a cura di Francesco Romani

Visualizzazione del moto dei pianeti

Il problema che trattiamo questo mese mi è stato proposto da alcuni insegnanti e consiste nella animazione del moto di punti materiali. Vedremo dapprima come animare un'orbita prefissata a partire dalle sue equazioni parametriche passando poi alla animazione delle soluzioni di equazioni differenziali ordinarie. Naturalmente le animazioni sulla rivista vengono un po' malino e ci dovremo accontentare di far vedere solo alcuni fotogrammi. I programmi sono però completi e chi li vuole provare si può accomodare... Come sempre lo scopo dell'articolo è quello di mostrare possibili applicazioni di *Mathematica* e chiedo scusa in anticipo per la superficiale trattazione del problema fisico.

Orbita ellittica

Definiamo due funzioni **x[t]** e **y[t]** che rappresentano le coordinate in funzione del tempo di un punto che si muove (in senso antiorario) su di un'ellisse.

In[1]:=

```
x[t_] := Cos[t];
y[t_] := 2 Sin[t];
```

La funzione **ParametricPlot** permette di ottenere il grafico a partire dalle equazioni parametriche.

In[2]:=

```
curva = ParametricPlot[{x[t],y[t]},{t,0, 2
Pi}];
```

Vedi Figura 1

Definiamo adesso una funzione di **t** che disegna la curva e un punto rosso nella posizione opportuna al tempo **t** (ricordo che per poter usare i nomi dei colori in inglese è necessario caricare il *package* **Graphics`Colors`**).

In[3]:=

```
Needs["Graphics`Colors`"];
param[t_] := Show[
  curva,
  Graphics[{Red,
            PointSize[0.08],
            Point[{x[t],y[t]}]}]]
```

Per ottenere l'animazione basta eseguire **param** per valori di **t** compresi tra **0** e **2π**. Attenzione: per evitare che il pallino si fermi un attimo nel punto iniziale solo uno dei due estremi deve comparire, ovvero l'animazione deve andare da **π/15** a **2π** oppure da **0** a **2π -π/15**.

In[4]:=

```
Do[param[t],{t,Pi/15, 2 Pi,Pi/15}]
```

In un testo scritto non si possono visualizzare animazioni. *Mathematica* fornisce però la possibilità di mostrare un *array* di grafici.

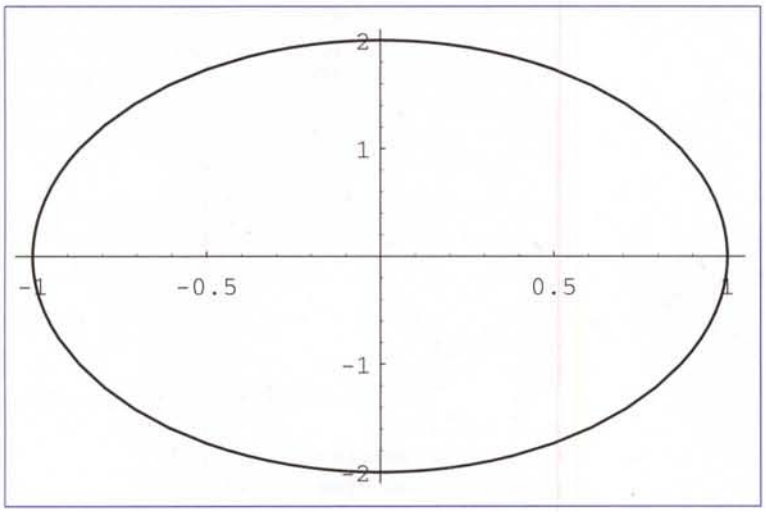


Figura 1

Si noti l'uso di **Partition** per avere un *array* bidimensionale e l'inclusione della chiamata alla funzione **param** in un blocco **Block[{$DisplayFunction = Identity}, ...]** che inibisce la stampa dei singoli fotogrammi, cambiando temporaneamente la funzione di visualizzazione nella funzione identica.

