di Francesco Romani

Suoni e Segnali

Questo è il primo di una serie di articoli dedicati al trattamento dei suoni e all'analisi dei segnali con *Mathematica*. Lo scopo è di mettere il lettore in grado di fare esperimenti di generazione di suoni e di trattamento numerico di segnali. L'uso di *Mathematica*, se da un lato limita drasticamente la dimensione dei problemi trattabili dall'altro permette di effettuare esperimenti molto avanzati con poca fatica e in modo indipendente dalla macchina su cui si lavora. Ovviamente la parte più interessante di questo articolo sono i suoni (che non possiamo stampare). Chi possiede una copia di *Mathematica* può sentirseli eseguendo i nostri programmi.

Trattamento dei suoni in *Mathematica*

Il trattamento dei suoni in *Mathematica* è simile a quello dei grafici monodimensionali. Esistono due funzioni base **Play** e **ListPlay** che sono le equivalenti di **Plot** e **ListPlot**, rispettivamente. Esse permettono di suonare una funzione oppure una lista di valori e contemporaneamente fanno un grafico dell'andamento dei segnali da suonare. Come le funzioni grafiche rendono un oggetto di tipo `-Graphics-` così le nostre rendono un oggetto di tipo `-Sound-`. Gli oggetti di tipo `-Sound-` possono essere mostrati e suonati con **Show**.

In[1]:=
```
?Play
Play[f, {t, tmin, tmax}] plays a sound whose
amplitude is given by f as a function of time
t in seconds between tmin and tmax.
```

Suoniamo un'onda a 50 Hz per un decimo di secondo.

In[2]:=
```
Play[Sin[2 Pi 50 x],{x,0,0.1}]
```

Vedi Figura 1

Vediamo ora la forma interna del risultato. **Sound** è la "testa" che determina il tipo di dato, il contenuto può essere un oggetto di tipo **SampledSoundFunction** oppure **SampledSoundList**. Nel nostro caso compare la forma (sia compilata che estesa) della funzione che abbiamo passato a **Play**.

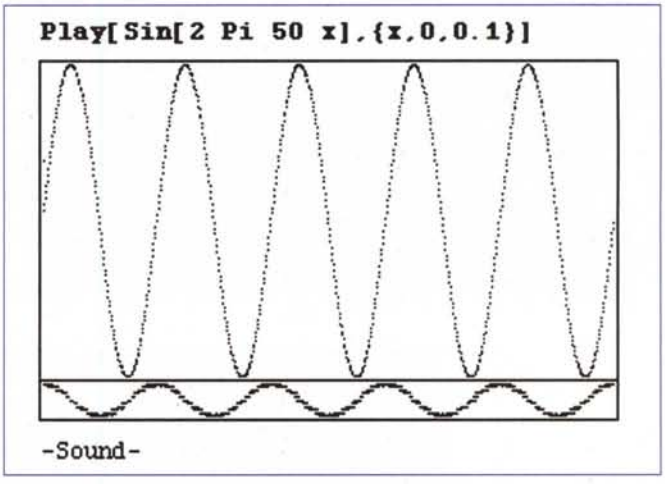

Figura 1

In[3]:=
```
InputForm[%]
```

Out[3]//InputForm=
```
Sound[SampledSoundFunction[
CompiledFunction[{_Integer}, <<…>>],
 Function[{Play`Time1}, Block[<<…>>], 819,
8192]]
```

In[4]:=

?SampledSoundFunction

```
SampledSoundFunction[f, n, r] is a sound primitive, which represents a sound whose amplitude sampled r times a second is generated by applying the function f to successive integers from 1 to n.
```

In modo analogo si può suonare una qualunque lista di numeri, anche casuali:

In[5]:=

?ListPlay

```
ListPlay[{a1, a2, ...}] plays a sound whose amplitude is given by the sequence of levels ai.
```

