Liste, vettori e matrici in Mathematica

Il terzo appuntamento con il sistema Mathematica *riguarda l'uso interattivo delle liste e dei vettori e delle matrici che, in* Mathematica, *sono rappresentati come liste. Per dare un'idea della potenza che il trattamento delle liste conferisce ad un linguaggio di programmazione basta ricordare che il linguaggio classico della elaborazione simbolica e dell'intelligenza artificiale il LISP (LISt Processing) è appunto basato sul trattamento di liste e alberi binari. La trattazione che segue potrà sembrare un poco arida e monotona, ma molte delle primitive che vengono esposte ora sono indispensabili per introdurre argomenti più «succosi» quali la grafica o la programmazione*

di Francesco Romani

Le espressioni

Mathematica è essenzialmente un linguaggio di manipolazione di espressioni. Un'espressione è un nome simbolico detto *Head* (testa), che rappresenta una funzione, e una serie di argomenti che sono anch'essi espressioni o simboli. Le costanti numeriche di vario tipo e le stringhe hanno una rappresentazione coerente con questo meccanismo. Le espressioni sono la forma interna dei dati; in *Mathematica*, ad alcune *Head* sono associate delle forme esterne che realizzano il cosiddetto «zucchero sintattico» ovvero delle rappresentazioni alternative che non hanno una motivazione matematica, ma solo tradizionale. Un esempio classico di zucchero sintattico è la somma di più elementi che ha come forma interna la funzione Plus applicata ai suoi argomenti, ma che viene stampata nel modo tradizionale in notazione infissa.

```
In[1]:=
    Plus[a,b,c]
Out[1]=
    a + b + c
```

Una *head* non precedentemente definita è priva di rappresentazione esterna e viene lasciata indicata come una funzione simbolica applicata ai suoi argomenti.

```
In[2]:=
    MCMicrocomputer[a,b,c]
Out[2]=
    MCMicrocomputer[a, b, c]
```

Liste

La rappresentazione esterna (OutputForm) di una lista è una sequenza di espressioni, separata da virgole, e racchiusa tra parentesi graffe.

```
In[3]:=
    {1,2,3,1}
Out[3]=
    {1, 2, 3, 1}
```

La forma interna (che può venire stampata con la funzione **FullForm**) è data dalla funzione **List** seguita dal contenuto della lista. Le liste sono, dunque, solo delle espressioni.

```
In[4]:=
    FullForm[%]
Out[4]=
    List[1, 2, 3, 1]
```

I vettori non sono altro che liste, il generico elemento si può accedere attraverso la funzione **Part** o attraverso la notazione compatta delle doppie parentesi quadre. Nel seguito useremo indifferentemente il termine vettore o lista.

```
In[5]:=
    vet={10,20,30,40,50}
Out[5]=
    {10, 20, 30, 40, 50}
In[6]:=
    vet[[4]]
Out[6]=
    40
```

La funzione **ColumnForm** stampa un vettore per colonne.

```
In[7]:=
    ColumnForm[vet]
Out[7]=
    10
    20
    30
    40
    50
```

La funzione **Apply** sostituisce alla *head* di una espressione un'altra *head*.

```
In[8]:=
    Apply[Plus,{a,b,c}]
Out[8]=
    a + b + c
In[9]:=
    Apply[Times,%]
Out[9]=
    a b c
In[10]:=
    Apply[List,%]
Out[10]=
    {a, b, c}
```

Il procedimento è più chiaro se si guardano le forme interne dei 3 risultati precedenti.

```
In[11]:=
    FullForm[%%%]
Out[11]=
    Plus[a, b, c]
In[12]:=
    FullForm[%%%]
Out[12]=
    Times[a, b, c]
In[13]:=
    FullForm[%%%]
Out[13]=
    List[a, b, c]
```

La funzione **Table** permette di costruire una lista a partire da un'espressione. Il secondo argomento della funzione è una lista che contiene da uno a 4 elementi che specificano il campo di variazione della variabile.