In[5]:=

```
Show[
 GraphicsArray[
  Block[{$DisplayFunction=Identity},
   Partition[
    Table[param[t],{t,Pi/6, 2 Pi,Pi/6}],4]]]];
```

Vedi Figura 2

Il problema dei 2 corpi

Consideriamo ora un corpo puntiforme di massa elevata (il Sole) collocato nell'origine e un corpo di massa minore (un satellite) posizionato ad una certa distanza da esso e con una data velocità iniziale.

Se fissiamo l'origine delle coordinate nel Sole, la posizione nel tempo del satellite è determinata risolvendo le equazioni differenziali che derivano dalla legge di gravitazione universale. Trascurando le costanti (ovvero supponendo che, nel nostro sistema di misura, il prodotto delle masse per la costante di gravitazione sia 1) la forza che agisce sul satellite ha le seguenti componenti:

In[1]:=

```
fx[x_,y_] := -x (x^2 + y^2)^(-3/2);
fy[x_,y_] := -y (x^2 + y^2)^(-3/2);
```

Vediamo il grafico del modulo del campo gravitazionale:

In[2]:=

```
Plot3D[(fx[x,y]^2 + fy[x,y]^2)^(1/2),
   {x,-2,2},{y,-2,2},
   PlotPoints->50,
   ClipFill->None];
```

Vedi Figura 3

In questo caso particolare il moto risultante è una conica e a seconda della velocità e della posizione al tempo **0** si ha una delle seguenti possibilità:

- il satellite cade sul Sole;
- il satellite entra in un'orbita ellittica;
- il satellite sfugge via con una traiettoria parabolica;
- il satellite sfugge via con una traiettoria iperbolica.

Per risolvere numericamente le equazioni differenziali ordinarie *Mathematica* mette a disposizione la funzione **NDSolve** (esiste anche la funzione **DSolve** che tenta una risoluzione analitica).

