Sopravvivere con Mathematica

Breve guida all'uso del piu' famoso software per

fare matematica col computer.

M.Andreoli

Ottobre 2001, ver 0.1

Questo documento e' stato scritto utilizzando Mathematica V3.0 (della Wolfram Researchs), e non un normale Word−processor. Non intende essere completo ne' potrebbe: i manuali dedicati a Mathematica sono normalmente costituiti di centinaia e centinaia di pagine! Si suppone, percio', che lo studente abbia accesso ad una copia funzionante del software e che consulti la guida on−lineper tutti i dettagli lasciati in ombra.

INDICE


```
Get::noopen : Cannot open Default.nb.
```

Regole di input

■ Formattazione

Un documento di *Mathematica* (detto notebook) e' un insieme di celle, riconoscibili dalla graffa rettangolare qui a fianco. Puoi selezionare una cella cliccando sopra la graffa con il mouse. Per cancellare una intera cella usa: CANC o ALT−X.
Per creare una nuova cella, devi puntare e cliccare col mouse nel primo posto libero in fondo al documento.
Le celle possono essere formattare in modo simile a Word: seleziona la cella e prova i vari formati: da ALT−1and ALT−9.Il tasto INVIO serve solo per rompere una linea in due e non per avviare il calcolo. Per questo scopo, e' previsto SHIFT−INVIO.
Ti consiglio di utilizzare INVIO per rompere le espressioni complicate su piu' linee, in maniera che siano meglio leggibili.

■ Formule

Tutte le funzioni matematiche disponibili internamente a *Mathematica* vanno inserite con la prima lettera in maiuscolo. Per la lista dei parametri di input alla funzione non si usano le parentesi tonde (), ma quelle quadrate []. Le parentesi tonde servono SOLO per raccogliere piu' addendi nelle espressioni algebriche.
Quindi Sin[x] va bene, mentre sin(x) NON e' accettato a *Mathematica*.
Per tutte le altre operazioni algebriche, *Mathematica* segue la sintassi di altri linguaggi come il Pascal, etc: il prodotto con *, la divisione con /, la potenza con ^.
Le parentesi graffe {}, molto importanti in *Mathematica*, non servono per raccogliere termini come le (), ma per creare liste, vettori e matrici (vedi in seguito). Per ottenerle sulla tastiera, devi usare i codici ESCAPE: alt−123e alt−125.

Per introrre un prodotto come x*y, devi usare l'asterisco. Non puoi scrivere xy, altrimenti *Mathematica* pensa di avere a che fare con una singola variabile di nome "xy". Se pero' uno dei fattori e' un numero, puoi scrivere 2x, e non 2*x, perche' in questo caso *Mathematica* non puo' equivocare.

Puoi assegnare un'espressione ad un altra variabile (per usarla successivamente) con
l'operatore =, come nel linguaggio Basic.

■ Help on line

Puoi ottenere informazioni su una funzione, scrivendola, marcandola col mouse e pigiando F1

Le tipiche funzioni matematiche

Le funzioni trigonometriche si indicano allo stesso modo: Sin[], Cos[], ma la tangente va indicata con Tan[]. La radice quadrato e' Sqrt[]. La funzione esponenziale viene indicata con Exp[x], come sulle macchinette calcolatrici.
Il logaritmo e' sempre quello naturale: Log[]

```
Log[Exp[5]]
```

```
5
```

Ma puoi avere i logaritmi in altre basi, specificando la base:

```
Log[10, 1000] (* logaritmo di 1000, in base 10 *)
```

```
3
```

In realta', Mathematica usa la nota formula di passaggio tra basi:

```
Log[b, x]
```

$$\frac{\text{Log}[x]}{\text{Log}[b]}$$

Algebra

■ Manipolazioni algebriche

Normalmente, *Mathematica* evita di sviluppare i calcoli algebrici, se non esplicitamente richiesto. Ecco alcuni esempi di tipiche operazioni algebriche:

```
(a - b)^5 // Expand
```

$$a^5 - 5\,a^4\,b + 10\,a^3\,b^2 - 10\,a^2\,b^3 + 5\,a\,b^4 - b^5$$

```
1 + 2 x + x^2 // Factor
```

$$(1 + x)^2$$

```
1 / x + 2 / (1 - x) // Together
```

$$\frac{-1 - x}{(-1 + x)\,x}$$

In questi esempi hai potuto vedere all'opera la cosidetta notazione "postfissa" di *Mathematica*: oggetto//operazione. Ma e' anche possibile la tradizionale notazione "prefissa": operazione[oggetto].

■ Sostituzione di variabli

Se hai un'espressione contenente (ad esempio) le variabli x,y e vuoi sostituire y con un'altra espressione, e' disponibile l'operatore /. Esempi di uso:

```
(x - 1)^4 + y^2 /. y -> 1 / x
```

$$(-1 + x)^4 + \frac{1}{x^2}$$

■ Numeri complessi

Puoi usare i numeri complessi come usi i reali, ricordando che l'unita' immaginaria e' indicata con I (cioe' "i" maiuscolo)

```
Sqrt[-1]
```

$$I$$

```
(x - I * y)^3 // Expand
```

$$x^3 - 3\,I\,x^2\,y - 3\,x\,y^2 + I\,y^3$$

```
Conjugate[ 2 - 3 I] (* coniugazione complessa *)
```

$$2 + 3\,I$$

```
Abs[2 - 3 I]  (* questo il modulo *)
```

$$\sqrt{13}$$

```
Arg[ 1 + I] (* questa e' la 'fase' *)
```

$$\frac{\pi}{4}$$

Calcoli numerici

Essendo *Mathematica* progettato per il calcolo simbolico, e quindi "esatto", normalmente non ritorna mai il risultato numerico di un'espressione, a meno che non sia obbligato con l'operatore postfisso //N. Esempi:

```
Sqrt[2]
```

