Università del Salento
Dipartimento di Matematica *"E. De Giorgi"*

Raffaele Vitolo

INTRODUZIONE A MAXIMA

Versione 1.0 (9 febbraio 2007)

ANNO ACCADEMICO 2006-2007

Indirizzo dell'autore.
Raffaele Vitolo,
Università di Lecce, Dipartimento di Matematica,
via per Arnesano, 73100 Lecce
email: `raffaele.vitolo@unile.it`
web: `http://poincare.unile.it/vitolo/`

Indice

1 Introduzione

Da un punto di vista metodologico, è conveniente imparare un linguaggio di programmazione *usandolo*, piuttosto che studiando il manuale. Quindi, in questa lezione il linguaggio di *Maxima* sarà introdotto tramite una serie di esempi.

NOTA. Il testo si riferisce alla versione 5.10 di *Maxima*.

2 Descrizione del programma

Maxima è un programma che può funzionare con varie modalità:

1. è utilizzabile come terminale che interpreta comandi in modo interattivo;

2. è utilizzabile come un vero e proprio linguaggio di programmazione (interpretato, non compilato come il `c`);

3. ha un'interfaccia grafica semplice dal terminale, di nome `xmaxima`;

4. ha un'interfaccia grafica più evoluta, `wxmaxima`, che facilita l'inserimento dei comandi dal terminale anche ai non esperti.

Il testo che segue si riferisce all'utilizzo interattivo di *Maxima*. Le istruzioni descritte nel testo possono essere inserite dal terminale in una modalità qualsiasi (grafica o no), questo non cambia il comportamento del programma. Le differenze tra le varie interfacce saranno indicate di volta in volta.

3 Installazione

Per Windows: bisogna scaricare ed installare la versione eseguibile per Windows (quella con l'estensione `.exe`), che si trova nella sezione 'Downloads' del sito web di *Maxima* `http://maxima.sourceforge.net/`. Inoltre, se si desidera fare grafici, è necessario il programma di disegno `gnuplot`, scaricabile in versione eseguibile per Windows (quella con l'estensione `.exe`), che si trova nella sezione 'Downloads' del sito web `http://www.gnuplot.info/`.

Per Linux: Tutte le maggiori distribuzioni hanno i pacchetti necessari. Ad esempio, per le distribuzioni basate su Debian (pacchetti `.deb`) si installino i pacchetti `maxima, maxima-doc, maxima-emacs, maxima-share, maxima-src, maxima-test, xmaxima, wxmaxima`, insieme a `gnuplot, gnuplot-doc, gnuplot-mode, gnuplot-nox, gnuplot-x11` per i disegni. Per le distribuzioni basate su RedHat (pacchetti `.rpm`), il discorso è analogo.

Si possono usare anche *live-CD* dedicati al calcolo scientifico (come Quantian) o all'educazione (come EduKnoppix, `http://www.eduknoppix.org/`.

4 Primi passi

1. *Come funziona il programma?* Ci sono due modalità: interattiva e di interpretazione di una lista di comandi (programma, o *file batch*).

   Nella modalità interattiva si scrivono i comandi sulle righe che cominciano per `(%i1), (%i2), ...` o simili. Queste sono le righe di *input*, e sono numerate progressivamente. Alla fine di una riga di *input*, si preme il tasto '`Invio`', e *Maxima* interpreta i comandi producendo, se è il caso, una riga di *output*, indicata con `(%o1), (%o2), ...` numerata progressivamente[1].

2. *Come si scrive un comando?* Bisogna sempre terminare i comandi, o le combinazioni di comandi, con un ';'. Se si omette il ';', il programma entra in condizione di errore, dalla quale si può uscire scegliendo la voce di menu '*Interrupt*' (anche attivabile con la sequenza di tasti '`ctrl-g`'). *Si noti che `wxmaxima` inserisce automaticamente il ';'.*

   Si può anche terminare un comando con '`$`', che però non mostra l'eventuale *output* del comando.

3. *Come si ottiene aiuto su di un comando?* Si usa la funzione `describe`. Ad esempio, per chiedere aiuto sulla funzione `sin` si ha:

```
(%i1) describe(sin);

 0: (maxima.info)Introduction to string processing.
 1: absint :Definitions for Fourier Series.
 2: asin :Definitions for Trigonometric.
 3: asinh :Definitions for Trigonometric.
 4: fourintsin :Definitions for Fourier Series.
 5: foursin :Definitions for Fourier Series.
 6: maxpsinegint :Definitions for Special Functions.
 7: maxpsiposint :Definitions for Special Functions.
 8: poisint :Definitions for Special Functions.
 9: sin :Definitions for Trigonometric.
 10: sinh :Definitions for Trigonometric.
 11: sinnpiflag :Definitions for Fourier Series.
 12: sinsert :Definitions for strings.
 13: sinvertcase :Definitions for strings.
Enter space-separated numbers, 'all' or 'none':
```

   inserendo il numero corrispondente alla voce richiesta (seguito da ';'), ad esempio `3;` si ottiene la documentazione richiesta.

   Inoltre, il comando `example` fornisce esempi di utilizzo, con la stessa sintassi di `describe`. Tutte queste funzioni sono accessibili dal menu `Help` di `wxmaxima`.

4. *Come si richiama un comando già eseguito?* Il comando precedente si richiama digitando '`%;`' il comando numerato con '`n`' si richiama digitando '`%in;`'.

[1] Si noti che, fino alla versione 5.9.0, la sequenza di inizio riga di *input/output* era indicata da `(C1), (C2), ..., (D1), (D2), ...`, rispettivamente.

5. *Come si usano le costanti standard?* Sono variabili predefinite, e sono '`%e`', '`%i`', '`%pi`'.

5 Radici di polinomi

Si introduca il polinomio

$$\frac{x^4}{2}+\frac{3x^3}{4}-\frac{5x^2}{4}-\frac{7x}{4}-\frac{1}{2}$$

con lo scopo di trovare le sue radici. La sintassi è la seguente:

```
(%i1) f(x) := x^4/2+3*x^3/4-5*x^2/4-7*x/4-1/2;
```

```
                             4      3           2
                            x    3 x    (- 5) x    (- 7) x    - 1
(%o1)               f(x) := -- + ---- + -------- + ------- + ---
                            2     4        4          4        2
```

in questo modo il programma definisce la funzione $f(x)$. Se, ad esempio, si vuole calcolare un valore specifico di f, ad esempio il valore in $x = 3$

```
(%i2) f(3);
```

```
                                        175
(%o2)                                   ---
                                         4
```

L'operazione funziona anche sostituendo un'espressione algebrica: si provi con

```
(%i3) f(y+3);
```

```
                     4            3            2
              (y + 3)    3 (y + 3)    5 (y + 3)    7 (y + 3)    1
(%o3)         -------- + ---------- - ---------- - --------- - -
                 2           4            4            4        2
```

