

Appendice 1.5

Mathematica: principali novità della versione 5

Fino alla precedente versione 4, *Mathematica* offriva prestazioni maggiori sul calcolo simbolico rispetto a quelle sul calcolo numerico. Con *Mathematica* 5 si inizia una nuova fase, grazie al raggiungimento di elevate capacità e prestazioni anche nelle applicazioni di calcolo numerico. Grazie alle diverse tecnologie introdotte negli ultimi anni, quali ad esempio *webMathematica*, *gridMathematica*, *JLink*, *.NET Link*, *Mathematica* può essere anche utilizzato per lavori di ricerca industriale complessa, sfruttando le sue caratteristiche di elevata velocità di calcolo e scalabilità, senza dover fare a meno delle sue caratteristiche di ampia applicabilità e general purpose. Tali connotazioni rendono *Mathematica* un sistema per il calcolo tecnico unico ed irrinunciabile per matematici, fisici, ingegneri e specialisti di qualsiasi settore tecnico-scientifico.

La caratteristica distintiva di *Mathematica* 5 è la presenza di nuovi algoritmi avanzati, nuove tecnologie chiave ridefinite e riprogettate, alcuni nuovi algoritmi numerici ed altri notevolmente migliorati ed un nuovo solutore per le equazioni differenziali con metodi numerici. Inoltre, *Mathematica* 5 conferma la leadership della Wolfram nel fornire uno strumento di calcolo potente e pienamente integrabile con altre tecnologie e standard all'avanguardia. Insieme ad una nuova versione di *J/Link* e *MathLink*, le connessioni con *Java* e *C/C++*, la versione 5 introduce la nuova tecnologia *.NET/Link* per integrare il motore di calcolo con le applicazioni che utilizzano il *Microsoft .NET Framework*.

Alcune operazioni con le matrici

Tra i pacchetti aggiuntivi (standard Add-ons) vi è anche la categoria `LinearAlgebra`` che aggiunge ulteriori funzioni utili per il calcolo matriciale. Ad esempio, `LinearAlgebra`MatrixManipulation`` implementa una serie di funzioni per la manipolazione di matrici, quali ad esempio

| Funzione | Descrizione |
|--|---|
| <code>AppendColumns[A, B]</code> | unisce le colonne di A con le colonne di B |
| <code>AppendRows[A, B]</code> | unisce le righe di A con le righe di B |
| <code>BlockMatrix[blocchi]</code> | unisce i <i>blocchi</i> per formare una nuova matrice |
| <code>TakeRows[M, n]</code> | restituisce la riga n della matrice M |
| <code>TakeColumns[M, n]</code> | restituisce la colonna n della matrice M |
| <code>SubMatrix[M, posizione, dimensione]</code> | restituisce la sottomatrice a partire dall'elemento in <i>posizione</i> e di dimensioni <i>dimensione</i> |
| <code>MatrixPlot[M]</code> | rappresenta la struttura della matrice mediante un grafico |
| <code>UpperDiagonalMatrix[f, n]</code> | genera una matrice diagonale superiore di dimensioni $n \times n$ |
| <code>LowerDiagonalMatrix[f, n]</code> | genera una matrice diagonale inferiore di dimensioni $n \times n$ |
| <code>TridiagonalMatrix[f, n]</code> | genera una matrice tridiagonale di dimensioni $n \times n$ |

Tabella 1.1. Alcune funzioni aggiuntive per la manipolazione delle matrici.

```
<< LinearAlgebra`MatrixManipulation`
```

Ad esempio la funzione `BlockMatrix` permette di costruire una matrice a blocchi indicandone i singoli blocchi. Se si desidera creare una matrice di dimensione 6×8 con dei valori nulli disposti secondo la diagonale principale a blocchi di 3×4 e valori casuali nelle rimanenti posizioni, si può scrivere

```
M = BlockMatrix[{
  {Table[0, {3}, {4}], Table[Random[Integer, {1, 5}], {3}, {7}], Table[0, {3}, {2}]},
  {Table[Random[Integer, {1, 5}], {5}, {4}],
   Table[0, {5}, {7}], Table[Random[Integer, {1, 5}], {5}, {2}]}]}
{{0, 0, 0, 0, 3, 5, 1, 3, 4, 2, 5, 0, 0}, {0, 0, 0, 0, 5, 5, 5, 2, 3, 5, 5, 0, 0},
 {0, 0, 0, 0, 5, 5, 1, 3, 1, 4, 5, 0, 0}, {4, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 4, 4},
 {2, 2, 2, 4, 0, 0, 0, 0, 0, 0, 1, 5}, {3, 2, 3, 3, 0, 0, 0, 0, 0, 0, 3, 2},
 {2, 4, 5, 3, 0, 0, 0, 0, 0, 0, 5, 4}, {2, 3, 4, 4, 0, 0, 0, 0, 0, 0, 2, 2}}
```

```
MatrixForm[M]
(
0 0 0 0 3 5 1 3 4 2 5 0 0
0 0 0 0 5 5 5 2 3 5 5 0 0
0 0 0 0 5 5 1 3 1 4 5 0 0
4 1 1 1 0 0 0 0 0 0 0 4 4
2 2 2 4 0 0 0 0 0 0 0 1 5
3 2 3 3 0 0 0 0 0 0 0 3 2
2 4 5 3 0 0 0 0 0 0 0 5 4
2 3 4 4 0 0 0 0 0 0 0 2 2
)
```

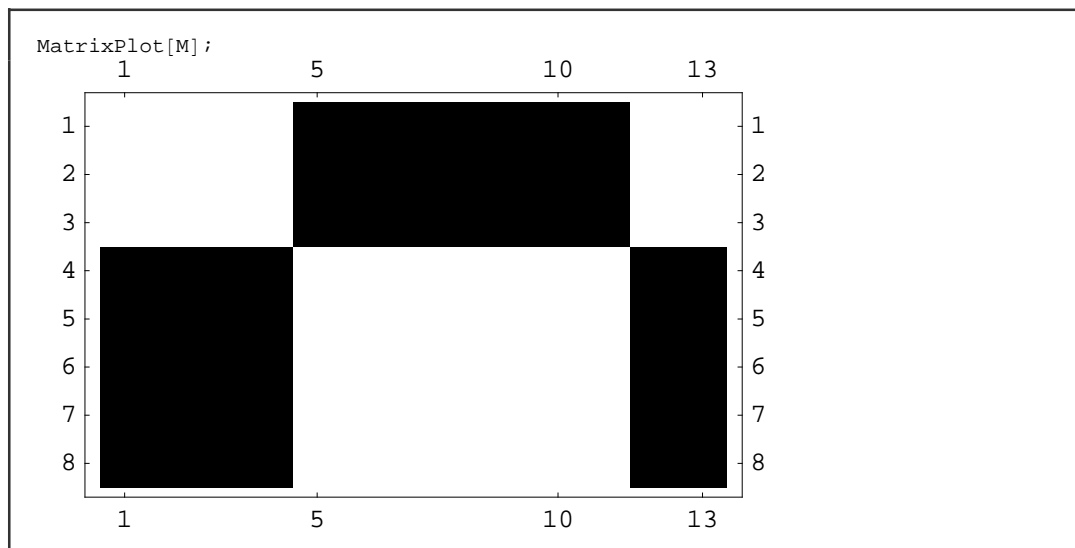
La funzione `SubMatrix` permette di prelevare una porzione di matrice specificando l'indice dell'elemento `{1, 1}` della sottomatrice e le sue dimensioni

```
SubMatrix[M, {4, 1}, {5, 4}] // MatrixForm

$$\begin{pmatrix} 4 & 1 & 1 & 1 \\ 2 & 2 & 2 & 4 \\ 3 & 2 & 3 & 3 \\ 2 & 4 & 5 & 3 \\ 2 & 3 & 4 & 4 \end{pmatrix}$$

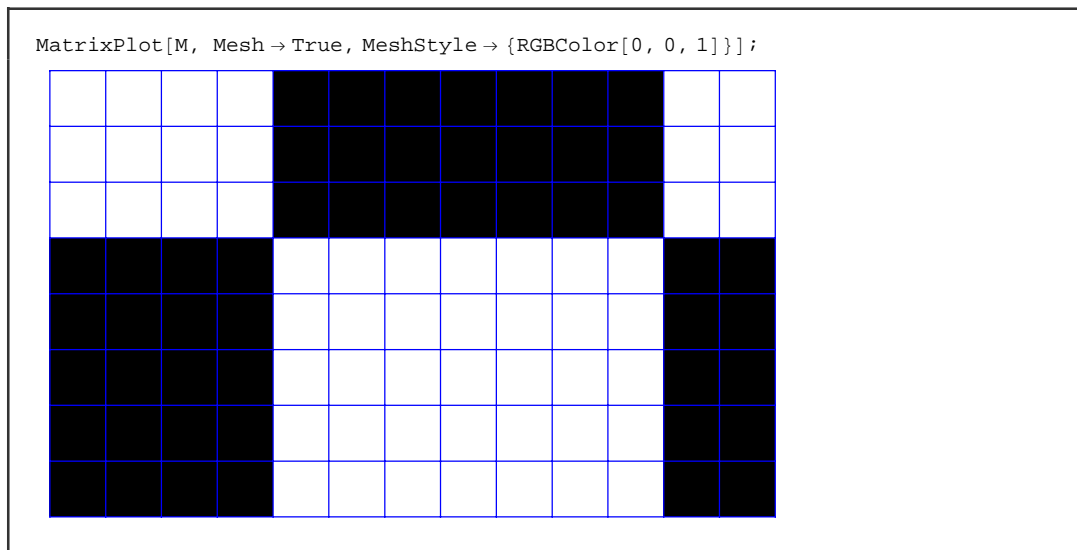
```

Nel pacchetto `LinearAlgebra`MatrixManipulation`` esiste anche la funzione `MatrixPlot`, molto utile per sintetizzare la struttura di una matrice mediante una rappresentazione grafica dei suoi elementi.



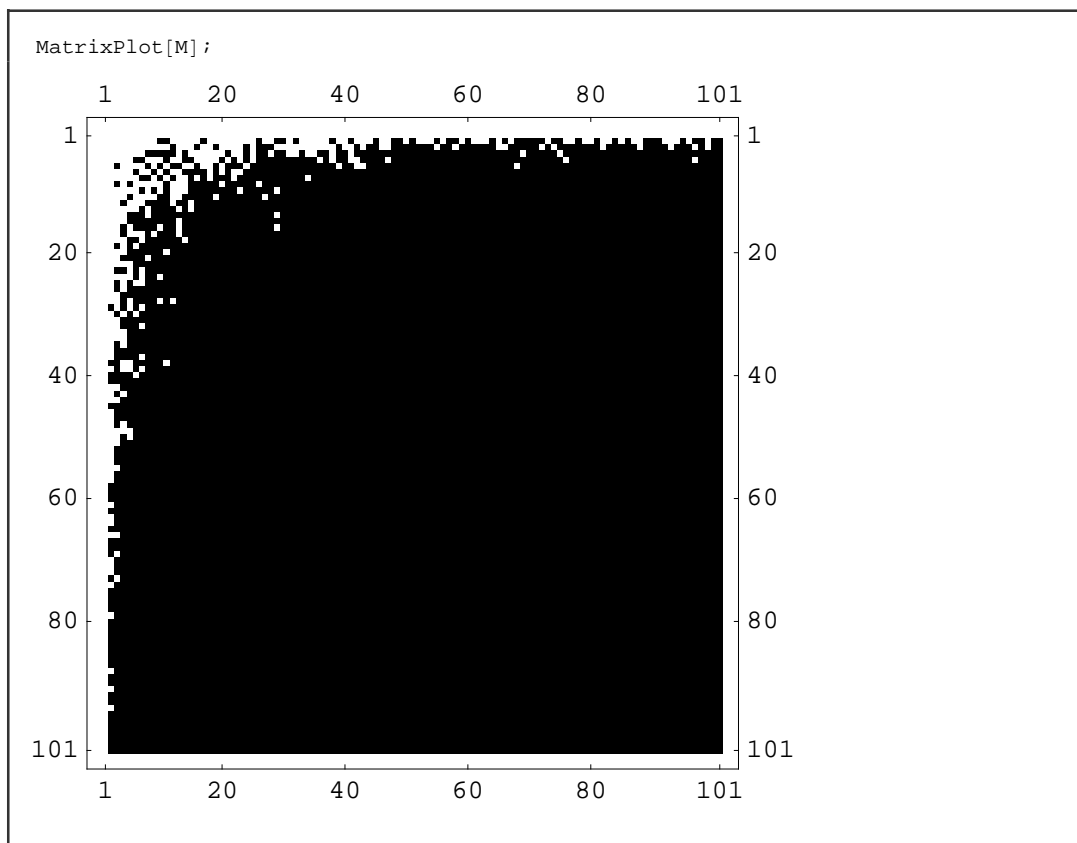
Il risultato ottenuto in questo caso, evidenzia come la matrice `M` sia una matrice a blocchi, infatti le aree in bianco nel grafico indicano i blocchi con elementi nulli.

Se si aggiunge l'opzione `Mesh`, si evidenzia maggiormente la struttura di righe e colonne.