$$\sqrt{2}$$

```
Sqrt[2] // N
```

```
1.41421
```

E' anche possibile specificare la precisione (e quindi il numero di cifre) del risultato, usando pero' la notazione prefissa:

```
N[ Sqrt[2], 20] (* la radice di 2, con 20 cifre *)
```

```
1.4142135623730950488
```

```
N[ Pi, 500]     (* pigreco, con 500 cifre *)
```

```
3.14159265358979323846264338327950288419716939937510582097494459230781640628620899
 86280348253421170679821480865132823066470938446095505822317253594081284811174502
 84102701938521105559644622948954930381964428810975665933446128475648233786783165
 12019091456485669234603486104543266482133936072602491412737245870066063155881748
 15209209628292540917153643678925903600113305305488204665213841469519415116094330
 57270365759591953092186117381932611793105118548074462379962749567351885752724891
 22
 79381830119491
```

Risoluzione di equazioni e di sistemi

Per tutto questo c'e' la funzione Solve[] (vedi help online). Puo' risolvere una o piu' equazioni simultanee, con un qualsiasi numero di variabili. Sintassi:

Solve[{eq1,eq2,...},{x,y,z, ...}].

Per il segno di uguale si usa == (l'uguale logico). Infatti, il singolo segno = in genera significa assegnare un valore a qualcosa, nei programmi per computer.

Esempi:

```
Solve[ x^2 - 5 x == 0 + 6, x] (* eq. di secondo grado *)
```

$$\{\{x \to -1\}, \{x \to 6\}\}$$

```
Solve[ a * x + b * y + c == 0, y] (* esplicitare rispetto ad

                una variabile *)
```

$$\left\{\left\{y \to -\frac{c + a\,x}{b}\right\}\right\}$$

```
Solve[ -8 Exp[x] + 2 * Exp[2 x] + 1 == 0, x]
```

```
Solve::ifun : Inverse functions are being used by Solve, so some solutions may not be found.
```

$$\left\{\left\{x \to \text{Log}\left[\frac{1}{2}\left(4 - \sqrt{14}\right)\right]\right\}, \left\{x \to \text{Log}\left[\frac{1}{2}\left(4 + \sqrt{14}\right)\right]\right\}\right\}$$

```
Solve[ {y - x^2 == 0, y - x == 0}, {x, y}] (* intersezione parabola
       e bisettrice *)
```

```
{{y → 0, x → 0}, {y → 1, x → 1}}
```

Trigonometria

Se stai usando *Mathematica* versione 3 o maggiori, potrai usare la funzione TrigExpand e osservare come vengono applicate le regole base della trigonometria:

```
Sin[2 * x] // TrigExpand
```

```
2 Cos[x] Sin[x]
```

```
Sin[x + y] // TrigExpand
```

```
Cos[y] Sin[x] + Cos[x] Sin[y]
```

```
Tan[2 x] // TrigExpand
```

$$\frac{2\,\mathrm{Cos}[x]\,\mathrm{Sin}[x]}{\mathrm{Cos}[x]^2 - \mathrm{Sin}[x]^2}$$

```
Sin[x - Pi / 2]  (* una ben nota regola *)
```

```
-Cos[x]
```

```
Sin[x]^2 + Cos[x]^2 // Simplify  (* Identita' Fondamentale *)
```

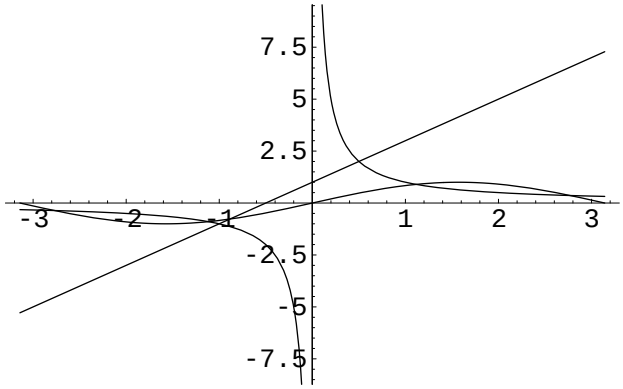
```
1
```

Grafici bi−dimensionali

■ Funzioni in una variabile

Per questo tipo di grafico esiste la funzione Plot[] (vedi help). Plot[] e' in grado di graficare piu' funzioni simultaneamente, ma il "range" della variabile indipendente deve essere lo stesso. Fai attenzione a come specifichi la funzione da plottare: se vuoi plottare y=f(x), devi inserire solo il secondo membro: f(x). Inoltre, se una delle funzioni "strafora", le altre potrebbero risultarne troppo "schiacciate".
Capirai da questi esempi la sintassi del comando Plot.

```
Plot[{1/x, 2x+1, Sin[x]}, {x, -Pi, Pi}]
```

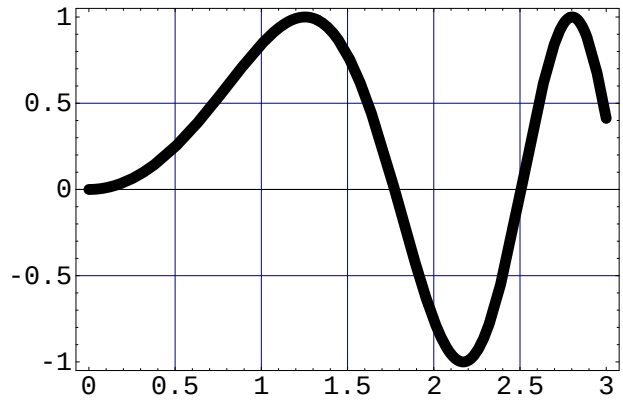



```
- Graphics -
```

■ Opzioni del grafico

Il comando Plot accetta tutta una serie di opzioni, nella forma " opzione−>valore",come
parte finale del comando. Potrai, ad esempio, specificare se usare o no una cornice (Frame−>True);se usare o no una griglia (GridLines−>Automatic)o specificare il colore e la dimensione del tratto. Studia questi esempi:

```
Plot[Sin[x^2], {x, 0, 3},
    Frame -> True,
    GridLines -> Automatic,
    AspectRatio -> Automatic,
    PlotStyle -> Thickness[0.02]
    ]
```




```
- Graphics -
```

