MAXIMA
a Computer Algebra System

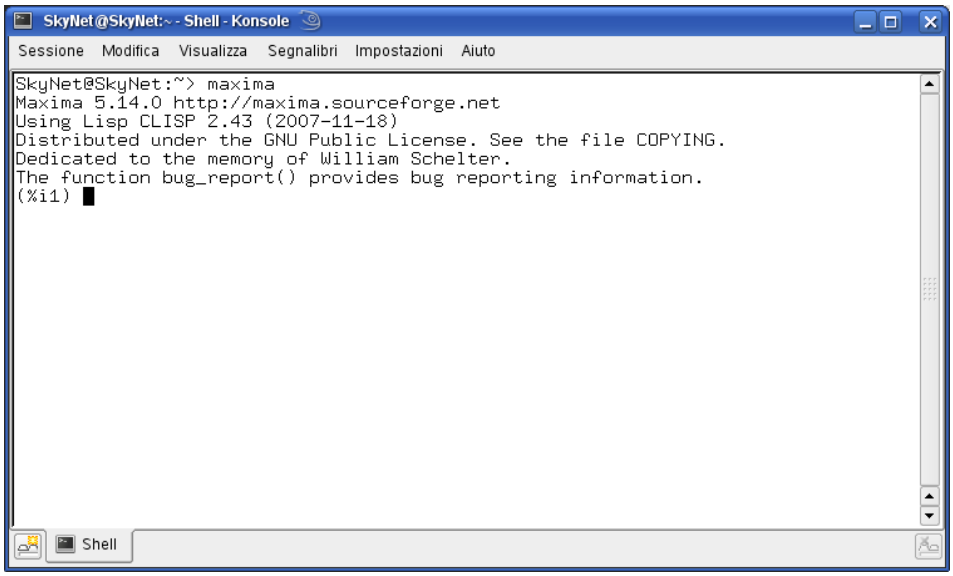
Questo è un piccolo sommario delle operazioni che si possono eseguire tramite (w)xMaxima presentate all'incontro con le classi del Liceo Scientifico sabato 1 Marzo presso la sede dell'associazione LUDAS.
Quanto ora riportato non ha pretesa di essere una guida vera e propria, ma serve solo come memoria ai partecipanti, oltre che permettere a chi vuole cominciare a conoscere il programma in oggetto, di poter valutarne le potenzialità con i comandi che ho reputato essere quelli di uso comune in questo ambiente.
L'approfondimento per arrivare alla piena conoscenza del/dei programmi spetta chiaramente al singolo, ma questo è chiaramente valido in ogni campo.