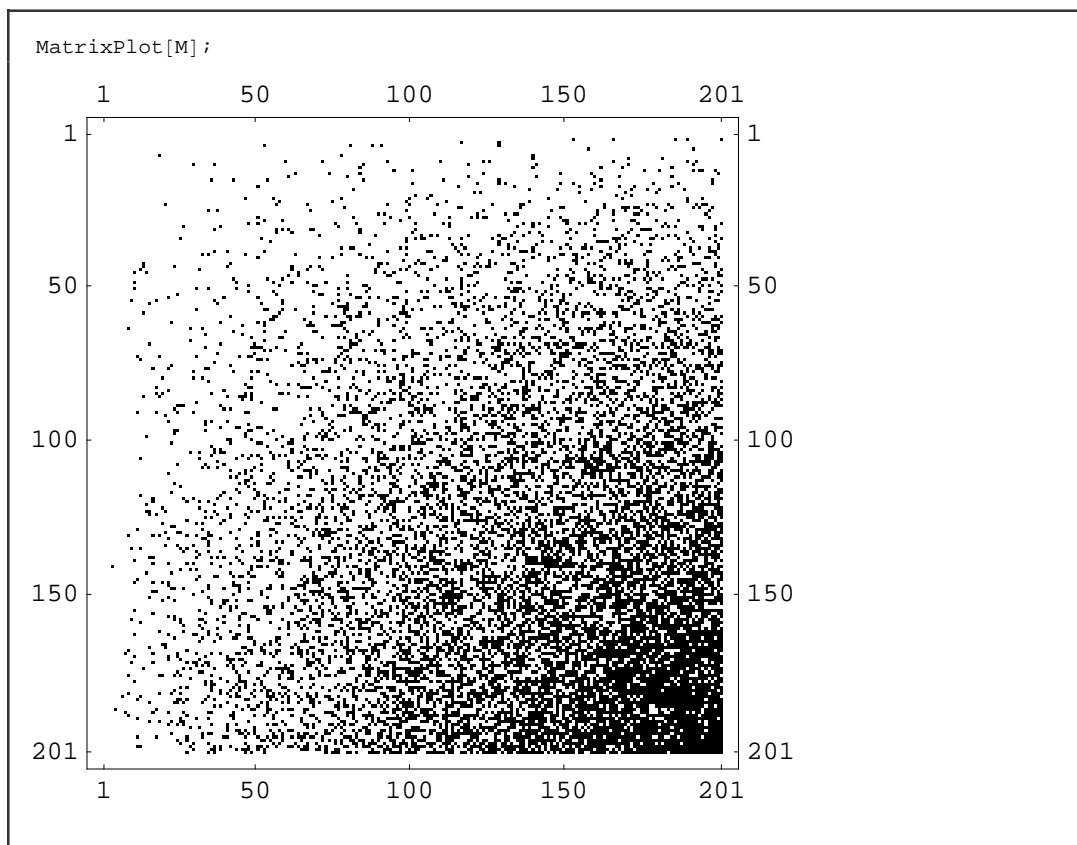
La `MatrixPlot` risulta maggiormente significativa per la visualizzazione di struttura per matrici molto grandi, come negli esempi che seguono. Si carica il pacchetto di statistica che offre una serie di funzioni per la generazione di numeri casuali secondo varie distribuzioni di probabilità

```
<< Statistics`  
  
M = Table[Random[BinomialDistribution[m, n]], {n, 0, 1, 0.01}, {m, 0, 100}];  
  
Dimensions[M]  
{101, 101}
```

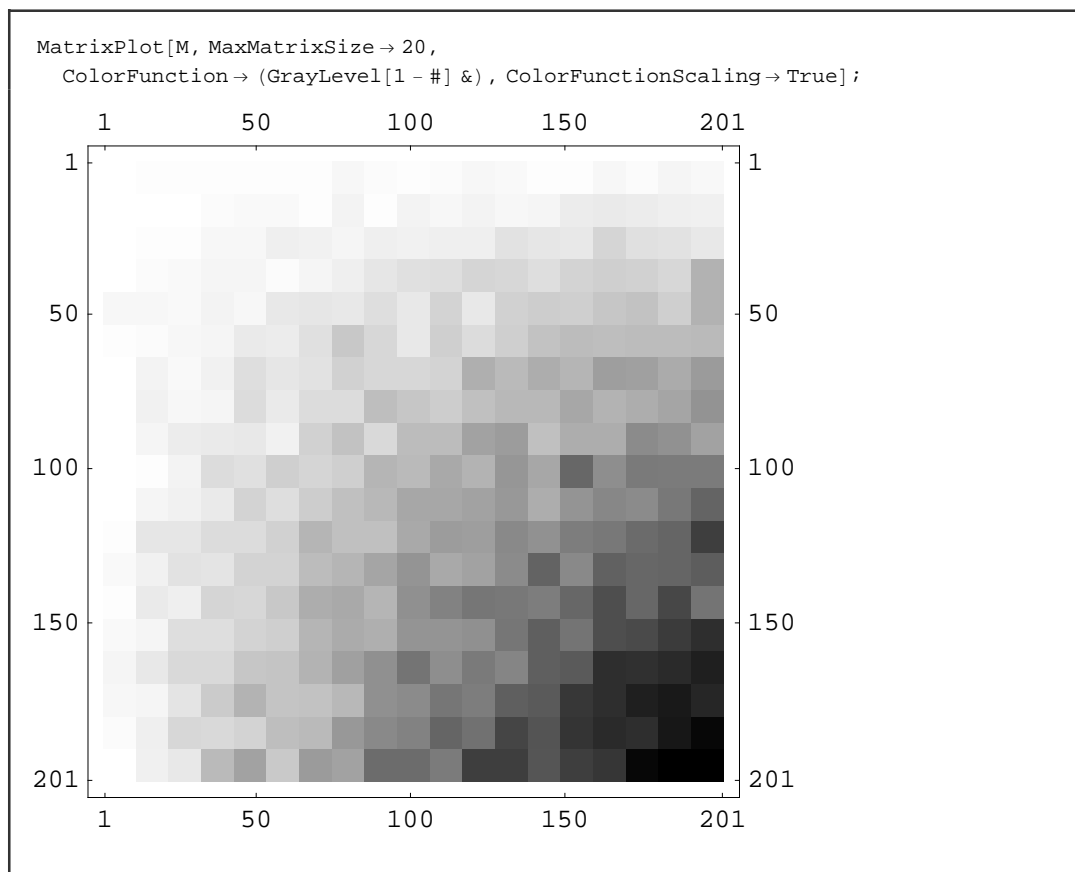


```
M = Table[Random[BernoulliDistribution[m*n]], {n, 0, 1, 0.005}, {m, 0, 1, 0.005}];
```

```
Dimensions[M]  
{201, 201}
```



Riducendo il numero di elementi da visualizzare è possibile anche evidenziare con diverse tonalità di grigio densità degli elementi non nulli.



Come ultimo argomento di questo paragrafo, si riporta una novità significativa introdotta con la versione 5 di *Mathematica*, nel campo dell'algebra delle matrici: la gestione ottimizzata per le matrici sparse. Un elevato numero di problemi reali si affronta con modelli matematici che presentano matrici sparse, ossia matrici dove la maggior parte degli elementi sono nulli. La gestione di tali matrici in *Mathematica 5* è totalmente integrata con le altre funzionalità del sistema e permette di lavorare con array di qualsiasi dimensione.

Per creare una matrice o vettore sparso, bisogna usare la funzione `SparseArray`, la cui struttura più comune è la seguente

```
SparseArray[regole, dimensione]
```

Il meccanismo delle regole permette di semplificare la creazione di particolari tipi di matrici. Infatti, per alcuni tipi di matrice, ad esempio quelle diagonali o triangolari, gli elementi non nulli sono disposti su particolari posizioni della matrice. Se, oltre alla posizione, anche il loro valore può essere determinato mediante una particolare funzione, allora conviene costruire una regola che a partire dagli indici di posizione i e j calcola automaticamente i valori delle entrate.

Ad esempio, per costruire la matrice identità basta scrivere

```
ID = SparseArray[{i_, i_} → 1, 4]
SparseArray[<4>, {4, 4}]
```

Come si osserva dall'output, l'oggetto generato è di tipo `SparseArray` e non lista. Gli `SparseArray` vengono gestiti dalle funzioni native come una lista di liste. L'utente non deve farsi carico di specificare alcuna informazione nel trattare tali oggetti come se fossero una vera e propria matrice.

```
Transpose[ID]
SparseArray[<4>, {4, 4}]
```

```
Det[ID]
1
```

```
MatrixForm[ID]

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

```

```
MatrixForm[ID.ID]

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

```

```
MatrixForm[Inverse[ID]]

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

```

Se si desidera convertire un oggetto di tipo `SparseArray` in una lista, si può utilizzare la funzione `Normal`.

```
Normal[ID]
{{1, 0, 0, 0}, {0, 1, 0, 0}, {0, 0, 1, 0}, {0, 0, 0, 1}}
```

Se si desidera conoscere le regole interne di una matrice sparsa, si può utilizzare la funzione `ArrayRules`.

```
ArrayRules[ID]
{{1, 1} → 1, {2, 2} → 1, {3, 3} → 1, {4, 4} → 1, {_, _} → 0}
```

Si noti che, anche se esiste il comando `IdentityMatrix` che genera matrici identità di qualsiasi dimensione, se le dimensioni della matrice sono notevoli è comunque preferibile usare la sua rappresentazione nella forma `SparseArray`.

Si calcola il numero di byte per memorizzare una matrice identità di dimensioni 1000×1000 in entrambi i modi.

```
IDsparsa = SparseArray[{i_, i_} → 1., 1000];
```

```
ByteCount[IDsparsa]
16376
```

```
ID = N[IdentityMatrix[1000]];
```

```
ByteCount[ID]
8000060
```

Se si deve costruire una matrice sparsa con elementi non nulli disposti su posizione non esprimibili in maniera diretta tramite un'unica regola, allora si può utilizzare una regola per ciascun elemento o una regola per quegli elementi che rientrano in un determinato criterio e delle singole regole per gli altri.

```
M = SparseArray[{{1, 2} → 1, {2, 3} → 4, {3, 1} → -2, {3, 3} → 2}]
SparseArray[<4>, {3, 3}]
```

```
MatrixForm[M]

$$\begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 4 \\ -2 & 0 & 2 \end{pmatrix}$$

```

Costruzione di una matrice con elementi non nulli sulla diagonale principale ed un altro elemento non nullo nella posizione (2,5).

```
SparseArray[{{i_, i_} → 2 i + 1, {2, 5} → -1}, 5]
SparseArray[<6>, {5, 5}]
```

```
MatrixForm[%]

$$\begin{pmatrix} 3 & 0 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 & -1 \\ 0 & 0 & 7 & 0 & 0 \\ 0 & 0 & 0 & 9 & 0 \\ 0 & 0 & 0 & 0 & 11 \end{pmatrix}$$

```

L'esempio che segue mostra come costruire una matrice tridiagonale.

```
SparseArray[{{i_, j_} /; Abs[i - j] ≤ 1 → Random[Integer, {1, 10}]}, {5, 5}]
SparseArray[<13>, {5, 5}]
```

```
MatrixForm[%]

$$\begin{pmatrix} 10 & 4 & 0 & 0 & 0 \\ 5 & 1 & 4 & 0 & 0 \\ 0 & 7 & 6 & 1 & 0 \\ 0 & 0 & 4 & 5 & 1 \\ 0 & 0 & 0 & 8 & 8 \end{pmatrix}$$

```

Si costruiscono due matrici triangolari, superiore ed inferiore, rispettivamente

```

SparseArray[{{i_, j_} /; i - j ≤ 0 => Random[Integer, {1, 10}]}, {5, 5}] // MatrixForm

```

$$\begin{pmatrix} 1 & 7 & 3 & 1 & 6 \\ 0 & 3 & 9 & 8 & 7 \\ 0 & 0 & 2 & 2 & 9 \\ 0 & 0 & 0 & 3 & 1 \\ 0 & 0 & 0 & 0 & 2 \end{pmatrix}$$

```

SparseArray[{{i_, j_} /; j - i ≤ 0 => Random[Integer, {1, 10}]}, {5, 5}] // MatrixForm

```

$$\begin{pmatrix} 9 & 0 & 0 & 0 & 0 \\ 7 & 1 & 0 & 0 & 0 \\ 3 & 9 & 8 & 0 & 0 \\ 10 & 7 & 3 & 7 & 0 \\ 8 & 4 & 7 & 3 & 8 \end{pmatrix}$$

Come ultimo esempio sulle matrici sparse, si riporta un ulteriore caso in cui la forma sparsa introdotta con `SparseArray` permette di eseguire calcoli praticamente impossibili con la equivalente forma completa. Si costruisce una matrice con 10^{10} elementi.

```
n = 105;
```

```

A = SparseArray[{{i_, j_} /; Abs[i - j] ≤ 1 => Random[]}, {n, n}]
SparseArray[<299998>, {100000, 100000}]

```

In questo caso non si può creare la forma completa in quanto non vi è memoria sufficiente sul computer. Si effettueranno solo i calcoli dei byte impiegati da entrambe le forme per verificare la convenienza della forma sparsa. La matrice in forma completa deve memorizzare $10^5 \times 10^5$ numeri reali generati casualmente con la `Random[]`, ciascuno dei quali richiede 16 byte di memoria.

```

ByteCount[Random[]]
16

```

Pertanto, il numero totale di byte necessari per salvare tutti gli elementi, compresi quelli nulli, è dato dal seguente calcolo:

```

MemoriaCompleta = N[106 106 16] Byte
1.6 × 1013 Byte

```

mentre la matrice sparsa occupa soltanto poco più di 40 MegaByte

```

MemoriaSparsa = N[ByteCount[A]] Byte
4.00035 × 106 Byte

```

In questo caso, oltre al vantaggio di ridurre la memoria, si ottiene anche quello del poter eseguire calcoli su matrici così grandi.