■ Grafici parametrici

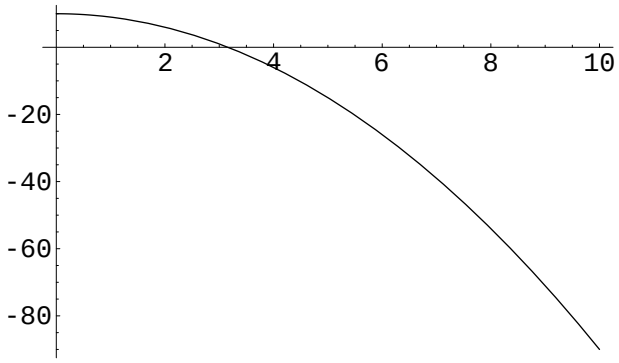
Adatti per rappresentare le "curve parametriche", cioe’ della forma:

x=X(t)

y=Y(t)

Ad esempio: le traiettorie dei moti piani in Meccanica, le curve di Lissajoux, etc

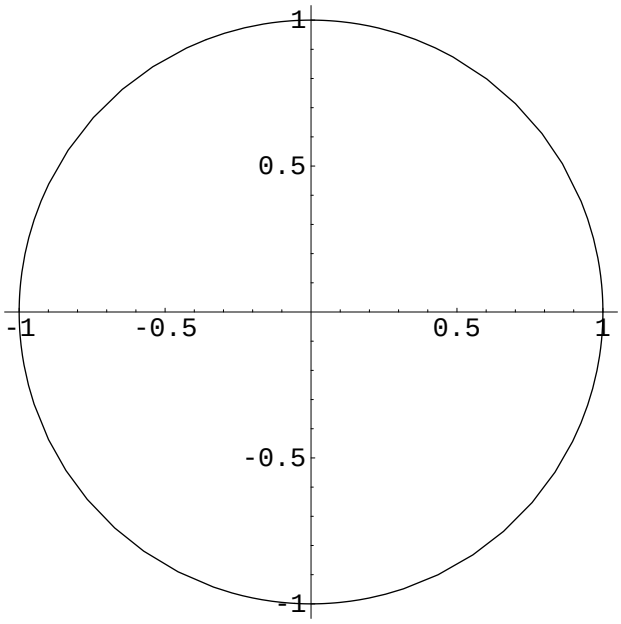
```
ParametricPlot[{t, 10 - t^2}, {t, 0, 10}]
```




```
- Graphics -
```

Il cerchio:

```
ParametricPlot[{Cos[t], Sin[t]}, {t, 0, 2 Pi},
AspectRatio -> Automatic
]
```



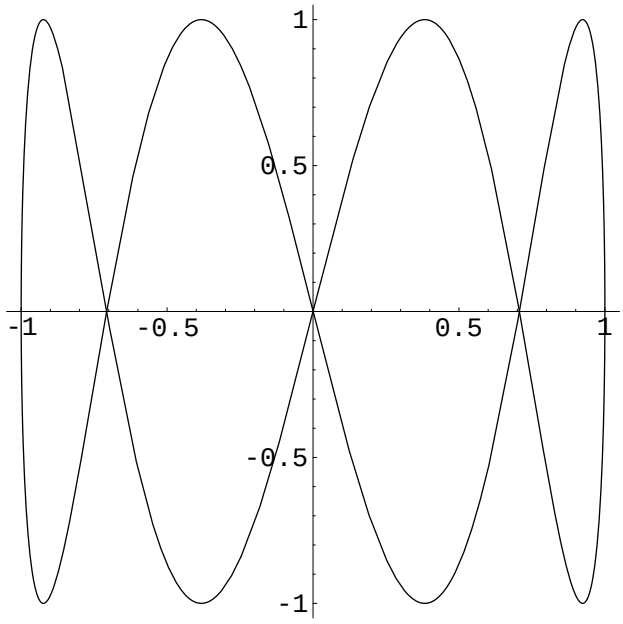

```
- Graphics -
```

Notare la presenza dell’opzione "AspectRatio−>Automatic":serve per correggere gli effetti della forma rettangolare dello schermo del computer.

```
- Graphics -
```

Curve di Lissajoux:

```
ParametricPlot[ { Cos[t], Sin[4 t]}, {t, 0, 2 Pi},
AspectRatio -> Automatic
]
```



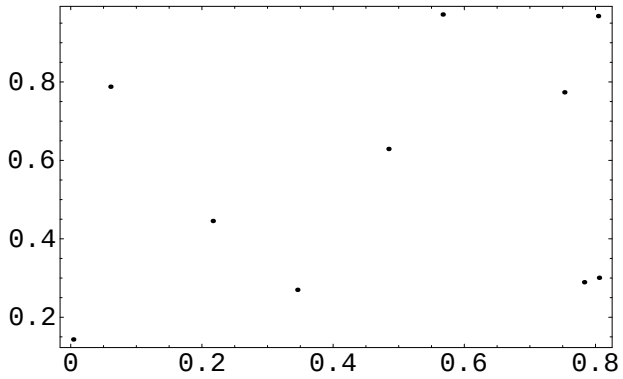
- Graphics -

■ Grafici fatti di singoli punti

```
t = Table[ {Random[], Random[]}, {i, 1, 10}
    ]
```

```
{{0.484414, 0.629163}, {0.567094, 0.972043},
 {0.0605094, 0.787855}, {0.345435, 0.269937},
 {0.00376167, 0.143437}, {0.782675, 0.28933}, {0.216544, 0.445534},
 {0.80396, 0.967951}, {0.805351, 0.300646}, {0.752521, 0.773591}}
```

```
ListPlot[t, Frame -> True]
```