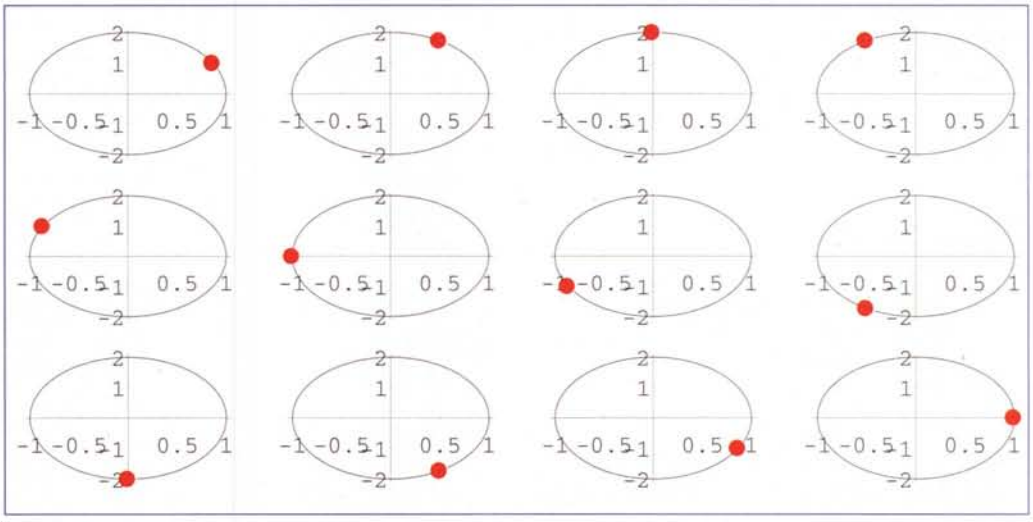


Figura 2

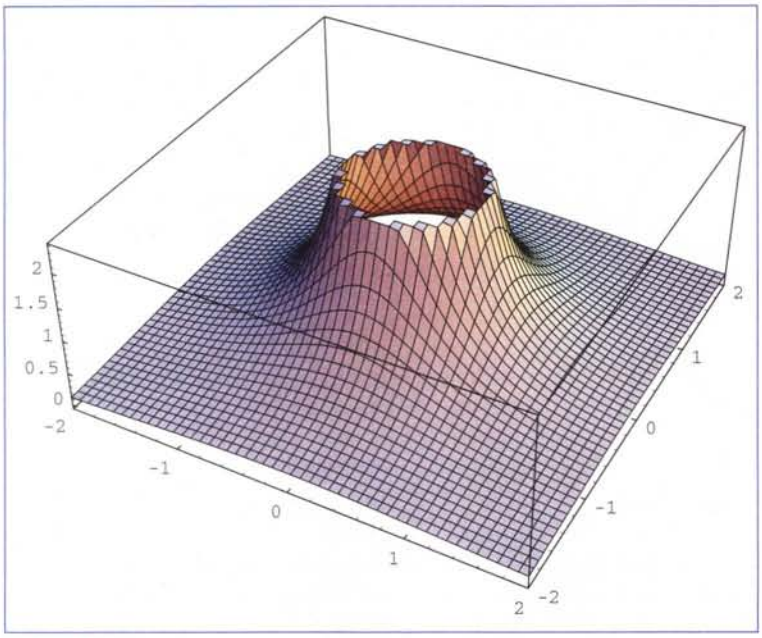


Figura 3

In[3]:=

```
?NDSolve
```

```
NDSolve[eqns, y, {x, xmin, xmax}] finds a numerical solution to the differential equations eqns for the function y with the independent variable x in the range xmin to xmax. NDSolve[eqns, {y1,y2,...}, {x,xmin,xmax}] finds numerical solutions for the functions yi.
```

Scriviamo le equazioni del moto ricavando l'accelerazione dalla forza che agisce sul satellite e dando velocità e posizione all'istante **0** come condizioni iniziali.

Il valore di **x''[0]** è stato determinato sperimentalmente per ottenere un'orbita ellittica.

In[4]:=

```
Clear[x,y];
Eq = {x''[t] == fx[x[t],y[t]],
      y''[t] == fy[x[t],y[t]],
      x'[0]  == 1.12,
      y'[0]  == 0,
      x[0]   == 0,
      y[0]   == 1};
```

Ora possiamo risolvere numericamente l'equazione. Come sempre la finzione di risoluzione restituisce un insieme di regole di sostituzione. Premettendo **{x[t],y[t]}/.** si ottiene la lista delle soluzioni che in questo caso sono funzioni interpolanti che passano per i punti forniti dalla integrazione numerica. Le singole componenti del risultato vengono usate per definire le funzioni x[t] e y[t] che rappresentano le coordinate del satellite in funzione del tempo.

Il tempo di integrazione corretto per ottenere che l'orbita si chiuda è stato determinato sperimentalmente.

In[5]:=

```
Clear[x,y];
sol = {x[t],y[t]}/.
                NDSolve[Eq,  {x[t],y[t]},
{t,0,10}][[1]];
x[t_]:=Evaluate[sol[[1]]];
y[t_]:=Evaluate[sol[[2]]];
```

Il risultato può essere disegnato con lo stesso procedimento usato per l'ellisse; sorge però il problema di determinare il periodo dell'orbita.

Facciamo dapprima il grafico di **x[t]** :

In[6]:=

```
Plot[x[t],{t,0,10}];
```

Vedi Figura 4

Ora possiamo determinare numericamente l'istante **tf** in cui **x[t]** ritorna al valore **0**.