Si calcola ora un sistema lineare la cui matrice dei coefficienti è quella creata in precedenza, ed il vettore dei termini noti è creato con 10^5 numeri reali casuali. La velocità ottenuta è anche dovuta alla funzione `LinearSolve` che è in grado di applicare algoritmi ottimizzati per le matrici sparse.

```
b = Table[Random[], {n}];
```

```
Dimensions[A]
{100000, 100000}
```

```
Dimensions[b]
{100000}
```

```
Timing[ LinearSolve[ A, b ]; ]
{0.344 Second, Null}
```

Si è appena risolto un sistema di 100.000 equazioni in 100.000 di incognite, con una matrice dei coefficienti tridiagonale.

In questo caso anche il prodotto di matrici sparse è sorprendentemente veloce:

```
Timing[ A.A; ]
{0.031 Second, Null}
```

Se si dispone di una matrice in forma completa e la si vuole convertire nella forma sparsa, si può ancora utilizzare `SparseArray`

```
A = {{0, 2, 1, 0}, {9, -3, 0, 2}, {0, -3, 0, 1}, {0, 0, 2, 2}}
    {{0, 2, 1, 0}, {9, -3, 0, 2}, {0, -3, 0, 1}, {0, 0, 2, 2}}
```

```
MatrixForm[A]

$$\begin{pmatrix} 0 & 2 & 1 & 0 \\ 9 & -3 & 0 & 2 \\ 0 & -3 & 0 & 1 \\ 0 & 0 & 2 & 2 \end{pmatrix}$$

```

```
B = SparseArray[A]
SparseArray[<9>, {4, 4}]
```

```
MatrixForm[B]

$$\begin{pmatrix} 0 & 2 & 1 & 0 \\ 9 & -3 & 0 & 2 \\ 0 & -3 & 0 & 1 \\ 0 & 0 & 2 & 2 \end{pmatrix}$$

```

```
A - B
{{0, 0, 0, 0}, {0, 0, 0, 0}, {0, 0, 0, 0}, {0, 0, 0, 0}}
```

Infine, si segnala che mediante `SparseArray` è anche possibile creare matrici non necessariamente sparse, dove una parte delle entrate è comunque sempre uguale da un certo valore.

```
A = SparseArray[{{2, 3} → 1, {3, 2} → -1, {3, 3} → 3}, 3, x]  
SparseArray[<3>, {3, 3}, x]
```

```
MatrixForm[A]  

$$\begin{pmatrix} x & x & x \\ x & x & 1 \\ x & -1 & 3 \end{pmatrix}$$

```

```
A = SparseArray[{{2, 3} → 1, {3, 2} → -1, {3, 3} → 3}, 3, -10]  
SparseArray[<3>, {3, 3}, -10]
```

```
MatrixForm[A]  

$$\begin{pmatrix} -10 & -10 & -10 \\ -10 & -10 & 1 \\ -10 & -1 & 3 \end{pmatrix}$$

```

Computazioni numeriche

Mathematica 5 introduce miglioramenti significativi proprio nel calcolo numerico, includendo una nuova generazione di algoritmi e solutori per le equazioni differenziali ordinarie e alle derivate parziali, così come un supporto estensivo per le variabili in forma di vettori e matrici nei solutori numerici.

NDSolve

La funzione `NDSolve` è stata completamente riscritta, risultando così molto più flessibile e con migliori prestazioni. Mentre la maggior parte dei miglioramenti della versione 5 sono trasparenti e fruibili da tutti gli utenti, la nuova `NDSolve` sarà particolarmente apprezzata dagli utenti esperti, che ora possono scegliere, con il noto meccanismo delle `Options`, quale metodo applicare, possono impostare alcuni parametri della valutazione e monitorare il procedimento del solutore. Inoltre, `NDSolve` permette anche di implementare propri algoritmi di soluzione ed includerli nella funzione `NDSolve` stessa.

Alcuni dei più significativi miglioramenti sono:

- Nuovi metodi di soluzione incluso i metodi espliciti di *RungeKutta*, i metodi impliciti di *RungeKutta* per qualsiasi ordine, i metodi di estrapolazione di *Deufhard* per qualsiasi ordine, i metodi basati sulle formule di differenziazione all'indietro (BDF-based methods) ed il double step.
- Implementazioni più efficienti con miglioramenti della velocità di esecuzione per molti tipi di equazioni differenziali
- `NDSolve` ora è in grado di lavorare anche con vettori e matrici
- `NDSolve` ora è in grado di risolvere alcune equazioni algebrico-differenziali
- Le nuove opzioni `EvaluationMonitor` e `StepMonitor` permettono di monitorare il procedimento di risoluzione e raffinare la procedura adottata.

– `NDSolve` fornisce una struttura per includere ulteriori metodi, definiti dall'utente, scritti in linguaggio *Mathematica*.

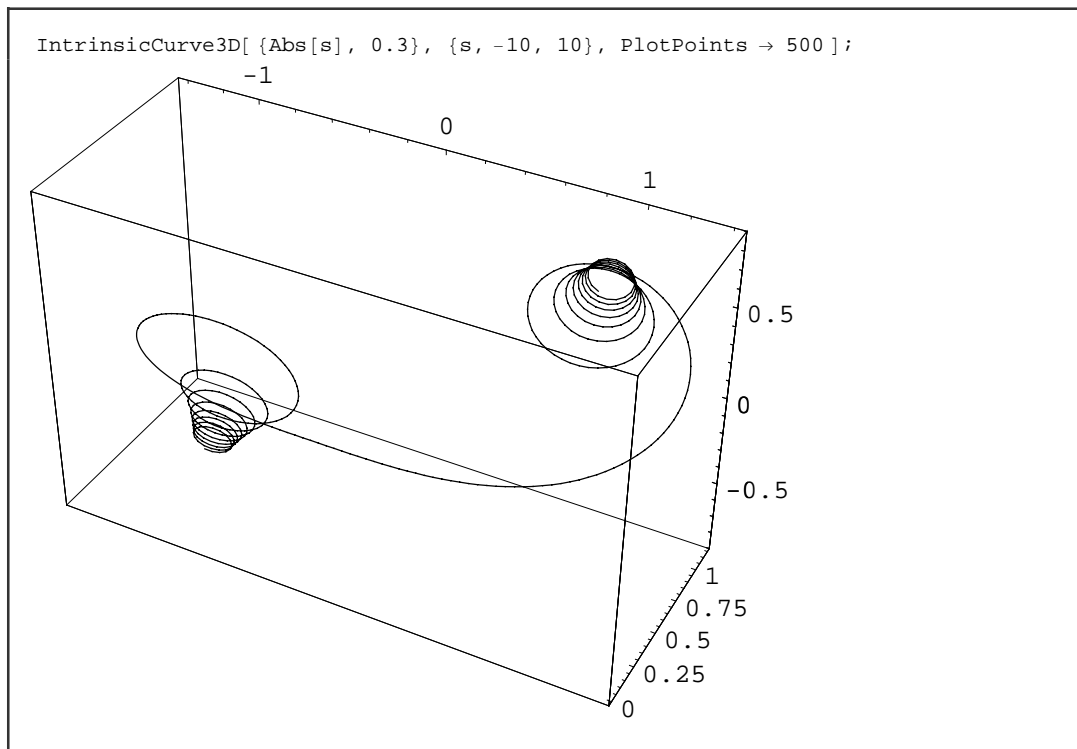
Esempio: risolvere un sistema di equazioni differenziali

L'esempio seguente mostra l'utilizzo di variabili vettoriali per specificare un sistema di equazioni differenziali di dimensione 12. Per visualizzare meglio i risultati ottenuti si utilizza una rappresentazione grafica in 3 dimensioni.

```
IntrinsicCurve3D[{κ_, τ_}, {s_, smin_, smax_},
  {s0_?NumericQ, x0_?VectorQ, t0_?VectorQ, n0_?VectorQ}, opts___?OptionQ] :=
Module[{x, t, n, b, s, c},
  c = x /. First@
  NDSolve[
  {
    x'[s] == t[s],           x[s0] == x0,
    t'[s] == κ n[s],        t[s0] == t0,
    n'[s] == τ b[s] - κ t[s], n[s0] == n0,
    b'[s] == -τ n[s],       b[s0] == Cross[t0, n0]
  },
  {x, t, n, b},
  {s, smin, smax}
  ];
  ParametricPlot3D[c[s], {s, smin, smax}, opts]
];
```

```
IntrinsicCurve3D[{κ_, τ_}, {s_, smin_, smax_}, opts___?OptionQ] :=
  IntrinsicCurve3D[{κ, τ},
    {s, smin, smax}, {0, {0, 0, 0}, {1, 0, 0}, {0, 1, 0}}, opts];
```

La soluzione seguente è una curva nello spazio definita da una curvatura ed una torsione.



Esempio: risolvere un'equazione differenziale matriciale

In questo esempio si risolve un'equazione differenziale matriciale con $X'(t) == A.X(t)$ con $A = -\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$ e la

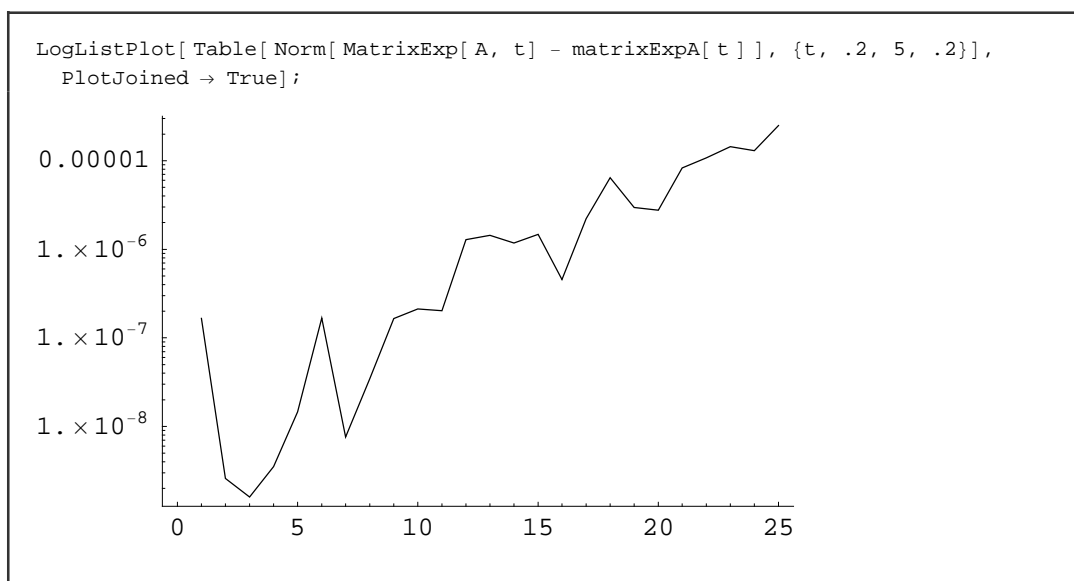
condizione iniziale $X(0) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$.

```
A = -{{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
```

```
matrixExpA =  
X /. First[NDSolve[X'[t] == A.X[t] ^ X[0] == IdentityMatrix[3], X, {t, 0, 5}]]  
InterpolatingFunction[{{0., 5.}}, <>]
```

Si può definire una particolare norma di matrice e osservare le differenze.

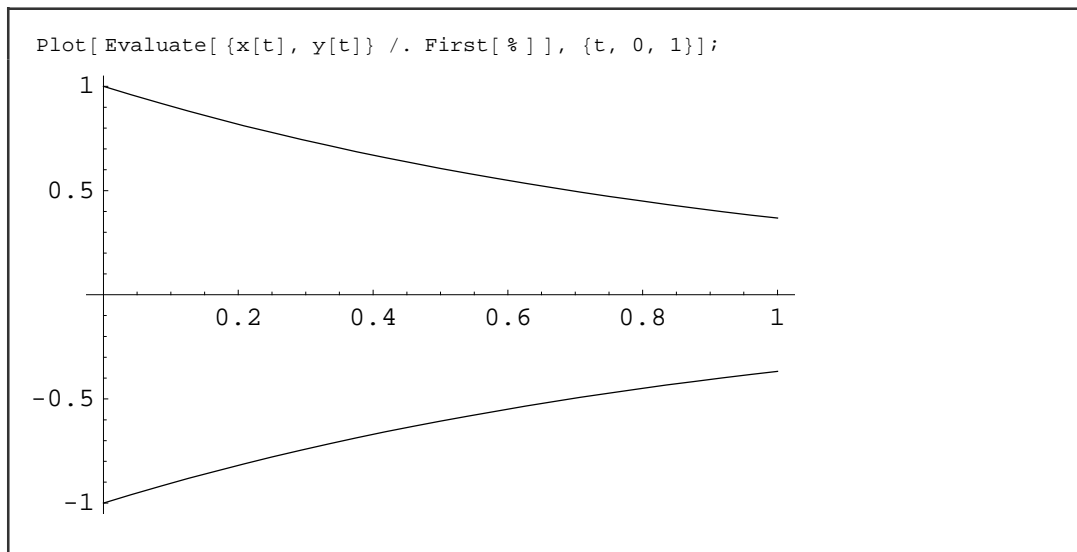
```
<< "Graphics`Graphics`"
```



Esempio: risolvere un'equazione algebrico-differenziale

In questo caso si ha un'equazione differenziale $x'(t) = x(t) + 2y(t)$ ed un'equazione algebrica esplicita $x(t) + y(t) = 0$. Si noti come, solo in questo caso, si imposta una equazione con valore iniziale $x(0) = 0$. La seconda equazione per la $y(0)$ viene implicitamente determinata dall'equazione algebrica.

```
NDSolve[ x'[t] == x[t] + 2y[t] ^ 0 == x[t] + y[t] ^ x[0] == 1, {x, y}, {t, 0, 1}]
{{x -> InterpolatingFunction[{{0., 1.}}, <>],
  y -> InterpolatingFunction[{{0., 1.}}, <>]}}
```



Esempio: risolvere un'equazione differenziale parziale

Un soliton è un'onda solitaria che supera le collisioni con altre onde solitarie. Ad esempio le onde degli oceani sono soliton. Le prime equazioni differenziali che hanno descritto un soliton sono quelle di *Korteweg-de Vries*. Le applicazioni tecniche dei soliton sono ad esempio di interesse per i sistemi di trasmissione a fibra ottica su lunghe distanze.