Introduzione

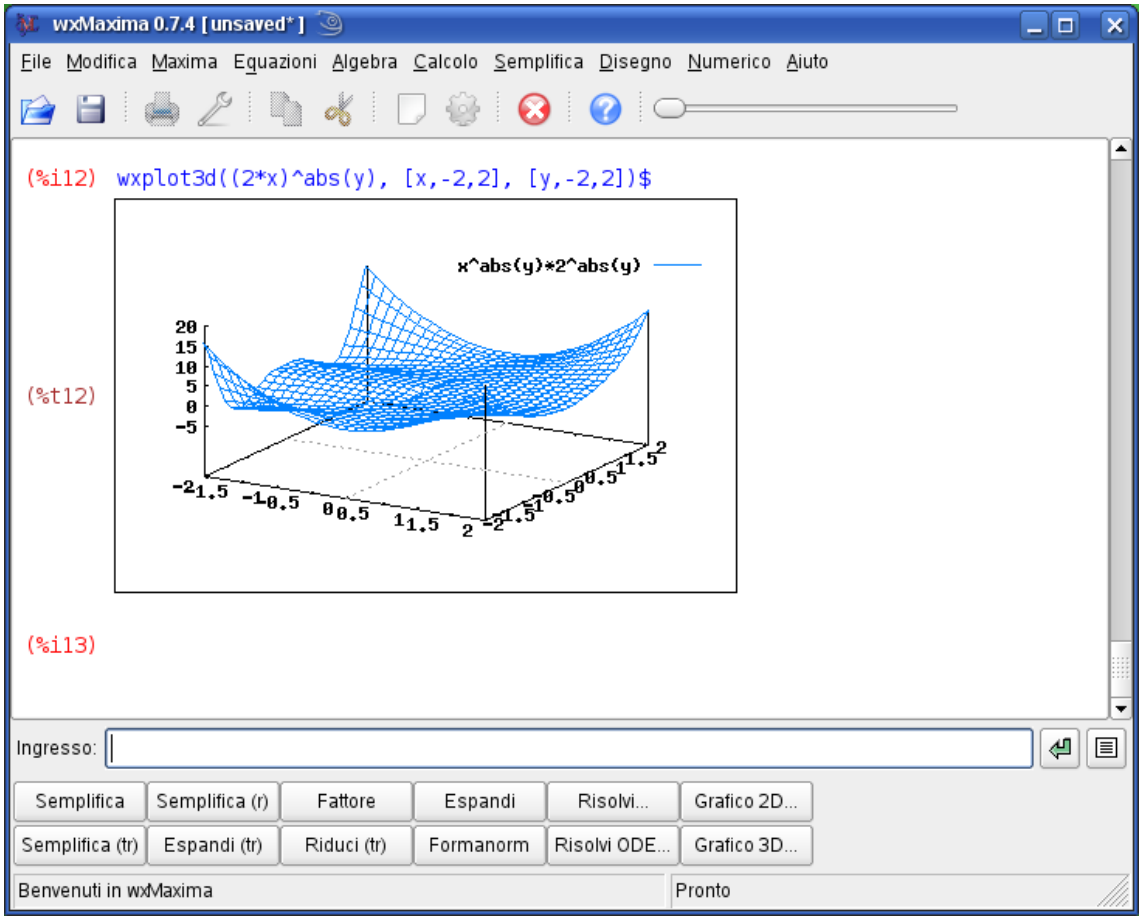
Maxima è un sistema per la manipolazione di espressioni matematiche sia algebriche che numeriche. Tra le espressioni che il programma è in grado di gestire e computare vi sono le operazioni di integrazione, derivazione, serie di Taylor, trasformate di Laplace, sistemi lineari di equazioni, gestione dei polinomi, ecc. Oltre a questo, Maxima permette di "plottare" le funzioni in 2 o 3 dimensioni.
Maxima deriva direttamente dal codice di Macsyma scritto al *Massachusetts Institute of Technology* (MIT) ed è una alternativa completamente libera a programmi come Derive, Maple o Mathematica.
Maxima è un programma completamente OpenSource che può essere compilato per la maggior i principali sistemi operativi UNIX (Linux e Mac), ma anche per Windows.

L'ambiente

L'ambiente nativo di Maxima è la shell (ossia la così detta "riga di comando") in quanto il programma rispetta lo standard POSIX; infatti un un sistema UNIX potete verificare ciò aprendo una shell e digitando il comando maxima. Per esempio in SuSE Linux 10.3 si ottiene un prompt come il seguente:



Come si nota vi è il tipico prompt di Maxima per la richiesta di input (%i). Gli utenti più smaliziati avranno già capito la potenza di istruire il programma a linea di comando, come per esempio la possibilità di controllare un sistema remoto andando a gravare non sulle risorse locali, ma solo sul sistema a cui si è connessi[1].
L'uso tipico è però fatto in locale ed è chiaro che un un'interfaccia grafica oltre che rendere l'ambiente più confortevole permette anche favorire l'avvicinamento dell'utente "novizio" e di semplificargli il lavoro, almeno nella fase iniziale di "esplorazione" del programma.
Le interfacce più famose sono xMaxima e wxMaxima. Questi due front-end non fanno altro che fare quello che si farebbe via shell, ossia istruire Maxima per ottenere il risultato desiderato. La figura seguente mostra l'interfaccia wxMaxima.



Rispetto a xMaxima questa interfaccia presenta qualche pulsante che permette di aiutare l'utente nelle operazioni più semplici. Inoltre la scrittura simbolica delle equazioni e dei vari simboli matematici viene fatta graficamente senza ricorrere ai caratteri testuali. Di contro, xMaxima permette di visualizzare una finestra di help contemporaneamente al programma con la quale è possibile interagire ed eseguire molte delle funzioni messe a disposizione, come integrali e derivate.