In[7]:=

```
tf=t/.FindRoot[x[t]==0, {t,9}]
```

Out[7]=

```
9.7593
```

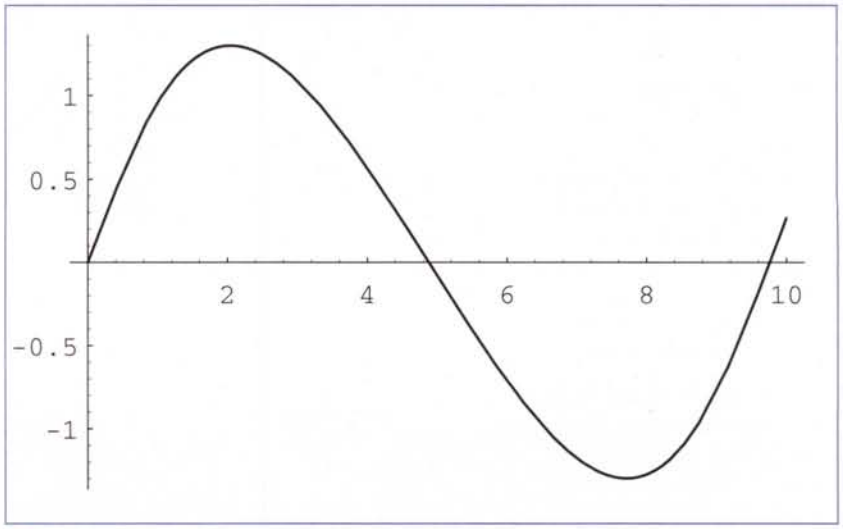


Figura 4

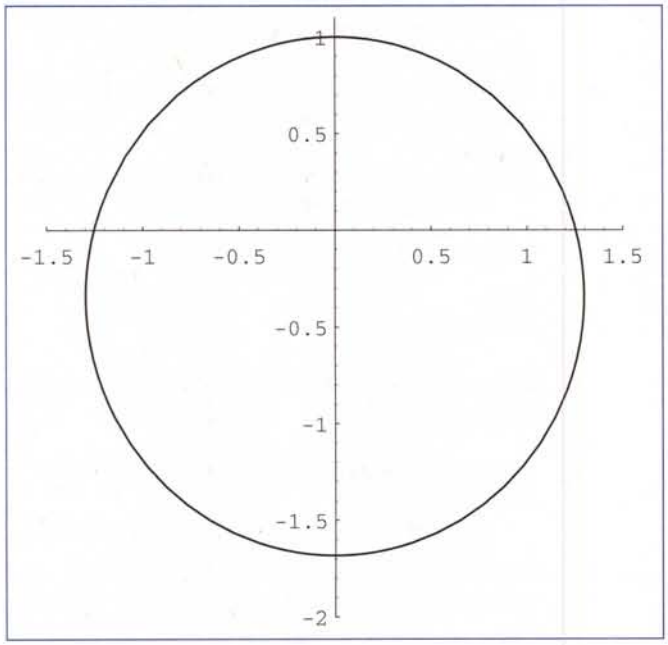


Figura 5

La curva dell'orbita si traccia come in precedenza.

In[8]:=
```
curva = ParametricPlot[{x[t],y[t]},{t,0,tf}];
```

Questo comando disegna una animazione in 20 fotogrammi.

In[9]:=
```
Do[param[t],{t,tf/20,tf,tf/20}]
```

Mentre questa sequenza di istruzioni mostra un *array* di 9 fotogrammi.

In[10]:=
```
Show[
 GraphicsArray[
  Block[{$DisplayFunction=Identity},
   Partition[
    Table[param[t],{t,tf/9,tf,tf/9}],3]]]];
```

Vedi Figura 5

Il problema dei 3 corpi

Se i corpi che interagiscono sono più di due, il moto risultante è molto più complicato ma può essere facilmente esplorato per via numerica.

La forza tra due corpi dipende dalla loro distanza e dalle loro masse (per semplicità abbiamo posto ad 1 la costante gravitazionale).

In[1]:=
```
fx[{x0_,y0_},{x1_,y1_},m0_,m1_]:=
     (x1-x0) m0 m1 ((x0-x1)^2 + (y0-y1)^2)^(-
3/2);
fy[{x0_,y0_},{x1_,y1_},m0_,m1_]:=
     (y1-y0) m0 m1 ((x0-x1)^2 + (y0-y1)^2)^(-
3/2);
```

Consideriamo 3 pianeti di massa **m1**, **m2** e **m3**, le loro posizioni rispetto al tempo sono `{p1x[t], p1[y]}`, `{p2x[t], p2[y]}` e `{p3x[t], p3[y]}`.

Scriviamo le equazioni del moto (la determinazione delle condizioni iniziali ha richiesto molti esperimenti).