- Graphics -

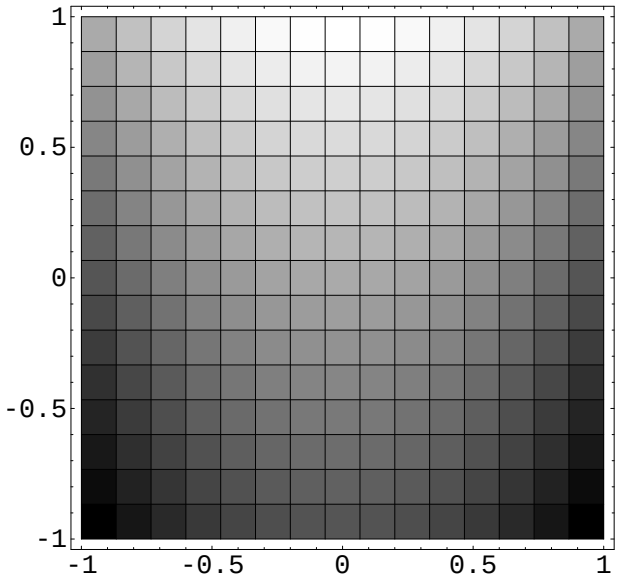
Grafici tri−dimensionali

Se nel caso bidimnsionale le funzioni avevano una sola variabile (x), nel caso tridimensionale ne hanno due (x,y). Volendo rappresentarli su un piano, l'informazione relativa alla terza variabilie (z) dovra' essere rappresentata con qualche trucco (colori, frecce, puntini, effetto profondita', etc). Analizzeremo alcune di queste tecniche, perche' utili nello studio della Fisica.

■ Density Plots

Come dice il nome stesso, possono essere utili a rappresentare la densita' o la temperatura in ogni punto di una lastra piana.

```
DensityPlot[y - x^2, {x, -1, 1}, {y, -1, 1}]
```



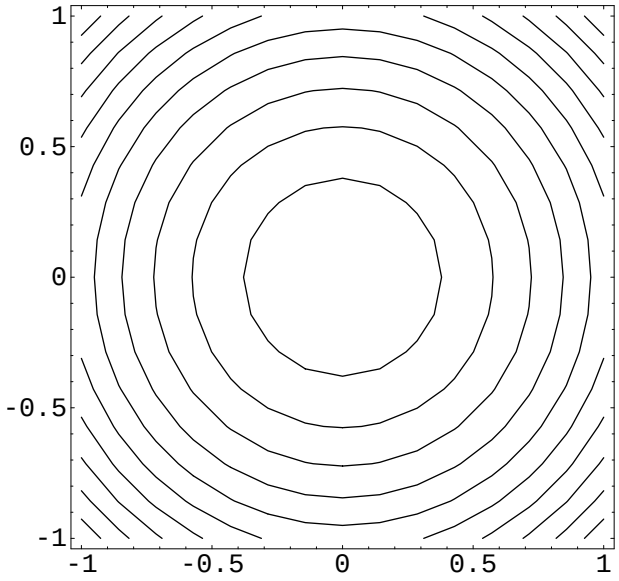

```
- DensityGraphics -
```

Questo tipo di grafico fa corrispondere uno stesso colore a valori simili della funzione f(x,y). In questo caso, i colori si dispongono lungo i fasci di parabole y−x^2=k

■ Contour Plots

Questo tipo di grafico e' simile ai DensityPlot. Invece di quadratini colorati, usa semplicemente delle linee che uniscono i punti con lo stesso valore della funzione f(x,y).
Utile quindi per rappresentazioni cartografiche, curve di livello e superfici equipotenziali.

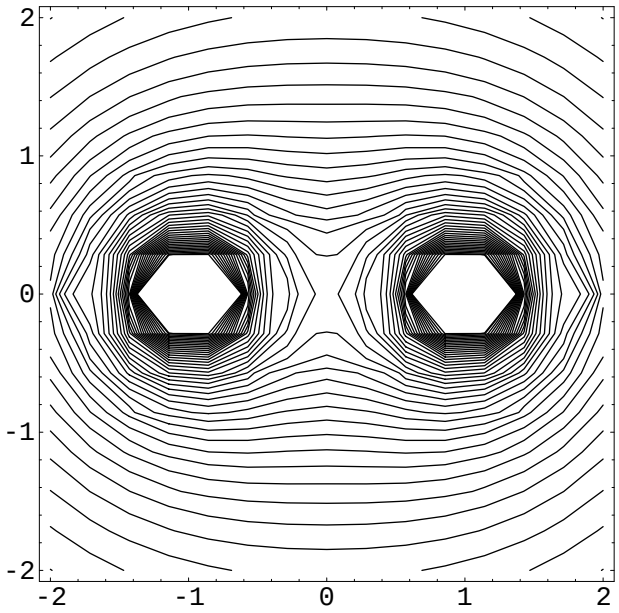
```
ContourPlot[ x^2 + y^2, {x, -1, 1}, {y, -1, 1},
ContourShading -> False, Contours -> 10]
```




```
- ContourGraphics -
```

Queste, ad esempio, sono le linee equipotenziali di un campo elettrico generato da due cariche puntiformi uguali in (−1,0)e (+1,0)

```
ContourPlot[ 1 / Sqrt[(x + 1) ^2 + y^2] + 1 / Sqrt[(x - 1) ^2 + y^2], {x, -2, 2}, {y, -2, 2},
 ContourShading -> False, Contours -> 40]
```