Se si vogliono calcolare le radici del polinomio, ovvero gli zeri di $f(x)$, si scriva

```
(%i4) f(x)=0;
```

```
                              4      3      2
                             x    3 x    5 x    7 x   1
(%o4)                        -- + ---- - ---- - --- - - = 0
                             2     4      4      4    2
```

questo definisce un'equazione. L'equazione può essere risolta con l'istruzione `solve`, in questo modo

```
(%i5) solve(f(x)=0,x);
```

```
                     1                     sqrt(5) - 1      sqrt(5) + 1
(%o5)        [x = - -, x = - 2, x = - -----------, x = -----------]
                     2                          2                2
```

il secondo argomento è la variabile rispetto alla quale si risolve l'equazione $f(x) = 0$. Si ottengono 4 radici. Si possono convertire le radici in numeri usando le istruzioni `float` o `bfloat`:

```
(%i6) float(%o6);
```

```
(%o6)                                       %o6
```

`bfloat` permette di fare calcoli con precisione arbitraria, controllata dalla variabile `fpprec`.

```
(%i7) fpprec;
```

```
(%o7)                                       16
```

Aumentando il valore di `fpprec` si può ottenere il valore di `%pi` con il numero di cifre desiderato:

```
(%i8) fpprec : 100;
```

```
(%o8)                                   100
(%i9) bfloat(%pi);
```

```
(%o9) 3.14159265358979323846264338327950288419716939937510582097494459230781
06286208998628034825342117068B0
```

Ora, per calcolare graficamente gli zeri della funzione $f(x)$ si può usare l'istruzione seguente:

```
(%i10) plot2d(f(x),[x,-5,5]);
```

```
(%o10)
```

Il risultato è la figura 1: Il grafico può essere anche salvato in un *file* di tipo PostScript:

```
(%i11) plot2d(f(x),[x,-5,5], [gnuplot_term,ps], [gnuplot_out_file, "polin.e
```

```
Output file "polin.eps".
(%o11)
```

Le opzioni del programma di disegno fornito a corredo con *Maxima*, che si chiama `GNUplot` [2], forniscono molti effetti grafici. Ad esempio, la seguente istruzione ridisegna il grafico con l'asse x:

```
(%i12) plot2d(f(x),[x,-5,5],[gnuplot_preamble,"set key off;set xzeroaxis;
```

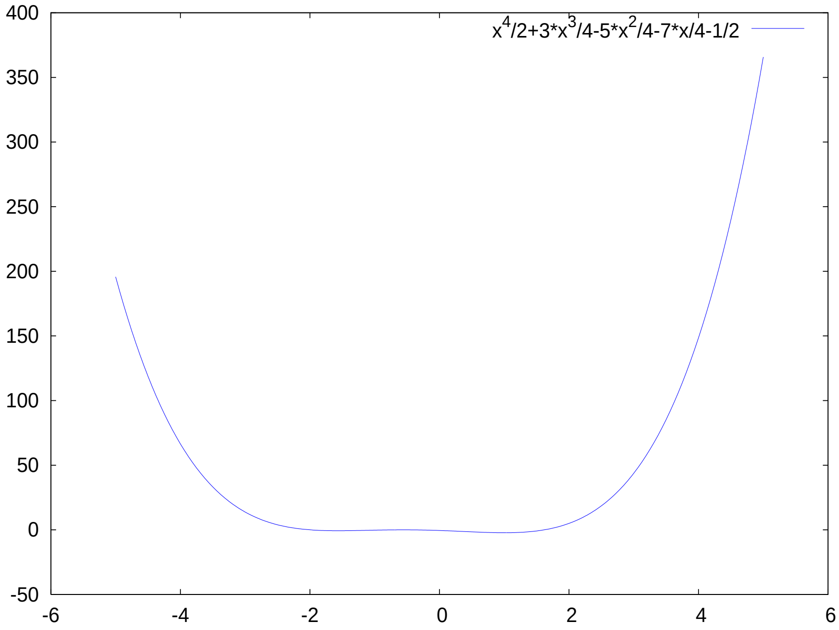

Figura 1: Grafico di $f(x)$

```
set yzeroaxis"]);
```

```
(%o13)
```

L'opzione di `gnuplot` '`set key off`' elimina la didascalia (come quella inclusa nella figura 1). Il risultato è la figura 2: Per individuare meglio gli zeri sul grafico, si ricorre ad un ingrandimento cambiando la scala:

```
(%i13) plot2d(f(x),[x,-2,0],[y,-2,2],[gnuplot_preamble,"set key off;set
xzeroaxis;set yzeroaxis"]);
```

```
(%o13)
```

con il risultato della figura 3.

Esercizi. Risolvere esplicitamente e graficamente le seguenti equazioni, ad esempio con il metodo di bisezione:

$$x^4 + 2x^3 + x^2 - 6x - 42 = 0, \qquad 4x^4 - 25x^2 + 20x - 4 = 0.$$

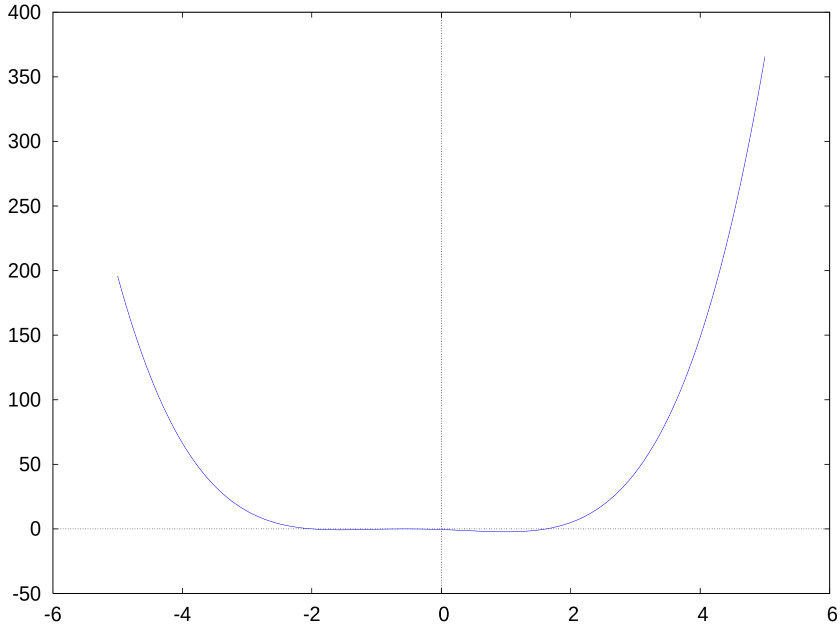

Figura 2: Grafico di $f(x)$

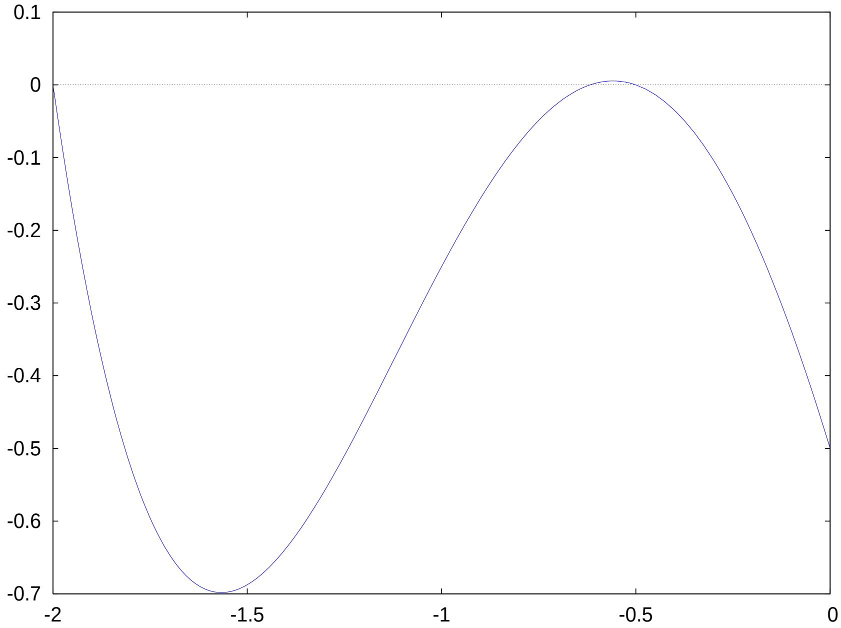

Figura 3: Grafico di $f(x)$

6 Raccoglimento e sviluppo

Si assegni alla variabile ‘`pol`’ il polinomio

$$\frac{a + (a - b)}{x} + \frac{a - b}{x} + \frac{(b - 1)^2}{x^2 - y^2}$$

```
(%i1) pol : (a + (a - b))/x + (b - 1)^2/(x^2 - y^2);
```

```
                                     2
                              (b - 1)    2 a - b
(%o1)                         -------- + -------
                               2    2       x
                              x  - y
```

Si può espandere il polinomio:

```
(%i2) expand(pol);
```

```
                 2
                b         2 b         1       b   2 a
(%o2)        ------- - ------- + ------- - - + ---
              2    2    2    2    2    2   x    x
             x  - y    x  - y    x  - y
```

I criteri con i quali funziona `expand` si possono cercare nella documentazione, nel paragrafo ‘Polynomials’. L’idea generale è quella di avere un numero massimo di addendi.

Si possono raccogliere fattori specifici nel polinomio:

```
(%i3) expandwrt(pol,a);
```

```
Warning - you are redefining the Maxima function intersection
                                     2
```

```
(%o3)                                  (b - 1)     b   2 a
                                       -------- - - + ---
                                        2    2     x     x
                                       x   - y
```

La funzione `factor` raccoglie a fattor comune (rispetto agli interi) i fattori in cui può essere scomposto il polinomio.

```
(%i4) factor(pol);
```

```
                 2         2       2        2    2
              b y  - 2 a y  - b x  + 2 a x  + b  x - 2 b x + x
(%o4)       - ---------------------------------------------------
                                x (y - x) (y + x)
```

Tuttavia, per operare su funzioni razionali si utilizzano i comandi `ratexpand` e `ratsimp`, che permettono di espandere in una somma di frazioni semplificate o raccogliere sotto un unico denominatore.

```
(%i5) ratexpand(pol);
```

```
           2           2                            2
        b y       2 a y       b x     2 a x      b
(%o5) - --------- + --------- + ------- - ------- - ------- +
          2    3     2    3    2    2    2    2    2    2
        x y  - x   x y  - x   y  - x   y  - x   y  - x

                                   2 b       1
                                + ------- - -------
                                   2    2    2    2
                                  y  - x    y  - x
(%i6) ratsimp(pol);
```

```
                          2               2     2
              (b - 2 a) y  + (2 a - b) x  + (b  - 2 b + 1) x
(%o6)       - ------------------------------------------------
                                   2    3
                                x y  - x
```

Se si vuole sostituire un'espressione algebrica ad una lettera del polinomio si usi il seguente comando:

```
(%i7) %o2, x=5/z;

                                                     2
                              (2 a - b) z   (b - 1)
(%o7)                         ----------- + --------
                                   5         25    2
                                             -- - y
                                              2
                                             z
```

Si consideri ora il polinomio

$$x^6 - 3x^4 - 8x^3 - 18x^2 + 12x + 16.$$

Usando l'istruzione `factor` si nota che il risultato è

$$(x - 1)(x + 2)(x - 2x - 2)^2(x + x + 4)^2.$$

Questo perché il programma fattorizza usando gli interi (o i razionali, il che è sostanzialmente equivalente). Tutti i principali programmi di calcolo simbolico hanno questo comportamento. Per fattorizzare usando un'estensione del campo dei razionali si usi l'istruzione `factor(polinomio,pmin)`, dove `pmin` è il polinomio minimo dell'elemento che estende il campo dei razionali. Ad esempio:

```
(%i8) poli :   x^4  - 4*x^3  + x^2  + 8*x - 6;

                             4      3    2
(%o8)                       x  - 4 x  + x  + 8 x - 6
(%i9) factor(poli);

                                                 2
(%o9)                       (x - 3) (x - 1) (x  - 2)
(%i10) factor(poli,A^2-2);
```

```
(%o10)                    (x - 3) (x - 1) (x - A) (A + x)
```

dove, ovviamente, `A` è l'elemento $\sqrt{2}$ che espande il campo dei razionali.

Analogamente si possono raccogliere e/o sviluppare espressioni contenenti funzioni trigonometriche:

```
(%i11) expr : sin(u+v)*cos(u)^3;

                                      3
(%o11)                             cos (u) sin(v + u)
(%i12) trigexpand(expr);

                              3
(%o12)                     cos (u) (cos(u) sin(v) + sin(u) cos(v))
(%i13) trigreduce(expr);

              sin(v + 4 u) + sin(v - 2 u)   3 sin(v + 2 u) + 3 sin(v)
(%o13)        --------------------------- + -------------------------
                           8                            8
```

Esercizi.

1. Semplificare l'espressione polinomiale

$$(1 + b/a + b^2/a^2)(a/(a^3 - b^3)).$$

2. Fattorizzare i polinomi al primo membro delle seguenti equazioni:

$$x^4 + 2x^3 + x^2 - 6x - 42 \; = 0, \qquad 4x^4 - 25x^2 + 20x - 4 = 0$$

risolvere esplicitamente e graficamente.