In[2]:=
```
m1  = 3;
m2  = 0.5;
m3  = 0.1;
Clear[p1x,p1y,p2x,p2y,p3x,p3y];
Eq = {
  p1x''[t] ==
    fx[{p1x[t],p1y[t]},{p2x[t],p2y[t]}, m1,m2]+
    fx[{p1x[t],p1y[t]},{p3x[t],p3y[t]}, m1,m3],
  p1y''[t] ==
    fy[{p1x[t],p1y[t]},{p2x[t],p2y[t]}, m1,m2]+
    fy[{p1x[t],p1y[t]},{p3x[t],p3y[t]}, m1,m3],
  p2x''[t] ==
    fx[{p2x[t],p2y[t]},{p1x[t],p1y[t]}, m2,m1]+
    fx[{p2x[t],p2y[t]},{p3x[t],p3y[t]}, m2,m1],
  p2y''[t] ==
    fy[{p2x[t],p2y[t]},{p1x[t],p1y[t]}, m2,m1]+
    fy[{p2x[t],p2y[t]},{p3x[t],p3y[t]}, m2,m1],
  p3x''[t] ==
    fx[{p3x[t],p3y[t]},{p1x[t],p1y[t]}, m3,m1]+
    fx[{p3x[t],p3y[t]},{p2x[t],p2y[t]}, m3,m2],
  p3y''[t] ==
    fy[{p3x[t],p3y[t]},{p1x[t],p1y[t]}, m3,m1]+
    fy[{p3x[t],p3y[t]},{p2x[t],p2y[t]}, m3,m2],
  p1x'[0] == 0,
  p1y'[0] == -1,
  p2x'[0] == 0.6,
  p2y'[0] == -1,
  p3x'[0] == 0.2,
  p3y'[0] == -1,
  p1x[0]  == 0,
  p1y[0]  == 0,
  p2x[0]  == 1,
  p2y[0]  == 1,
  p3x[0]  == 0.5,
  p3y[0]  == 0.5};
```

Di nuovo risolviamo numericamente l'equazione.

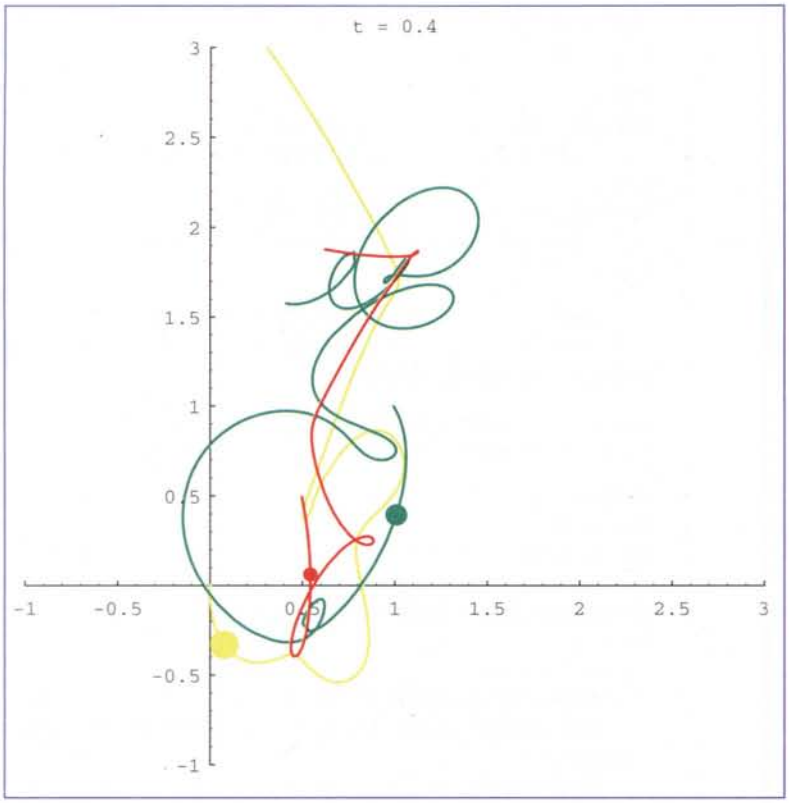


Figura 6

Questa volta aggiungiamo due opzioni che forzano una valutazione più accurata: **MaxSteps** eleva a 2000 il numero massimo di iterazione (il valore *default* è 500); **MaxStepSize** pone un limite superiore al passo di integrazione.

