a cura di Francesco Romani

La grafica di *Mathematica*

Dopo sei anni e settanta articoli (tutti o quasi pieni di disegni più o meno elaborati) riprendiamo la teoria di base della generazione di grafici con Mathematica. Oltre che un aiuto per chi comincia adesso, queste note abbastanza elementari possono servire come spunto per alcune applicazioni sofisticate che vedremo in futuro.

Introduzione

Mathematica è (anche) un linguaggio di programmazione e il principale vantaggio/svantaggio della grafica di *Mathematica* è la sua totale programmabilità.

A differenza di programmi specifici per il disegno artistico, il ritocco fotografico o il CAD, nel nostro caso anche il più semplice disegno va programmato (un po' come si faceva nella preistoria con i vecchi plotter pilotati da programmi FORTRAN).

Questa caratteristica complica di molto la vita nel caso di disegni elementari (un ellisse verde in campo rosso) ma permette agevolmente applicazioni di complessità straordinaria (come i frattali o le mappe stellari).

Distinguiamo tre livelli di programmazione grafica:

- uso diretto delle programmazione grafica.
- uso dei programmi di plottaggio predefiniti.
- combinazione dei due metodi per ottenere risultati più sofisticati.

Nel seguito vediamo esempi dei due primi casi, vedremo in altri articoli le applicazioni avanzate.

Uso diretto della programmazione grafica

Per comprendere l'uso delle primitive grafiche bisogna notare che in *Mathematica* molte funzioni hanno un significato puramente simbolico; ad esempio, la primitiva `Line[{{a,b},{c,d}}]` è la rappresentazione simbolica di una linea che congiunge il punto **(a,b)** con il punto **(c,d)** del piano cartesiano, e la funzione **`Line`** non ha alcuna implementazione.

La funzione **`Graphics`** (anch'essa priva di implementazione) riceve come argomento una lista di primitive grafiche e costituisce la rappresentazione simbolica di un grafico bidimensionale. In modo analogo esistono altre funzioni (**`Graphics3D`**, **`SurfaceGraphics3D`**, **`ContourGraphics`**, **`DensityGraphics`**, **`GraphicsArray`**) che costituiscono la rappresentazione simbolica di grafici di altro tipo.

La funzione **`Show`** (per cui esiste una implementazione interna) mostra sullo schermo il grafico associato ad una rappresentazione simbolica. **`Show`** riceve una rappresentazione simbolica di un grafico chiama la funzione **`$DisplayFunction`** e rende gli argomenti non valutati.

`$DisplayFunction` di solito vale **`Display`**: una funzione di sistema che disegna nel *Front-End* il grafico associato a quella rappresentazione. Giocando sul valore di **`$DisplayFunction`** è possibile sia evitare la rappresentazione (comodo per raggruppare più grafici in uno stesso disegno) sia dirigere la stampa su apparecchiature diverse dal monitor (ad esempio un plotter).

Ad esempio, il seguente programma disegna un ellisse verde in campo rosso.

In[1]:=

```
f1=Show[Graphics[{
     Green, Rectangle[{-2,-2},{2,2}],
     Red,   Disk[{0,0},1]
  }],AspectRatio->1/2];
```

L'uso delle primitive grafiche unito alla potenza della programmazione permette di ottenere agevolmente grafici di grande complessità come i frattali. L'esempio che segue traccia una delle tante varianti del triangolo di Sierpinski, disegnando un punto in corrispondenza dei valori dispari del triangolo di Tartaglia **`AspectRatio`** e **`PlotLabel`** sono opzioni di **`Show`** e permettono di specificare il rapporto tra altezza e larghezza dell'area di plottaggio e il titolo da porre in testa al grafico

In[2]:=

```
Show[
 Graphics[Point/@First/@Select[Flatten[
  Table[{{k-n/2.,-n},Binomial[n,k]},
                {n,0,63},{k,0,n}],1],
```