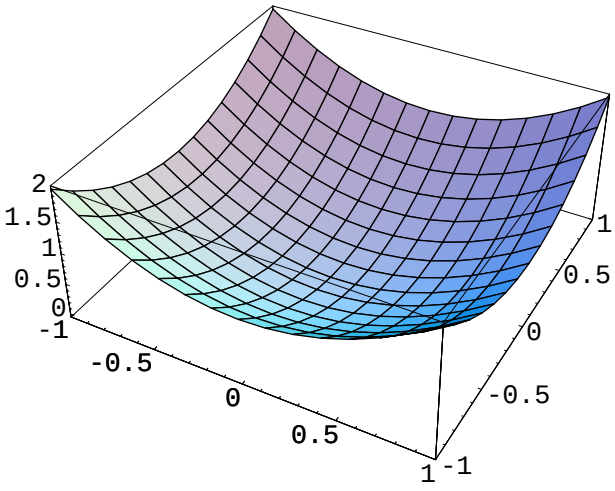



```
- ContourGraphics -
```

■ Plot 3D

Qui la terza dimensione e' ottenuta mediante il senso della profondita', inclinando la figura nello spazio, prospetticamente:

```
Plot3D[ x^2 + y^2, {x, -1, 1}, {y, -1, 1}]
```




```
- SurfaceGraphics -
```

Derivate ed Integrali

Per il calcolo integro−differenziale,*Mathematica* impiega due operatori: D[] e Integrate[], con questa sintassi:

D[f(x), x]	* Derivata prima rispetto ad x
D[f(x),{x,n}]	* Derivata n−esima,rispetto ad x
Integrate[f(x), x]	* Integrali indefiniti *
Integrate[f(x), {x,a,b}]	* Integrali definiti sull'intervallo [a,b]

Gli estremi di integrazione a,b possono prendere il valore Infinity, se l'integrale lo permette (cioe' se c'e' convergenza)

■ Esempi:

```
D[ Sin[x], x]
```

```
Cos[x]
```

```
D[ 1 / x, {x, 4}]
```

$$\frac{24}{x^5}$$

```
Integrate[ Sin[x] * x, x] (* integrale per parti *)
```

```
-x Cos[x] + Sin[x]
```

```
Integrate[ Sin[a * x], {x, 0, Pi}]
```

$$\frac{1}{a} - \frac{\text{Cos}[a\,\pi]}{a}$$

```
Integrate[ Sin[x] / x, {x, -Infinity, Infinity}]        (* integrale improprio *)
```

```
π
```

■ Esempi piu’ astratti:

```
D[Sin[f[x]], x]
```

$$\text{Cos}[f[x]]\, f'[x]$$

```
D[f[g[x]], x] (* funzioni composte *)
```

$$f'[g[x]]\, g'[x]$$

```
Integrate[f'[x]/f[x], x]
```

$$\text{Log}[f[x]]$$

```
D[f[x^2], {x, 5}]
```

$$120\, x\, f^{(3)}[x^2] + 160\, x^3\, f^{(4)}[x^2] + 32\, x^5\, f^{(5)}[x^2]$$

```
Integrate[x^(x^x), x] (* qui, Math. abbandona *)
```

$$\int x^{x^x}\, dx$$

Successioni e Serie

Con *Mathematica* e’ possibile creare facilmente "sequene finite" mediante il comando Table[]. Esempi:

```
Table[x^n, {n, 0, 5}]  (* successione di potenze *)
```

$$\{1, x, x^2, x^3, x^4, x^5\}$$

```
Table[1/(2 n + 1), {n, 0, 10}] (* inversi numeri dispari *)
```

$$\left\{1, \frac{1}{3}, \frac{1}{5}, \frac{1}{7}, \frac{1}{9}, \frac{1}{11}, \frac{1}{13}, \frac{1}{15}, \frac{1}{17}, \frac{1}{19}, \frac{1}{21}\right\}$$

Per sommare i termini di una successione, e’ disponibile il comando Sum[]:

```
Sum[k * x^k, {k, 0, 5}]
```

$$x + 2\, x^2 + 3\, x^3 + 4\, x^4 + 5\, x^5$$

Il comando Sum[], in qualche caso, e’ abbastanza intelligente da saper sommare sequenze senza conoscere a priori il numero di termini. Esempio:

```
Sum[q^k, {k, 0, n}]  (* progressione geometrica *)
```

$$\frac{-1 + q^{1+n}}{-1 + q}$$

```
Sum[a + k * q, {k, 0, n}] (* progressione aritmetica *)
```

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

Talvolta, Sum[] e’ in grado di calcolare la somma si sequenze infinite, quando sono di forme particolari:

```
Sum[1/2^k, {k, 1, Infinity}] (* la famosa serie armonica *)
```

$$1$$

o come nella somma degli inversi di tutti i quadrati perfetti:

```
Table[1/k^2, {k, 1, 10}]  (* eccone alcuni .... *)
Sum[1/k^2, {k, 1, Infinity}]
```

$$\left\{1, \frac{1}{4}, \frac{1}{9}, \frac{1}{16}, \frac{1}{25}, \frac{1}{36}, \frac{1}{49}, \frac{1}{64}, \frac{1}{81}, \frac{1}{100}\right\}$$

$$\frac{\pi^2}{6}$$

... ma se la serie e' veramente complicata, anche *Mathematica* abbandona, e semplicemente la riscrive intatta:

```
Sum[Sin[x^k]/(1+x)^k, {k, 0, Infinity}]
```

$$\sum_{k=0}^{\infty} \frac{\text{Sin}[x^k]}{(1+x)^k}$$

```
Sum[1/k, {k, 0, Infinity}] (* una serie non convergente *)
```

```
Sum::div : Sum does not converge.
```

$$\sum_{k=0}^{\infty} \frac{1}{k}$$

Mathematica e' in grado di produrre automaticamente lo sviluppo in serie di Taylor, semplicemente aggiungendo il simbolo di infinitesimo:

```
s = Sin[x] + O[x]^9
```

$$x - \frac{x^3}{6} + \frac{x^5}{120} - \frac{x^7}{5040} + O[x]^9$$

```
c = Cos[x] + O[x]^9
```

$$1 - \frac{x^2}{2} + \frac{x^4}{24} - \frac{x^6}{720} + \frac{x^8}{40320} + O[x]^9$$