7 Sistemi di equazioni

Per dare un sistema di equazioni lineari, il formato è quello di un vettore di equazioni:

```
(%i1) sist : [x-2*y=3,x+y=1];
```

```
(%o1)                      [x - 2 y = 3, y + x = 1]
```

al quale si può applicare il comando `solve`:

```
(%i2) solve(sist,[x,y]);
```

```
                                     5        2
(%o2)                          [[x = -, y = - -]]
                                     3        3
```

anche se l'istruzione `linsolve` è quella ottimizzata per la soluzione dei sistemi lineari.

Si può visualizzare la soluzione graficamente, disegnando le due rette corrispondenti alle due equazioni lineari:

```
(%i3) plot2d([(x-3)/2,x-1],[x,-5,5],[gnuplot_preamble,"set xzeroaxis;
set yzeroaxis"]);
```

```
(%o3)
```

con il risultato della figura 4.

Esercizi. Risolvere esplicitamente e graficamente i seguenti sistemi di equazioni:

$$\begin{cases} x + 2y = 8 \\ x - y = -1 \end{cases} \qquad \begin{cases} xy = \sqrt{2} \\ x^2 + y^2 = 4 \end{cases}$$

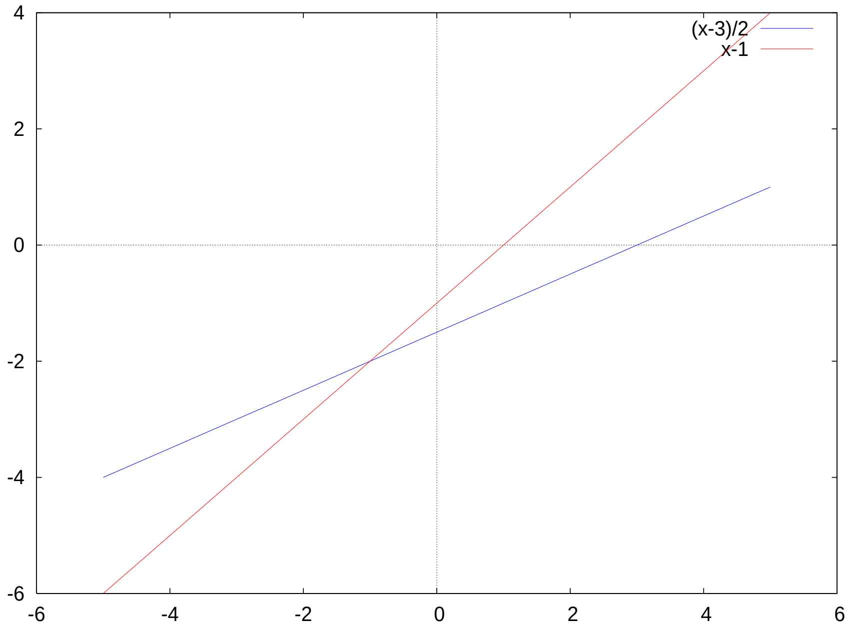

Figura 4: Soluzione grafica di un sistema

8 Analisi con Maxima

8.1 Regole di derivazione

Maxima può essere utile per ripassare le regole di derivazione. Si ponga

```
(%i6) depends(a,x);

(%o6)                                   [a(x)]

(%i7) depends(b,x);

(%o7)                                   [b(x)]
```

allora la regola del quoziente si ottiene come

```
(%i8) quoziente:diff(a/b,x);
```

```
                                    da     db
                                    --   a --
                                    dx     dx
(%o8)                               -- - ----
                                    b      2
                                          b
```

La regola deve essere semplificata per ottenere la forma più usuale:

```
(%i9) ratsimp(quoziente);
```

```
                                      db   da
                                    a -- - -- b
                                      dx   dx
(%o9)                             - -----------
                                         2
                                        b
```

9 Studio di funzioni

9.1 Funzioni elementari

Si inserisca una funzione in *Maxima*

```
(%i1) f(x) := ((x-1)/(x-2))*(%e^(x - 1));
```

```
                                      x - 1   x - 1
(%o1)                        f(x) := ----- %e
                                      x - 2
```

I limiti per $x \to +\infty$ e per $x \to -\infty$ sono

```
(%i2) limit(f(x),x,inf);

(%o2)                                inf
(%i3) limit(f(x),x,minf);

(%o3)                                 0
```

Analogamente si possono calcolare i limiti per valori finiti, usando `plus` e `minus` come ultimo argomento quando si vuole il limite destro e sinistro:

```
(%i4) limit(f(x),x,2,plus);

(%o4)                                inf
```

Si può controllare la presenza di un asintoto obliquo calcolando il limite di $f(x)/x$.

Per comprendere meglio l'andamento della funzione può essere necessario ingrandire parti di grafico:

```
(%i5) plot2d(f(x),[x,-5,1.5],[gnuplot_preamble,
"set xzeroaxis; set yzeroaxis"]);

(%o5)
(%i6) plot2d(f(x),[x,2.1,4],[gnuplot_preamble,
"set xzeroaxis; set yzeroaxis"]);

(%o6)
```

con i risultati della figura 5.

Si calcoli la derivata prima:

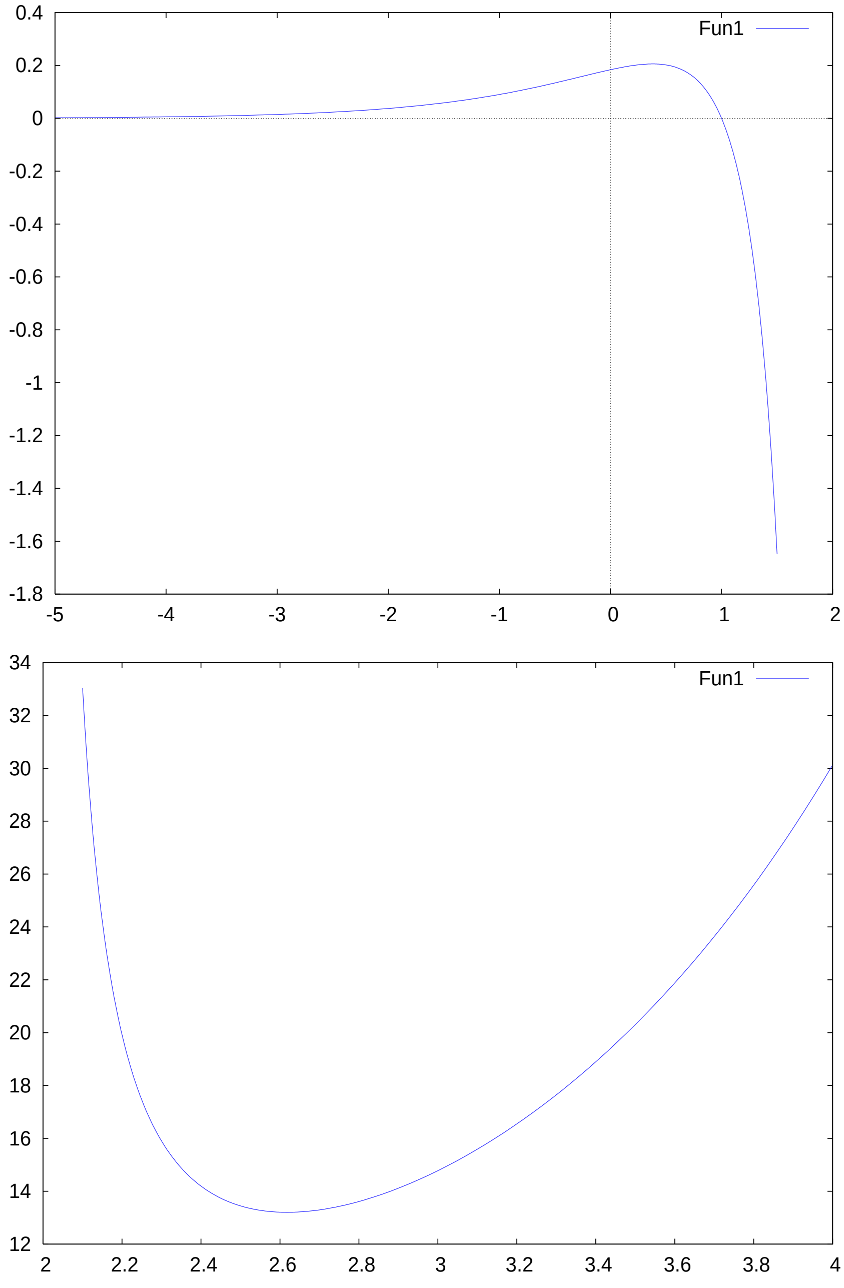

Figura 5: Grafico della funzione

```
(%i7) diff(f(x),x);
```

```
                              x - 1              x - 1     x - 1
                    (x - 1) %e         (x - 1) %e        %e
(%o7)               --------------- - --------------- + -------
                         x - 2                  2        x - 2
                                          (x - 2)
```

purtroppo il risultato non è in forma semplificata, si deve procedere con un'ulteriore comando:

```
(%i8) factor(%);
```

```
                                  2              x - 1
                              (x   - 3 x + 1) %e
(%o8)                         ----------------------
                                            2
                                     (x - 2)
```

Usando `solve` si trovano le radici della derivata prima, che sono due (come era da aspettarsi).

```
(%i9) solve(diff(f(x),x)=0,x);
```

```
                      sqrt(5) - 3      sqrt(5) + 3    x - 1
(%o9)           [x = - -----------, x = -----------, %e      = 0]
                           2                2
```

Iterando il procedimento si trova la derivata seconda, con le sue radici.

È molto interessante disegnare il grafico di una funzione sovrapposto al grafico della sua derivata:

```
(%i10) plot2d([f(x),diff(f(x),x)],[x,-2,1.5],[y,-1,1],
```

```
[gnuplot_preamble,"set xzeroaxis; set yzeroaxis"]);
```

```
(%o10)
```

con i risultati della figura 6.

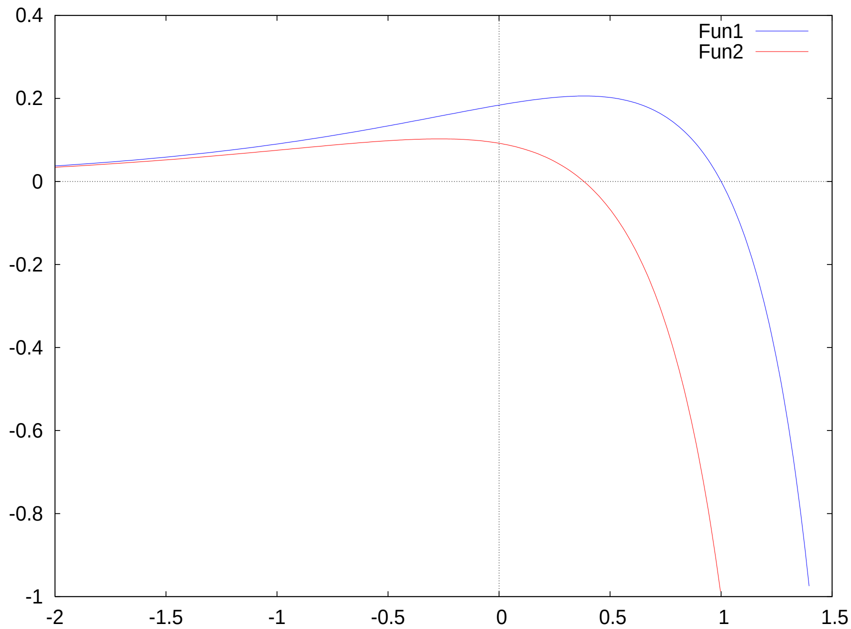

Figura 6: Grafico della funzione

Esercizi.

1. Studiare la funzione $(4 + \log x)x \log^3 x$;
2. Studiare la funzione $e^{\sin x}|\sin x|$
3. Disegnare la tangente nel punto di flesso della funzione studiata nel paragrafo.

9.2 Funzioni definite "a blocchi"

Si possono definire e studiare funzioni assegnate mediante una lista di espressioni, ognuna valida in un certo intervallo, in questo modo. Si consideri, ad esempio, la funzione

$$f(x) := \begin{cases} \frac{1+\sqrt[3]{x}}{2-\sqrt[3]{x}} & x \in (-\infty, 0], \\ e^{-x}\sqrt[3]{(e^x - 1)} & x \in [0, +\infty). \end{cases}$$

Questa si può assegnare in Maxima come

```
(%i1) f1(x):=(1+(x)^(1/3))/(2-(x)^(1/3));
```

```
                                                     1/3
                                                1 + x
(%o1)                               f1(x) := --------
                                                     1/3
                                                2 - x
```

```
(%i2) f2(x):=(%e)^(-x)*((%e)^x-1)^(1/3);
```

```
                                         - x     x      1/3
(%o2)                       f2(x) := %e     (%e  - 1)
```

```
(%i3) f(x):=block([],if (x<=0) then return(f1(x)) else return(f2(x)));
```

```
(%o3) f(x) := block([], if x <= 0 then return(f1(x)) else return(f2(x)))
```

Il grafico della funzione così definita è ottenuto con

```
(%i4) plot2d(f,[x,-5,5]);
```

```
(%o4)
```

con i risultati della figura 7. Ovviamente qui è interessante studiare i limiti sinistro e destro nei punti in cui le funzioni cambiano.