In[6]:=

```
ListPlay[Table[Random[],{100}]]
```

Vedi Figura 2

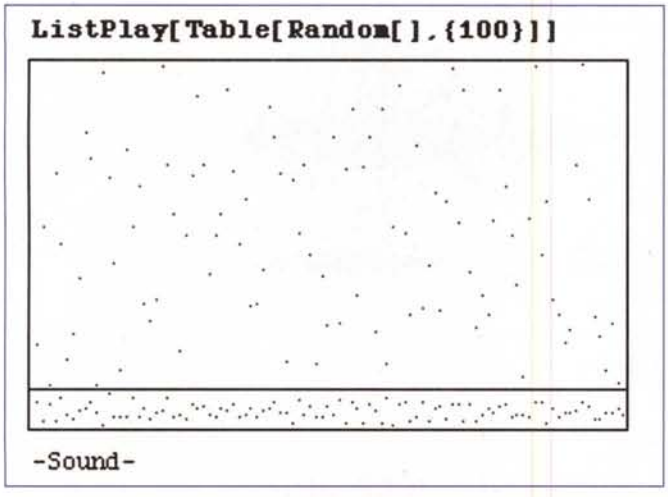

Figura 2

La forma interna del risultato è un oggetto di tipo **SampledSoundList**. Il numero 8192 rappresenta la frequenza di campionamento.

In[7]:=

```
InputForm[%]
```

Out[7]//InputForm=

```
Sound[SampledSoundList[
  {0.604526…, 0.623793…, 0.352284…, <<…>> -
0.404894…}, 8192]]
```

In[8]:=

?SampledSoundList

```
SampledSoundList[{a1, a2, ...}, r] is a sound primitive, which represents a sound whose amplitude has levels ai sampled r times a second.
```

Un esempio di sintesi

Il pacchetto **Miscellaneous`Audio`** contiene molte funzioni utili alla sintesi di segnali musicali. Tanto per cominciare vediamo sia la modulazione di ampiezza che la modulazione di frequenza

In[1]:=

```
Needs["Miscellaneous`Audio`"]
```

In[2]:=

?AmplitudeModulation

```
AmplitudeModulation[fc, fm, mi, d, opts] returns a Sound object that is an amplitude modulated sinusoid, with fc and fm being the carrier and modulating frequencies in Hertz, mi the modulation index, and d the duration of the sound in seconds.  If the option  Ring is set to True, the sound will be ring-modulated.
```

In[3]:=

```
Show[AmplitudeModulation[300,30,0.5,0.1]]
```

Vedi Figura 3

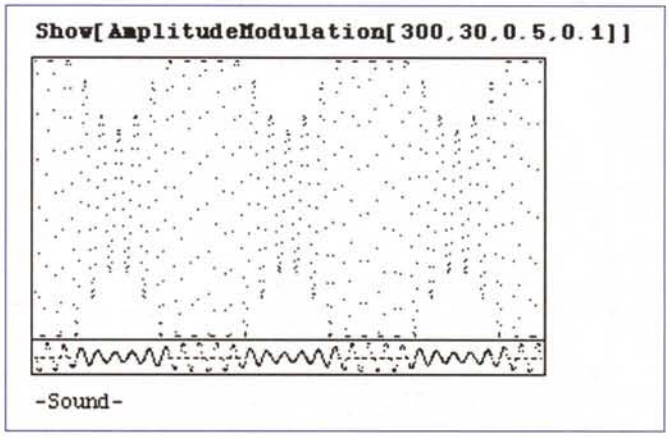

Figura 3

In[4]:=

?FrequencyModulation

```
FrequencyModulation[fc, {fm, pd}, d, opts] returns a Sound object that is a frequency modulated sinusoid, where fc, fm, and pd are the carrier, modulator, and peak deviation frequencies in Hertz, and d is the duration in seconds. The option Type can be set to Standard (default), Cascade, or Parallel. For Cascade and Parallel, the second argument must be a list of pairs of the form {{fm1, pd1}, {fm2, pd2}, ...}, where pdi is the peak deviation associated with modulating frequency fmi, and both values are measured in Hertz.
```

In[13]:=

```
Show[FrequencyModulation[300,{30,100},0.1]]
```

Vedi Figura 4

Il comando Waveform permette di generare un'onda con una forma predeterminata (sinusoide, dente di sega, quadra, triangolare)