Nelle variabili "s" e "c" sono ora memorizzati gli sviluppi in potenze fino all'ordine 9 del seno e del coseno. Se ora coinvolgete le due variabili in altri calcoli, il risultato verra' automaticamente sviluppato. Questa, ad esempio, e' la tangente:

```
s/c
```

$$x + \frac{x^3}{3} + \frac{2\,x^5}{15} + \frac{17\,x^7}{315} + O[x]^9$$

Algebra Lineare: vettori e matrici

■ Liste

Per lista, in *Mathematica*, si intende un insieme ("set") di elementi, e viene indicata con la notazione {.... }. Una lista puo' contenere altre liste. La lista vuota e' {}.
Se ho una lista, posso aggiungere facilmente elementi in fondo, con la funzione AppendTo:

```
vecchia = {1, 2, 3};
nuova = AppendTo[vecchia, 4]
```

```
{1, 2, 3, 4}
```

Per accedere alla posizione "3" di una lista L, basta scrivere L[[3]]. Esempio:

```
nuova[[3]]
```

```
3
```

■ Vettori

I vettori vengono visti come semplici liste ordinate di numeri reali o complessi. Esempio:

```
B = {4, 5} ; X = {x, y};
```

In particolare, per *Mathematica* questi sono vettori colonna. Infatti:

```
X // MatrixForm
```

$$\begin{pmatrix} x \\ y \end{pmatrix}$$

■ Matrici

Dato che una lista puo' contenere altre liste, e' comodo rappresentare una matrice come "una lista costituita dalle liste di ogni sua riga".

```
Esempio
```

```
Esempio
```

```
A = { {1, 2}, {3, 4}};
A // MatrixForm
```

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

Notare l'operatore MatrixForm: il suo scopo, puramente tipografico, e' quello di mostrare la matrice nella forma tradizionale .
Mathematica permette di usare le matrici come se fossero oggetti algebrici qualsiasi, ma il prodotto va indicato con il punto (.), non con l'asterisco.
Esempio:

```
A.A // MatrixForm
```

$$\begin{pmatrix} 7 & 10 \\ 15 & 22 \end{pmatrix}$$

```
2 * A + A.A.A // MatrixForm
```

$$\begin{pmatrix} 39 & 58 \\ 87 & 126 \end{pmatrix}$$

```
{Inverse[A] // MatrixForm, Transpose[A] // MatrixForm}
```

$$\left\{\begin{pmatrix} -2 & 1 \\ \frac{3}{2} & -\frac{1}{2} \end{pmatrix}, \begin{pmatrix} 1 & 3 \\ 2 & 4 \end{pmatrix}\right\}$$

```
Det[A]  (* determinante *)
```

$$-2$$

```
Eigenvalues[A]  (* autovalori *)
```

$$\left\{\frac{1}{2}\left(5-\sqrt{33}\right), \frac{1}{2}\left(5+\sqrt{33}\right)\right\}$$

■ Prodotti tra matrici e vettori

```
A.X // MatrixForm
```

$$\begin{pmatrix} x+2y \\ 3x+4y \end{pmatrix}$$

Cosi', se io volessi risolvere il sistema lineare AX=B, potrei calcolare l'espressione X=A^−1*B:

```
Z = Inverse[A].B
```

$$\left\{-3, \frac{7}{2}\right\}$$

Verifica:

```
A.Z - B // MatrixForm
```

$$\begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Grafica avanzata

(Saltare in prima lettura −−e anche in seconda ....)
Una "primitiva grafica" e' una funzione con certi parametri di input, il cui scopo e' definire in maniera del tutto astratta un elemento rappresentabile su una superficie bidimensionale. In parole povere, qui il "astratto" significa che queste funzioni non disegnano realmente line e cerchi, ma servono solo a rappresentare il concetto in un modo che *Mathematica* possa comprendere.

■ Alcune primitive grafiche piu' usate

Nel seguito, con p1, p2, ... si intende punti nel nel piano o nello spazio.

p=Point[{x,y}]
Line[{ p1,p2}]
Polygon[{ p1,p2,p3,p4,...}]
Circle[{x,y], r]

■ Combinare le primitive grafiche

Un grafico non e' altro che un insieme di primitive grafiche e puo' essere quindi rappresentato come una lista {} di primitive:

graf= Graphics[{ primitiva1, primitiva2, ...}]

Come si vede, per distinguere questo tipo di liste da altre, si racchiude tutto nella direttiva Graphics[]. Anche le altre funzioni grafiche viste prima (Plot, etc) producono un oggetto dello stesso tipo, cioe' un Graphics[].

Oltre che primitive, un grafico puo' contenere anche le cosidette "direttive grafiche", che non sono altro che opzioni di tracciamento, quali:

PointSize[r] (* dimensione punti *)
Thickness[r] (* dimensione linee *)
RGBColor[rosso,verde,blue] (* composizione colore , formato RGB *)
Hue[h] (* composizione colore, formato [0,1] *)

In conclusione, un "grafico" puo' essere ottenuto combinando in una lista "primitive" e "direttive", con la convenzione che una direttiva "agisce" solo sulle primitive che seguono nella lista di definizione.