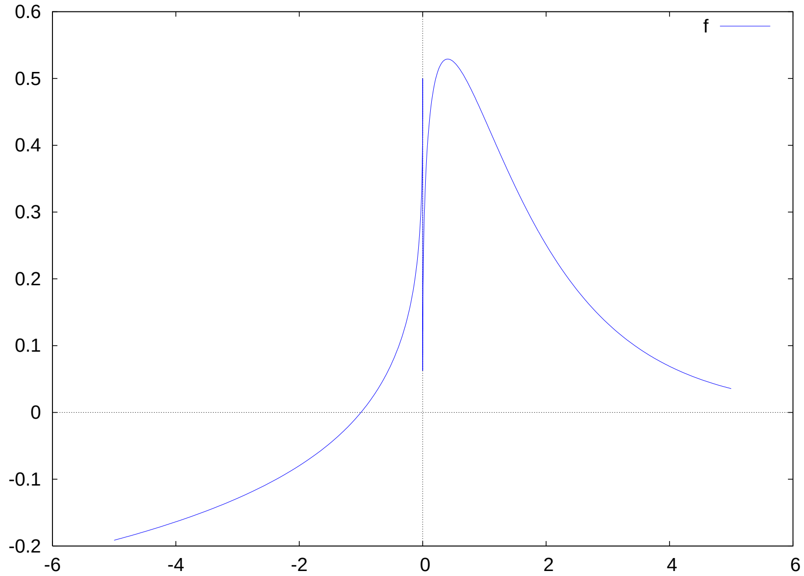

Figura 7: Funzione definita "a blocchi"

Esercizio. Studiare la funzione definita da

$$f(x) := \begin{cases} x & x \in (-\infty, 0], \\ x^3 & x \in [0, 1], \\ \sqrt[3]{x} & x \in [1, +\infty). \end{cases}$$

9.3 Funzioni composte

Siano f e g due funzioni per le quali abbiano senso le composizioni $f \circ g$ e $g \circ f$. Allora *Maxima* è in grado di calcolare le espressioni corrispondenti alle due composizioni.

```
(%i1) f(x):= x^3-1;
```

```
                                    3
(%o1)                     f(x) := x  - 1
```

```
(%i2) g(y):=2*sin(y);
```

```
(%o2)                          g(y) := 2 sin(y)
```

```
(%i3) comp1(x):=g(f(x));
```

```
(%o3)                         comp1(x) := g(f(x))
```

```
(%i4) comp2(x):=f(g(y));
```

```
(%o4)                         comp2(x) := f(g(y))
```

```
(%i5) [comp1(x),comp2(y)];
```

```
                                   3             3
(%o5)                    [2 sin(x  - 1), 8 sin (y) - 1]
```

È interessante confrontare i grafici delle composizioni di due funzioni in ordine inverso.

Si può inoltre verificare la regola della derivata della funzione composta.

```
(%i6) depends(y,x);
```

```
(%o6)                               [y(x)]
```

```
(%i7) diff(g(y),x);
```

```
                                       dy
(%o7)                          2 cos(y) --
                                       dx
```

Esercizio. Si considerino le funzioni

$$f(x) = x^2 - x + 1, \qquad g(x) = log(x);$$

si traccino i grafici di $f \circ g$ e $g \circ f$.

10 Integrazione

L'istruzione a cui fare riferimento è `integrate`. Può essere usata con limiti di integrazione (integrale definito) o senza (integrale indefinito). *Maxima* non integra funzioni arbitrarie come $f(x)$.

```
(%i1) integrate(log(x),x);


(%o1)                         x log(x) - x



(%i2) integrate(log(x),x,1,2);


(%o2)                         2 log(2) - 1
```

È possibile integrare funzioni contenenti parametri:

```
(%i3) integrate (x/ sqrt (b^2 - x^2), x);

                                      2    2
(%o3)                        - sqrt(b  - x )
```

Si può usare il comando `integrate` per ripassare la formula di integrazione di un polinomio:

```
(%i4) integrate(a[3]*x^3+a[2]*x^2+a[1]*x+a[0],x);

                                  4       3       2
                               a  x    a  x    a  x
                                3       2       1
(%o4)                          ----- + ----- + ----- + a  x
                                 4       3       2      0
```

Si noti come è stato introdotto il vettore dei coefficienti del polinomio.

```
(%i5) integrate(x/(x^2+x+1),x,0,2);

                                   5 sqrt(3)
                    sqrt(3) atan(---------)
                                       3       log(7)   sqrt(3) %pi
(%o5)             - ----------------------- + ------ + -----------
                                3                 2         18
```

e se si vuole un'approssimazione numerica:

```
(%i6) %,numer;


(%o6)                            0.5608861122167
```

A volte *Maxima* chiede informazioni sui parametri introdotti:

```
(%i7) integrate(asin(x),x,0,u);


Is  u  positive, negative, or zero?

pos;
                                                 2
(%o7)                          u asin(u) + sqrt(1 - u ) - 1
```

A volte, l'integrando semplicemente non si integra. . .

```
(%i8) expand ((x-4) * (x^3+2*x+1));

                          4      3      2
(%o8)                    x  - 4 x  + 2 x  - 7 x - 4


(%i9) integrate(1/%,x);

                                     /  2
                                     [ x  + 4 x + 18
                                     I ------------- dx
                                     ]  3
                        log(x - 4)   / x  + 2 x + 1
(%o9)                   ---------- - ------------------
                            73               73
```

L'istruzione `grind` mostra il risultato con la sintassi di *Maxima*:

```
(%i10) grind(%);


log(x-4)/73-('integrate((x^2+4*x+18)/(x^3+2*x+1),x))/73
(%o10)                               done
```

Esercizi. Calcolare i seguenti integrali.

- $\int \log^2 x \, dx$;
- $\int_{-1}^{1} \frac{2x^2-1}{x^3-2x^2+x-2} dx$;
- $\int \frac{x+1}{x-1} dx$ (suggerimento: si usi la funzione `logcontract` sul risultato). Si provi ad integrare la stessa funzione su diversi intervalli.
- $\int \cos^q x \, dx$.