Quella che segue è un esempio di equazione di *Korteweg-de Vries* (KdV) ad una dimensione spaziale.

$$\text{KdV}[\{\alpha_{-}, t_{-}, x_{-}\} := \text{D}[\psi[t, x], t] + \alpha \psi[t, x] \text{D}[\psi[t, x], x] + \text{D}[\psi[t, x], \{x, 3\}] = 0;$$

Segue una soluzione di onda solitaria per l'equazione KdV sopra riportata.

$$\text{S}[\{\alpha_{-}, c0_{-}, x0_{-}\}, t_{-}, x_{-}] := \frac{3 c0}{\alpha} \left(\text{Sech} \left[\frac{\sqrt{c0}}{2} (x - x0 - c0 t) \right] \right)^2;$$

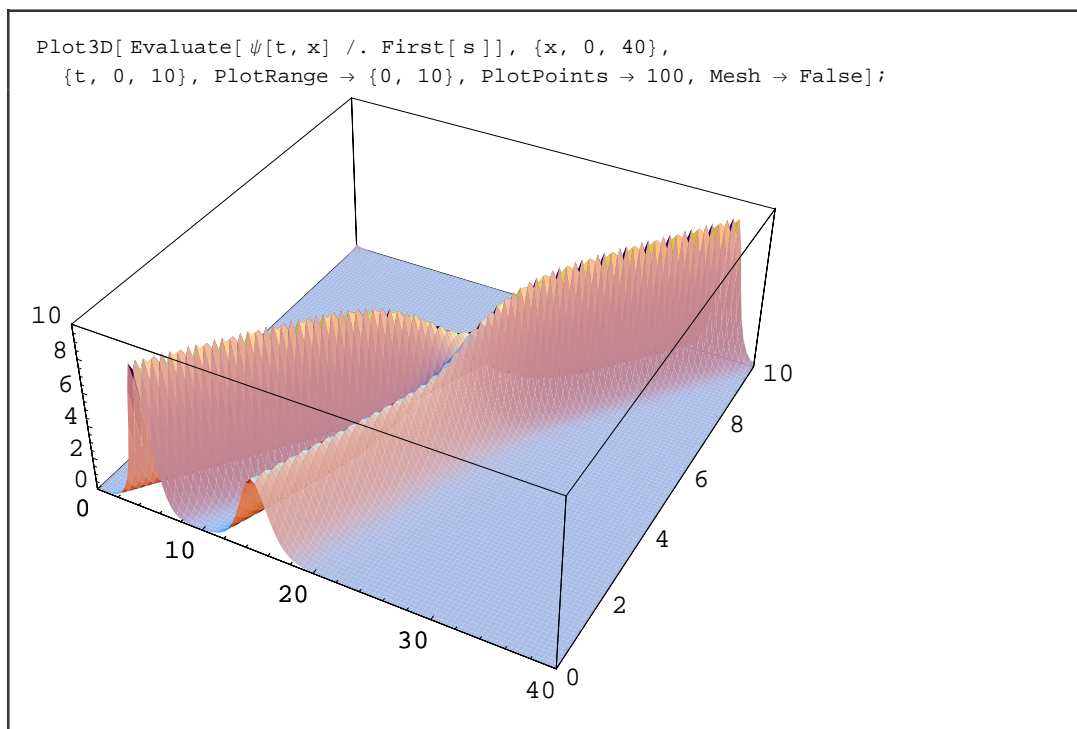
Poiché si dovranno utilizzare condizioni al contorno periodiche, si aggiungeranno tutti i contributi periodici per ottenere le seguenti condizioni iniziali.

$$\Phi = \text{S}[\{1, 3, 5\}, 0, x] + \text{S}[\{1, 3/2, 15\}, 0, x] + \\ \text{S}[\{1, 3, 5 + 40\}, 0, x] + \text{S}[\{1, 3/2, 15 + 40\}, 0, x] + \\ \text{S}[\{1, 3, 5 - 40\}, 0, x] + \text{S}[\{1, 3/2, 15 - 40\}, 0, x];$$

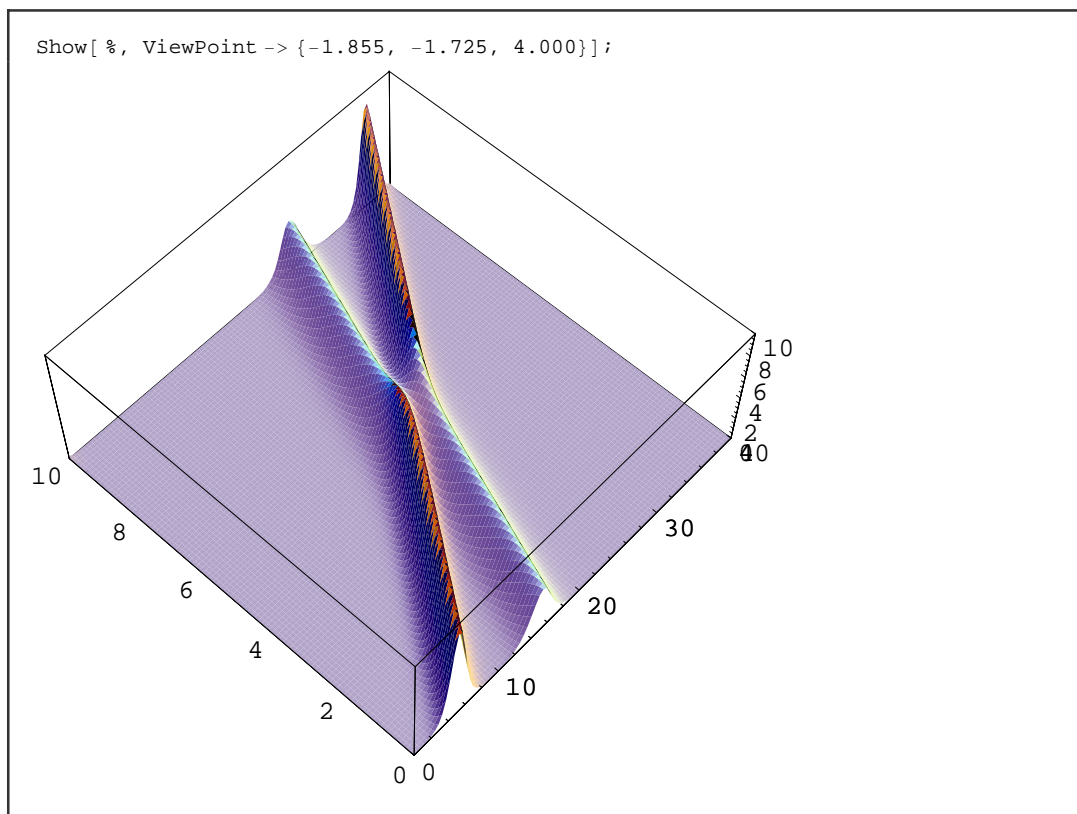
$$(\Phi /. x \rightarrow 0.) == (\Phi /. x \rightarrow 40.) \\ \text{True}$$

Ora si può osservare come queste onde solitarie interagiscono.

```
s = NDSolve[KdV[{1}, t, x] && \psi[0, x] == \Phi && \psi[t, 0] == \psi[t, 40],
  \psi, {t, 0, 10}, {x, 0, 40}, Method -> StiffnessSwitching]; // Timing
{5.516 Second, Null}
```



Si noti l'interazione non lineare quando l'una sorpassa l'altra.



FindRoot

La funzione `FindRoot` ora supporta variabili vettoriali e matriciali ed include nuovi algoritmi ed alcuni miglioramenti che permettono di manipolare problemi su larga scala anche su computer con risorse di memoria limitate.

Esempio: risolvere l'equazione di Riccati in forma matriciale

Questo esempio mostra la soluzione di un'equazione matriciale, un'equazione algebrica di Riccati $Q + A^T P + P A - P B R^{-1} B^T P = 0$.

Si supponga di avere un sistema e dei vincoli:

```
A = {{0, 1}, {0, 0}}; B = {{0}, {1}};
```

```
Q = {{1, 0}, {0, 1}}; R = {{1}};
```

Il comando seguente trova la matrice radice dell'equazione data

```
FindRoot[Transpose[A] . P + P . A - P . B . Inverse[R] . Transpose[B] . P == - Q,
  {P, {{10, 2}, {2, 10}}}]
{P -> {{1.73205, 1.}, {1., 1.73205}}}
```

FindFit

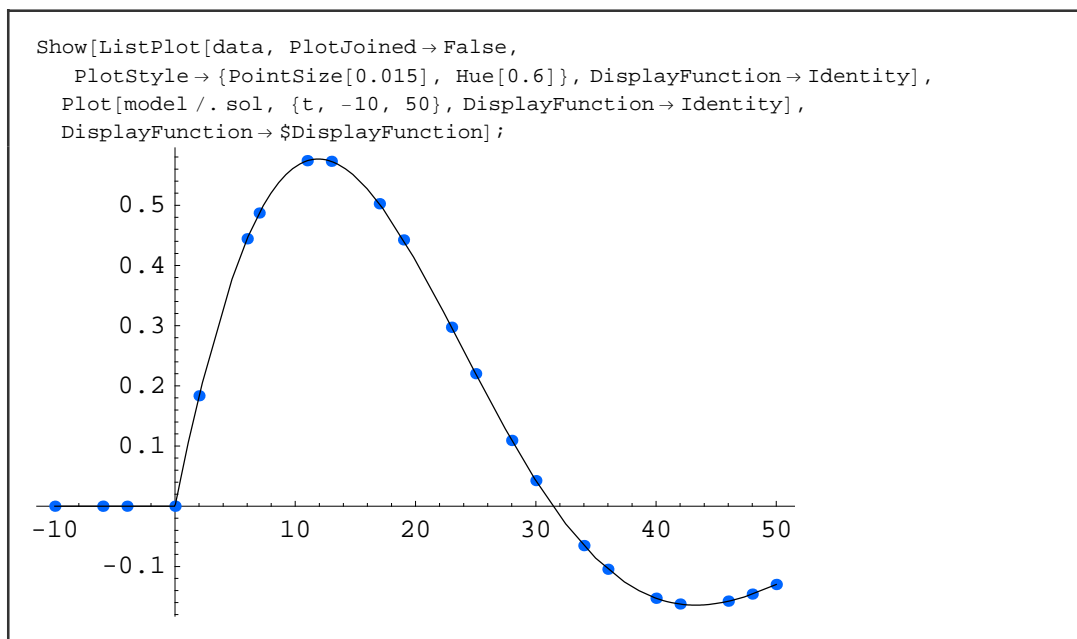
Questa nuova funzione supera ed estende le funzionalità della coppia `Fit` e `NonlinearFit`. `FindFit` supporta variabili in forma di vettori e matrici, fornisce più metodi e permette di utilizzare le opzioni `StepMonitor` e `EvaluationMonitor` per tenere traccia del processo.

Esempio: data-fitting

`FindFit` può eseguire il data-fitting anche utilizzando una funzione continua a tratti

```
data = Table[{t, Which[t < 0, 0, True, e-0.04 t Sin[0.1 t]]}, {t, -10, 50}];
model = Which[t < 0, 0, True, e-a t Sin[b t]];
sol = FindFit[data, model, {{a, 1}, {b, 0.1}}, {t}]
{a -> 0.04, b -> 0.1}
```

I dati possono essere mostrati insieme al risultato ottenuto.

