a cura di Francesco Romani

Errori di calcolo

Questo mese trattiamo un argomento che sta alla base della Analisi Numerica: la teoria della rappresentazione dei numeri e dei conseguenti errori di calcolo. Come al solito la trattazione è solo un antipasto; alcuni tempi verranno ripresi in puntate successive e per chi volesse approfondire l'argomento, ottimi testi specialistici sono citati in bibliografia.

Le rappresentazioni dei numeri

Non si possono comprendere i tipi e le cause degli errori nella aritmetica del *computer* se prima non si studia come sono rappresentati i numeri. Le rappresentazioni possibili sono praticamente infinite, vediamone le principali.

Numeri interi a precisione fissa

I numeri interi sono di solito rappresentati in binario su 2 o 4 *byte*. Ciò implica che c'è un limite invalicabile al massimo numero rappresentabile. Se si tenta di superarlo avviene un errore di *overflow* che spesso sui *computer* attuali non è segnalato e causa fastidiose "incomprensioni" come nel programma Pascal che segue.

```
program int;
  var  a, i: integer;
begin
 a := 16000;
 for i := 1 to 3 do begin
     writeln(a);
      a := a * 2;
 end;
end.
```

Quando il ciclo **for** viene percorso per la terza volta il valore a=32000 viene raddoppiato e si ottiene un numero negativo (in questo esempio gli interi sono rappresentati su 2 *byte* e il massimo numero rappresentabile supera di poco 32000).

```
16000
32000
-1536
```

Numeri interi a precisione illimitata

Memorizzando gli interi in vettori di lunghezza arbitraria invece che in parole di lunghezza fissa si evita il rischio di *overflow* a scapito del tempo di elaborazione e di un'occupazione di memoria che può divenire eccessiva. *Mathematica* passa automaticamente a questo tipo di rappresentazione non appena i calcoli lo richiedano. Proviamo a calcolare il fattoriale di 200. (la barra \ indica che il numero segue nella linea successiva)

In[1]:=

```
200!
```

Out[1]=

```
788657867364790503552363213932185062\
 295135977687173263294742533244359 4\
 499634033429203042840119846239041 7\
 721213891963883025764279024263710 5\
 061926624952829931113462857270763 3\
 172373969889439224456214516642402 5\
 403329186413122742829485327752424 2\
 407573903240321257405579568660226 0\
 319041703240623517008587961789222 2\
 278962370389737472000000000000000 0\
 000000000000000000000000000000000
```

Numeri razionali a precisione illimitata

Con una coppia di interi a precisione illimitata si possono rappresentare tutti i numeri razionali. *Mathematica* offre tra l'altro la funzione **Rationalize** che cerca il numero razionale più vicino ad un numero reale dato. Cerchiamo un'approssimazione razionale di π con un errore minore di 10^{-40}.

In[2]:=

```
Rationalize[N[Pi,40],10^-40]
```

Out[2]=

```
26509932346052150374 3
-----------------------
84383735478118508040
```

In[2]:=

```
N[%-Pi,50]
```

Out[2]=

```
-4.06672232 10^-41
```

Numeri reali a precisione fissa

I numeri reali irrazionali (per esempio la radice quadrata di 2) non ammettono rappresentazioni finite e sono quindi "tabù" per ogni calcolatore. Quella che viene usata di solito è la rappresentazione **a virgola mobile** (***Floating Point***). Un numero *floating point* viene scritto come $\pm b^e m$ dove

- $\pm$ vale +1 o -1, rappresenta il **segno** e sta in un *bit*;
- b è la **base** della rappresentazione (per esempio 2) e non viene memorizzata;
- e è **l'esponente** a cui viene elevata la base, è un numero relativo che rappresenta la posizione della virgola mobile nella scrittura del numero in base b; e viene memorizzata (per esempio in 7 *bit*);
- m è la **mantissa**: sono le cifre più significative del numero da rappresentare e viene memorizzata nei restanti *byte* (per esempio 3 se il numero deve stare in 4 *byte*, 7 se deve stare in 8).