11 Grafica 3D

La grafica in *Maxima* è gestita dal programma `GNUplot`. Dunque, i comandi eseguiti dal terminale di *Maxima* sono poi passati a `GNUplot` che disegna la figura richiesta e restituisce il controllo a *Maxima*. La documentazione di `GNUplot` è reperibile in rete [2]; qui saranno indicati alcuni esempi specifici per la grafica 3D.

```
(%i1) plot3D(sin(sqrt(x^2+y^2)),[x,-2*%pi,2*%pi],[y,-2*%pi,2*%pi]);
```

```
(%o1)
```

con i risultati della figura 8. Volendo eliminare la sovrapposizione con le li-

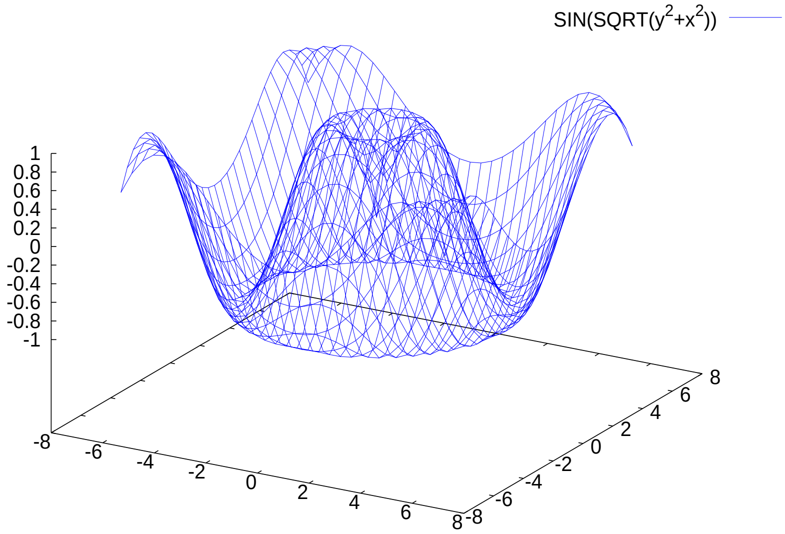

Figura 8: Grafico di $\sin\sqrt{x^2+y^2}$

nee nascoste, bisogna passare a `GNUplot` il parametro `set hidden3D`, come segue (figura 9):

```
(%i2) plot3D(sin(sqrt(x^2+y^2)),[x,-2*%pi,2*%pi],[y,-2*%pi,2*%pi],
[gnuplot_preamble,"set hidden3d"]);
```

(%o2)

Inoltre, è possibile disegnare il grafico come griglia di poligoni, in questo mo-

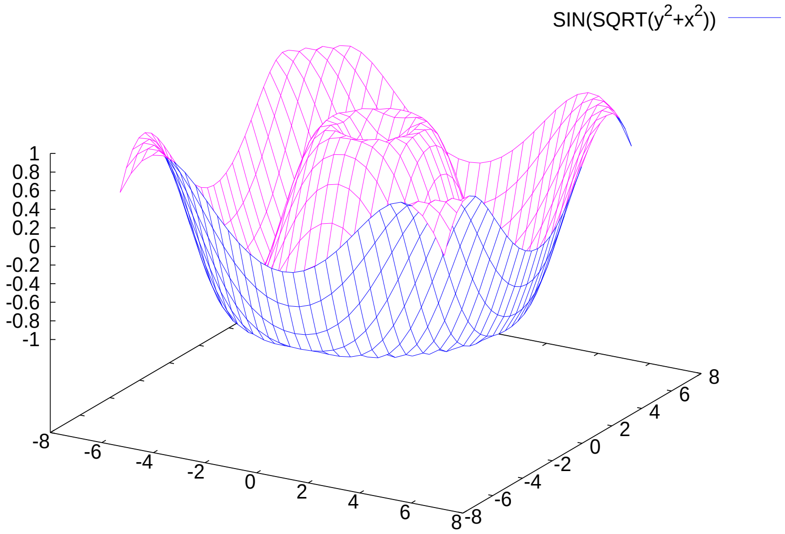

Figura 9: Grafico di sin $\sqrt{x^2 + y^2}$, rimozione delle linee nascoste

do:

```
(%i3) plot3D(sin(sqrt(x^2+y^2)),[x,-2*%pi,2*%pi],[y,-2*%pi,2*%pi],
[gnuplot_preamble,"set pm3d;set hidden3d"]);
```

(%o3)

con il risultato nella figura 10. È possibile aumentare il numero di punti della griglia in questo modo:

```
(%i4) plot3D(sin(sqrt(x^2+y^2)),[x,-2*%pi,2*%pi],[y,-2*%pi,2*%pi],
[grid,100,100],[gnuplot_preamble,"set pm3d;set hidden3d"]);
```

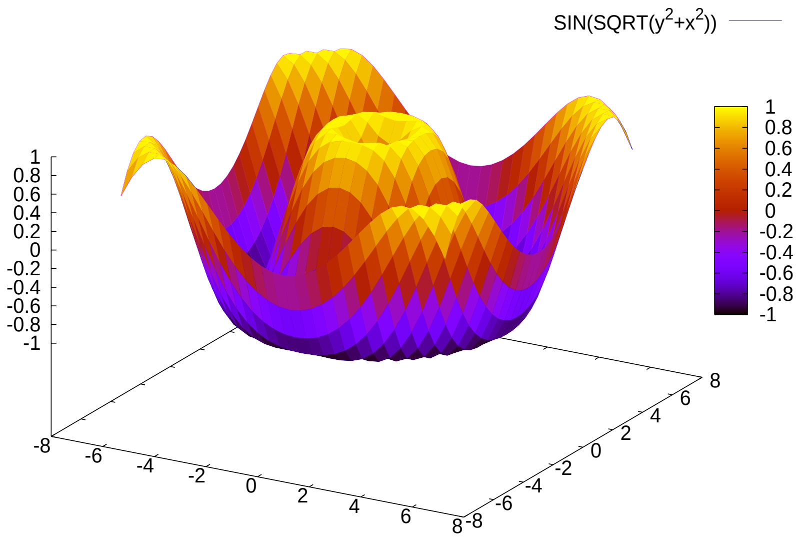

Figura 10: Grafico di $\sin\sqrt{x^2+y^2}$, con poligoni

```
(%o4)
```

con il risultato della figura 11.

Si possono disegnare le curve di livello di una superficie (*contour plot*, in inglese)

```
(%i5) plot3D(sin(sqrt(x^2+y^2)),[x,-2*%pi,2*%pi],[y,-2*%pi,2*%pi],
[grid,100,100],[gnuplot_term,ps],[gnuplot_out_file, "sinxy5.eps"],
[gnuplot_preamble,"set pm3d;set hidden3d;unset colorbox;set contour;
set cntrparam levels 20"]);
```

```
(%o5)
```

con il risultato della figura 12.

```
(%i6) plot3D(sin(sqrt(x^2+y^2)),[x,-2*%pi,2*%pi],[y,-2*%pi,2*%pi],
```