In[14]:=

?Waveform

```
Waveform[t, f, d, opts] returns a Sound object of
```

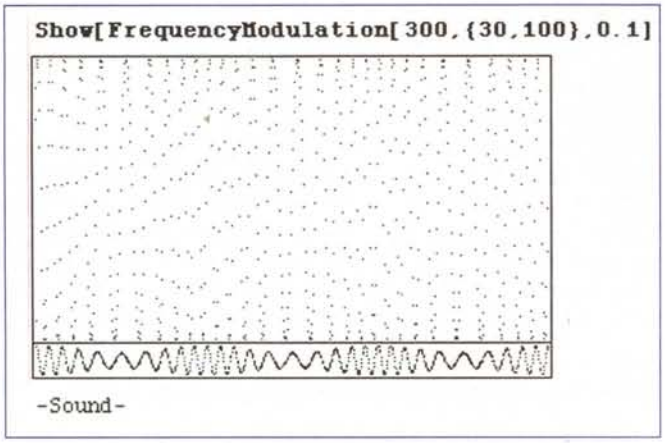

Figura 4

waveform type t, having a fundamental frequency f Hertz, and duration of d seconds. The type t must be one of the following: Sinusoid, Sawtooth, Square, Triangle. The option Overtones sets the number of overtones that will be present in the sound.

In[15]:=

```
Show[Waveform[Sawtooth,50,0.1]]
```

Vedi Figura 5

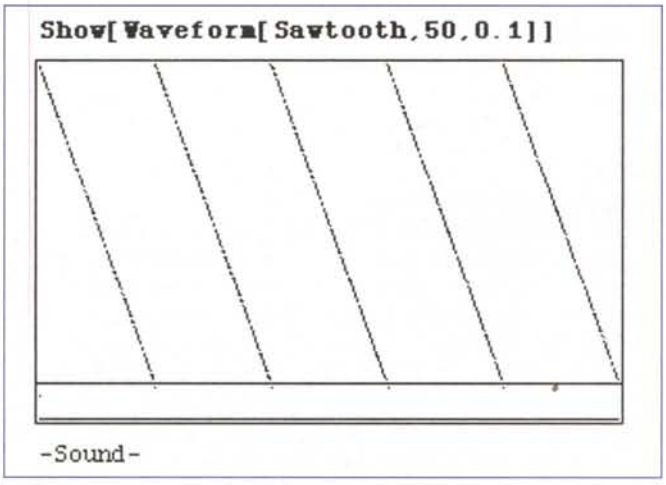

Figura 5

In[16]:=

```
InputForm[%]
```

Out[16]//InputForm=

```
Sound[SampledSoundFunction[
  CompiledFunction[{_Integer}, {0, 5, 15, 0},
<<...>> {8, 14}},
 Function[{Play`Time4}, Block[...], 819, 8192]]
```

Un esempio di generazione di un segnale complesso abbastanza gradevole si può trovare nel file fornito con la versione 3.0 di *Mathematica*. (Attenzione per l'esecuzione servono molto tempo e molta memoria!).

In[28]:=

```
Play[
 Sin[Pi t/2] Sum[
  If[v/12<t<v/12+1/4,
   Exp[-(3+2v-24t)^2/36]*
    v Sin[1625 v t - 13000t]/24,
      0],{v,0,24}],
 {t,0,2}]
```

Vedi Figura 6

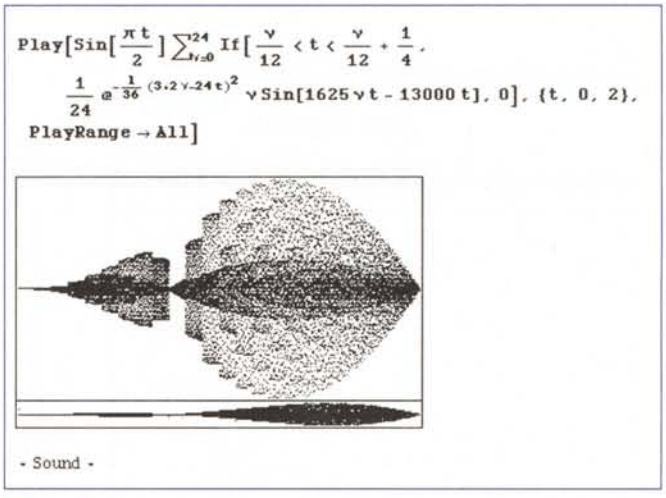

Figura 6

Lettura di tracce audio

La parte più interessante del pacchetto **Miscellaneous`Audio`** è certamente la possibilità di leggere tracce audio da un Hard Disk. I formati più interessanti sono il formato .waw per Windows e il formato AIFF per Macintosh.

