y$$

MuPAD (5)



- La funzione expand() espande una espressione p utilizzando una serie di regole;
- La funzione factor() fattorizza un'espressione data in input;
- La funzione normal() trova un denominatore comune per le espressioni razionali;
- La funzione simplify() serve a semplificare un'espressione:

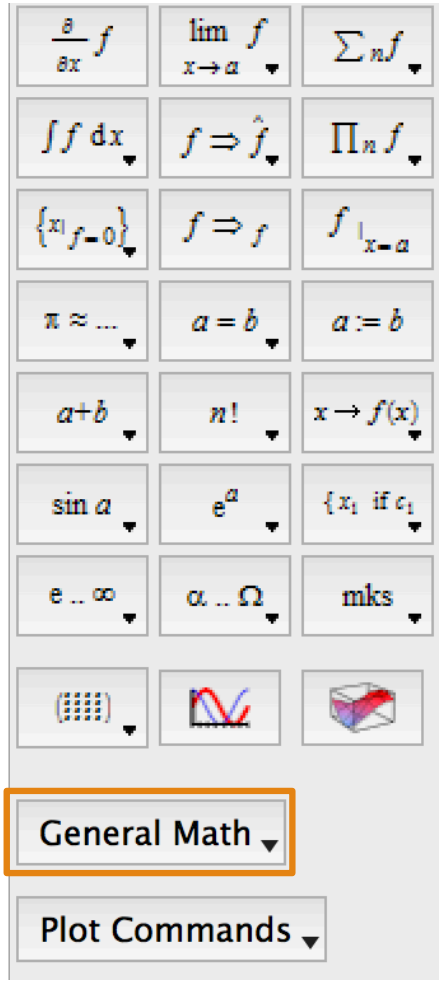
```
p1:= cos(x)*sin(y) + cos(y)*sin(x)
```

$$\cos(x)\sin(y)+\cos(y)\sin(x)$$

```
simplify(p1)
```

$$\sin(x+y)$$

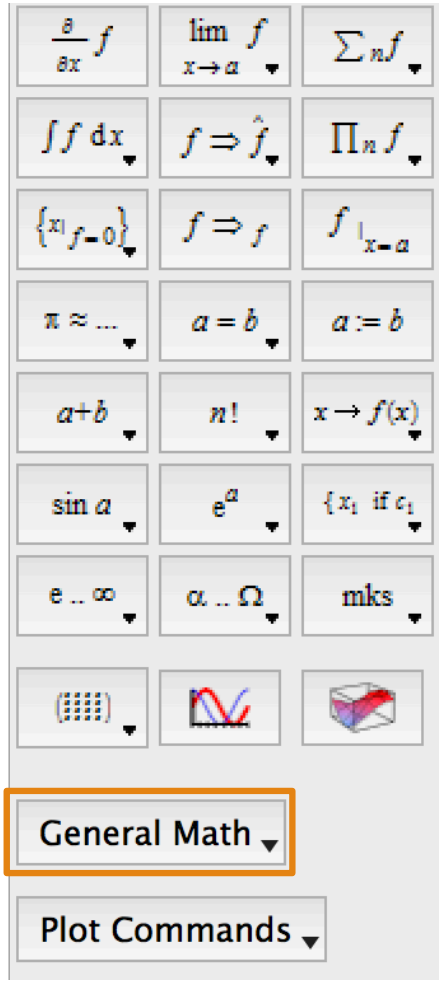
MuPAD (5)



- La funzione expand() espande una espressione p utilizzando una serie di regole;
- La funzione factor() fattorizza un'espressione data in input;
- La funzione normal() trova un denominatore comune per le espressioni razionali;
- La funzione simplify() serve a semplificare un'espressione;
- La funzione solve() serve a risolvere un'equazione simbolica:

```
equ := 5*x+25=0
  5 x + 25 = 0
solve(equ)
  {[x = -5]}
```

MuPAD (6)



- La funzione diff() calcola la derivata simbolica di una espressione:

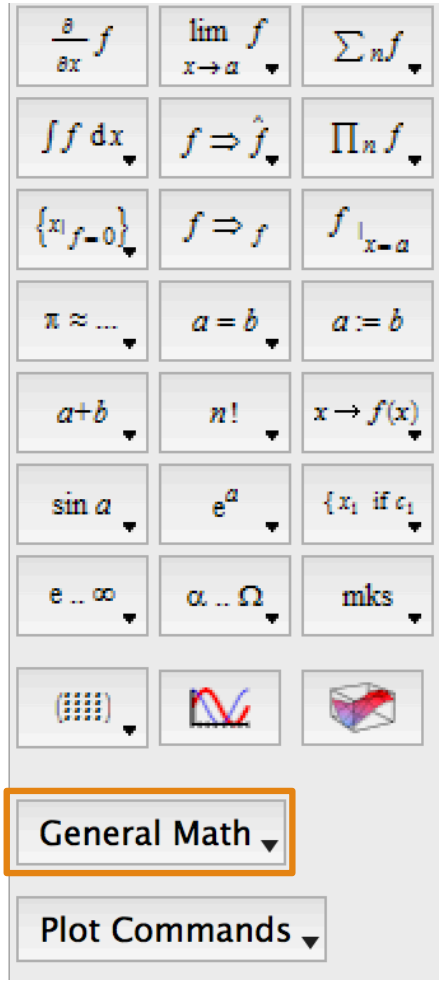
```
p1 := sin(x)^2
```

$$\sin(x)^2$$

```
d1 = diff(p1, x)
```

$$d1 = 2\cos(x)\sin(x)$$

MuPAD (6)



