Recen… *Soft*

a cura di Luca Barletta

Mathematica 5.2 Wolfram research

Non è semplice dare una definizione concisa ed esaustiva del software *Mathematica*, dato che esso integra un potente *kernel* per il calcolo simbolico e numerico ma non si limita solo a questo: elaborazione grafica, documentazione, linguaggio di programmazione e interfaccia verso altre applicazioni sono gli altri aspetti dell'anima di *Mathematica*.

Si possono dare comandi al *kernel* e riceverne le relative risposte attraverso la consueta interfaccia grafica presente nel software originale, oppure si può utilizzarlo come motore di altre applicazioni, come ad esempio *web-applications*.

Considerando il software originale, il *front end* del *kernel* è una pagina di testo interattiva che prende il nome di *notebook*: da questa pagina vengono dati gli input al *kernel*, e sempre su questa pagina vengono visualizzati i risultati sotto forma di formule, grafici, testo o audio; tutti questi elementi vengono automaticamente formattati grazie ad un sistema di documentazione integrato, permettendo così all'utente di ottenere con pochi sforzi un prodotto ordinato e facilmente maneggiabile. Il *notebook*, una volta salvato, assume estensione *.nb* e non è altro che un file ASCII.

A prima vista, l'ambiente di lavoro di *Mathematica*, per come si presenta, potrebbe sembrare ostile: in alto il menu dei comandi, a sinistra un *notebook* ancora bianco, a destra una *palette* con i comandi principali per la matematica simbolica. Un principiante, di fronte a questo scenario, potrebbe trovarsi spiazzato; in questo caso, il consiglio è di consultare la funzionalità *help browser*, oppure il *tutorial* messo a disposizione, entrambi molto ben documentati.

Qual è la filosofia che sta dietro a *Mathematica*? Ogni formula, grafico, documento, e in generale ogni oggetto, viene rappresentato come un'espressione simbolica e trattata come tale dal *kernel*, previa opportuna manipolazione di ogni espressione immessa dall'utente.

Vediamo un esempio: tramite la *palette* dei simboli matematici immettiamo l'espressione:

$$\texttt{FullForm[}\, Ae^{i\omega t} \,\texttt{]}$$

La funzione `FullForm` restituisce la rappresentazione simbolica dell'espressione tra parentesi quadre dopo la manipolazione; se si dà ora il comando *shift+invio* compare:

```
Times[A,Power[E,Times[Complex[0,1],t,\[Omega]]]]
```

Come si nota, l'espressione iniziale è stata decomposta in particelle elementari (`A, E, 0, 1, t, Omega`), che vengono opportunamente passate come parametri a funzioni annidate (`\, Complex, Times, Power`). Tutto ciò viene trattato nel *kernel* come una struttura ad albero.

Facciamo ora qualche esempio semplice per capire come muovere i primi passi.

1° esempio

Vogliamo verificare la formula risolutiva delle equazioni algebriche di 2° grado:

$$ax^2+bx+c=0$$

$$x=\frac{-b\pm\sqrt{b^2-4ac}}{2a}$$

Immettiamo l'equazione nel notebook dichiarando una variabile *eq*:

```
eq=a x^2+b x+c==0
```

Da notare che il primo uguale indica un assegnamento, invece il doppio uguale è proprio dell'equazione algebrica; premere invio e digitare il comando:

```
Solve[eq,x]
```

In questo modo, diciamo al *kernel* di risolvere l'equazione *eq* nella variabile *x*; premendo la combinazione di tasti shift+invio si ha l'output:

$$\left\{\left\{x\to\frac{-B-\sqrt{B^2-4AC}}{2A}\right\},\left\{x\to\frac{-B+\sqrt{B^2-4AC}}{2A}\right\}\right\}$$

che è la risposta che ci aspettavamo. Notate come le soluzioni vengono presentate sotto forma di *array*.

2° esempio

Vogliamo calcolare la soluzione $u(x,t)$ dell'equazione differenziale di diffusione:

$$u_t=\sigma u_{xx}$$

con la condizione iniziale $$u(x,0)=\delta(x)=\begin{cases}1 & x=0\\ 0 & x\neq 0\end{cases}$$

dove $\delta(x)$ è la funzione delta di Dirac.

Un metodo risolutivo noto in analisi è quello di considerare il problema trasformato tramite trasformazione di Fourier: applichiamo questo metodo.