In[1]:=

```
?ReadSoundfile
ReadSoundfile["soundfile"] reads the specified sound file, and returns a list of amplitudes between -32768 and +32767. If the option PrintHeader is set to True, the header information in the sound file (the sampling rate, sample width, etc.) will be displayed.
```

Con un lettore di CD e un'utilità per copiare le tracce audio su Hard Disk è possibile estrarre una qualunque traccia musicale (a circa 10 Mbytes al minuto).

Se la traccia è piccola può essere letta direttamente altrimenti è bene ridurla con un qualunque editor musicale. Per esempio abbiamo copiato una traccia di un disco test Hi-Fi.

In[18]:=

```
snd=ReadSoundfile[":sound:CD:Fade to Noise"]
  PrintHeader->True];
Format: Apple AIFF
```

```
Duration: 1.04667 seconds
Channels: 2
Sampling rate: 44100.
Bits per sample: 16
Data size: 369264 bytes
Number of samples: 92316
```

Mathematica riconosce il tipo del file, legge le specifiche interne del formato AIFF, individua la frequenza di campionamento (44.100 Hz standard CD), la lunghezza delle parole (16 bit) e rende due liste che contengono rispettivamente i campioni del canale sinistro e del canale destro. Le due liste possono essere suonate oppure elaborate.

In[19]:=
```
left=snd[[1]];
ListPlay[left]
```

Vedi Figura 7

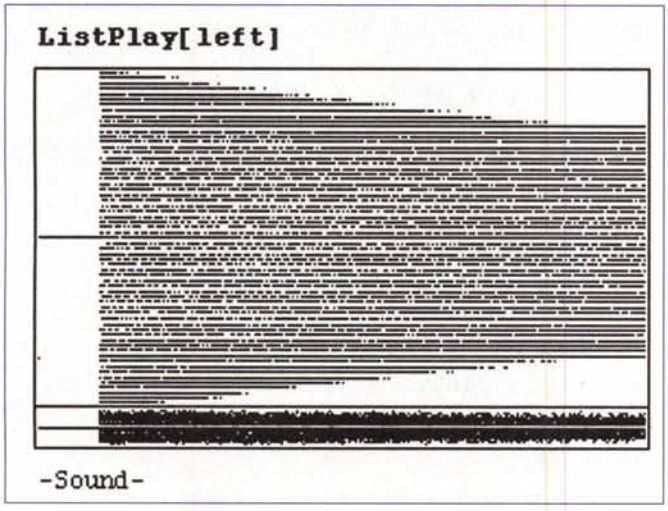

Figura 7

Analisi ed elaborazione dei segnali

Una volta che un segnale (musicale) campionato è stato caricato in memoria è possibile elaborarlo. Si distinguono due grandi filoni: l'analisi del segnale e l'elaborazione vera e propria. Nel primo caso l'interesse è quello di ricavare dal segnale parametri numerici in forma numerica oppure sotto forma di grafico. Nel secondo caso il segnale, una volta rielaborato, viene salvato in forma opportuna per essere suonato di nuovo. Vedremo nelle prossime puntate come uno studio non banale della elaborazione non possa prescindere dai fondamenti matematici della analisi dei segnali, in particolare sarà necessario introdurre quello strumento fondamentale che è l'analisi nel dominio della frequenza.

In questa puntata ci limitiamo al più semplice esempio di analisi nel dominio del tempo; la misura del livello del segnale.

Tipicamente un segnale audio rappresenta la tensione o la corrente in funzione del tempo. Se si suppone che questo segnale sia amplificato in modo lineare e applicato ad un trasduttore (per esempio un altoparlante) di impedenza costante e resistiva si vede che la potenza irradiata è (con molta approssimazione) proporzionale al prodotto della tensione per la corrente e quindi al quadrato del segnale audio.

Poiché la sensibilità dell'orecchio umano è logaritmica si preferisce misurare i livelli in termini logaritmici invece che lineari. L'unità di misura di un rapporto di segnali audio è il **Bel** che rappresenta un rapporto 10, in pratica si usa solo il suo sottomultiplo, il **decibel** (abbreviato **dB**). Il rapporto di potenza tra due segnali di valore v1 e v2 è quindi

$20 \log_{10} v1/v2$ dB.

Come livello 0 si può avere un livello convenzionale. Ad esempio i livelli di rumore ambientali sono misurati rispetto ad uno zero prefissato e si indicano in **dBA**, ed è in questa scala che si esprimono per esempio i limiti delle normative antirumore.