Una volta scelta la rappresentazione, solo un sottoinsieme finito di numeri razionali (detto insieme dei **numeri di macchina**) può essere rappresentato. Ogni altro numero reale viene **approssimato** scrivendo un numero di macchina ad esso vicino. Ciò implica, tra l'altro, un curioso fenomeno: esistono numeri di macchina ε tali che $1+\varepsilon$ assume il valore 1. Il più grande numero ε che soddisfa questa proprietà viene detto **precisione di macchina**.

Il seguente programma Pascal divide per due il contenuto della variabile a finché non si trova un numero con la proprietà sopra enunciata.

```
program real;
  var  a, b: real;
begin
 a := 0.000001;
 repeat
    writeln(a : 15);
    b := 1 + a;
    a := a / 2;
 until b = 1;
end.
```

```
 1.000000e-6
 5.000000e-7
 2.500000e-7
 1.250000e-7
 6.250000e-8
 3.125000e-8
```

Il valore che si ottiene dà un'idea del numero di cifre con cui vengono fatti i calcoli. Sostituendo double al posto di real si ottiene:

```
 1.000000e-6
 ...
 1.164153e-16
 5.820766e-17
```

Lo stesso calcolo si può fare con *Mathematica* con l'operatore **FixedPoint** usato per iterare una funzione pura fino a trovarne un punto fisso.

In[3]:=
```
x=1.;
FixedPoint[(x=x/2.;1.+x)&, ];
x
```

Out[3]=
```
4.44089 10^-16
```

Numeri reali a precisione illimitata

Anche per i numeri reali è possibile una memorizzazione in vettori che garantisce un numero illimitato di cifre di precisione. E anche in questo caso la memoria e il tempo di elaborazione crescono inesorabilmente.

In[4]:=
```
N[Pi,200]
```

Out[4]=
```
3.14159265358979323846264338327950288\
 84197169399375105820974944592307816\
 40628620899862803482534211706798 2\
 14808651328230664709384460955058 22\
 31725359408128481117450284102701 93\
 85211055596446229489549303819 6
```

Aritmetica degli intervalli

Si tratta di un tipo di aritmetica molto sofisticato: invece di un valore approssimato di un numero reale si utilizza un intervallo che lo contiene. Questo modo di procedere (anch'esso implementato in *Mathematica*) permette di tenere sotto controllo in maniera rigorosa gli errori di calcolo, a spese di un notevole tempo di elaborazione e di alcune difficoltà matematiche non banali.