Command Line

La linea di comando è la vera essenza dell'ambiente Maxima. Essa è descritta in modo esaustivo e corredata di esempi nel manuale (tasto *F1* oppure menù Aiuto->Guida di Maxima in wxMaxima). Quindi per ora basti sapere che il prompt in cui compare la scritta *%in* (con n numero intero positivo) indica la linea di input, ossia il comando che Maxima dovrà interpretare, mentre *%on* indica l'output. Seguono quindi alcuni comandi (o meglio espressioni) di base come:

- Definizione di una costante con il carattere ":", per esempio

```
(%i1) aa: 10;
```

1 Per esempio il Centro di Super-Calcolo del Politecnico di Milano è un cluster Linux sul quale sono installati vari programmi (per esempio MatLab) controllabili appunto con questa tecnica via SSH.

restituirà il valore:

```
(%o1) 10
```

- Definizione di una espressione letterale, per esempio

```
(%i2) a^2;
```

restituirà

```
(%o2) a²
```

- Il mix delle due restituirà:

```
(%i3) a^aa;
(%o3) a^10
```

- Per quanto riguarda la sintassi, l'apostrofo "'" prima di una parentesi che raggruppa un'espressione, permette di valutare le espressioni letteralmente anche se all'interno vi è una costante:

```
(%i4) '(sqrt(aa));
(%o4)  √aa
```

- Mentre per valutare l'espressione appena scritta si usa il doppio apostrofo seguito da %:

```
(%i4) ''%;
(%o4) 32
```

- Il simbolo "%" permette quindi di valutare alcune costanti come pi-greco (%pi) e la costante di Nepero (%e)
- I due punti ":" definiscono una costante o espressione, mentre ":=" definisce una equazione. "=" definisce una espressione come per esempio una equazione.
- il punto e virgola ";" è un terminatore che permette di avere un output una volta premuto <enter>, mentre "$" on restituisce output. I terminatori sono necessari.
- Per eliminare una entità (espressione, costante, ecc) usare **kill**(a,b,c,...) o **kill**(all)

Esempi

Passiamo ora in rassegna alcuni esempi per valutare le peculiarità dell'applicazione.

Polinomi (cap 12 del mauale)

Espansione di un polinomio:

```
(%i) expand((x+3)^4);
(%o) x^4+12*x^3+54*x^2+108*x+81
```

Supponendo ora di voler "valutare" l'espressione è sufficiente ricorrere al comando **ev** (*evaluate*):

```
(%i) ev(%,x=0.99)
(%o) 253.44958401
```

Il comando in questione è molto complesso. Per vedere tutte le peculiarità si rimanda al capitolo 4 del manuale. Si vuole però notare una cosa: Maxima esegue i calcoli in "altissima precisione". Ciò vuol semplicemente dire che se si lancia immediatamente dopo **expand** il comando:

```
(%i) ev(%, x=3/7)
```

il risultato sarà un numero "perfetto" ossia:

```
(%o) 331776/2401
```

Divisione di un polinomio:

```
(%i) divide(x^2 + 2*x*y+y^2, x+y);
(%o) [x+y, 0]
```

Il comando **divide** permette di definire un primo polinomio da dividere mediante il secondo. Il risultato sono dei coefficiente che moltiplicati per il polinomio divisore danno il polinomio dividendo.

Equazioni (cap 21 del mauale)

Radici di un polinomio:

```
(%i) eqn: (1 + 2*x)^3 = 13.5*(1 + x^5)$
(%o) allroots (eqn);
```

(%o2) [x = .8296749902129361, x = - 1.015755543828121,
x = .9659625152196369 %i - .4069597231924075,
x = - .9659625152196369 %i - .4069597231924075, x = 1.0]

Come si nota in questo caso il risultato viene dato in formato floating. Si noti che pur essendo una funzione polinomiale, questa è definita nelle equazioni perché di fatto l'espressione scritta è una equazione.

Trovare le soluzioni di un sistema di equazioni:

(%i) e1: x^2+y^2$
(%i) e2: -1 - y + 2*y^2 - x + x^2$
(%i) solve([e1, e2], [x, y]);

(%o) [$[x = \frac{-1}{\sqrt{3}}, y = \frac{1}{\sqrt{3}}]$, $[x = \frac{1}{\sqrt{3}}, y = \frac{-1}{\sqrt{3}}]$,
$[x = \frac{-1}{3}, y = \frac{1}{3}]$, [x = 1, y =1]]

Questo comando permette di ottenere tutte le soluzioni date *n* equazioni in *n* incognite. Sia le equazioni che le incognite vengono passate come vettori; le prime sono il primo vettore, le seconde il secondo. Vedere anche la funzione **algsys** la quale è molto simile.