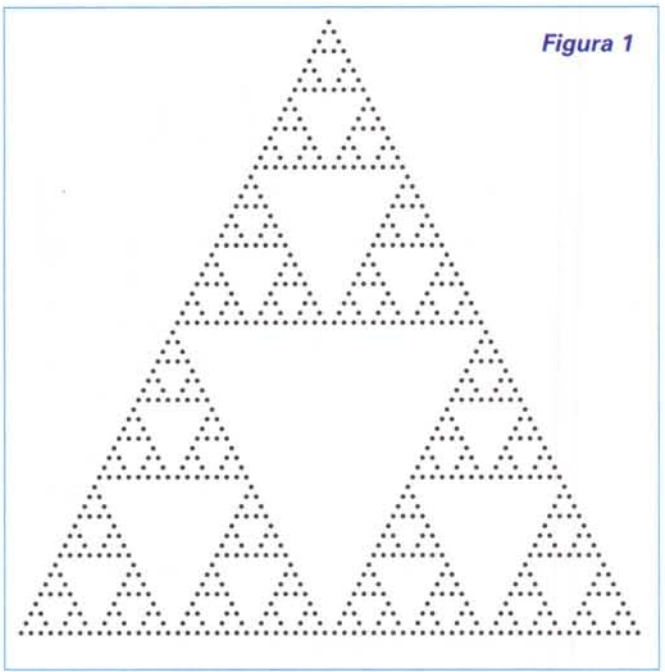

Figura 1

```
  OddQ[#[[2]]]&]],
PlotRange->{{-32,32},{-64,1}},
AspectRatio->1,
PlotLabel->"Figura 1"];
```

(Vedi Figura 1)

Programmi di plottaggio predefiniti

La applicazione grafica più naturale per il matematico è il semplice plottaggio di una funzione di variabile reale (anche con eventuali singolarità) in un intervallo dato. La funzione **Plot** soddisfa tale necessità.

Per esempio il comando seguente traccia il grafico del polinomio di Chebyshev $T_4(x)$ tra -1 e 1.

In[1]:=

```
f2=Plot[ChebyshevT[4,x],{x,-1,1},
   PlotStyle->Red,
   PlotLabel->"Figura 2"]
```

(Vedi Figura 2)

In realtà **Plot** è un programma di livello superiore che valutando in modo adattivo la funzione da plottare genera le opportune primitive grafiche, come si vede dalla forma interna del risultato.

In[2]:=

```
f2//InputForm
```

Out[2]

```
Graphics[{RGBColor[1.,0.,0.], {Line[{
{-1., 1.},
{ - 0 . 9 5 8 3 3 3 3 3 3 3 3 3 3 3 3 3 ,
0.4004870756172832},
...,
{0.916666666666667,                    -
0.07368827160493641},
{0.958333333333334, 0.4004870756172857},
{1., 1.}}]}},
{PlotRange -> Automatic, AspectRatio -> Gol-
denRatio^(-1), PlotLabel -> "Figura 2", Axe-
sLabel -> None, ...}]
```

Nel *package* **Graphics`Graphics`** sono presenti molti diversi programmi di plottaggio di funzioni di una variabile o di liste monodimensionali.

Nel *package* **Graphics`Colors`** (che conviene sempre caricare all'inizio dell'elaborazione) sono presenti le definizioni dei colori.

Ciò permette di chiamare i colori con il loro nome inglese (**Red**, **Green**, etc,) invece che con la tripla dei valori RGB (**RGBColor[1,0,0]**, **RGBColor[0,1,0]**, etc,).

Altri programmi di grafica sono presenti negli *Standard packages*, conviene leggerne la descrizione nell *Help* in linea o sull'apposito manuale.

Raggruppamento di grafici

Capita spesso di dovere raggruppare in un unico disegno grafici diversi (per esempio generati in momenti successivi. In questo caso basta rieseguire la **Show** dando come argomenti i risultati dei disegni precedenti.

Questa sequenza di comandi traccia dapprima separatamente i tre grafici e poi li sovrappone.

In[1]:=

```
p1=Plot[ChebyshevT[4,x],{x,-1,1},PlotStyle-
>Red];
p2=Plot[ChebyshevT[5,x],{x,-1,1},PlotStyle-
>Green];
p3=Plot[ChebyshevT[6,x],{x,-1,1},PlotStyle-
>Blue];
```