In[5]:=
```
a=Interval[{1.1, 1.2}];
b=Interval[{3.17, 3.175}];
```

In[6]:=
```
a+b
```

Out[6]=
```
Interval[{4.27, 4.375}]
```

In[7]:=
```
a b
```

Out[7]=
```
Interval[{3.487, 3.81}]
```

In[8]:=
```
a-b
```

Out[8]=
```
Interval[{-2.075, -1.97}]
```

In[9]:=
```
a/b
```

Out[9]=
```
Interval[{0.346457, 0.378549}]
```

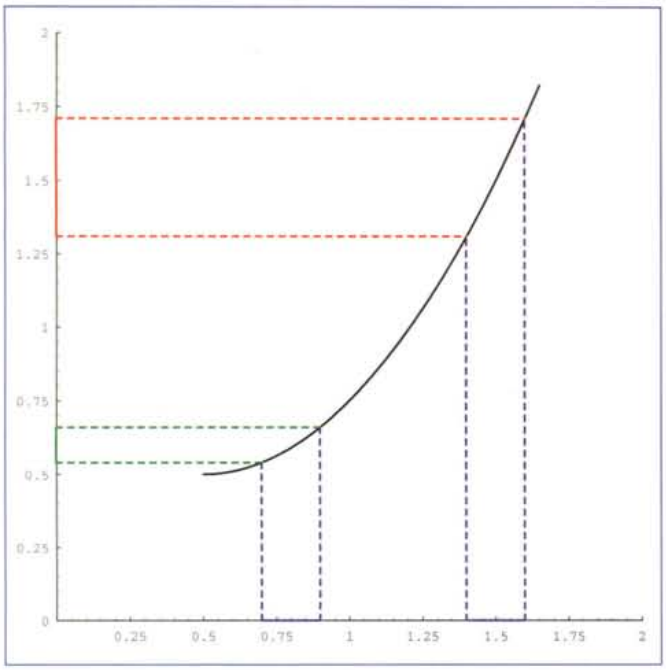

Figura 1

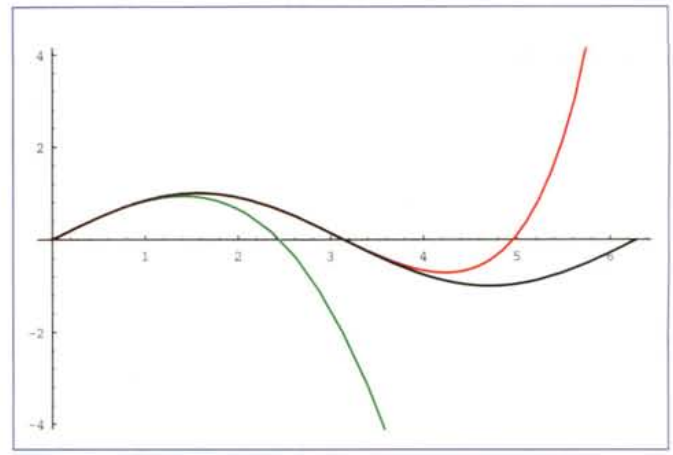

Figura 2

Le cause di errore

Consideriamo una funzione f(x), un programma che la calcola in un punto x_0. Generalmente il risultato è affetto da tre tipi di errori.

Errore inerente

La prima causa di errore è quella dovuta alla imprecisione dei dati: se ad un programma per il calcolo della superficie di un appartamento fornisco dati misurati a mano col metro (e quindi affetti di un errore dell'ordine del centimetro) difficilmente riuscirò ad ottenere più di due, tre cifre di precisione anche se il calcolatore internamente lavora con 16 cifre decimali. L'errore indotto dalla imprecisione sui dati è detto **errore inerente** e la sua grandezza dipende sia dalla imprecisione dei dati che dalla derivata prima della funzione che dobbiamo calcolare.

In **Figura 1** si vede come uno stesso errore di 0.2 sul dato (segmenti in blu sull'asse dell'ascissa) viene diversamente amplificato a seconda si debba calcolare un funzione crescente lentamente (in verde) o molto velocemente (in rosso).

L'errore inerente non può essere stimato fino a che non è nota la (im)precisione dei dati in ingresso.

Errore analitico

Con un calcolatore in un tempo finito può essere calcolata solo una sequenza finita di operazioni aritmetiche (+, -, *, /), quindi i programmi possono calcolare solo **funzioni razionali**. Se la funzione che si deve calcolare non è esprimibile come risultato di un numero finito di operazioni aritmetiche si deve **approssimare** il risultato, limitandosi a compiere un numero finito di operazioni. Nell'interrompere il calcolo ad un certo punto della somma si commette un errore (detto di **troncamento** o **analitico**) che di solito viene stimato con le tecniche della Analisi Matematica.

Per esempio il "famigerato" algoritmo per calcolare la radice quadrata che si imparava alle scuole medie inferiori non calcola esattamente la radice quadrata di un numero che non sia un quadrato perfetto, ma solo una sua approssimazione.