Nol nostro caso conviene prendere come livello 0 il massimo livello rappresentabile nei 16 bit della parola discretizzata ovvero il numero $32768=2^{15}$. Quindi il livello istantaneo di un segnale si può esprimere come

In[1]:=
```
levist[x_] := 20 Log[10, Abs[x]/2^15]
```

Poiché i segnali audio sono onde il livello istantaneo ha ben poco significato. La potenza media su un certo numero di campioni è data dalla media della somma dei quadrati dei campioni stessi (potenza RMS). Allora il livello medio in dB su un gruppo di campioni, riferito al massimo livello possibile, vale:

In[2]:=
```
levRMS[v_]:=10. Log[10.,
  Plus@@(v^2)/Length[v]/2^30.];
```

Si noti il gioco delle costanti: siccome i campioni vengono elevati al quadrato il termine davanti al logaritmo è 10 e non 20 e l'esponente di 2 nel fattore di normalizzazione è 30 e non 15.

In[3]:=
```
gruppi = Partition [left,441];
```

In[4]:=
```
ListPlot[levRMS/@gruppi,PlotJoined->True,
    PlotLabel->"Livello RMS",
    AxesLabel->{"sec/100","dB"}]
```

From In[4]:=
```
Graphics::gptn: Coordinate -Infinity in {1, -In-
finity} is not a floating-point number.
Graphics::gptn: Coordinate -Infinity in {2, -In-
finity} is not a floating-point number.
Graphics::gptn: Coordinate -Infinity in {3, -In-
finity} is not a floating-point number.
General::stop: Further output of Graphics::gptn
will be suppressed during this calculation.
```

Vedi Figura 8

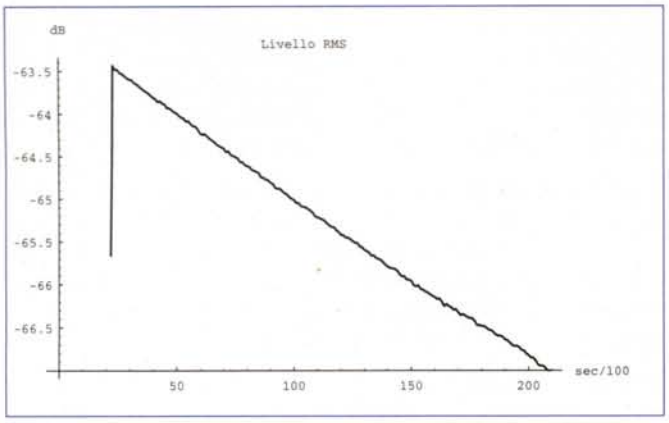

Figura 8

Il grafico mostra chiaramente quale è la natura del segnale in esame. Si noti che il segnale inizia con molti campioni nulli che, se non danno fastidio al grafico, fanno generare molti messaggi di errore alla funzione logaritmo. Inoltre, in realtà, il segnale è oscillante e una diversa scelta del numero di campioni su cui fare la media lo rivela chiaramente. Scriviamo una funzione che fa la prova su **x** campioni.

In[5]:=

```
test[x_]:=ListPlot[levRMS/@Partition[left,x],
  PlotJoined->True,
  PlotLabel->"Media su "<>
             ToString[x]<>" campioni",
  AxesLabel->{"","dB"}]
```

E vediamo il grafico per x=200, 300, 700 e 800

In[6]:=

```
Show[GraphicsArray[Partition[
 {test[200],test[300],
  test[700],test[800]},2]]];
```

Vedi Figura 9

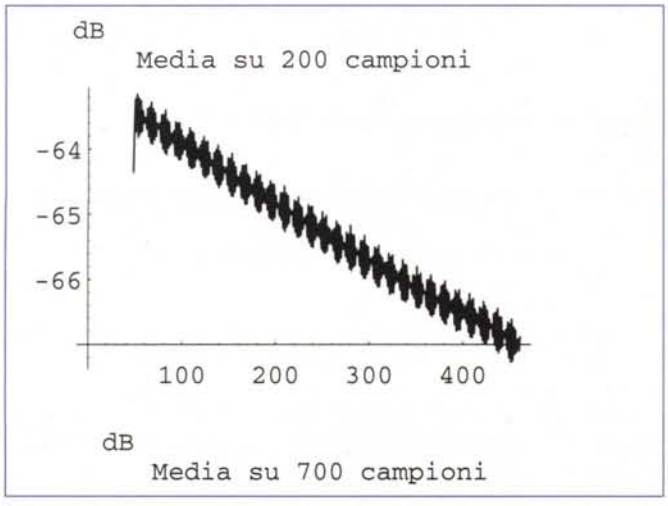

Figura 9

MC

Bibliografia ragionata

Oltre al già citato pacchetto **Miscellaneous`Audio`** fornito con ogni copia di *Mathematica* esistono alcuni pacchetti adatti alla elaborazione di segnali di cui parleremo nelle prossime puntate.