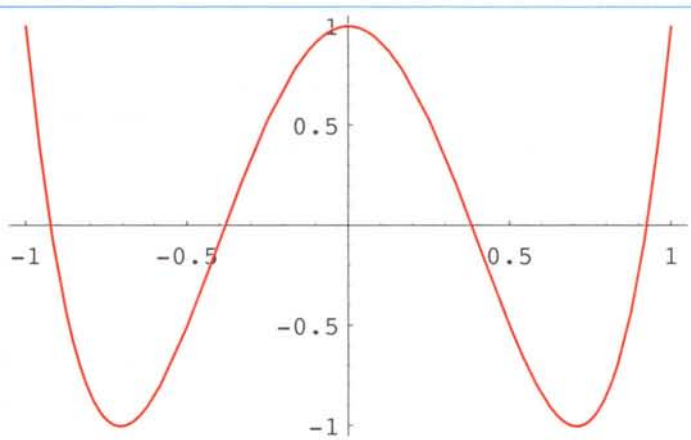

Figura 2

```
Show[p1,p2,p3, PlotLabel->"Figura 3"];
```

(Vedi Figura 3)

Se si vuole evitare i plottaggi dei singoli grafici prima della sorapposizione conviene alterare provvisoriamente il valore della funzione **$DisplayFunction**. **N.B.** in questo caso usare **Module** invece che **Block** non avrebbe funzionato, (perché?)

In[2]:=

```
Show[
 Block[{$DisplayFunction=Identity},
  Table[
   Plot[ChebyshevT[i,x],{x,-1,1},
     PlotStyle->Hue[i/6]],
   {i,6}]],
 PlotLabel->"Figura 4"];
```

(Vedi Figura 4)

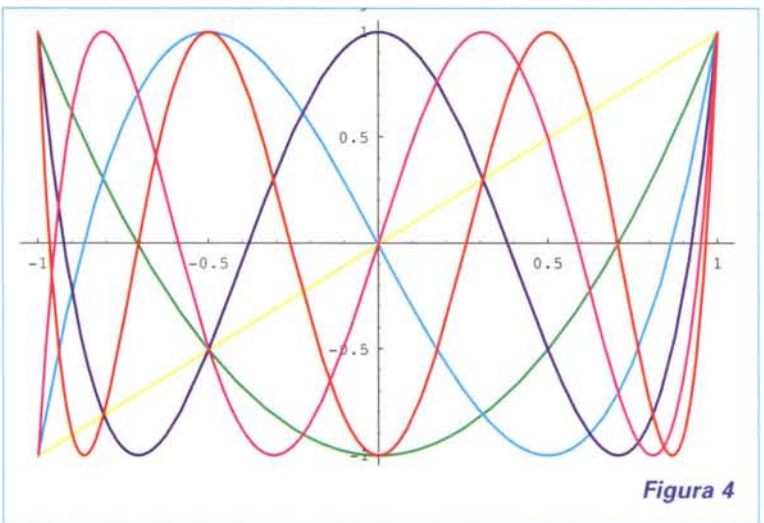

Figura 4

Animazioni

Un altro modo per rappresentare più grafici è fare un animazione. Se il risultato di una elaborazione è un gruppo di grafici consecutivi basta selezionarli tutti e dare il comando **Animate Selected Graphics** per vederli visualizzare uno dopo l'altro con un effetto di movimento. Il comando seguente genera una breve animazione dei polinomi di Chebyshev.

In[3]:=

```
Do[
 Plot[ChebyshevT[i,x],{x,-1,1},
   PlotStyle->Hue[i/6]],
 {i,6}]
```

GraphicsArray

Quando si scrive un articolo (o un libro o una tesi) le animazioni sono poco pratiche, la funzione **GraphicsArray** permette

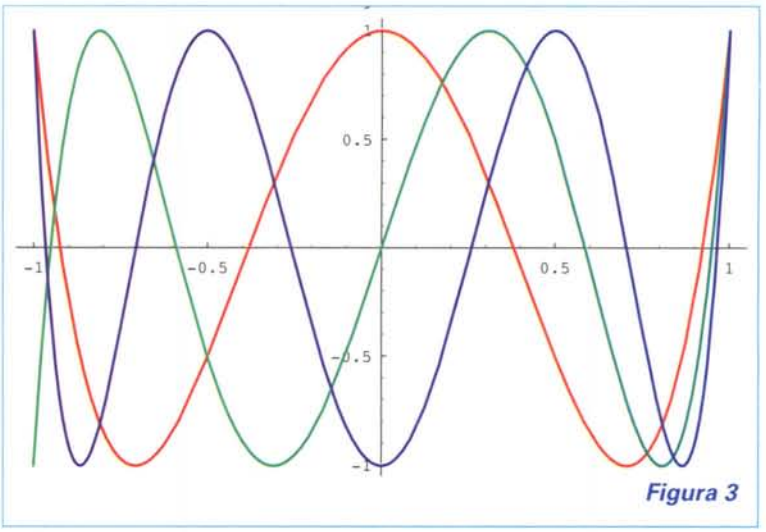

Figura 3

disegnare un **Array** di grafici come nel caso seguente.