Un altro esempio è dato dalla formula di Taylor per la funzione seno:

$$\sin x = x - \frac{x^3}{6} + \frac{x^5}{5!} + \ldots + (-1)^k \frac{x^{2k+1}}{(2k+1)!} + \ldots$$

che può essere facilmente calcolata con *Mathematica*, per esempio con due termini:

In[1]:=
```
s1=Normal[Series[Sin[x],{x,0,3}]]
```

Out[1]=
```
      3
     x
x - --
     6
```

e con cinque

In[2]:=
```
s2=Normal[Series[Sin[x],{x,0,10}]]
```

Out[2]=
```
     3    5     7      9
    x    x     x      x
x - -- + --- - ---- + ------
    6    120   5040   362880
```

La formula di Taylor approssima una funzione periodica (il seno) con dei polinomi, che periodici non sono di certo. È evidente quindi che le funzioni approssimanti possono essere buone solo in intervalli limitati. In **Figura 2** sono disegnate nell'intervallo $[0,2\pi]$, la funzione seno (in nero), la sua approssimazione di Taylor con 2 termini (in verde) e quella con 5 (in rosso).

Errore di arrotondamento

Un *computer* che esegue il programma numerico lavora di soli-

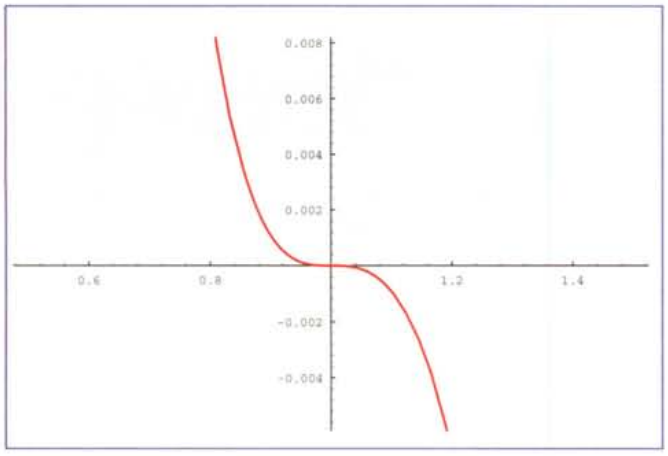

Figura 3

to con un'aritmetica *floating point* a lunghezza finita prefissata (4 o 8 *byte* per un numero reale approssimato) e quindi ogni operazione è affetta da un errore **di arrotondamento** che dipende dalla lunghezza della rappresentazione usata. L'errore di arrotondamento è casuale ma può essere stimato con una certa sicurezza mediante tecniche molto raffinate. È importante notare che l'errore di arrotondamento si propaga durante il calcolo secondo le regole dell'errore inerente, infatti i risultati di un calcolo intermedio sono a tutti gli effetti i dati di ingresso dei calcoli successivi.

Studiamo l'errore di arrotondamento nel calcolo del polinomio $(x-1)^3(x-2)$ che attraversa l'asse delle ascisse nel punto x=1. Nella **Figura 3** è disegnato il grafico nell'intervallo [0.5,1.5]. Ingrandiamo al "microscopio" questa figura intorno al punto x=1, **(Figura 4)**. L'origine è centrata in 1, le ascisse sono amplificate di un fattore 10^5 e le ordinate di un fattore 10^{15}; i 4 fotogrammi ingrandiscono sempre di più la parte centrale mostrando la "grana" della funzioni. Poiché il risultato deve essere un numero di macchina la funzione può assumere solo valori discreti.

Un riassunto dei vari tipi di errore è essere illustrato nella **Figura 5**:

- La linea verde rappresenta la funzione esatta f(x);
- La linea blu rappresenta la funzione razionale g(x) che approssima analiticamente f(x);
- La linea rossa rappresenta la funzione g(x) + l'errore di arrotondamento dovuto all'algoritmo (la funzione è seghettata per indicare che questo errore è casuale);

Le tre funzioni sono state artificiosamente separate per apprezzarne la differenza nella realtà non sono distinguibili ad occhio nudo (almeno se le cose vanno bene).