{x, x0, x1, step}	significa «per x che va da x0 a x1 passo step»;
{x, x0, x1}	significa «per x che va da x0 a x1 passo 1»;
{x, x1}	significa «per x che va da 1 a x1 passo 1»;
{x1}	significa che l'espressione va valutata x1 volte senza alcuna variabile che cambia valore.

```
In[14]:=
    lista=Table[x^2,{x,14}]
Out[14]=
    {1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121, 144, 169, 196}
```

La funzione **Select** in unione ad una funzione booleana (in questo caso il predicato **OddQ** che dice se un numero è dispari) permette di selezionare solo gli elementi della lista per cui la condizione è vera.

```
In[15]:=
    Select[lista,OddQ]
Out[15]=
    {1, 9, 25, 49, 81, 121, 169}
```

La funzione **Map** in unione ad una funzione che assume un solo argomento (in questo caso la radice quadrata **Sqrt**) applica la funzione ad ogni elemento della lista.

```
In[16]:=
    Map[Sqrt,lista]
Out[16]=
    {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14}
```

Le funzioni **Take** e **Drop** prendono o cancellano alcuni elementi da una lista.

```
In[17]:=
    Take[lista,5]
Out[17]=
    {1, 4, 9, 16, 25}
In[18]:=
    Drop[lista,5]
Out[18]=
    {36, 49, 64, 81, 100, 121, 144, 169, 196}
```

La funzione **Join** concatena una o più liste.

```
In[19]:=
    Join[{2,3,2,1,4},{65,4,3,7},{4,5,3,67}]
Out[19]=
    {2, 3, 2, 1, 4, 65, 4, 3, 7, 4, 5, 3, 67}
```

La funzione **Sort** effettua l'ordinamento crescente (con eventuali ripetizioni).

```
In[20]:=
    Sort[%]
Out[20]=
    {1, 2, 2, 3, 3, 3, 4, 4, 4, 5, 7, 65, 67}
```

La funzione **Union** effettua la concatenazione e l'ordinamento crescente senza ripetizioni, di una o più liste, creando quindi un insieme.

```
In[21]:=
    Union[{2,3,2,1,4},{65,4,3,7},{4,5,3,67}]
Out[21]=
    {1, 2, 3, 4, 5, 7, 65, 67}
```

Esempio di applicazione: la distribuzione dei primi

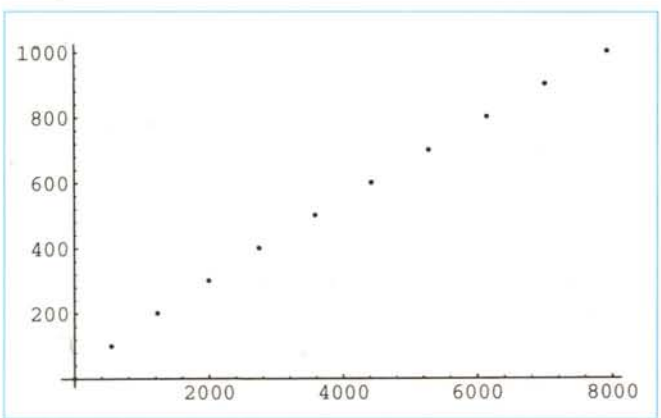
La funzione **Prime[k]** restituisce il k-esimo numero primo, è quindi facile generare una lista di primi. Nell'esempio seguente si chiama la funzione 11 volte con l'indice i che va da 1 a 1001 con passo 100, generando una lista di coppie primo-posizione.

```
In[22]:=
    lp=Table[{Prime[i],i},{i,1,1001,100}]
Out[22]=
    {{2,1}, {547,101}, {1229,201}, {1993,301}, {2749,401},
     {3581,501}, {4421,601}, {5281,701}, {6143,801},
     {7001,901}, {7927,1001}}
```

Questo formato può venire interpretato come una lista di coppe ascissa ordinata per generare un insieme di punti su di un piano cartesiano

```
In[23]:=
    ListPlot[lp];
Out[23]=
```