In[4]:=

```
Show[GraphicsArray[Partition[
 Block[{$DisplayFunction=Identity},
  Table[
   Plot[ChebyshevT[i,x],{x,-1,1},
     PlotStyle->Hue[i/6]],
   {i,6}]], {2}]],
 PlotLabel->"Figura 5"];
```

(Vedi Figura 5)

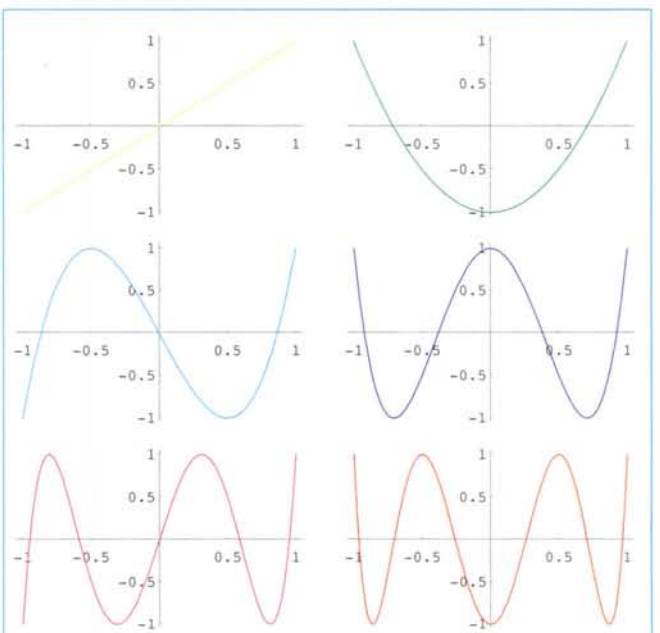

Figura 5

Rappresentazioni di funzioni di due variabili

Una funzione di due variabili (ad esempio **z = sin(x y)** può essere rappresentata in due dimensioni in molteplici modi diver-

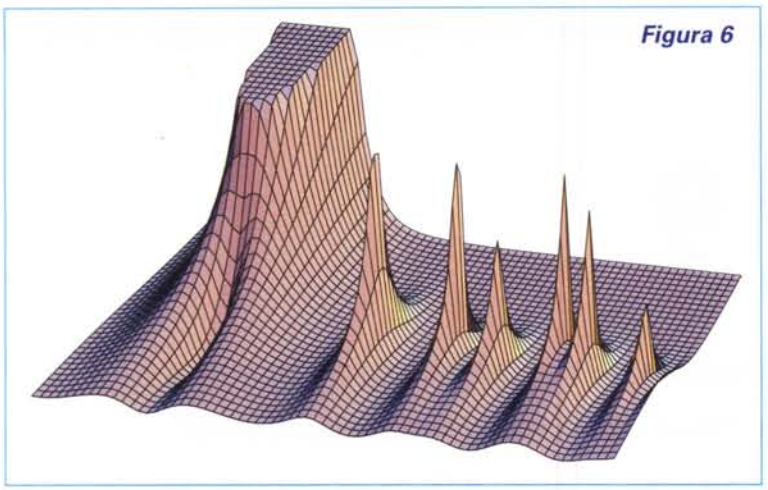

Figura 6

si, ognuno dei quali evidenzia alcune informazioni, nascondendone altre. Prendiamo come esempio la funzione ζ di Riemann (una funzione speciale di fondamentale importanza matematica). Questa funzione possiede infiniti zeri nel campo complesso lungo la linea **1/2+ i y**. La funzione **`1/Abs[Zeta[x + I*y]]`** è reale in due variabili x e y e ha poli (picchi infiniti) in corrispondenza degli zeri di ζ. Vediamo il suo grafico tracciato in tre modi diversi.

`Plot3D` genera un oggetto di tipo **`SurfaceGraphics`** che può essere rappresentato in bianco e nero o a colori con le ombreggiature e il colore che rappresentano a scelta, o il valore di una ulteriore variabile, o una illuminazione simulata.