- Il punto verde in ascissa è il valore x_0 vero e la linea tratteggiata verde conduce al valore corretto $f(x_0)$ in ordinata; la linea tratteggiata blu conduce al valore approssimato $g(x_0)$ in ordinata;
- Il punto rosso in ascissa è il valore x_1 dato in input al programma e la linea tratteggiata celeste conduce al valore $f(x_1)$ in ordinata, la linea tratteggiata rosa conduce al valore $g(x_1)$ in ordinata, la linea tratteggiata rossa conduce al valore effettivamente calcolato dal programma tenendo conto di tutti gli errori.

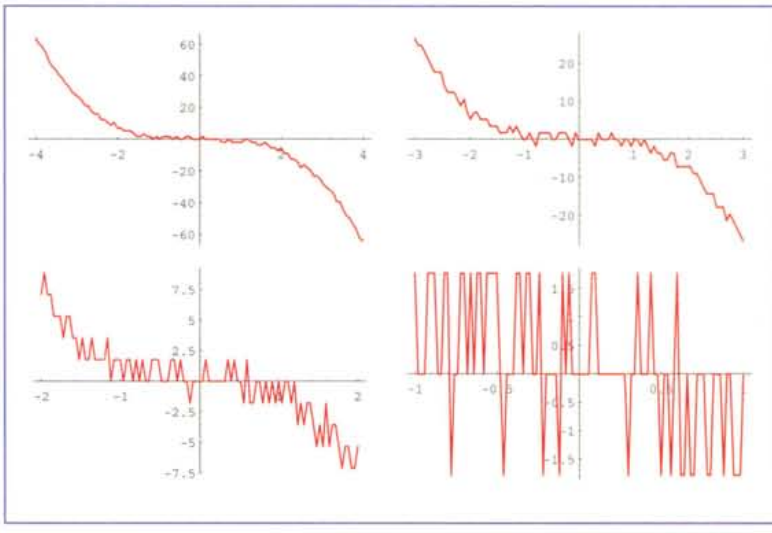

Figura 4

Catastrofi numeriche

Esiste l'ingenua credenza che basti lavorare con una certa precisione (per esempio 10 cifre) per ottenere risultati con quella precisione. Niente di più falso: il numero di cifre esatte del risultato può essere stimato solo dopo un'accurata analisi matematica del problema. Alcune volte i vari tipi di errore possono congiurare per farci ottenere risultati privi di ogni parentela con quanto desiderato. Vediamo le tre possibilità più significative.

Cancellazione

La sottrazione tra due quantità vicine tra loro **cancella** un grande numero di cifre significative e il risultato può essere zero, ottenendo quindi un risultato privo di cifre esatte. Consideriamo l'espressione seguente

In[1]:=
```
f[b_]:=1-Sqrt[1+b^2]
```

che per **b** piccolo ottiene un valore piccolo come differenza di due valori prossimi ad 1. Un calcolo diretto su numero di macchina produce cancellazione totale.

In[2]:=
```
f[10.^-8]
```

Out[2]=
```
0.
```

In[3]:=
```
N[f[10.^-8],30]
```

Out[3]=
```
0.
```

Il risultato giusto si può ottenere lavorando con precisione infini-

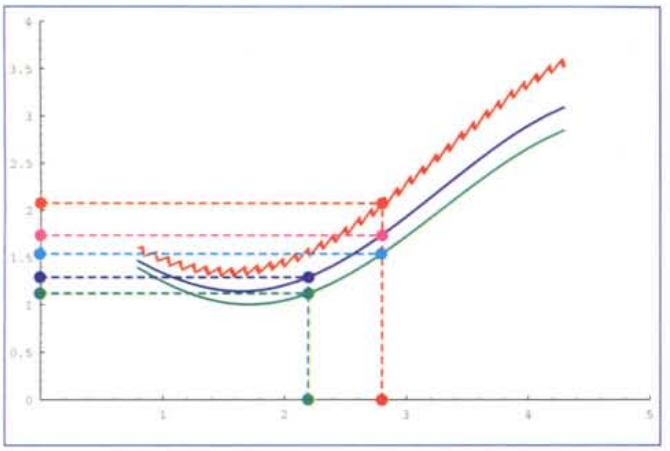

Figura 5

ta e poi chiedendo trenta cifre del risultato.