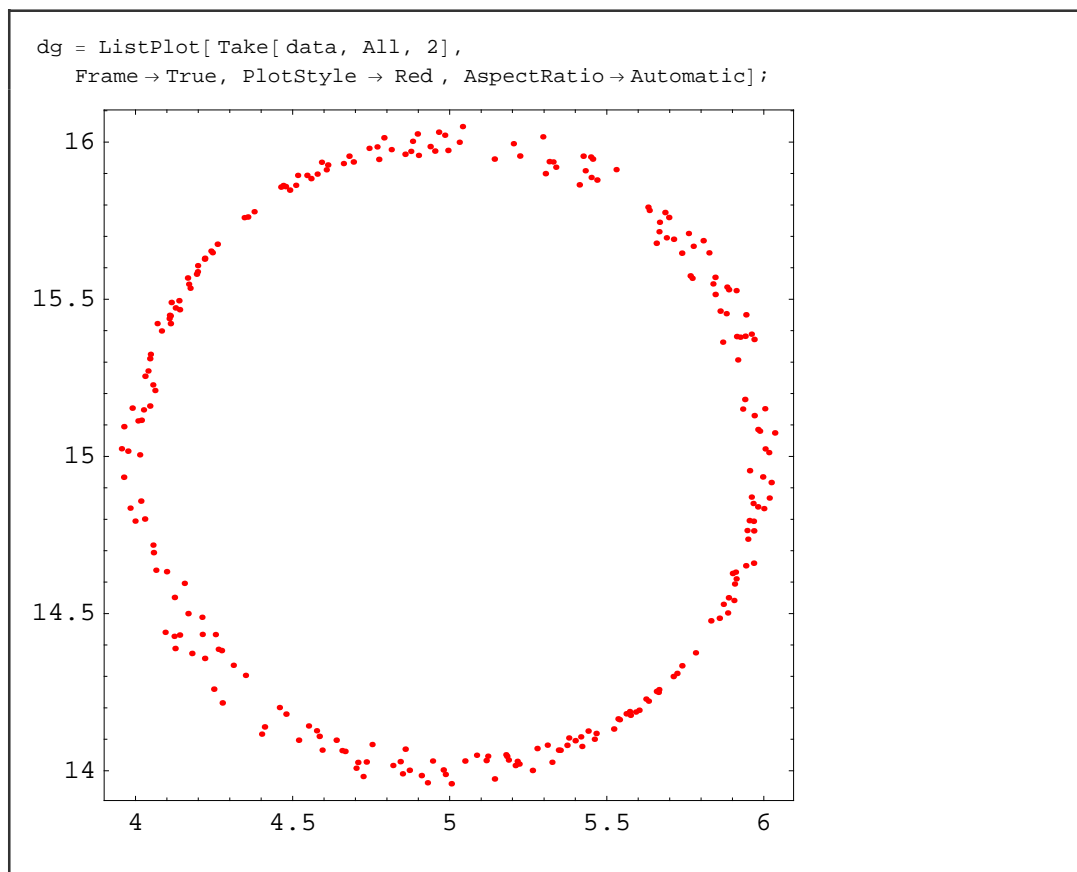


Esempio: dati sperimentali su una circonferenza

Si supponga di eseguire delle misurazioni lungo una circonferenza

```
data = Table[
  {Cos[t] + 5, Sin[t] + 15, 0.} + Random[Real, {-.05, 0.05}], {t, 0., 2 Pi, 0.01}];

<< "Graphics`Colors`"
```



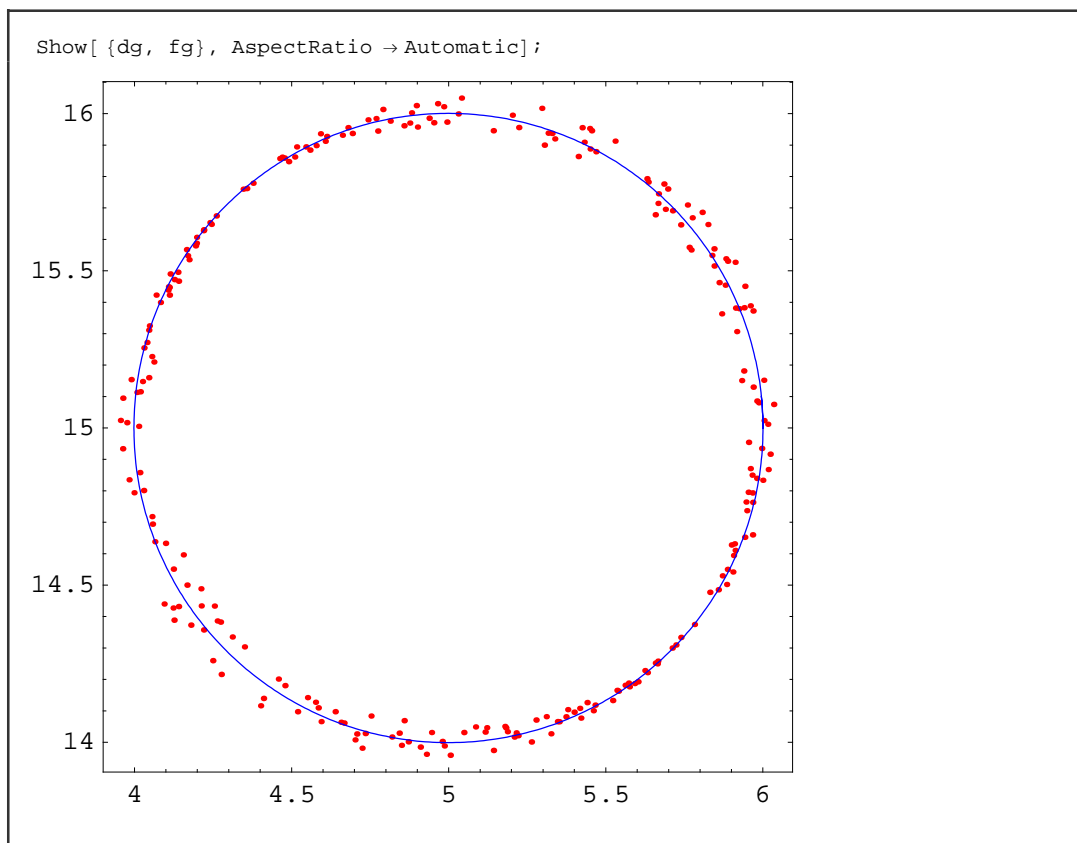
Il codice seguente cerca di adattare una circonferenza con raggio R e centro in $\{x_0, y_0\}$ ai dati misurati

```
Clear[R]
```

```
s = FindFit[data, (x - x0)2 + (y - y0)2 - R2, {x0, y0, R}, {x, y}]
{x0 -> 4.99928, y0 -> 14.9997, R -> 1.00093}
```

```
fg = Show[Graphics[{RGBColor[0, 0, 1], Circle[{x0 /. s, y0 /. s}, R /. s]}],
  DisplayFunction -> Identity];
```

Si mostrano i dati misurati e la circonferenza.



Computazioni simboliche

Anche per il calcolo simbolico, *Mathematica* 5 introduce nuove funzionalità. Si evidenziano tra questi, i miglioramenti nel calcolo simbolico di equazioni e disequazioni con numeri complessi, reali e interi, il solutore di equazioni differenziali algebriche ed equazioni di ricorrenza ed un migliore e più ampio supporto per le specifiche del dominio da parte dell'utente.

DSolve

La funzione `DSolve` è ora in grado di individuare tutte le soluzioni razionali di un sistema di equazioni lineari con coefficienti razionali e può risolvere i sistemi lineari di equazioni differenziali con coefficienti costanti.

Esempio: sistema lineare con coefficienti razionali

L'esempio che segue mostra la soluzione di un sistema di equazioni differenziali i cui coefficienti sono funzioni razionali e la cui soluzione è razionale.

```
DSolve[
{
  u'[t] ==  $\frac{(6 + 2t - 3t^3 - t^5) u[t]}{t(-3 - 2t + t^2)(-1 + t^3)} + \frac{(5 + t) v[t]}{(-3 - 2t + t^2)(-1 + t^3)}$ ,
  v'[t] ==  $-\frac{4t(-3 - 2t + t^2) u[t]}{(5 + t)(-1 + t^3)} + \frac{(1 + 20t^2 + 3t^3) v[t]}{(5 + t)(-1 + t^3)}$ ,
},
{u, v},
t
]
{{u -> Function[{t},  $\frac{t C[1]}{-3 - 2t + t^2} + \frac{t^2 C[2]}{-3 - 2t + t^2}$ ], v -> Function[{t},  $\frac{C[1]}{5 + t} + \frac{t^4 C[2]}{5 + t}$ ]}}
```

In aggiunta, `DSolve` può anche risolvere alcune equazioni singolari o algebrico-differenziali. In particolare, sistemi di equazioni lineari della forma

$$A \cdot x'[t] + B \cdot x[t] = g[t]$$

dove A è singolare. Si noti che la soluzione generale di questa equazione algebrico-differenziale ha un solo parametro sebbene sia un sistema di ordine due. Un grado di libertà viene rimosso per via dell'equazione algebrica che lega le variabili.

```
DSolve[x'[t] == x[t] + 2y[t] & 0 == x[t] + y[t], {x, y}, t]
{{x -> Function[{t},  $\frac{1}{4} e^{-t} C[1]$ ], y -> Function[{t},  $-\frac{1}{4} e^{-t} C[1]$ ]}}
```

In questo caso si ha la libertà di scegliere un solo valore iniziale con valore arbitrario

```
DSolve[x'[t] == x[t] + 2y[t] & 0 == x[t] + y[t] & x[0] == 1, {x, y}, t]
{{x -> Function[{t},  $e^{-t}$ ], y -> Function[{t},  $-e^{-t}$ ]}}
```

Quello che segue è il valore iniziale implicato dal valore iniziale dell'altra variabile

```
y[0] /. First[%]
-1
```

RSolve

`RSolve` può risolvere sistemi di equazioni alle differenze lineari e non lineari, equazioni algebriche alle differenze ed equazioni alle differenze con derivate parziali. Può anche risolvere equazioni del tipo q-difference e divide-and-conquer.

Proprio come nel caso di equazioni differenziali, le equazioni alle differenze anche se appaiono semplici, potrebbero generare soluzioni molto complicate, e non ci sono funzioni abbastanza generali per esprimere tutte le soluzioni in forma chiusa. In molti casi, comunque, possono essere trovate soluzioni in forma chiusa, come negli esempi che seguono.

Example: Legendre Polynomials

`RSolve` riconosce l'equazione alle differenze per i polinomi di Legendre.

```
RSolve[u[x] - ((3 + 2*x) * (d + c*z) * u[1 + x]) / (1 + x) +
((2 + x) * u[2 + x]) / (1 + x) == 0, u, x]
{{u -> Function[{x}, C[1] LegendreP[x, d + c z] + C[2] LegendreQ[x, d + c z]]}}
```

Esempio: soluzioni generali e particolari per l'equazione $y_{k+1} = (k+1) y_k$

Di seguito si riporta la soluzione generale dell'equazione $y_{k+1} = (k+1) y_k$ con una costante arbitraria `C[1]`.

```
RSolve[y[k + 1] == (k + 1) y[k], y, k]
{{y -> Function[{k}, C[1] Gamma[1 + k]]}}
```

Si risolve l'equazione alle differenze $y_{k+1} = (k+1) y_k$ con l'equazione iniziale $y_0 = 1$

```
RSolve[y[k + 1] == (k + 1) y[k] && y[0] == 1, y, k]
{{y -> Function[{k}, Gamma[1 + k]]}}
```

Si mostra ora che la soluzione soddisfa l'equazione alle differenze e la condizione iniziale

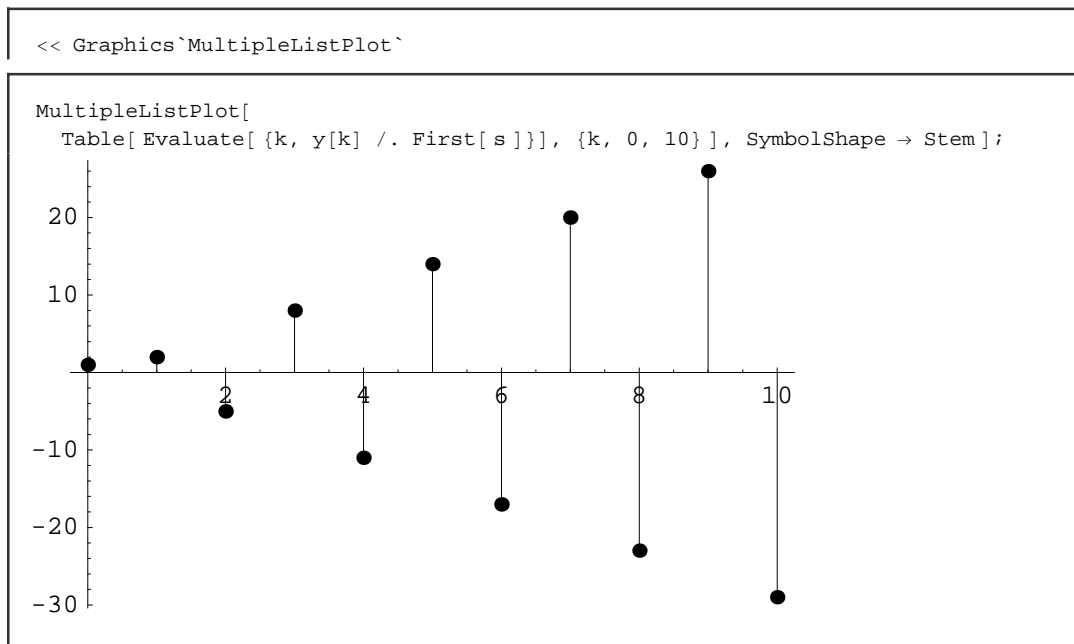
```
y[k + 1] == (k + 1) y[k] && y[0] == 1 /. % // FullSimplify
{True}
```

Esempio: equazione alle differenze del secondo ordine con condizioni al contorno

Si risolve un'equazione alle differenze del secondo ordine con due condizioni al contorno

```
s = RSolve[ y[k+2] + 2 y[k+1] + y[k] == 0 && y[0] == 1 && y[1] == 2, y, k ]
{{y -> Function[{k}, -(-1)^k (-1 + 3 k)]}}
```

È possibile visualizzare graficamente la soluzione con un tipico diagramma stem (diagramma a rami).