Soluzione di un sistema non lineare:

(%i) load(newton1)
(%o) /usr/share/maxima/5.14.0/share/numeric/newton1.mac
(%i) newton(log(x)-cos(x),x,1,1/1000);
(%o) 1.302955472926695

Questo comando applica il metodo di Newton per arrivare alla convergenza del sistema. Il sistema (primo parametro) è passato in forma normale (ossia è come se fosse eguagliato a 0), il parametro su cui effettuare i calcoli è il secondo (x), segue il punto di inizio a cui effettuare i calcoli e infine la tolleranza di errore al di sotto della quale si assume che sia avvenuta la convergenza dell'iterazione.

Limiti *(cap 18 del mauale)*

Sui limiti non vi è molto da dire. La funzione è **limit** e richiede in ingresso come primo termine una funzione, mentre gli altri termini sono facoltativi e sono la variabile su cui effettuare il limite, il valore in cui calcolare il limite e la direzione.

(%i) limit(1/(y-3),y,3,minus);
(%o) -∞

La direzione può essere minux, plus, +inf, -inf.

Integrazione *(cap 20 del mauale)*

L'integrazione in Maxima può essere eseguita sia in ambito classico (Reimann) sia distribuzionale. Gli esempi riportati sono solo in ambito classico.
La funzione principale è **integrate**, e può essere usata in alcuni modi:

Integrazione indefinita:

(%i) integrate(a*x,x);
(%o) $a\frac{x^2}{2}$

richiede solo la funzione da integrare rispetto a una precisa variabile.

Integrazione definita:

(%i23) integrate(%e^(-x),x,0,+inf);
(%o23) 1

Vengono definiti i limiti i quali possono anche essere “infiniti”.

Integrazione numerica: è possibile anche definire l'integrazione numerica, ossia calcolare il valore di una funzione integrale qualunque. Per esempio data una funzione il cui integrale non sia una funzione nota di ha:

```
(%i) integrate(cos(1/x^3), x, 1,2);
```

Questa funzione non verrà mai calcolata se non in forma letterale. Utilizzando invece il pacchetto QUADPACK è possibile integrare numericamente la funzione:

```
(%i) quad_qag(cos(1/x^3), x, 1,  2, 6);
(%o) [0.9068303994012, 1.006783988845239*10^-14, 61, 0]
```

Si noti che l'ultimo parametro (di valore 6 in questo caso) indica solo le come eseguire il processo di integrazione e deve essere compreso tra 1 e 6. Il risultato sono quattro numeri: il primo è il risultato, il secondo è l'approssimazione, il terzo sono il numero di integrazioni per e il quarto è un codice di errore (vedere il manuale).

Nota: non scrivere l'equazione nella forma *expression1=experession2* per esempio cos(x)-log(x)=1 perché manda in crisi il sistema. L'espressione scritta è già da considerarsi posta pari a 0.

Derivazione (cap 19 del mauale)

La funzione principale è chiaramente l'operazione di derivata definita con **diff** (*differentiate*):

```
(%i) diff(x^x);
```

(%o) $x^x(\log(x)+1)dx$

questo risultato vede la presenza del differenziale di *x* (vedere funzione differenziale **del**) in quanto non è definita la variabile di derivazione. Se questa fosse definita si avrebbe:

```
(%i) diff(x^x,x);
```

(%o) $x^x(\log(x)+1)$

La derivata ennesima viene definita come terzo parametro. Per esempio la derivata seconda dell'espressione precedente sarà:

```
(%i) diff(x^x,x,2);
```

(%o) $x^x(\log(x)+1)^2+x^{x-1}$

Plotting (cap 8 del mauale)

Il sistema di plot dei grafici si appoggia a una variabile di ambiente di Maxima detta **plot_options** la quale se richiamata restituisce una cosa simile alla seguente:

*[[x,-1.75555970201398*10^+305,1.75555970201398*10^+305],[y,-1.75555970201398*10^*
*+305,1.75555970201398*10^+305],[*
t,-3,3],[grid,30,30],[transform_xy,false],[run_viewer,true],[plot_format,gnuplot_pipes],[gnuplot_term,
default],[gnuplot_out_file,false],[nticks,10],[adapt_depth,10],[gnuplot_pm3d,false],[gnuplot_preamble,],[
gnuplot_curve_titles,[default]],[gnuplot_curve_styles,[with lines 3,with lines 1,with lines 2,with lines 5,
with lines 4,with lines 6,with lines 7]],[gnuplot_default_term_command,set term x11 font "Helvetica,16"],[
gnuplot_dumb_term_command,set term dumb 79 22],[gnuplot_ps_term_command,
set size 1.5, 1.5;set term postscript eps enhanced color solid 24],[gnuplot_pipes_term,x11],
[plot_realpart,false]]