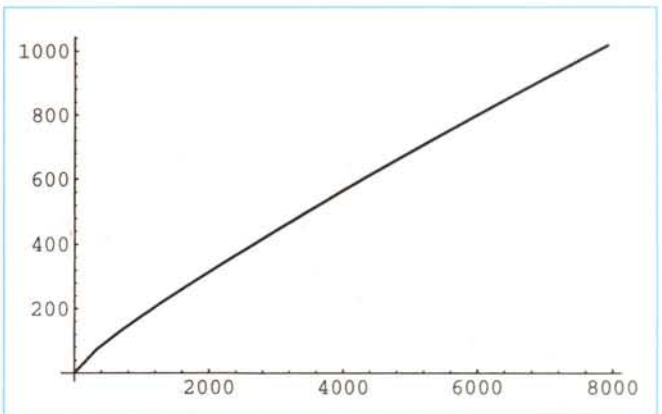
Vedi figura 1. ▼



Una approssimazione abbastanza accurata della funzione $\pi(x)$, che denota i numero dei primi minori di x, è da una funzione speciale detta Logaritmo Integrale (definita come la parte principale dell'integrale $\int_0^x dt/\log t$ Tale funzione è già definita in *Mathematica* e può essere calcolata direttamente. Il grafico seguente mostra l'andamento della funzione. La scrittura **Last[lp][[1]]** rappresenta la prima componente dell'ultima coppia della lista **lp**, ovvero l'ultima ascissa del grafico precedente.

```
In[24]:=
    Plot[LogIntegral[x],{x,0,Last[lp][[1]]}];
Out[24]=
```

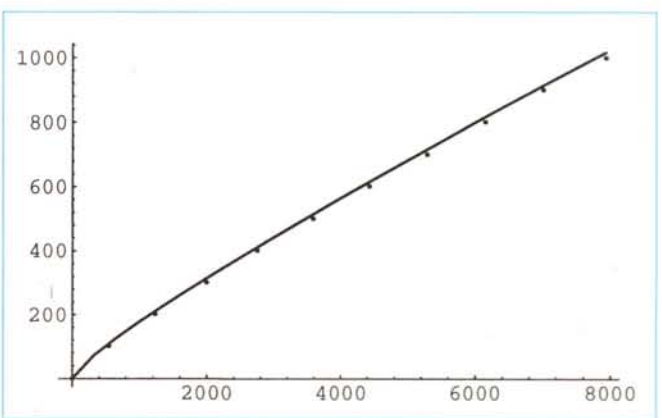
Vedi figura 2. ▼



I due grafici possono essere combinati insieme e si nota

```
In[25]:=
    Show[%,%%];
Out[25]=
```

Vedi figura 3. ▼



Vettori e matrici

Il prodotto scalare tra vettori si indica con un punto (o esplicitamente con la funzione **Dot**).

```
In[26]:=
    {a,b,c} . {x,y,x}
Out[26]=
    a x + c x + b y
```

Le matrici sono semplicemente liste di liste.

```
In[27]:=
{{1,2,3},{4,5,6},{7,8,9}}
Out[27]=
{{1, 2, 3}, {4, 5, 6}, {7, 8, 9}}
In[28]:=
MatrixForm[%]
Out[28]=
1  2  3
4  5  6
7  8  9
```

Il prodotto scalare funziona anche tra matrici e tra matrici e vettori.

```
In[29]:=
{{a,b},{c,d}} .{u,v}
Out[29]=
{a u + b v, c u + d v}
In[30]:=
{{a,b},{c,d}} .{{u,v},{w,z}}
Out[30]=
{{a u + b w, a v + b z}, {c u + d w, c v + d z}}
```