Esempio: equazioni lineari del secondo ordine con coefficienti variabili

Questo esempio mostra la soluzione dell'equazione $y(n) n^2 + (n^2 + 1) y(n+1) + y(n+2) = 0$, un'equazione lineare del secondo ordine con coefficienti variabili.

```
RSolve[ y[n+2] + (n^2 + 1) y[n+1] + n^2 y[n] == 0, y, n ]
{{y -> Function[{n}, -(-1)^n C[1] - (-1)^n C[2] (HypergeometricPFQ[{1, 1, 1}, {}, 1] - Gamma[n]^2 HypergeometricPFQ[1, n, n], {}, 1])}}
```

Esempio: sistema di equazioni alle differenze e algebriche

Si risolve un sistema lineare di equazioni alle differenze e equazioni algebriche. Si noti che questo sistema ha solo una costante generale nella soluzione, poiché un grado di libertà è stato rimosso dall'equazione algebrica.

```
RSolve[ x[k+1] == x[k] + 2 y[k] ^ 0 == x[k] + y[k], {x, y}, k ]
{{x -> Function[{k}, 1/4 (-1)^k C[1]], y -> Function[{k}, -1/4 (-1)^k C[1]}}
```

Di seguito si riporta il problema del valore iniziale con un solo valore iniziale assegnato

```
RSolve[ x[k+1] == x[k] + 2 y[k] ^ 0 == x[k] + y[k] ^ x[0] == 1, {x, y}, k]
{{x -> Function[{k}, (-1)^k], y -> Function[{k}, -(-1)^k]}}
```

Questo è il valore iniziale di y .

```
y[0] /. First[%]
-1
```

Esempio: equazioni alle differenze parziali

Le equazioni alle differenze parziali hanno diverse variabili indipendenti. In tali casi, la soluzione generale è ottenuta parametrizzando la funzione generale.

Segue una equazione alle differenze parziali lineare, con una funzione arbitraria generale $C[l][l - k]$.

```
RSolve[ u[k+1, l+1] == a u[k, l], u, {k, l}]
{{u -> Function[{k, l}, a^{-1+k} C[l] [-k+1] ]}}
```

Reduce

La funzione `Reduce` è stata estesa ed ora risolve equazioni che comprendono qualsiasi combinazione di uguaglianze, disuguaglianze, operatori esistenziali, operatori generali e specifiche a riguardo del dominio delle variabili.

Esempio: risolvere l'equazione $x^2 - 2y^2 = 1$ su differenti domini

In questo esempio si risolve l'equazione $x^2 - 2y^2 = 1$ su differenti domini.

Risoluzione dell'equazione nel campo dei complessi.

```
Reduce[ x^2 - 2 y^2 == 1, {x, y}, Complexes ]
y == -\frac{\sqrt{-1+x^2}}{\sqrt{2}} || y == \frac{\sqrt{-1+x^2}}{\sqrt{2}}
```

Risoluzione della stessa equazione nel campo dei reali; la soluzione viene descritta in termini di disuguaglianze.

```
Reduce[ x^2 - 2 y^2 == 1, {x, y}, Reals ]
\left( x \leq -1 \ \&\& \left( y == -\frac{\sqrt{-1+x^2}}{\sqrt{2}} || y == \frac{\sqrt{-1+x^2}}{\sqrt{2}} \right) \right) || \left( x \geq 1 \ \&\& \left( y == -\frac{\sqrt{-1+x^2}}{\sqrt{2}} || y == \frac{\sqrt{-1+x^2}}{\sqrt{2}} \right) \right)
```

Risolvere la stessa equazione sul campo degli interi richiede l'introduzione di ulteriori parametri per descrivere la soluzione. Queste equazioni sono chiamate equazioni diofantine. L'esempio seguente mostra una forma speciale chiamata equazione di Pell.

$$\begin{aligned}
 & \text{Reduce}[x^2 - 2y^2 = 1, \{x, y\}, \text{Integers}] \\
 & \left(C[1] \in \text{Integers} \ \&\& \ C[1] \geq 0 \ \&\& \right. \\
 & \quad \left. x = \frac{1}{2} \left(-(3 - 2\sqrt{2})^{C[1]} - (3 + 2\sqrt{2})^{C[1]} \right) \ \&\& \ y = -\frac{(3 - 2\sqrt{2})^{C[1]} - (3 + 2\sqrt{2})^{C[1]}}{2\sqrt{2}} \right) \ || \\
 & \left(C[1] \in \text{Integers} \ \&\& \ C[1] \geq 0 \ \&\& \ x = \frac{1}{2} \left(-(3 - 2\sqrt{2})^{C[1]} - (3 + 2\sqrt{2})^{C[1]} \right) \ \&\& \right. \\
 & \quad \left. y = \frac{(3 - 2\sqrt{2})^{C[1]} - (3 + 2\sqrt{2})^{C[1]}}{2\sqrt{2}} \right) \ || \\
 & \left(C[1] \in \text{Integers} \ \&\& \ C[1] \geq 0 \ \&\& \ x = \frac{1}{2} \left((3 - 2\sqrt{2})^{C[1]} + (3 + 2\sqrt{2})^{C[1]} \right) \ \&\& \right. \\
 & \quad \left. y = -\frac{(3 - 2\sqrt{2})^{C[1]} - (3 + 2\sqrt{2})^{C[1]}}{2\sqrt{2}} \right) \ || \left(C[1] \in \text{Integers} \ \&\& \ C[1] \geq 0 \ \&\& \right. \\
 & \quad \left. x = \frac{1}{2} \left((3 - 2\sqrt{2})^{C[1]} + (3 + 2\sqrt{2})^{C[1]} \right) \ \&\& \ y = \frac{(3 - 2\sqrt{2})^{C[1]} - (3 + 2\sqrt{2})^{C[1]}}{2\sqrt{2}} \right)
 \end{aligned}$$

Si risolve ora un'equazione nel campo degli interi modulo 12

$$\begin{aligned}
 & \text{Reduce}[x^2 - 2y^2 = 1, \{x, y\}, \text{Modulus} \rightarrow 12] \\
 & (x = 1 \ \&\& \ y = 0) \ || \ (x = 1 \ \&\& \ y = 6) \ || \ (x = 3 \ \&\& \ y = 2) \ || \ (x = 3 \ \&\& \ y = 4) \ || \\
 & (x = 3 \ \&\& \ y = 8) \ || \ (x = 3 \ \&\& \ y = 10) \ || \ (x = 5 \ \&\& \ y = 0) \ || \ (x = 5 \ \&\& \ y = 6) \ || \\
 & (x = 7 \ \&\& \ y = 0) \ || \ (x = 7 \ \&\& \ y = 6) \ || \ (x = 9 \ \&\& \ y = 2) \ || \ (x = 9 \ \&\& \ y = 4) \ || \\
 & (x = 9 \ \&\& \ y = 8) \ || \ (x = 9 \ \&\& \ y = 10) \ || \ (x = 11 \ \&\& \ y = 0) \ || \ (x = 11 \ \&\& \ y = 6)
 \end{aligned}$$

Ora, invece, si risolve la stessa equazione sul dominio delle x reali e delle y complesse

$$\begin{aligned}
 & \text{Reduce}[x^2 - 2y^2 = 1 \ \&\& \ x \in \text{Reals} \ \&\& \ y \in \text{Complexes}, \{x, y\}] \\
 & \left(x < -1 \ \&\& \ \left(y = -\frac{\sqrt{-1+x^2}}{\sqrt{2}} \ || \ y = \frac{\sqrt{-1+x^2}}{\sqrt{2}} \right) \right) \ || \\
 & (x = -1 \ \&\& \ y = 0) \ || \ \left(-1 < x < 1 \ \&\& \ \left(y = -\frac{i\sqrt{1-x^2}}{\sqrt{2}} \ || \ y = \frac{i\sqrt{1-x^2}}{\sqrt{2}} \right) \right) \ || \\
 & (x = 1 \ \&\& \ y = 0) \ || \ \left(x > 1 \ \&\& \ \left(y = -\frac{\sqrt{-1+x^2}}{\sqrt{2}} \ || \ y = \frac{\sqrt{-1+x^2}}{\sqrt{2}} \right) \right)
 \end{aligned}$$

Esempio: equazioni con radici multiple infinite

Reduce genera le soluzioni complete. Di seguito si riporta un esempio nel quale ci sono radici multiple infinite.

```
Reduce[ Tan[2 x]^2 + Cot[2 x]^2 + 2 Tan[2 x] + 2 Cot[2 x] == 6 && x ∈ Reals, x ]
C[1] ∈ Integers &&
( x ==  $\frac{1}{8} (\pi + 4 \pi C[1])$  || x ==  $\frac{1}{24} (-5 \pi + 12 \pi C[1])$  || x ==  $\frac{1}{24} (-\pi + 12 \pi C[1])$  )
```

Esempio equazioni che includono operatori

Un altro tipo di equazione che la `Reduce` è in grado di trattare è quello che comprende i quantificatori come ad esempio quello esistenziale (\exists) e quello universale (\forall), definiti da *Mathematica* rispettivamente con i termini `Exists` (\exists) and `ForAll` (\forall).

L'esempio mostra la determinazione dei valori di a e b che rendono il polinomio positivo per tutti i valori reali della x .

```
Reduce[ ForAll[ x, x ∈ Reals, x^2 + a x + b ≥ 0 ], {a, b}, Reals ]
b ≥  $\frac{a^2}{4}$ 
```

Resolve

La funzione `Resolve` è in grado di eliminare i quantificatori da qualsiasi sistema polinomiale in variabili reali o complesse utilizzando gli stessi metodi della funzione `Reduce`. Nei casi in cui è più semplice individuare una forma senza la presenza dei quantificatori piuttosto che calcolare le soluzioni esplicitamente, `Resolve` restituisce la forma implicita. `Resolve` può anche eliminare quantificatori relativi a variabili booleane.

The function `Resolve` can eliminate quantifiers from arbitrary polynomial systems in complex or real variables using the same methods used by the function `Reduce`. For cases in which obtaining an implicit quantifier-free form of the system is easier than computing explicit solutions, `Resolve` returns the implicit form. `Resolve` can also eliminate quantifiers involving Boolean variables.

Esempio: coefficienti reali

In questo caso `Resolve` restituisce le condizioni sui coefficienti reali (a, b, c) per cui il polinomio nella variabile reale x è sempre positivo.

```
Resolve[ ForAll[ x, {x, a, b, c} ∈ Reals, c x^2 + b x + a > 0 ] ]
(a | b) ∈ Reals && ((c > 0 && -b^2 c + 4 a c^2 > 0) ||
(a > 0 && b = 0 && c = 0) || (a > 0 && b = 0 && c ≥ 0 && -b^2 c + 4 a c^2 > 0) )
```

Esempio: la disequazione $x^2 + y^2 < 1$ implica che $y > x^4 - 2$?

In questo caso si desidera verificare $x^2 + y^2 < 1$ implica che $y > x^4 - 2$.

```
Resolve[  $\forall_{\{x,y\}}$  Implies[ $x^2 + y^2 < 1$ ,  $y > x^4 - 2$ ] ]  
True
```

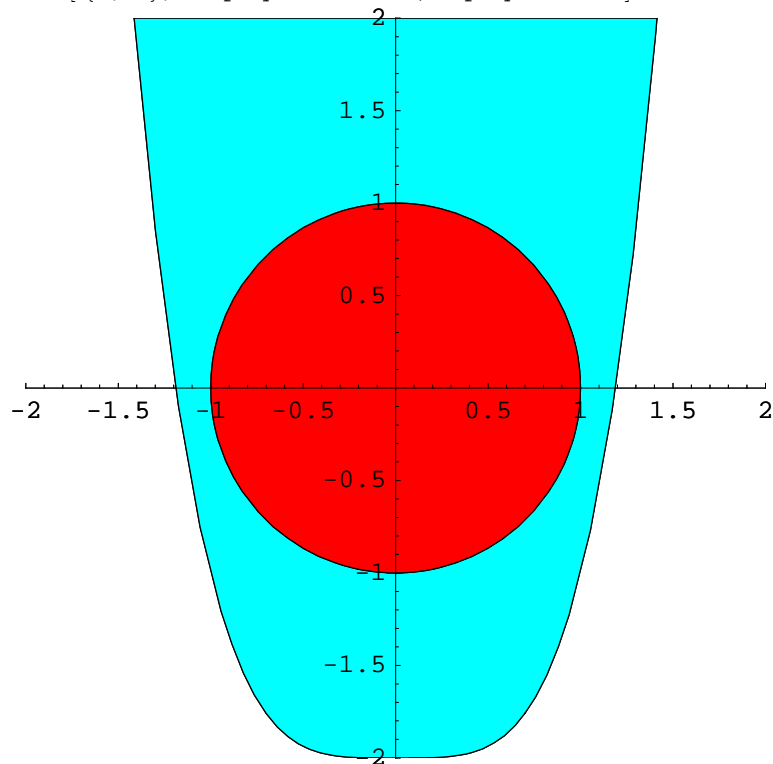
La stessa verifica può anche essere eseguita graficamente.

```
<< "Graphics`InequalityGraphics`"
```

```
A = InequalityPlot[  $x^2 + y^2 < 1$ , {x}, {y},  
PlotRange → {{-2, 2}, {-2, 2}},  
Fills → {{1, 2}, RGBColor[1, 0, 0]}},  
DisplayFunction → Identity];
```

```
B = InequalityPlot[  $y > x^4 - 2$ , {x, -2, 2}, {y, -2, 2},  
PlotRange → {{-2, 2}, {-2, 2}},  
DisplayFunction → Identity];
```

```
Show[ {B, A}, DisplayFunction → $DisplayFunction];
```



FindInstance

La nuova funzione `FindInstance` accetta gli stessi input della `Reduce` ma restituisce, al massimo, un numero prefissato dall'utente, di possibili soluzioni. Mentre la `Reduce` restituisce le condizioni sulle variabili che indicano la forma completa della soluzione, `FindInstance` restituisce un insieme di valori soluzione. `FindInstance` lavora su tutti i problemi per i quali la `Reduce` è in grado di restituire soluzioni complete. Per problemi in cui la soluzione completa non è necessaria, trovare un insieme di soluzioni può essere più veloce del cercare di ottenere la forma generale della soluzione. In alcuni casi `FindInstance` è in grado di individuare soluzioni per alcuni problemi che `Reduce` non riesce a risolvere.