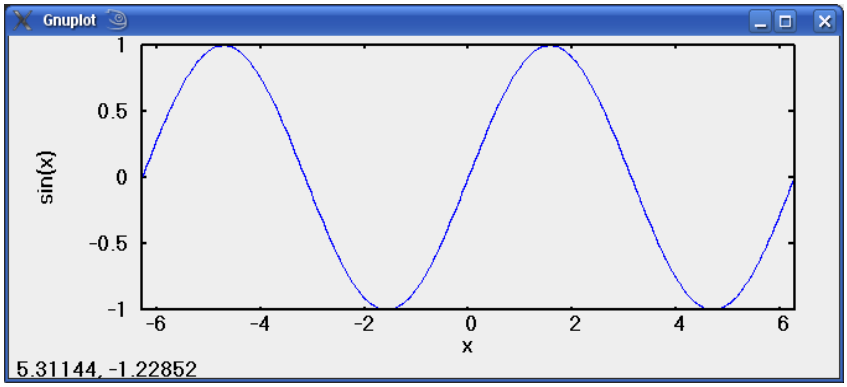
ossia tutti i parametri necessari per effettuare particolari plot. Settando questa variabile, i valori impostati diventano di default per la sessione corrente, ma possono essere modificati per il singolo grafico solamente. Il sistema di plot utilizzato di default è *gnuplot*.

Per settare le impostazioni si ricorre al comando **set_plot_option** il quale richiede in ingresso una sintassi simile a quella della variabile *plot_options*, per esempio *set_plot_option([grid,50,50])*;

plot2d è il primo comando per il plot di una o più funzioni. Tipicamente richiede in ingresso una funzione definita secondo una variabile e il range di variazione di tale variabile. Seguono alcuni esempi:

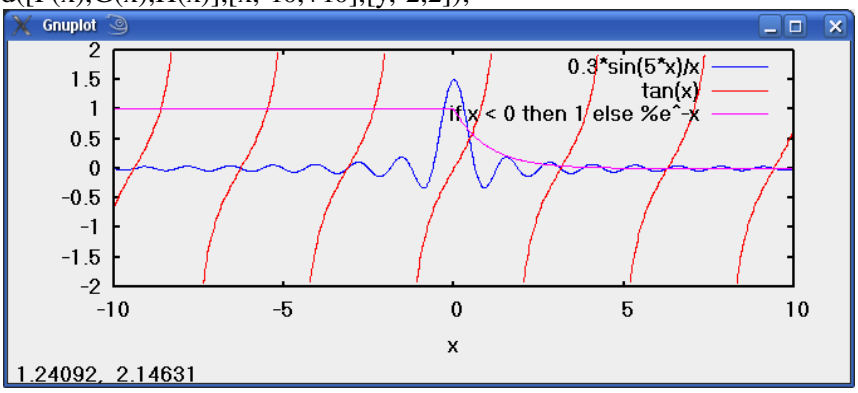
Plot singola funzione:

(%i) plot2d(sin(x),[x,-2*%pi,2*%pi]);



Plot funzioni multiple: in questo esempio viene visualizzato anche come contenere il grafico nelle ordinate (asse y) e anche come definire una funzione *splain* (la funzione H).

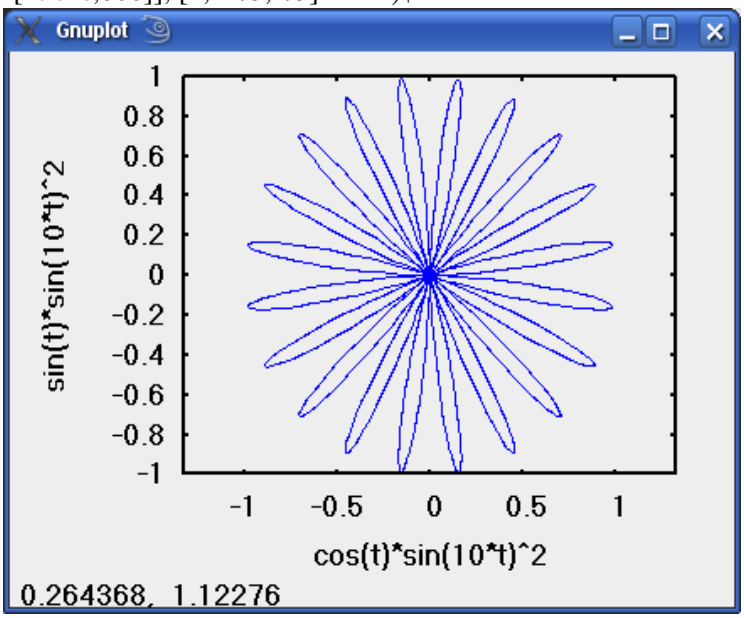
(%i) F(x):=0.3*sin(5*x)/x$
(%i) G(x):=tan(x)$
(%i) H(x):=if x<0 then 1 else %e^-x$
(%i) plot2d([F(x),G(x),H(x)],[x,-10,+10],[y,-2,2]);