■ Visualizzare il grafico (rendering)

Per trasformare il "grafico astratto" in un grafico realmente visualizzato su un dispositivo di output, quale lo schermo del computer, e' disponibile la funzione Show[]. Questa funzione ha molti parametri e modalita' sofisticate (vedi l'help). Ecco qualche esempio, nei casi piu' semplici:

Show[graf]
Show[graf, Axes->True,AspectRatio->Automatic]

Talvolta, specie creando animazioni, si vuole soltanto produrre le bitmap per il grafico, con l'intenzione di visualizzarlo successivamente. Si puo' disabilitare momentaneamente la visualizzazione dell'immagine in questo modo:

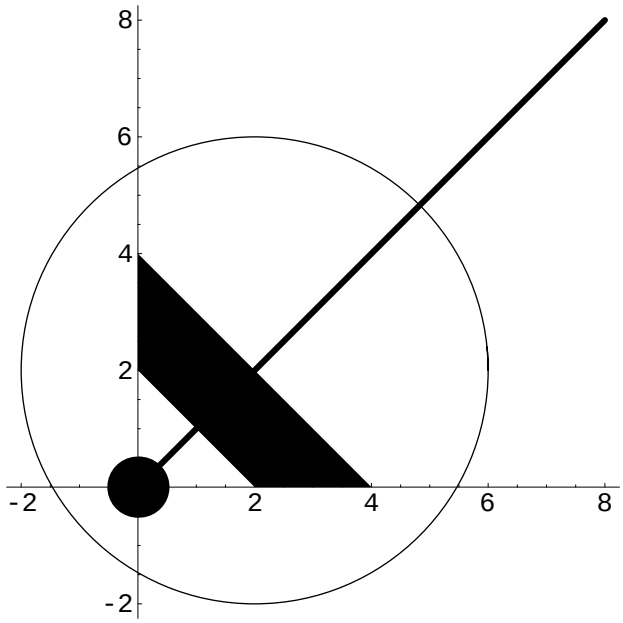
Show[graf, DisplayFuntion->Identity]

■ Esempi

```
grafico = Graphics [
{
            Circle[ {2, 2}, 4],
            Thickness[0.01], Line[ { {0, 0}, {8, 8}}],
            PointSize[0.1], Point[{0, 0}],
            Polygon[ { {0, 2}, {0, 4}, {4, 0}, {2, 0}}]
        }
    ]

- Graphics -
```

```
Show[grafico, Axes -> True, AspectRatio -> Automatic]
```




```
- Graphics -
```

Animazioni

In questo esempio, per creare un’animazione, viene costruita una lista di fotogrammi chiamata "film"; ogni fotogramma e’ un oggetto grafico prodotto (ad esempio) dalla funzione Plot[], o con altri metodi.

```
film = {};
    fotogramma :=
 Plot [Sin[t] * Sin[x], {x, 0, 2 * Pi}, PlotRange -> {-1, 1}, DisplayFunction -> Identity]

    For [ t = 0,  t < 2 Pi,  t = t + 0.4,
        AppendTo[film, fotogramma]
]
```

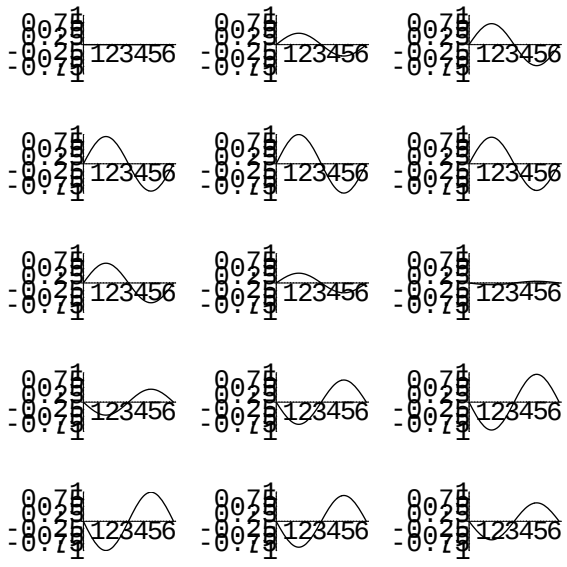
Col comando seguente, i vari fotogrammi vengono "srotolati" nella pagina. Per visualizzare effettivamente l’animazione, devi selezionarli ed attivare l’opzione Cell−>Animate,col tuo mouse.

```
ShowAnimation[ film]
```

(io non lo faccio, altrimenti la pagina si riempie di centinaia di diagrammi della funzion seno)

I vari fotogrammi possono anche essere rappresentati in forma di matrice grafica, come nel seguente esempio:

```
Show[ GraphicsArray[ Partition[film, 3] ] ]
```



```
- GraphicsArray -
```

Programmazione con Mathematica

Talvolta e necessario creare dei "moduli di calcolo indipendenti", costruiti per scopi particolari e tali da poter essere chiamati a loro volta da altri moduli dello stesso tipo. Esattamente come in altri linguaggi di programmazione (quali il Pascal), in *Mathematica* e' possibile costruirsi le proprie funzioni e le proprie subroutines. Una volta create, le nuove funzioni possono entrare a far parte dei tuoi schemi di calcolo, esattamente come le funzioni interne di *Mathematica*. E' necessario pero' rispettare una precisa sintassi.

■ Semplici esempi di funzioni

Una funzione non e' altro che una procedura di calcolo capace di produrre dei risultati a partire da un certo insieme di dati iniziali (i parametri di input). Le "variabili formali", cioe' quelle che indicano l'input alla funzione, devono essere del tipo "Simbolo_":

```
quadrato[x_] := x^2   (* definizione di quadrato[] *)
quadrato[2]
```

```
4
```

Una funzione puo' avere piu' dati di input e puo' produrre anche oggetti multipli, tipo le liste, i vettori. Esempio:

```
EqCirconferenza[x0_, y0_, R_] := (x - x0)^2 + (y - y0)^2 - R^2 // Expand
```

Uso:

```
EqCirconferenza[2, -3, 4]
```

$$-3 - 4\,x + x^2 + 6\,y + y^2$$

Nell'esempio seguente, la funzione SP costruisce un punto nel piano, ottenuto sommando e moltiplicando i due valori x,y forniti:

```
SP[x_, y_] := {x + y, x^y}
```

Uso:

```
SP[a, b]
```

$$\{a + b,\ a^b\}$$

■ Funzioni con un corpo piu' voluminoso

Negli esempi precedenti la funzione veniva implementata con una semplice riga. Se dovessero occorrere piu' righe e, magari, anche delle variabili locali, e' piu' comodo usare il costrutto Module[], con questa sintassi:

funzione[x_,y_,z_, ...]:= Module[{var. locali}, riga1; riga2; riga3; ....]