Costruiamo ora la matrice di Hilbert di ordine 5.

```
In[31]:=
matrice=Table[1/(1+i+j),{i,1,5},{j,1,5}];
In[32]:=
MatrixForm[%]
Out[32]=
 1   1   1   1   1
 -   -   -   -   -
 3   4   5   6   7

 1   1   1   1   1
 -   -   -   -   -
 4   5   6   7   8

 1   1   1   1   1
 -   -   -   -   -
 5   6   7   8   9

 1   1   1   1   1
 -   -   -   -   --
 6   7   8   9   10

 1   1   1   1   1
 -   -   -   --  --
 7   8   9   10  11
```

Sono presenti numerose funzioni che operano su matrici con l'aritmetica indotta dai dati. Per esempio l'inversa di una matrice razionale, come quella appena creata, viene calcolata in aritmetica razionale esatta e (siccome l'inversa di una matrice di Hilbert è una matrice di interi) nel nostro caso si ottiene un risultato esatto negli interi.

```
In[33]:=
inv=Inverse[matrice];
In[34]:=
MatrixForm[%]
Out[34]=
3675      -29400     79380      -88200     34650

-29400    250880     -705600    806400     -323400

79380     -705600    2041200    -2381400   970200

-88200    806400     -2381400   2822400    -1164240

34650     -323400    970200     -1164240   485100
In[35]:=
MatrixForm[inv . matrice]
Out[35]=
1  0  0  0  0
0  1  0  0  0
0  0  1  0  0
0  0  0  1  0
0  0  0  0  1
```

Il prodotto esterno o tensoriale è definito tra un operatore e una sequenza di liste e applica l'operatore agli elementi delle liste in tutti i modi possibili, conservando l'ordine. Il risultato è una lista a tanti livelli quante sono le liste di partenza. Nel caso più semplice, un operatore simbolico e due liste, viene generata una matrice.

```
In[36]:=
Outer[ff,{a,b},{c,d}]
Out[36]=
{{ff[a, c], ff[a, d]}, {ff[b, c], ff[b, d]}}
In[37]:=
MatrixForm[%]
Out[37]=
ff[a, c]   ff[a, d]

ff[b, c]   ff[b, d]
```

Se l'operatore è quello della moltiplicazione (**Times**) si realizza il prodotto esterno classico.

```
In[38]:=
Outer[Times,{a,b,c},{x,y,z}]
Out[38]=
{{a x, a y, a z}, {b x, b y, b z}, {c x, c y, c z}}
In[39]:=
MatrixForm[%]
Out[39]=
a x   a y   a z

b x   b y   b z

c x   c y   c z
```

Particolarmente interessante è il caso che l'operatore sia quello della derivazione simbolica: **D[f,x]** realizza la derivata parziale $\delta f/\delta x$.

```
In[40]:=
D[x^2+y^2,x]
Out[40]=
2 x
```

Outer[D,{f1,f2, ...} ,{x1,x2, ... }], dove f1,f2, ... è un vettore di funzioni nelle variabili x1,x2, ... realizza la cosiddetta matrice Jacobiana, ovvero la matrice delle derivate parziali $\delta fi/\delta x_j$

```
In[41]:=
    MatrixForm[Outer[D,{x^2+y^2,x y},{x,y}]]
Out[41]=
    2 x   2 y

    y     x
```

definiamo ora una matrice simbolica 3x3.

```
In[42]:=
    mat={{0,a,b},{b,0,a},{a,b,0}};
    MatrixForm[mat]
Out[42]=
    0   a   b

    b   0   a

    a   b   0
```

Il determinante e il polinomio caratteristico possono venire calcolati in forma simbolica.

ERRATA CORRIGE

Per un disguido interno alcune formule nell'articolo dello scorso mese sono state stampate in modo non corretto. Ci scusiamo del problema e riproponiamo qui di seguito le formule così come sono state pubblicate e nella loro forma corretta.