In[4]:=
```
N[f[10^-8],30]
```

Out[4]=
```
-5. 10^-17
```

La spiegazione è la seguente: il secondo addendo è molto vicino ad 1 e se non c'è spazio viene trattato come 1.

In[5]:=
```
N[Sqrt[1+b^2],40]/.b->10^-8
```

Out[5]=
```
1.0000000000000000499999\
 9999999999875
```

In questi casi è necessario cambiare algoritmo, per esempio sviluppando in serie la funzione da calcolare e valutando la serie in un intorno del punto b=0.

In[6]:=
```
Series[1-Sqrt[1+b^2],{b,0,2}]
```

Out[6]=
```
  2
-b
-- + O[b]^3
 2
```

In[7]:=
```
f1[b_]:=-b^2/2
f1[10.^-8]
```

Out[8]=
```
-5. 10^-17
```

Accumulo degli errori

Sommare tanti numeri piccoli ad un numero grosso può essere perfettamente inutile: si perde tutto! Il programma Pascal:

```
program accumulo;
  var a, b: real;
      i: longint;
begin
  b := 1;
  for i:= 1 to 25 do b:=b/2;
  writeln(b:17);

  a := 1;
  for i := 1 to 10000000 do
    a := a + b;
  a := a - 1;
  writeln(a : 15 : 9);
  a := 0;
  for i := 1 to 10000000 do
    a := a + b;
  writeln(a : 15 : 9);
end.
```

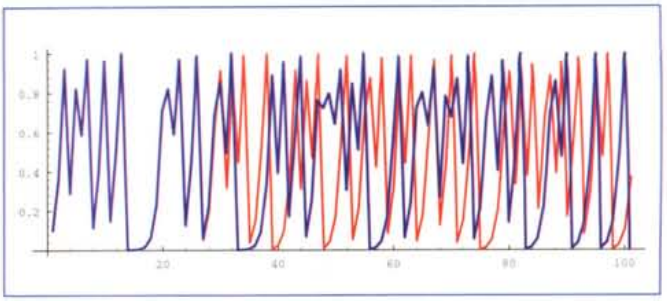

Figura 6

Produce come uscita:
```
2.98023224e-8
0.000000000
0.298023224
```

Questo esperimento non sono riuscito a farlo in *Mathematica* (probabilmente vi sono controlli speciali proprio per evitare questo fenomeno).

Problemi stabili e instabili

In alcuni casi una piccolissima variazione dei dati di ingresso produce risultati completamente differenti (**instabilità del problema**). Il seguente esempio è tratto dal libro di Wagon *Mathematica in Action*. Consideriamo la funzione `f[x_]:=4.x(1-x)` e applichiamola 100 volte a se stessa partendo da 0 (linea blu) e da 10^{-10} (linea rossa). Nonostante l'estrema vicinanza dei due punti di partenza dopo poco le due liste di valori sono totalmente differenti (**Figura 6**).

Bibliografia

R. Bevilacqua, D. Bini, M. Capovani, O. Menchi. **Introduzione alla Matematica Computazionale.** (Zanichelli, 1987)

R. Bevilacqua, D. Bini, M. Capovani, O. Menchi. **Metodi Numerici.** (Zanichelli, 1992)

R. D. Skeel, J. Keiper, **Elementary Numerical computing with *Mathematica*.** (McGraw-Hill International Editions, 1993).