Si noti che se le funzioni vengono passate come [F,G,H] e non come [F(x),G(x),H(x)] il sistema va a "calcolare" le divisioni per 0 nel seno cardinale e il plot non viene effettuato.

Plot di funzioni parametriche: linea nel piano

(%i) plot2d ([parametric, cos(t)*(sin(10*t))^2, sin(t)*(sin(10*t))^2, [t,-%pi,%pi], [nticks,800]], [x, -4/3,4/3])$

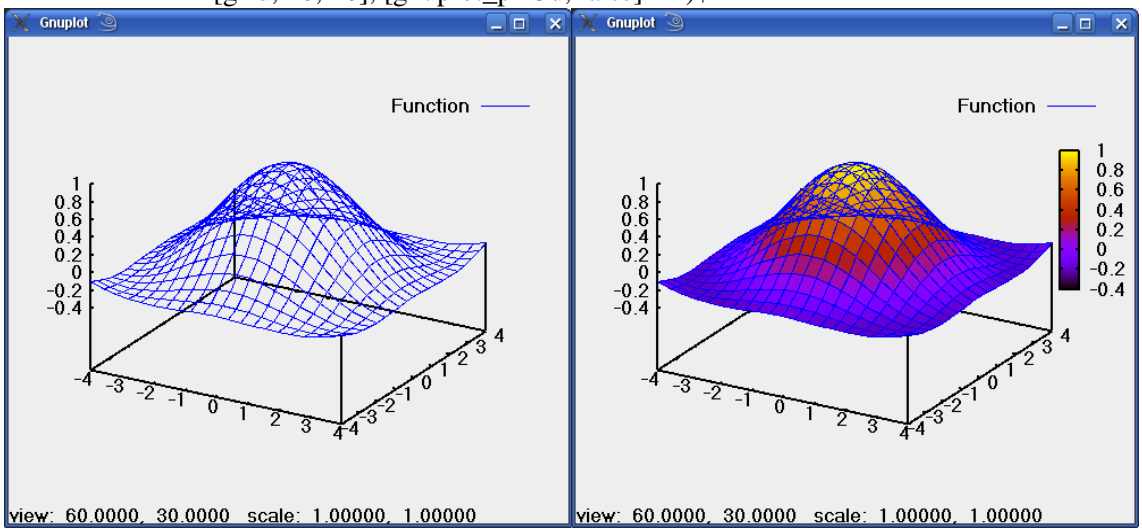


nticks indica il passo con cui la funzione è definita: più è alto, più la funzione è precisa; il valore dato a x serve invece per avere una corretta visione della geometria della figura (per esempio se si plotta un cerchio in modo che non sembri un'ellisse).

plot3d: in questo caso in ingresso è richiesto sia il range di variazione dell'ascissa che della ordinata.

Plot del seno cardinale a 2 dimensioni: la funzione è tale per cui partendo dal punto di massimo della funzione tridimensionale, muovendosi in una qualsiasi direzione la variazione della coordinata z sarà sempre come quella del seno cardinale.