In[3]:=

```
Clear[p1x,p1y,p2x,p2y,p3x,p3y];
sol                                                    =
{p1x[t],p1y[t],p2x[t],p2y[t],p3x[t],p3y[t]}/.
                                          (NDSolve[Eq,
{p1x[t],p1y[t],p2x[t],p2y[t],p3x[t],p3y[t]},
     {t,0,5},
    MaxStepSize->0.05,
    MaxSteps->2000][[1]]);
p1x[t_]:=Evaluate[sol[[1]]];
p1y[t_]:=Evaluate[sol[[2]]];
p2x[t_]:=Evaluate[sol[[3]]];
p2y[t_]:=Evaluate[sol[[4]]];
p3x[t_]:=Evaluate[sol[[5]]];
p3y[t_]:=Evaluate[sol[[6]]];
```

Tracciamo ora le traiettorie dei tre pianeti:

In[4]:=

```
curva1                                                 =
ParametricPlot[{p1x[t],p1y[t]},{t,0,5},
   PlotStyle->Yellow];
curva2                                                 =
ParametricPlot[{p2x[t],p2y[t]},{t,0,5},
   PlotStyle->Green];
curva3                                                 =
ParametricPlot[{p3x[t],p3y[t]},{t,0,5},
   PlotStyle->Red];
```

Definiamo tre punti colorati mobili:

In[5]:=

```
p1[u_,v_] :=

{Yellow,PointSize[0.04],Point[{u,v}]};
p2[u_,v_] :=

{Green,PointSize[0.03],Point[{u,v}]};
p3[u_,v_] :=
   {Red,PointSize[0.02],Point[{u,v}]};
```

Combinando le traiettorie con i punti si può disegnare il fotogramma all'istante **t**.

In[6]:=

```
mostra[t_]:= Show[
 curva1,
 curva2,
 curva3,
 Graphics[{
  p1[p1x[t],p1y[t]],
  p2[p2x[t],p2y[t]],
  p3[p3x[t],p3y[t]]}],
 PlotRange->{{-1,3},{-1,3}},
 AspectRatio->1,
 Axes->True,
 PlotLabel->"t = "<>ToString[t]]
```

Per generare l'animazione basta stampare molti fotogrammi.

In[7]:=

```
Do[mostra[t],{t,0,5,0.05}]
```

In **Figura 6** ne mostriamo uno solo (provate poi a girare questo programma per godervi l'effetto).

MC

Bibliografia

VisualDSolve: Visualizing Differential Equations with Mathematica, Springer Verlag/TELOS (1996)