Cominciamo a scrivere il termine che rappresenta la derivata seconda rispetto a x di u:

```
termine=D[u[x,t],x,x]
```

shift+invio e otteniamo

$$u^{(2,0)}[x,t]$$

ora calcoliamo la trasformata di Fourier di questo termine rispetto a `x`:

```
tf1=FourierTransform[termine,x,k]
```

shift+invio e otteniamo

$$-k^2\,\text{FourierTransform}[u[x,\,t],\,x,\,k]$$

Il risultato scritto così non ci piace, vorremmo vedere scritta una forma più compatta e non la scritta per esteso FourierTrasform[…] sostituendola, ad esempio, con U[t]:

```
tf2=tf1/.{FourierTransform[u[x,t],x,k]→U[t]}
```

In pratica, abbiamo scritto una *regola* che associa ad ogni occorrenza di `FourierTransform[u[x,t],x,k]` in `tf1` la scrittura simbolica `U[t]`.

Premendo shift+invio abbiamo:

$$-k^2\,U[t]$$

Ora possiamo scrivere l'equazione trasformata:

```
eq=D[U[t],t]-σ tf2==0
```

Il *kernel* ci risponde con: $k^2\,\sigma\,U[t] + U'[t] == 0$

Trasformiamo anche la condizione iniziale:

```
tfci=FourierTransform[DiracDelta[x],x,k]
```

La risposta è: $\dfrac{1}{\sqrt{2\pi}}$

Finalmente si può risolvere l'equazione differenziale con il comando `DSolve`:

```
sol=DSolve[{eq,U[0]==tfci},U[t],t]
```

Il risultato è: $\left\{\left\{U[t] \to \dfrac{e^{-k^2 t\sigma}}{\sqrt{2\pi}}\right\}\right\}$

Anche in questo caso il risultato viene dato come elemento di un *array* e con una regola di sostituzione indicata dalla freccia; assegniamo il risultato ad una variabile di comodo:

```
sol2=U[t]/.sol[[1]]
```

Ora dobbiamo ricavare la soluzione nel dominio antitrasformato, quindi:

```
antisol=InverseFourierTransform[sol2,k,x]
```

Premendo la solita combinazione shift+invio otteniamo il risultato finale:

$$\frac{e^{-\frac{x^2}{4t\sigma}}}{2\sqrt{\pi}\,\sqrt{t\sigma}}$$

3° esempio

Questa volta vogliamo mettere alla prova le capacità di calcolo numerico e di elaborazione grafica di *Mathematica*; un frattale è quello che serve per testare congiuntamente queste due caratteristiche.

Consideriamo un frattale di Barnsley regolato da queste equazioni ricorsive:

$$(x_{n+1}, y_{n+1}) = \begin{cases} (0.05x_n, 0.6y_n) & 0 < r < 0.1 \\ (0.05x_n, -0.5y_n + 1) & 0.1 < r < 0.2 \\ (0.46x_n - 0.15y_n, 0.39x_n + 0.38y_n + 0.6) & 0.2 < r < 0.4 \\ (0.47x_n - 0.15y_n, 0.17x_n + 0.42y_n + 1.1) & 0.4 < r < 0.6 \\ (0.43x_n + 0.28y_n, -0.25x_n + 0.45y_n + 1) & 0.6 < r < 0.8 \\ (0.42x_n + 0.26y_n, -0.35x_n + 0.31y_n + 0.7) & 0.8 < r < 1 \end{cases}$$

dove r è un numero casuale generato uniformemente tra 0 e 1. Come condizione iniziale scegliamo $(x_0, y_0) = (0.5, 0)$ ed effettuiamo $N = 25000$ iterazioni.

In generale, le relazioni che governano le iterazioni saranno del tipo:

$$\begin{cases} x_{n+1} = a_i x_n + b_i y_n + e_i \\ y_{n+1} = c_i x_n + d_i y_n + f_i \end{cases}$$

Dove sarà scelta l'equazione i qualora $r < p_i$.

Possiamo immettere i parametri dell'istanza del frattale e le condizioni iniziali

```
a[1]=0.05;a[2]=0.05;a[3]=0.46;a[4]=0.47;a[5]=0.43;a[6]=0.42;
b[1]=0;b[2]=0;b[3]=-0.15;b[4]=-0.15;b[5]=0.28;b[6]=0.26;
c[1]=0;c[2]=0;c[3]=0.39;c[4]=0.17;c[5]=-0.25;c[6]=-0.35;
d[1]=0.6;d[2]=-0.5;d[3]=0.38;d[4]=0.17;d[5]=0.45;d[6]=0.31;
e[1]=0;e[2]=0;e[3]=0;e[4]=0;e[5]=0;e[6]=0;
f[1]=0;f[2]=1.0;f[3]=0.6;f[4]=1.1;f[5]=1.0;f[6]=0.7;
p[1]=0.1;p[2]=0.2;p[3]=0.4;p[4]=0.6;p[5]=0.8;p[6]=1;
total=25000;x[0]=0.5;y[0]=0;
```