Esempio: trovare le soluzioni reali dell'equazione $x^2 - 2y^2 = 1$

In questo esempio si utilizza la `Reduce` per trovare la forma generale per tutte le soluzioni reali dell'equazione $x^2 - 2y^2 = 1$.

```
Reduce[ x^2 - 2 y^2 == 1, {x, y}, Reals ]
( x ≤ -1 && ( y == - $\frac{\sqrt{-1+x^2}}{\sqrt{2}}$  || y ==  $\frac{\sqrt{-1+x^2}}{\sqrt{2}}$  ) ) || ( x ≥ 1 && ( y == - $\frac{\sqrt{-1+x^2}}{\sqrt{2}}$  || y ==  $\frac{\sqrt{-1+x^2}}{\sqrt{2}}$  ) )
```

Con `FindInstance` si richiede di individuare tre coppie di soluzioni numeriche dell'equazione sopra riportata.

```
FindInstance[ x^2 - 2 y^2 == 1, {x, y}, Reals, 3]
{ {x → -122, y → 11  $\sqrt{\frac{123}{2}}$  }, {x → -99, y → -70}, {x → 1, y → 0} }
```

Maximize/Minimize

Le funzioni `Maximize` e `Minimize` trovano esattamente il massimo ed il minimo globali per una funzione su una determinata regione. Generalmente queste funzioni operano su input polinomiali; comunque possono trattare anche input trascendentali.

Esempio: proprietà geometriche di base

In questo caso `Maximize` mostra che il rettangolo con la massima area su di una circonferenza è un quadrato.

```
Maximize[ {x y, 2 x + 2 y == 1}, {x, y}]
{  $\frac{1}{16}$ , {x →  $\frac{1}{4}$ , y →  $\frac{1}{4}$  } }
```

Calcolo del massimo volume del cilindro quando l'area della superficie è 24π .

```
Maximize[{ $\pi r^2 h$ ,  $2 \pi r^2 + 2 \pi r h == 24 \pi$  &&  $r \geq 0$  &&  $h \geq 0$ }, {r, h}]
```

```
{16  $\pi$ , {r  $\rightarrow$  2, h  $\rightarrow$  4}}
```

Esempio: problema senza vincoli

Minimize e Maximize restituiscono le liste dei valori ottenuti nei punti di minimo e di massimo, così come le regole che specificano quando tali valori vengono raggiunti.

Il minimo di una funzione quadratica

```
Minimize[x^2 - 3 x + 6, x]
```

```
{ $\frac{15}{4}$ , {x  $\rightarrow$   $\frac{3}{2}$ }}
```

Per verificare il risultato basta applicare la regola ottenuta in output alla funzione e si ottiene il valore del minimo.

```
x^2 - 3 x + 6 /. Last[%]
```

```
 $\frac{15}{4}$ 
```

Calcolo del massimo rispetto ad x ed y .

```
Maximize[5 x y - x^4 - y^4, {x, y}]
```

```
{ $\frac{25}{8}$ , {x  $\rightarrow$   $-\frac{\sqrt{5}}{2}$ , y  $\rightarrow$   $-\frac{\sqrt{5}}{2}$ }}
```

Esempio: problema con vincoli

Minimize[expr, x] minimizza l'espressione expr considerando la x che varia da $-\infty$ a $+\infty$.

Minimize[{expr, cons}, x] consente di indicare dei vincoli sulla variabile x che devono essere rispettati nell'individuazione del minimo dell'espressione expr. I vincoli possono essere inseriti sotto forma di qualsiasi combinazione di equazioni e disequazioni.

Si calcola il minimo soggetto al vincolo $x \geq 3$.

```
Minimize[{x^2 - 3 x + 6, x  $\geq$  3}, x]
```

```
{6, {x  $\rightarrow$  3}}
```

Si calcola il massimo all'interno della circonferenza unitaria.

```
Maximize[{5 x y - x^4 - y^4, x^2 + y^2  $\leq$  1}, {x, y}]
```

```
{2, {x  $\rightarrow$   $-\frac{1}{\sqrt{2}}$ , y  $\rightarrow$   $-\frac{1}{\sqrt{2}}$ }}
```

Si calcola il massimo all'interno di un'ellisse. Il risultato è abbastanza complicato.

```

Maximize[{5 x y - x^4 - y^4, x^2 + 2 y^2 ≤ 1}, {x, y}]
{-Root[-811219 + 320160 #1 + 274624 #1^2 - 170240 #1^3 + 25600 #1^4 &, 1],
{x → Root[25 - 102 #1^2 + 122 #1^4 - 70 #1^6 + 50 #1^8 &, 2],
y → Root[25 - 264 #1^2 + 848 #1^4 - 1040 #1^6 + 800 #1^8 &, 1]}}

```

Si calcola il massimo lungo una retta.

```

Maximize[{5 x y - x^4 - y^4, x + y == 1}, {x, y}]
{9/8, {x → 1/2, y → 1/2}}

```

Minimize e Maximize possono risolvere qualsiasi problema di programmazione lineare nel quale la funzione obiettivo ed i vincoli siano espressioni lineari nelle variabili x_i . Di seguito si mostra un tipico problema di programmazione lineare

```

Minimize[{x + 3 y, x - 3 y ≤ 7 && x + 2 y ≥ 10}, {x, y}]
{53/5, {x → 44/5, y → 3/5}}

```

Teoricamente, Minimize e Maximize possono risolvere qualsiasi problema in cui la funzione obiettivo ed i vincoli sono espressi come funzioni polinomiali arbitrarie nelle variabili x_i . Molti importanti problemi, ad esempio molti problemi geometrici, possono essere formulati in tale modo.

L'esempio seguente mostra un problema geometrico molto semplice, dove si chiede di massimizzare l'area di un rettangolo con un perimetro fissato.

```

Maximize[{x y, x + y == 1}, {x, y}]
{1/4, {x → 1/2, y → 1/2}}

```

Si calcola il massimo volume di un cuboide interno alla sfera unitaria.

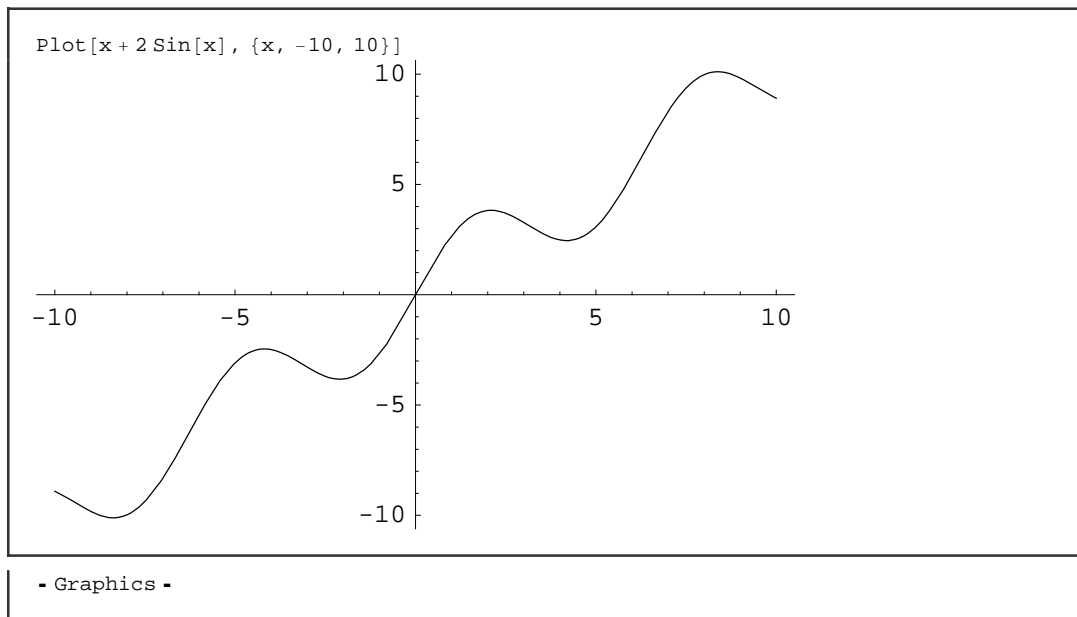
```

Maximize[{8 x y z, x^2 + y^2 + z^2 ≤ 1}, {x, y, z}]
{8/(3√3), {x → 1/√3, y → -1/√3, z → -1/√3}}

```

Una importante caratteristica delle funzioni Minimize e Maximize è che esse individuano i minimi e massimi globali. Spesso le funzioni hanno diversi minimi o massimi locali nei quali la derivata si annulla; Minimize e Maximize utilizzano metodi globali per individuare minimi e massimi assoluti e non solo estremi locali.

La funzione seguente ha molti minimi e massimi locali.



Maximize trova il massimo globale.

```
Maximize[{x + 2 Sin[x], -10 ≤ x ≤ 10}, x]
{√3 + 8π/3, {x → 8π/3}}
```

Se la funzione non è limitata inferiormente o superiormente, Minimize e Maximize restituiscono $-\infty$ e $+\infty$ come valori di minimo e massimo, rispettivamente. Se si impostano dei vincoli che non possono essere soddisfatti Minimize e Maximize restituiscono ancora $-\infty$ e $+\infty$ come minimi e massimi assoluti ed il valore Indeterminate come valore per le variabili.

Una sottile precisazione: Minimize e Maximize permettono di specificare sia disequazioni del tipo $x \leq v$ che del tipo $x < v$. Nel primo caso non ci sono problemi nell'individuazione dei minimo e massimi che possono essere esattamente sui punti di frontiera. Nel secondo caso, invece, un minimo o un massimo dovrebbe teoricamente porsi all'interno del contorno ad una distanza infinitesima dalla frontiera.

In questo caso *Mathematica* mostra un messaggio di "attenzione" (warning) e restituisce il punto della frontiera.

```
Minimize[{x^2 - 3 x + 6, x > 3}, x]
Minimize::wksol : Warning: There is no minimum in the region
described by the constraints; returning a result on the boundary. More...
{6, {x → 3}}
```

Minimize e Maximize assumono normalmente che le variabili siano tutte reali. Eventuali indicazioni in merito alle variabili possono essere inserite come vincoli del tipo $x \in \text{Integers}$, che permette di specificare che la variabile x deve essere considerata come un intero.

Si calcola la massimizzazione dell'espressione per i valori interi di x ed y .

```
Maximize[{x y, x^2 + y^2 < 120 && (x | y) ∈ Integers}, {x, y}]
{56, {x → -8, y → -7}}
```

Assuming/Refine

La nuova funzione `Refine[expr, assum]` restituisce la forma che assume l'espressione `expr` se i simboli in essa compresi vengono sostituiti da valori numerici che soddisfano le ipotesi indicate con `assum`.

Esempio: semplificazione di una radice quadrata

$$\text{var} = \frac{\sqrt{x y^2 z^4}}{\sqrt{x y^2 z^4}}$$

Si può richiedere la forma che assume la variabile `var` nel caso in cui `x` sia positiva.

$$\text{Refine}[\text{var}, x > 0]$$

$$\sqrt{x} \sqrt{y^2 z^4}$$

Se si assume che anche `y` è positivo allora $\sqrt{y^2}$ si semplifica in `y`.

$$\text{Refine}[\text{var}, x > 0 \ \&\& \ y > 0]$$

$$\sqrt{x} \ y \ \sqrt{z^4}$$

```
Clear[var]
```

Esempio: utilizzo dell'opzione `Assumptions` in altre funzioni di Mathematica

Diverse funzioni di *Mathematica* accettano l'opzione `Assumptions`, ad esempio la `Limit` e la `Integrate`, che permettono di specificare ulteriori informazioni sulle variabili che intercorrono nelle espressioni da manipolare.