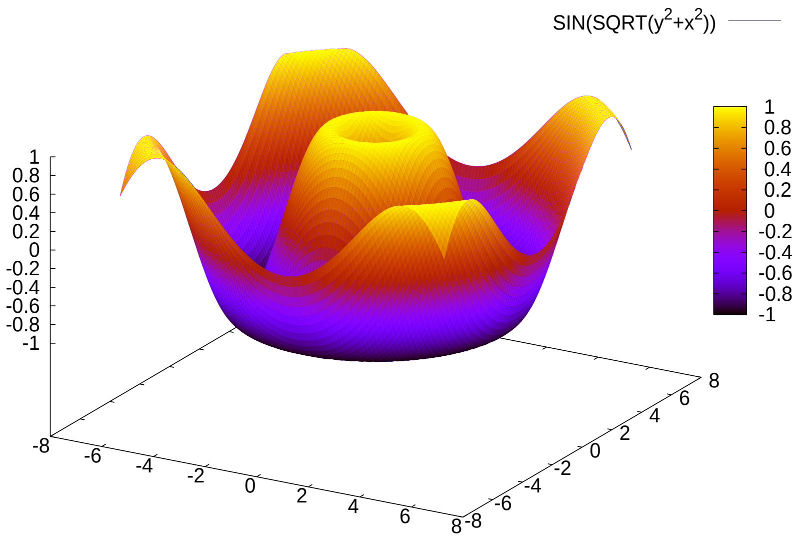

Figura 11: Grafico di $\sin\sqrt{x^2+y^2}$, griglia più fitta

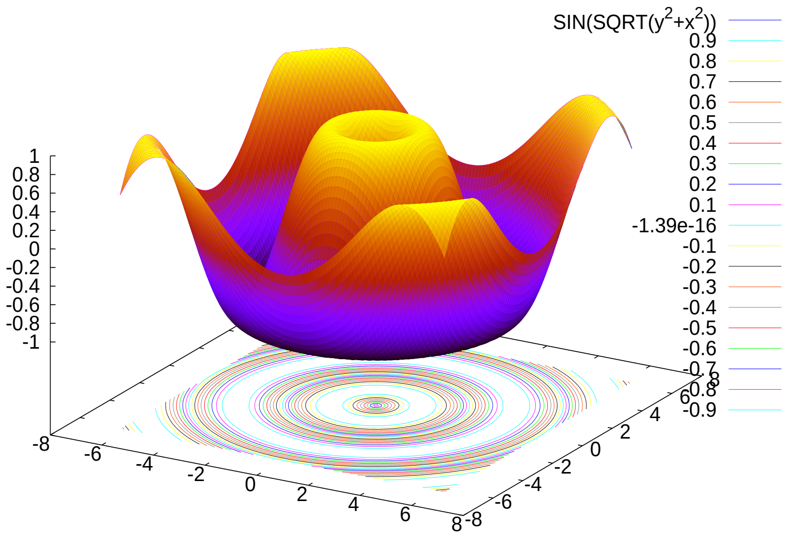

Figura 12: Grafico di $\sin\sqrt{x^2+y^2}$, con curve di livello

```
[grid,150,150],
[gnuplot_term,ps],[gnuplot_out_file, "sinxy6.eps"],
[gnuplot_preamble,"set pm3d;set view map;unset surface"]);
```

```
(%o6)
```

con il risultato della figura 13.

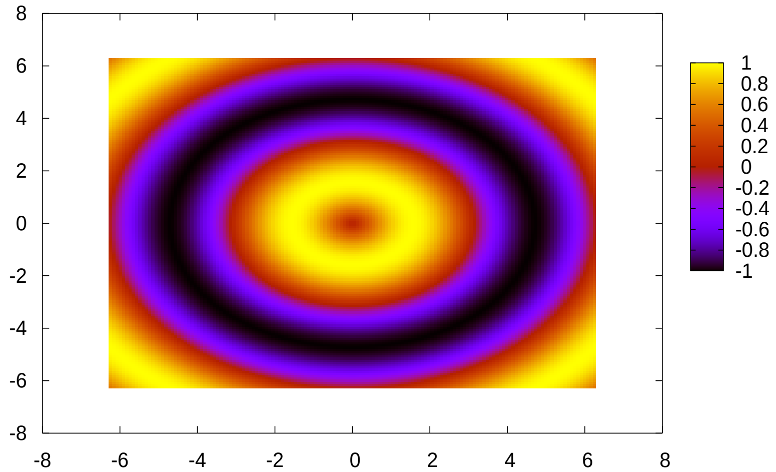

Figura 13: Grafico di sin $\sqrt{x^2+y^2}$, sole curve di livello

```
(%i7) plot3D([u*cos(v), u*sin(v), u], [u, -10, 10], [v, 0, 2*%pi],
[gnuplot_term,ps],[gnuplot_out_file, "param.eps"],
[gnuplot_preamble,"set parametric;set pm3d;set hidden3d"]);
```

```
(%o7)
```

con il risultato della figura 14.

Nota. Non tutte le funzionalità di `GNUplot` sono state raggiungibili da *Maxima*: ad esempio l'animazione dei grafici, o la sovrapposizione di due grafici di superficie,

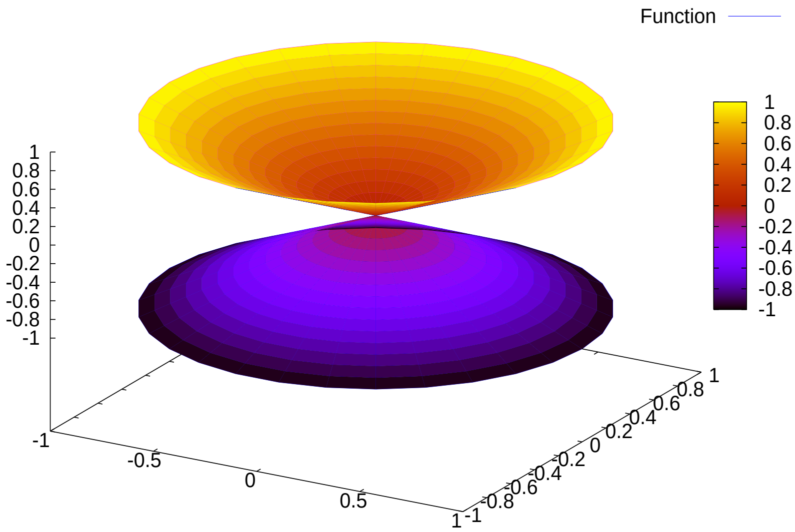

Figura 14: Grafico di un cono in forma parametrica

sono possibili solo usando direttamente `GNUplot`. In altri termini, se si vuole usare molta grafica, sfruttando tutte le possibilità di `GNUplot`, è meglio usare direttamente `GNUplot`, senza utilizzare *Maxima* come interfaccia.

Esercizi.

1. Disegnare la superficie
$$z = \frac{x^3}{8} - \frac{y^3}{27}.$$

2. Disegnare la superficie (in forma parametrica)
$$\begin{cases} x = \sin u \cos v, \\ y = \cos u \cos v, \\ z = \sin v \end{cases}$$
dove $u \in [0, 2\pi]$, $v \in [-\pi/2, \pi/2]$.

Riferimenti bibliografici

[1] *Maxima* website: `http://maxima.sourceforge.net/`

[2] `GNUplot` website: `http://www.gnuplot.info/`