- La funzione diff() calcola la derivata simbolica di una espressione;
- La funzione int() computa l'integrale di un'espressione simbolica:

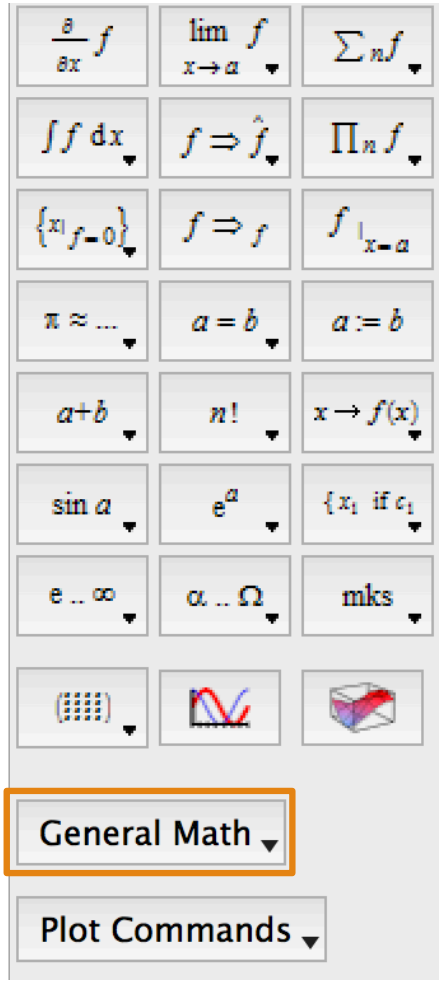
```
p1 := 1/(x-1)
```

$$\frac{1}{x-1}$$

```
int1 = int(p1, x)
```

$$int1 = \ln(x-1)$$

MuPAD (6)



- La funzione diff() calcola la derivata simbolica di una espressione;
- La funzione int() computa l'integrale di un'espressione simbolica;
- La funzione sum() determina la sommatoria di una data espressione in un determinato intervallo:

```
s1 := x^2
```

$$x^2$$

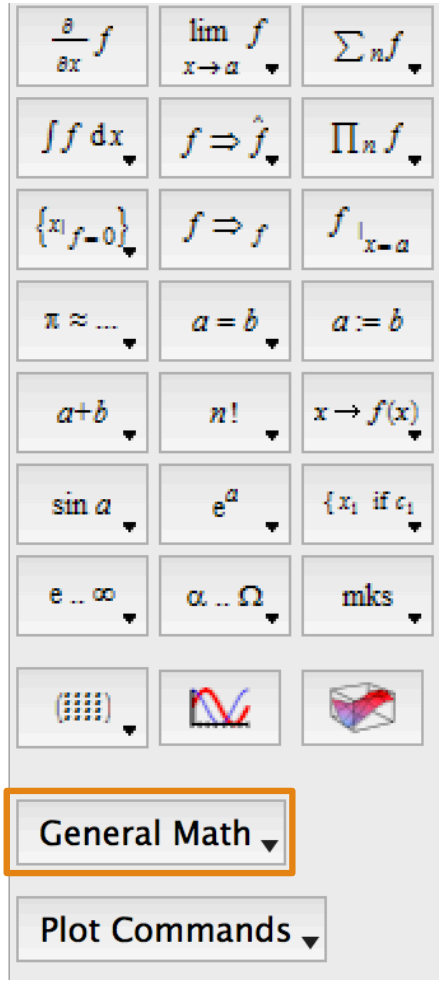
```
sum(s1, x = 0..10)
```

$$385$$

```
sum(s1, x = 0..n)
```

$$\frac{n\,(2\,n+1)\,(n+1)}{6}$$

MuPAD (6)



- La funzione diff() calcola la derivata simbolica di una espressione;
- La funzione int() computa l'integrale di un'espressione simbolica;
- La funzione sum() determina la sommatoria di una data espressione in un determinato intervallo;
- La funzione limit() serve a determinare il limite di una espressione simbolica:

```
f := (x-3)/(x^2-9)
```

$$\frac{x-3}{x^2-9}$$

```
limit(f, x=3)
```

$$\frac{1}{6}$$

```
limit(f, x=Inf, Left)
```

$$\frac{Inf-3}{Inf^2-9}$$

Riferimenti

- Capitolo 8
  - Paragrafi 1 [**Metodi matriciali per risolvere le equazioni lineari**], e 2 [**Il metodo della divisione a sinistra**].
- Capitolo 9
  - Paragrafi 1 [**Integrazione numerica**], e 2 [**Derivazione numerica**].
- Capitolo 11
  - Paragrafi 1-5 (cenni)