Il limite di x^α dipende dal dominio di α e di x .

$$\text{Limit}[x^\alpha, x \rightarrow \infty, \text{Assumptions} \rightarrow \alpha > 0]$$

$$\infty$$

$$\text{Limit}[x^\alpha, x \rightarrow \infty, \text{Assumptions} \rightarrow \alpha < 0]$$

$$0$$

È anche possibile stabilire una condizione su una variabile e fare in modo che tale condizione si mantenga anche in espressioni che comprendono diverse funzioni di *Mathematica*, come nell'esempio che segue.

```
Assuming[σ > 0,
```

$$F = \int_{-\infty}^y \frac{1}{\sqrt{2\pi}\sigma} \text{Exp}\left[-\frac{(x-\mu)^2}{2\sigma^2}\right] dx;$$

```
Limit[F, y → ∞]
```

$$]$$

```
1
```

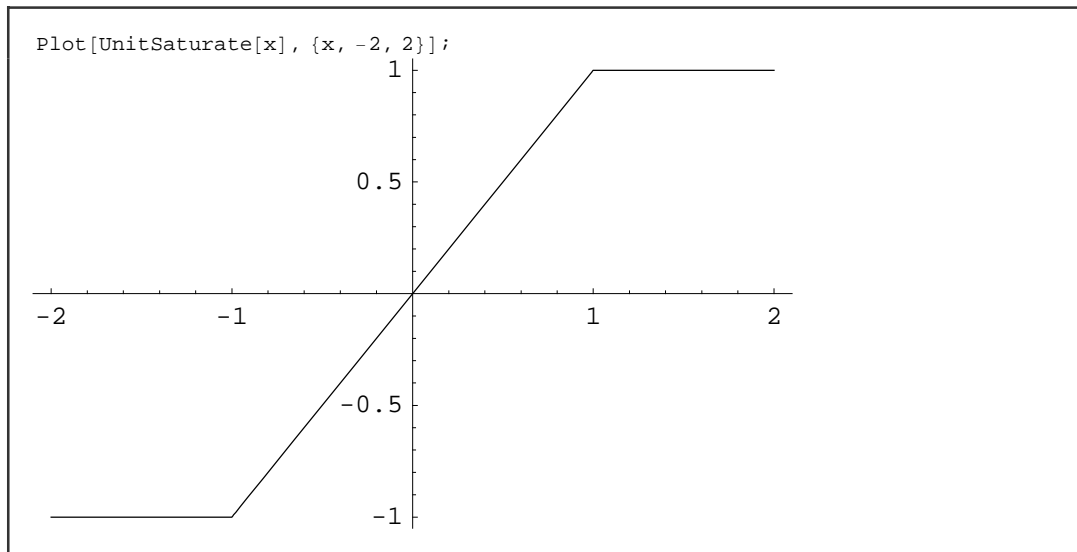
Funzioni definite a tratti

Definizione e valutazione

Questo definisce una funzione a tratti

```
UnitSaturate[x_] := { -1 x < -1
                    x  -1 ≤ x ≤ 1
                    1  x > 1
```

Questo è il grafico



Questa è una semplificazione

```
Simplify[UnitSaturate[x], x < -2]
-1
```

Espansione

Qualsiasi combinazione di funzioni definite a tratti può essere ricondotta ad una sola funzione definita a tratti, le così dette funzioni a tratti in forma normale.

```
PiecewiseExpand[Piecewise[{{Piecewise[{{f1, d1}, {f2, d2}}], c1}, {e2, c2}]]]
{ f1 c1 && d1
  f2 c1 && d2
  0  c1 || !c2
  e2 True
```

Funzioni a tratti con un numero finito di casi

Max e Min hanno due casi di funzione a tratti. Questa è una composizione che viene ricondotta ad una forma normale

```
PiecewiseExpand[Max[x, Min[y, z]]]
{ x (x - y ≥ 0 && y - z ≤ 0) || (y - z > 0 && x - z ≥ 0)
  y x - y < 0 && y - z ≤ 0
  z True
```

Questa espansione si applica solo nel caso in cui le variabili sono reali, poiché nel caso complesso `Abs` non è una funzione definita a tratti

```
PiecewiseExpand[Max[Abs[x], Abs[y]], Reals]
{ -x (x < 0 && y < 0 && x - y ≤ 0) || (x < 0 && y ≥ 0 && x + y ≤ 0)
  x (x ≥ 0 && y < 0 && x + y ≥ 0) || (x ≥ 0 && y ≥ 0 && x - y ≥ 0)
  -y (x - y > 0 && x < 0 && y < 0) || (x ≥ 0 && y < 0 && x + y < 0)
  y True
```

Funzioni a tratti con un numero infinito di casi

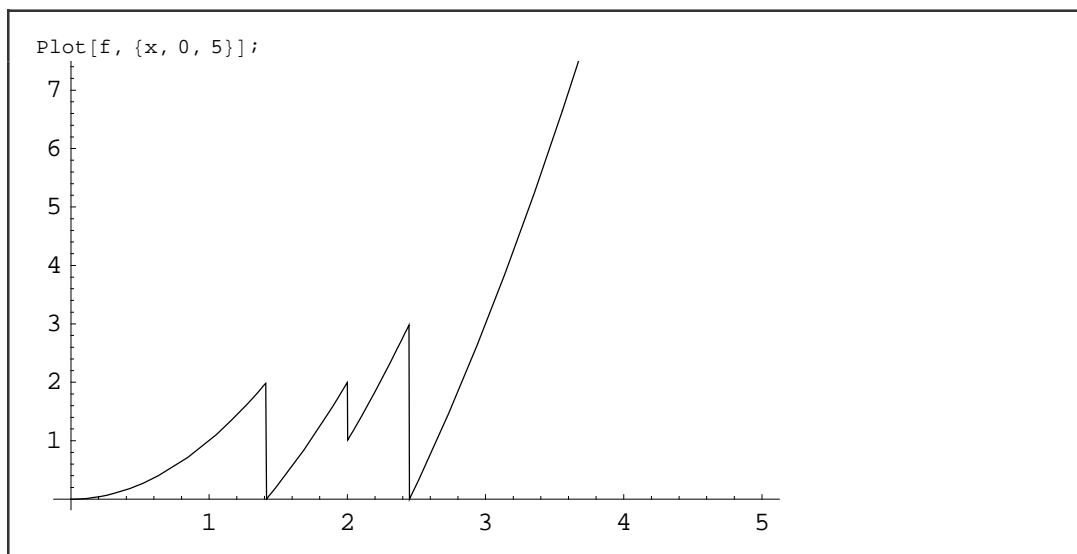
`Floor`, `Ceiling`, `FractionalPart` e `IntegerPart` possono avere un numero arbitrario di casi

```
PiecewiseExpand[0 ≤ x - Floor[x]^2 ≤ 3, Reals]
0 ≤ x < 2
```

In questo caso viene dato esplicitamente un intervallo finito per la variabile `x`.

```
f = PiecewiseExpand[Mod[x^2, Ceiling[x]], 1 < x < 3]
{ x^2 x < √2
  -6 + x^2 x ≥ √6
  -4 + x^2 x == 2
  -3 + x^2 2 < x < √6
  -2 + x^2 True
```

```
f
{ x^2 x < √2
  -6 + x^2 x ≥ √6
  -4 + x^2 x == 2
  -3 + x^2 2 < x < √6
  -2 + x^2 True
```



Equazioni e disequazioni con funzioni definite a tratti

Si possono risolvere equazioni, disequazioni ed eliminazioni di quantificatori con espressioni che comprendono funzioni a tratti

Questo risolve due disequazioni

```
Reduce[1 ≤ x + Ceiling[x + Floor[x]] < 10, x]
0 < x ≤ 3
```

Questo risolve un'equazione

```
Reduce[x2 + Piecewise[{{x, 0 < x < 1}, {x3, 1 ≤ x < 2}}] == 5, {x}]
x == -√5 || x == √5 || x == Root[-5 + #12 + #13 &, 1]
```

Questo trova una particolare soluzione di un sistema con disequazioni ed equazioni

```
FindInstance[2 ≤ Max[x2, y2] ≤ 3 && y == Floor[x] + 1, {x, y}, Reals]
{{x → -25/16, y → -1}}
```

Questo trova una condizione su α tale che la disequazione abbia una soluzione

```
Piecewise[{{0 < α < 1, x2 + α < 2}}, x2 + 2 ≤ α]
{ 0 < α < 1    x2 + α < 2
  2 + x2 ≤ α   True
```

```
Resolve[Exists[x, %]]
α ≥ 2 || 0 < α < 1
```

Ottimizzazione di funzioni a tratti

Questo minimizza una funzione a tratti su un intervallo dato

```
Minimize[ {3 + (x - 2) UnitStep[1 - Abs[x - 3/2]] , 1 ≤ x < 4} , {x} ]
{2, {x → 1}}
```

Questo massimizza una funzione a tratti su una disequazione definita a tratti

```
Maximize[ {e^Floor[x] Sign[x] , 0 < x + Ceiling[x]^3 < 12} , {x} ]
{e^2, {x → 2}}
```

Derivate di funzioni a tratti

La derivata è indeterminata nell'origine, poiché non esiste in questo punto

```
D[Piecewise[{{x, x < 0}, {x^2, x ≥ 0}}] , x ]
{ 1          x < 0
  2 x        x > 0
  Indeterminate True
```

La funzione Sinc ha una derivata in ogni punto, incluso l'origine

```
Sinc[x_] := Piecewise[{{1, x == 0}},  $\frac{\text{Sin}[x]}{x}$  ]
General::spell : Possible spelling error: new symbol
name "Sinc" is similar to existing symbols {Sin, Sinh}. More...
```

```
D[Sinc[x], x]
{ 0          x == 0
   $\frac{\text{Cos}[x]}{x} - \frac{\text{Sin}[x]}{x^2}$  True
```

Limiti di funzioni a tratti

In generale, le funzioni a tratti hanno delle discontinuità, pertanto i limiti direzionali potrebbero differire

```
Limit[Piecewise[{{x^2, x^2 ≤ 1}},  $\frac{\text{Sin}[x]}{x}$  ] + 1, x → 1, Direction → 1]
2
```

```
Limit[Piecewise[{{x^2, x^2 ≤ 1}},  $\frac{\text{Sin}[x]}{x}$  ] + 1, x → 1, Direction → -1]
1 + Sin[1]
```

Integrali di funzioni a tratti

Questo è l'integrale di una funzione a tratti

```
 $\int \text{Max}[\text{Min}[x, e^x], x^2] dx$ 
{ e^x      e^x - x^2 > 0 && e^x - x ≤ 0
   $\frac{x^2}{2}$    e^x - x > 0 && -x + x^2 < 0
   $\frac{x^3}{3}$    True
```

Questa funzione non può essere integrata nel campo dei complessi, in quanto la funzione Abs non ha primitive

$$\int \text{Abs}[2 - \text{Abs}[x]] \, dx$$

$$\int \text{Abs}[2 - \text{Abs}[x]] \, dx$$

Comunque, la funzione può essere integrata sui reali, in tal caso la primitiva di $\text{Abs}[x]$ è $\frac{x \text{Abs}[x]}{2}$.

$$\text{Assuming}[x \in \text{Reals}, \int \text{Abs}[2 - \text{Abs}[x]] \, dx]$$

$$\left\{ \begin{array}{ll} -2x - \frac{x^2}{2} & x \leq -2 \\ 4 + 2x + \frac{x^2}{2} & -2 < x \leq 0 \\ 4 + 2x - \frac{x^2}{2} & 0 < x \leq 2 \\ 8 - 2x + \frac{x^2}{2} & \text{True} \end{array} \right.$$

Integrali definiti di funzioni a tratti su regioni

Quando si integra su regioni definite da combinazioni logiche di disequazioni, è spesso conveniente utilizzare la funzione a tratti `Boole`. La complessità dipende non solo dalla funzione che deve essere integrata, ma anche dalla complessità della regione.

$$\int_0^1 \int_0^1 \text{Boole}[a x^2 + y^2 < 1] \, dy \, dx$$

$$\left\{ \begin{array}{ll} 1 & a \leq 0 \\ \frac{\pi}{4\sqrt{a}} & a > 1 \\ \frac{\sqrt{a-a^2} + \text{ArcSin}[\sqrt{a}]}{2\sqrt{a}} & \text{True} \end{array} \right.$$

$$\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \text{Boole}\left[\frac{x^2}{a^2} + \frac{y^2}{b^2} < 1\right] \, dy \, dx$$

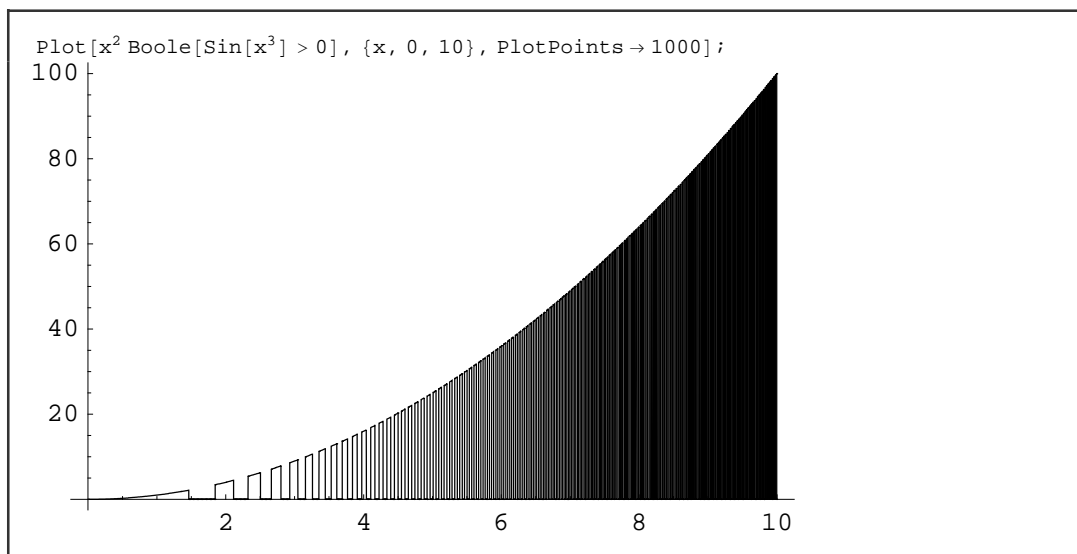
$$\left\{ \begin{array}{ll} -a b \pi & (a > 0 \ \&\& \ b < 0) \ || \ (a < 0 \ \&\& \ b > 0) \\ a b \pi & (a > 0 \ \&\& \ b > 0) \ || \ (a < 0 \ \&\& \ b < 0) \end{array} \right.$$

Questa è una regione definita da una disequazione trascendentale

$$\int_{-\infty}^{\infty} e^x \text{Boole}[x + 2 > e^x] \, dx$$

$$\text{ProductLog}\left[-\frac{1}{e^2}\right] - \text{ProductLog}\left[-1, -\frac{1}{e^2}\right]$$

Questa funzione a tratti ha 319 discontinuità nell'intervallo da 0 a 10.



Questo è l'integrale

$$\int_0^{10} x^2 \text{Boole}[\text{Sin}[x^3] > 0] dx$$

$$\frac{1}{3} (1000 - 159 \pi)$$

Si possono integrare molte altre funzioni a tratti

$$\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} e^{\text{Floor}[x^2+y]} \text{Boole}[\text{Abs}[x] + \text{Abs}[y] < 1] dy dx$$

$$\frac{-7 + 5\sqrt{5} + 19e - 5\sqrt{5}e}{6e}$$