In[1]:=

```
Plot3D[1/Abs[Zeta[x +
I*y]], {x, -3, 3}, {y,
-10, 40},
    PlotPoints -> {40,
70},
   ViewPoint -> {7, 2,
3},
   PlotRange -> {0, 5},
   Boxed -> False,
   Axes ->  None];
```

(Vedi Figura 6)

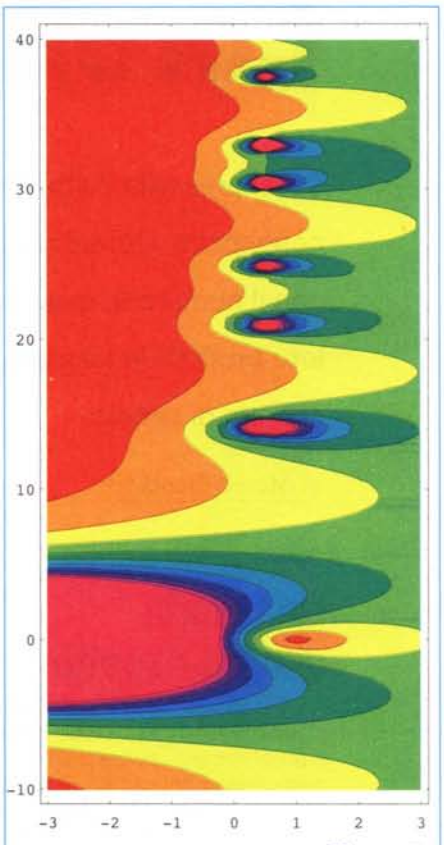

Figura 7

`ContourPlot` mostra una mappa a curve di livello, le aree tra le curve possono venire colorate con livelli di grigio o di colore differenti. Per generare un grafico di questo tipo il programma usa un approccio adattivo per agganciare e seguire le curve di livello.

In[2]:=

```
ContourPlot[1/Abs[Zeta[x + I*y]], {x, -3,
3},{y, -10, 40},
      ColorFunction->(Hue[0.9#]&),
      PlotPoints->100,AspectRatio->2];
```

(Vedi Figura 7)

`DensityPlot` campiona invece la funzione nei nodi di un reticolo e colorare di conseguenza i vari quadrâtini.. I risultati sono tanto più accurati quanto maggiore è il numero dei punti usati.

In[3]:=

```
DensityPlot[1/Abs[Zeta[x + I*y]], {x, -3,
3},{y, -10, 40},
      ColorFunction->(Hue[0.9#]&),
      Mesh->False,
      PlotPoints->100,AspectRatio->2];
```

Il risultato è molto simile ma meno definito per la mancanza delle linee di contorno.

MC

E' uscita *Mathematica* 4.0

Mentre finivo questo articolo mi è arrivata la notizia dell'uscita di *Mathematica* 4.0. insieme ad una copia di saggio. Le principali novità di questa versione, secondo il comunicato stampa, sono:

- Notevole miglioramento della velocità e dell'efficienza nel calcolo numerico;
- Ampliamento della gamma dei file da importare ed esportare da Mathematica, con oltre 20 tipi di formati standard di dati, grafici e suoni;
- Inclusione, nell'interfaccia del *notebook*, delle funzioni di correzione ortografica e di giustificazione;
- Estensione delle funzioni di generazione degli output e del codice HTML;
- Miglioramento delle funzioni ed estensione del range per l'analisi dei dati;
- Implementazione del supporto per la gestione di calcoli in domini specifici;
- Oltre 100 funzioni di Mathematica nuove o perfezionate.

Prossimamente un intero articolo sarà dedicato alla prova sul campo del prodotto. I prezzi iva esclusa di una licenza per Windows, Macintosh e Linux sono i seguenti:

a) Listino Educational

- Professional Version L. 2.500.000;
- Professional Version per Laboratorio Didattico L. 800.000;
- High School Teacher's Edition L. 650.000;
- Student Version L. 280.000;

b) Listino Commerciale

- Professional Version L. 3.500.000.

Chi desidera avere informazioni sulla disponibilità e sulle possibilità di aggiornamento dalla versione 3.0 può rivolgersi al distributore italiano: **`info@scisoft.it`**