Attenzione all'uso delle virgole e dei punti−e−virgolanella riga precedente: la loro posizione e' cruciale. Le varie righe del programma (riga1, riga2, ...) conviene metterle su linee separate, utilizzando il tasto INVIO, cercando possibilmente di mantenere il codice leggibile. Ricorda: puoi andare a capo ogni volta che vuoi, all'interno di Module[]:

```
funzione[...] := Module [ {.....},
        riga1;
        riga2;
        .....
]
```

Ma, dei vari calcoli effettuati, qual'e' quello effettivamente ritornato dalla funzione?
Semplice: l'ultimo.

Come Esempio, il programma seguente genera "n" numeri interi compresi tra 1 e 10 e calcola la frazione di essi che risulta un numero pari. Un tale programma si potrebbe fare in maniera assai piu' elegante sfruttando meglio le caratteristiche di *Mathematica* , ma qui viene affrontato in modo tradizionale, per illustrare i costrutti For[], If[] etc. per i quali comunque ti consiglio di consultare l'help on−line.

```
conta[n_] := Module [ {i, pari, casuale},
        pari = 0;
        For[ i = 1, i <= n, i = i + 1,
            casuale = Random[Integer, {1, 20}];
            pari = pari + If[ Mod[casuale , 2] == 0, 1, 0];
            ];
    pari / n  // N
        ]

Table[ conta[100 * i], {i, 1, 10}]
```

$$\{0.58, 0.505, 0.453333, 0.485, 0.49, 0.51, 0.527143, 0.505, 0.508889, 0.516\}$$

■ Funzioni astratte

In certi casi e' comodo poter definire delle funzioni utilizzando una sintassi leggermente diversa, in modo poterle creare direttamente dove servono. La sintassi e' la seguente:

f[#]&

Esempi:

```
f = (1 + Sin[#]) &;
g = 1 / (1 + #^2) &;
```

Le funzioni definite in questo modo si possono "comporre" in senso matematico:

```
h = Composition[ f, g];
```

Si ha percio':

```
h[x]
```

$$1 + \text{Sin}\left[\frac{1}{1 + x^2}\right]$$

```
h'[x]
```

$$-\frac{2\,x\,\text{Cos}\left[\frac{1}{1+x^2}\right]}{(1 + x^2)^2}$$

■ Calcoli iterativi

Mathematica dispone di una serie di costrutti in grado di applicare delle funzioni ripetutamente, a partire da un oggetto iniziale x (composizione di funzioni). In questi esempi puoi vedere all'opera il concetto di "funzione astratta":

```
Nest[ f, x, 4]
```

```
f[f[f[f[x]]]]
```

```
Nest [ 1 / (2 + #) &,  x, 4]
```

$$\cfrac{1}{2 + \cfrac{1}{2 + \cfrac{1}{2 + \cfrac{1}{2 + x}}}}$$

```
Nest [ Sqrt[1 + 2 #] &,  x, 8]
```

$$\sqrt{1 + 2\sqrt{1 + 2\sqrt{1 + 2\sqrt{1 + 2\sqrt{1 + 2\sqrt{1 + 2\sqrt{1 + 2\sqrt{1 + 2\,x}}}}}}}}$$

Uno dei metodi di ricerca degli zeri di una funzione si base sulla "metodo del punto fisso", in sostanza sulla soluzione di un'equazione del tipo x=f(x). Cosa succede se inserite nella vostra macchinetta tascali il numero 0.1 e poi cominciate a schiacciare continuamente il tasto COS del coseno? Dopo un po' il risultato non cambia piu' (apparentemente), e otterrete:

```
FixedPoint[ Cos, 0.1]
```

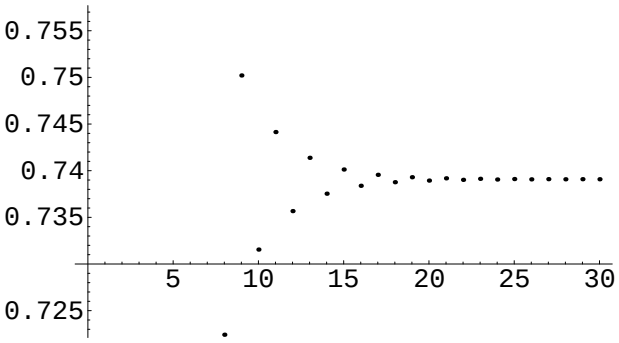
```
0.739085
```

che e' appunto un "punto fisso" del coseno. Naturalmente, non e' detto che il processo converge sempre, ma sotto precise condizioni sulla derivata di f(x), si.
Per osservare meglio come il "processo" evolve, si puo' richiedere alla funzione FixedPoint di operare solo un numero "n" di cicli, come in questo esempio:

```
punti = Table [ FixedPoint[ Cos, 0.1, n], {n, 1, 30}]
```

```
{0.995004, 0.544499, 0.855387, 0.655927, 0.792483, 0.702079, 0.763501, 0.72242,
 0.750208, 0.731547, 0.744142, 0.735669, 0.741382, 0.737536, 0.740128,
 0.738383, 0.739558, 0.738766, 0.7393, 0.73894, 0.739183, 0.73902, 0.739129,
 0.739055, 0.739105, 0.739072, 0.739094, 0.739079, 0.739089, 0.739082}
```

```
ListPlot[punti]
```



- Graphics -