S. Wolfram, **The *Mathematica* Book, 3rd ed.** (Cambridge University Press, 1996)

S. Wagon, ***Mathematica* in Action.** (W. H. Freeman, 1991),

hardware

UNIBIT

The Telematic & Multimedial Planet

usare la testa
molto spesso
NON BASTA

così come è
FONDAMENTALE
SCEGLIERE
strumenti adeguati

CA&G ELETTRONICA

HEWLETT PACKARD

Canon

FAST

JVC PROFESSIONAL

TRAXDATA

matrox

UMAX

LOGITECH

KOSS

MOTOROLA

US Robotics

SONY

CREATIVE

SHARP

Microsoft Windows 95

PER INFORMAZIONI SUL PROGRAMMA DI AFFILIAZIONE: Numero Verde 167-018116

Pelikan Hardcopy

• **TRENTO** - *Gruppo per l'Informatica s.r.l. (0461) 934.611* • **VIGO DI FASSA** (TN) - *Fassa Computer (0462) 763.744* • **BADIA POLESINE** (RO) - *Hakitval s.a.s (0425) 51.136* •
• **BASSANO DEL GRAPPA** (VI) - *Eurosoft (0424) 522.810* • **ALTAVILLA** (VI) - *Progetto CAD (0444) 574.799* • **CORNEDO VIC.** (VI) - *Unibit Planet (0444) 446.501* •
• **BRESSANVIDO** (VI) - *Soluzioni Inform. (0444) 660.950* • **SCHIO** (VI) - *Pitagora s.r.l. (0445) 576.223* • **DUEVILLE** (VI) - *Tuttoufficio Cortese (0444) 750.170* •
• **THIENE** (VI) - *Genero Anna (0445) 380.433* • **PADOVA** - *C.R. Elettronica (049) 601.066* • **PEDEMONTE** (VR) - *Service (045) 680.10.56* • **TREVISO** - *Computerware (0422) 422.422* •
• **NOALE** (VE) - *Computer House s.a.s. (041) 442.968* • **VENEZIA** - *K551 Jupiter (041) 523.80.59* • **CERRO MAGGIORE** (MI) - *Master Bit Line (0331) 421.360* •
• **CARAVAGGIO** (BG) - *NTM Computers (0363) 350.610* • **FOSSANO** (CN) - *System Service (0872) 635.365* • **UDINE** - *Eurojapan s.r.l. (0432) 479.884* • **TRIESTE** - *T.H.E. 90 (040) 824.974* •
• **SALUZZO** (CN) - *Expo Computer (0175) 43.443* • **PARMA** - *Meccanografica (0521) 994.250* • **CHIAVARI** (GE) - *Computer Service (0185) 323.213* • **LA SPEZIA** - *Copitecnica (0187) 509.566* •
• **RAPALLO** (GE) - *Mario Bottazzi s.r.l. (0185) 50.185* • **BORDIGHERA** (IM) - *Full Stop (0184) 264.353* • **MADONNA DELL'ACQUA** (PI) - *Eurotec Pisa (050) 890.839* •
• **S. BENEDETTO DEL TRONTO** (AP) - *MAEN Computer Service (0735) 751.295* • **PESCARA** - *Il Pianeta del Computer (085) 692.349* • **ARPINO** (FR) - *Sisteminformatici (0776) 84.219* •
• **PUTIGNANO** (BA) - *Lonuzzo Domenico (080) 491.19.33* • **FOGGIA** - *S.I.M. (0881) 720.475* • **CAMPOBASSO** - *Ecom System (0874) 411.330* • **COSENZA** - *Hard & Soft (0984) 413.450* •
• **SCALEA** (CS) - *General Office (0985) 90.069* • **POGGIOMARINO** (NA) - *R.B.F. (081) 528.59.63* • **ALCAMO** (TP) - *Coelda Info (0924) 507.497* •
• **ROSOLINI** (SR) - *Tecnosystem (0931) 502.110* • **BAGHERIA** (PA) - *C.S. di Corrao Antonino (091) 963.970* • **QUARTU S. ELENA** (CA) - *3Pi Informatica (070) 826.892* •

CONCEPT BY CAMPIONE OF PROFESSIONAL ADVERTISING PHOTOS BY URNATO