Implementiamo le ricorsioni:

```
xx[i_,n_]:=x[n+1]=a[i] x[n]+b[i] y[n]+e[i]
yy[i_,n_]:=y[n+1]=c[i] x[n]+d[i] y[n]+f[i]
```

dove il simbolo `:=` permette di definire una funzione; il simbolo di sottolineatura che segue i parametri significa che ogni volta che richiameremo le funzioni i parametri andranno a sostituire ogni occorrenza nella funzione. Definiamo r come numero casuale

```
r:=Random[]
```

Tramite il comando `Table` creiamo una lista dei risultati:

```
pts=Table[Which[
r<p[1],{xx[1,n],yy[1,n]},
r<p[2],{xx[2,n],yy[2,n]},
r<p[3],{xx[3,n],yy[3,n]},
r<p[4],{xx[4,n],yy[4,n]},
```

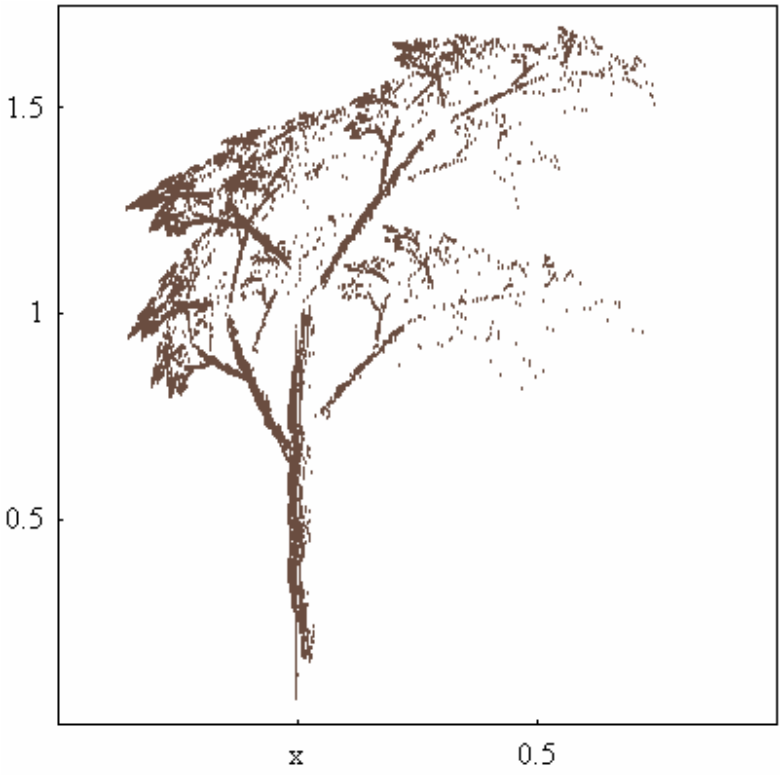
```
r<p[5],{xx[5,n],yy[5,n]},
r<p[6],{xx[6,n],yy[6,n]}],{n,0,total}];
```

dove il comando `Which` ci indirizza sulla branca corretta della nostra ricorsione. Ora, premendo la combinazione shift+invio e utilizzando il comando `ListPlot` con alcune opzioni per rendere più gradevole la visualizzazione:

```
ListPlot[pts,AspectRatio→1,Axes→False,Frame→True,
PlotRange→{{-.5,1.},{0,1.75}},
FrameTicks->{{-.5,{0,"x"},.5},{{-.001,"0"},.5,{-.75,"y"},1,1.5},{},{}},
PlotStyle→{RGBColor[106/255,77/255,64/255],PointSize[.007]},
TextStyle→{FontFamily->"Times",FontSize→16},ImageSize→{400,400}];
```

Otteniamo una figura del genere:

Cosa vi ricorda?



Come visto in questi semplici esempi, *Mathematica* è uno strumento di calcolo, sia numerico che simbolico, molto potente ed efficace. Per un utente alle prime armi probabilmente sarà difficile prendere confidenza con l'ambiente di lavoro, ma una volta fatta qualche esperienza diretta e tenendo sott'occhio l'*help* dovrebbe essere semplice acquisire le competenze necessarie per utilizzare questo software.

Mathematica rappresenta un efficace strumento di didattica ma costituisce soprattutto un valido supporto per la ricerca, non solo nelle università ma anche nelle aziende di tutto il mondo: i settori maggiormente interessati sono quelli dell'ingegneria, dell'economia e delle biotecnologie.

Il fornitore italiano di Mathematica è CreActive (www.creactive.net).