Il pacchetto a pagamento **Signal and Systems** offre molti degli strumenti classici di *signal processing* e di analisi con una completa integrazione tra il calcolo numerico e quello simbolico.

Il pacchetto a pagamento **Wavelet Explorer** offre le tecniche più avanzate di analisi di segnali ed immagini basati sulle *Wavelet*.

Pacchetti e *Notebook* più rudimentali ma gratuiti sono reperibili nella raccolta ***MathSource;*** anche di alcuni di questi parleremo in maggiore dettaglio in seguito.

Nel *directory* `/Applications/Audio` del CD del 1994 ho trovato i seguenti *items:*

0201-777: **Sounds from Mathematical Functions** (1991) di Arun Chandra

0202-217: **A Composer's Guide to Sound Production with *Mathematica*** (1992) di Arun Chandra

0204-321: **Filtering of SampledSoundList Objects** (1992) di Ben Cox

0205-489: **Touch Tones** (1993) di Arun Chandra

0201-856: **Polyphonic Composition** (1994) di Arun Chandra

Nel *directory*

`/Applications/Engineering/ Electrical/ Signals`

dello stesso CD vi sono:

0200-327: **Digital Signal Processing Tools** (1991) di Kevin McIsaac

0202-240: **Signal Processing Packages and Notebooks** (1994) di Brian Evans, James McClellan, Kevin West, Wallace McClure, Lena Karam, and Jim Proctor

0202-644: **Rectangular, Hanning, and Hamming Window Transforms** (1991) di Julius O. Smith III

0205-265: **A Review of Filter Design** (1993) di Malcolm Slaney

0205-276: **An Efficient Implementation of the Patterson-Holdsworth Auditory Filter Bank** (1993) di Malcolm Slaney

0205-502: **The Kaiser Window** (1992) di Julius O. Smith III

0205-535: **Make Your Own Convolution** (1992) di H. Joel Trussell

0205-546: **Filter Design by Pole-Zero Placement** (1992) di H. Joel Trussell

AVM

ISDN world

INTERNET, FAX, SEGRETERIA TELEFONICA...

Il marchio leader dei prodotti ISDN in Germania, ti propone la migliore soluzione per le connessioni ISDN, oggi ad un prezzo ancora più vantaggioso...

design by filax

FRITZ! CARD

La tecnologia 32bit-VxD
permette contemporaneamente la connessione ad Internet e la ricezione/trasmissione Fax.

Supporta i principali Sistemi-Operativi
Windows 3.1/95/NT, Novell, Linux, ecc.

Gamma hardware completa
è possibile scegliere tra la versione ISA/AT PnP e non, e la versione Bus PCI e PCMCIA.

Internet, Fax, Segreteria Telefonica
Il software in dotazione permette di Tx/Rx Fax dal proprio PC, di accedere ad Internet e a BBS, di utilizzare il PC come segreteria telefonica, tutto attraverso una interfaccia utente semplice ed immediata.

Con la Fritz! Card puoi risparmiare senza rinunciare alla qualità!

BUONO SCONTO DI LIRE 50.000

Fritz! Card ISDN, la scheda numero uno in Germania, per festeggiare la milionesima installazione, Ti mette a disposizione un buono sconto del valore di £. 50.000 per entrare nel mondo della comunicazione veloce... Compila in ogni sua parte il seguente coupon, ed invialo per **Fax** al num.verde: 167-865109

Nome	**Cognome**	
Società	**P.IVA**	
Indirizzo		
Città	**CAP**	**Tel.**

Con riferimento alle vigenti normative sulla riservatezza dei dati personali, Vi autorizzo ad utilizzare le informazioni contenute nel presente coupon per la sola finalità di essere aggiornato sulle Vs. iniziative commerciali.
Firma (leggibile)

CoFax® TELEMATICA

http://www.cofax.it
Roma 00151 Viale 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**