ISDN world

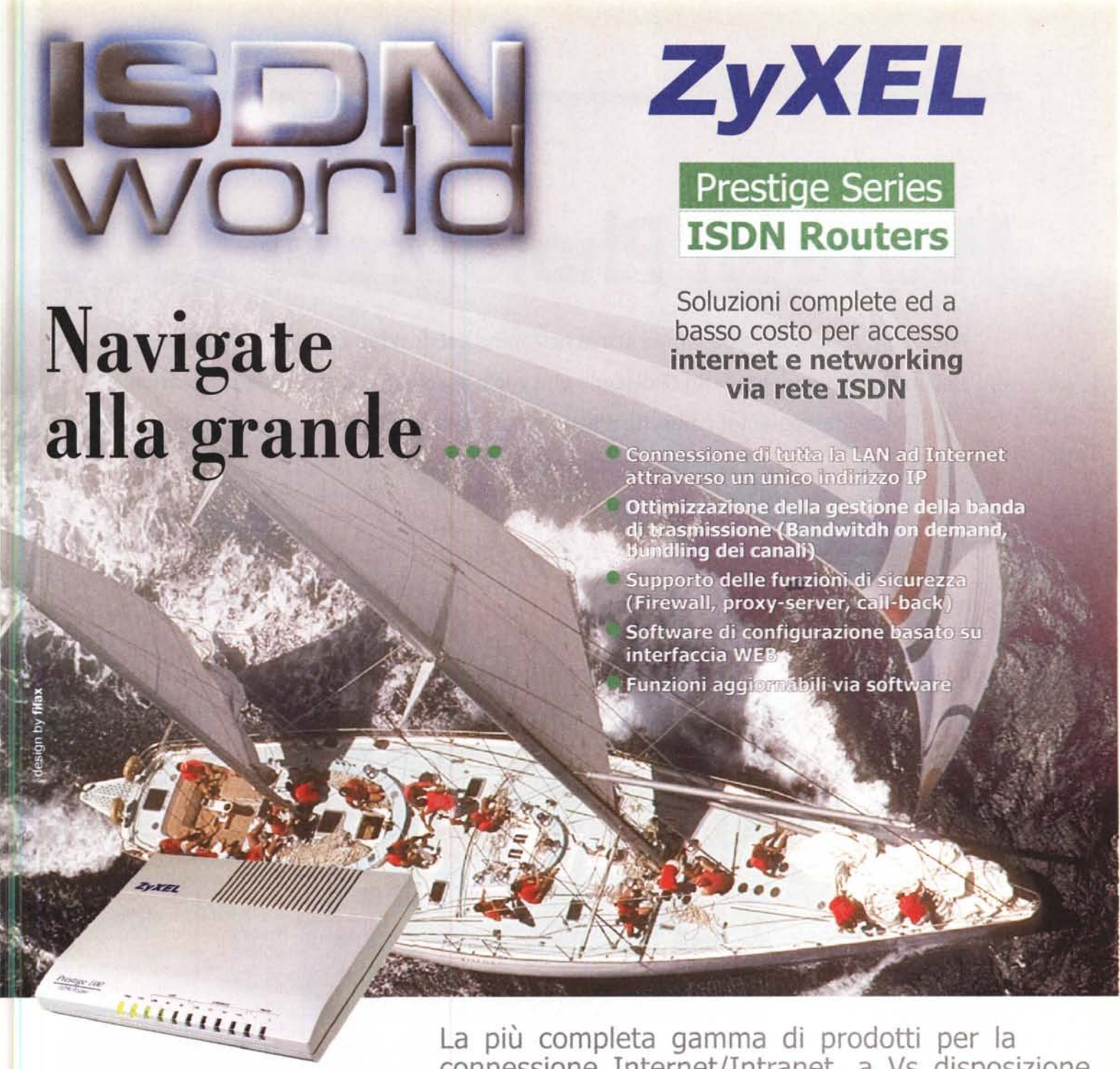
ZyXEL

Prestige Series
ISDN Routers

Soluzioni complete ed a basso costo per accesso **internet e networking via rete ISDN**

Navigate alla grande ...

- Connessione di tutta la LAN ad Internet attraverso un unico indirizzo IP
- **Ottimizzazione della gestione della banda di trasmissione (Bandwitdh on demand, bundling dei canali)**
- **Supporto delle funzioni di sicurezza (Firewall, proxy-server, call-back)**
- **Software di configurazione basato su interfaccia WEB**
- **Funzioni aggiornabili via software**



design by fifax

La più completa gamma di prodotti per la connessione Internet/Intranet, a Vs disposizione dal distributore italiano specializzato in **ISDN.**

CoFax®
TELEMATICA

DA 10 ANNI AL SERVIZIO DELLA COMUNICAZIONE VELOCE

Per maggiori informazioni - **http://www.cofax.it**
Roma - 00151 V.le dei Colli Portuensi, 110/a
Tel. 06/58201362 r.a. **Fax 06/58201550**
Milano - 20129 C.so Buenos Aires, 37
Tel. 02/29526100 r.a. **Fax 02/29520884**

Numero Verde **167-865108**

BVQI ISO 9002

Deutscher Akkreditierungs Rat DAR Cert. N°48135

ISDN world by CoFax Telematica ®

NET working by CoFax Telematica ®