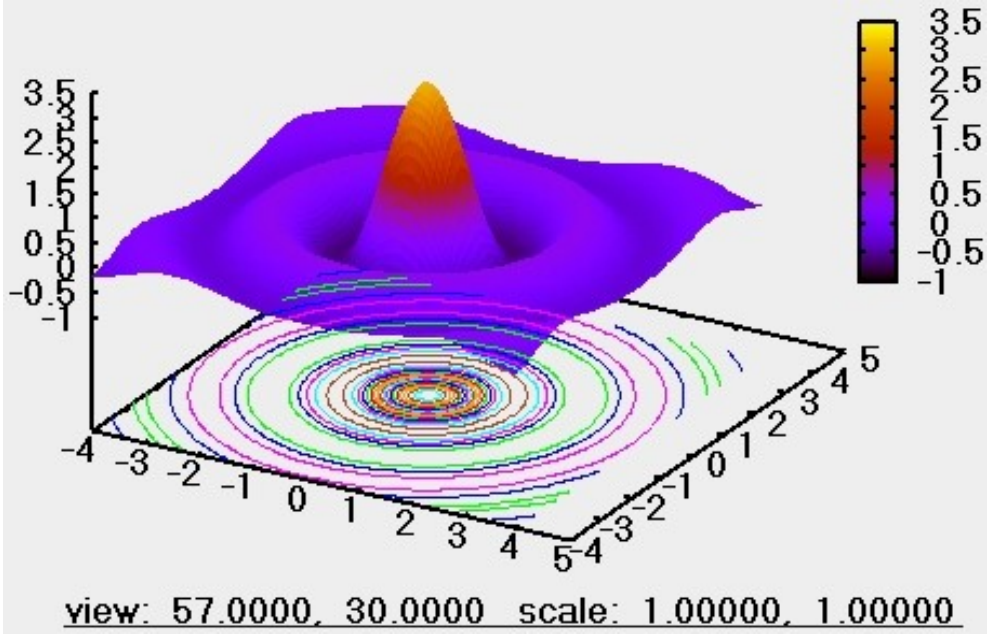
```
(%i) plot3d (     sin(sqrt(x^2+y^2))/sqrt(x^2+y^2), [x, -4, 4], [y, -4, 4],
                  [grid, 20, 20], [gnuplot_pm3d, false]     )$
```



Nel primo caso *gnuplot_pm3d=false*, nel secondo *gnuplot_pm3d=true*. In entrambi i casi si vede la parte retrostante del grafico. Inoltre si vede la "rete" dovuta all'intersezione dei piani *XZ* e *YZ*.

Plot del seno cardinale a frequenza maggiore: la funzione precedente non sembra avere la forma del seno cardinale per questo ora verrà aumentata la frequenza moltiplicando l'argomento del seno. Viene ora anche mostrato come visualizzare il grafico delle isobare sul piano XY. Per fare questo si ricorre al *preambolo*.

```
(%i) my_preamble: "set pm3d at s;unset surface;set contour; set cntrparam levels 20;unset key"$
(%i) plot3d (     sin(3*sqrt(x^2+y^2))/sqrt(x^2+y^2), [x, -4, 4], [y, -4, 4], [grid, 100, 100],
                  [gnuplot_preamble, my_preamble]     )$
```