Formule a pag. 262

\Isu(h=0,n,)\B\bc\((\S(k,n)) a^h b^(n-k) = (a+b)^k,

$$\sum_{h=0}^{n} \binom{k}{n} a^h b^{(n-k)} = (a+b)^k,$$

Formule a pag. 263

$$\binom{h,k}{} p^h (1-p)^{k-h},$$

$$\binom{h}{k} p^h (1-p)^{k-h},$$

\Isu(h=m,k,)\B\bc\((\S(h,k)) p^h (1-p)^k-h,

$$\sum_{h=m}^{k} \binom{h}{k} p^h (1-p)^{k-h},$$

\Isu(h=0,k,)\B\bc\((\S(h,k)) p^h 1-p)^k-h = 1

$$\sum_{h=0}^{k} \binom{h}{k} p^h\ 1-p)^{k-h} = 1$$

```
In[43]:=
    Det[mat]
Out[43]=
     3    3
    a  + b
In[44]:=
    Det[mat - u IdentityMatrix[3]]
Out[44]=
     3    3              3
    a  + b  + 3 a b u - u
```

Se la dimensione della matrice lo consente, ovvero se il polinomio caratteristico ha grado sufficientemente basso (oppure se è riducibile), si possono calcolare in forma simbolica anche gli autovalori.

```
In[45]:=
    Eigenvalues[mat]
Out[45]=
                -a-b + I Sqrt[3] (-a+b)  -a-b - I Sqrt[3] (-a+b)
    {a+b,----------------------, ----------------------}
                       2                        2
```

La scrittura **expr /.{ a->b}** sostituisce ogni occorrenza di **a** con **b** in **expr**. In questo modo è possibile attribuire valori particolari alle variabili simboliche della nostra matrice.

```
In[46]:=
    mat=mat/.{a->0.5,b->1.17};
    MatrixForm[mat]
Out[46]=
    0       0.5     1.17

    1.17    0       0.5

    0.5     1.17    0
```

Adesso la nostra matrice è numerica e i calcoli vengono effettuati nella aritmetica di macchina producendo risultati approssimati.

```
In[47]:=
    Eigenvalues[mat]
Out[47]=
    {1.67, -0.835 + 0.580237 I, -0.835 - 0.580237 I}
In[48]:=
    Det[mat - u IdentityMatrix[3]]
Out[48]=
                          3
    1.72661 + 1.755 u - u
```

Nelle prossime puntate vedremo esempi di grafica e di programmazione nonché un esempio di come alcuni concetti elementari di calcolo numerico possano venire illustrati con *Mathematica*. MC

Francesco Romani è raggiungibile tramite Internet all'indirizzo romani@di.unipi.it

PUZZLE SOFT

LA NUOVA DIMENSIONE NEL DISEGNO ARCHITETTONICO

Il rivoluzionario applicativo di AUTOCAD® che non genera linee, ma sistemi costruttivi predefiniti.

PUZZLE SOFT è un programma applicativo architettonico per la realizzazione di progetti esecutivi. Le sue caratteristiche innovative aprono una nuova dimensione nella progettazione CAD.
Flessibilità - Possibilità di inserire modifiche in cantiere senza dover abbandonare il progetto originario.

Potenza - Creazione automatica di ***44 tipologie di sistemi costruttivi parametrici, con materiali, finiture, spessori diversi***, con possibilità di inserimento di porte, serramenti, sezioni complessive per ogni tipologia di muro.
Semplicità d'uso - Operatività intuitiva con l'utilizzo di icone; disponibilità di ampie librerie tecniche e architettoniche.
Velocità - Grazie al sistema parametrico, PUZZLE SOFT semplifica e velocizza le operazioni di disegno. Il risparmio di tempo in progettazione, ovviamente enorme rispetto al disegno a mano, arriva fino all'80% anche rispetto ad un CAD tradizionale.

AUTOCAD® è un marchio AUTODESK

PUZZLE SOFT è distribuito in Italia dalla ***CIDAS SpA*** *al prezzo di £ 2.500.000.* *

CIDAS SpA - Via Ferrovia, 7 - 31020 San Fior (TV) - Tel. 0438/260150

* La ROSADA SpA si è inserita nel progetto PUZZLE SOFT con una propria libreria. Citando il nome ROSADA nella richiesta, potrete ottenere una riduzione del 35% sul prezzo di listino.