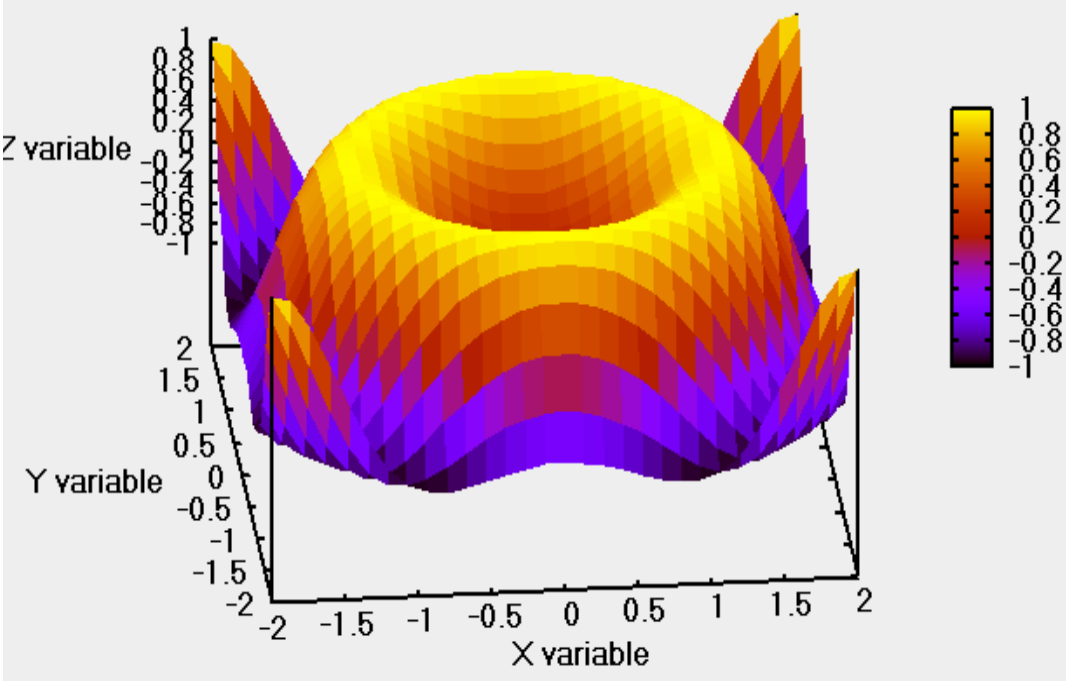


Per la scrittura di un *preambolo* corretto si rimanda alla documentazione generale di *gnuplot* in quanto rappresenta uno script di configurazione per questo programma.

Animation (cap 48 del mauale)

In questo capitolo viene spiegata la libreria *draw* la quale ha funzioni simili a quelle per il plotting, ma presenta qualche funzione in più per esempio permette di intersecare le figure e riempire le aree sottese dalle figure, cambiare spessori di linea, salvare in file di vario formato, ecc. Oltre a questo presenta qualche peculiarità legata alle animazioni.
Una animazione può essere infatti molto utile per capire come il sistema evolve nel tempo; insomma, se un'immagine parla più di mille parole, figuriamoci un filmato!
Questa libreria non è caricata di default e occorre caricarla almeno una volta durante la sessione di Maxima (se la si vuole usare) tramite il comando *load(draw)*. L'esempio che segue permette di fare il plot3d di una funzione usando draw:

```
(%i) load(draw);
(%i) my_pr2: "set pm3d at s; unset surface; set cntrparam levels 20; unset key"$
(%i) draw3d(    zlabel = "Z variable", ylabel = "Y variable", xlabel = "X variable",
                explicit(sin(x^2+y^2),x,-2,2,y,-2,2), user_preamble=my_pr2      )$
```



La sintassi è leggermente differente rispetto a plot3d, ma le funzionalità sono un po' più estese. Chiaramente esiste anche la versione 2D del comando.
Passando ora alle animazioni, la sintassi del comando draw diventa:

```
(%i) draw(
delay     = 40,
file_name = "zzz",
terminal  = 'animated_gif,
gr3d(    surface_hide=true,enhanced3d = true,explicit(sin(sqrt(x^2+y^2))/sqrt(x^2 +y^2),
         x,-3,3,y,-3,3)),
gr3d(    surface_hide=true,enhanced3d = true,explicit(sin(1.5*sqrt(x^2+y^2))/sqrt(x^2+y^2),
         x,-3,3,y,-3,3)),
gr3d(    surface_hide=true,enhanced3d = true,explicit(sin(2*sqrt(x^2+y^2))/sqrt(x^2+y^2),
         x,-3,3,y,-3,3)))$
```

surface_hide e *enhanced3d* sono i comandi equivalenti a quelli che in gnuplot permettono di avere un 3D più realistico nascondendo la parte retrostante della superficie rispetto al punto di visione.

Info & Credits

Questa piccola introduzione a Maxima è stata scritta per l'incontro con le classi del Liceo Scientifico di Mantova nell'incontro tenutosi in via Oberdan 7 Sabato 1° Marzo 2008 organizzato dalle associazioni ARCO e LUGMan da Calzoni Pietro aka Calzo, membro dell'associazione LUGMan.

I formati distribuiti sono PDF e ODT (OpenOffice)

Questo documento è stato scritto utilizzando Open Office 2.3 sotto sistema operativo Linux openSuSE 10.3. Altri software utilizzati: Maxima e GNUPlot.

Link Utili

www.lugman.org
www.clubvirgiliano.altervista.org

Software:

http://maxima.sourceforge.net
http://www.opensuse.org
http://www.openoffice.org

Documentazione:

http://sodilinux.itd.cnr.it/cd/scheda1.php?ID=3
http://poincare.unile.it/vitolo/progmat.shtml
http://maxima.sourceforge.net/docs.shtml
http://sodilinux.itd.cnr.it/cd/documenti/espmaxima.pdf

Altri link didattici:

http://sodilinux.itd.cnr.it/
http://www.edubuntu.org/ *oppure* http://www.ubuntu-it.org/index.php?page=edubuntu
https://linux.ing.unibo.it/progetti/Engineerix/dettagli