GRÁFICOS

GNUPLOT es un programa interactivo para dibujar dirigido mediante comandos. Los comandos y funciones de GNUPLOT deberán escribirse con minúsculas o mayúsculas según se indique ya que el programa reconoce como diferentes estos caracteres. Todos los comandos pueden abreviarse siempre y cuando la abreviatura no se confunda con la de otro comando. Varios comandos pueden aparecer en la misma línea siempre que se separen por punto y coma (;). Las cadenas de caracteres se indican mediante comillas (simples o dobles). Un comando puede ocupar muchas líneas, basta con finalizar cada línea con un \ (backslash), que deberá ser el último carácter de la línea. También podemos añadir comentarios al final de una línea (que no serán ejecutados) precedidos por el carácter #. Esta opción la utilizaremos en lo sucesivo para explicaciones simples de la acción de un comando particular.

GNUPLOT tiene dos modos de ejecución: batch e interactivo. El modo batch se accede ejecutando:

```
$ gnuplot fichero1 fichero2 ... ficheroN
```

donde se supone que los argumentos de `gnuplot` son ficheros que contienen comandos válidos. En este modo se ejecutarán las instrucciones de cada fichero en el orden establecido y el programa finalizará. Si en el ejemplo anterior se ejecutara tan sólo `gnuplot` sin ficheros a cargar, entonces el programa entrará en un modo interactivo. Es decir, ejecutando:

```
$ gnuplot
```

aparecerá en la misma ventana lo siguiente:

```
        G N U P L O T
        Version 4.2 patchlevel 0
        last modified March 2007
        System: Linux 2.6.22.18

        Copyright (C) 1986 - 1993, 1998, 2004, 2007
        Thomas Williams, Colin Kelley and many others

        Type ‘help‘ to access the on-line reference manual.
        The gnuplot FAQ is available from
                http://www.gnuplot.info/faq/

        Send comments and help requests to  <gnuplot-info@lists.sourceforge.net>
        Send bug reports and suggestions to <gnuplot-bugs@lists.sourceforge.net>

Terminal type set to 'x11'

gnuplot>
```

donde los comandos deberán escribirse a partir de `gnuplot>`.

Una vez entremos en modo interactivo, `gnuplot` dispone de un completísimo manual interactivo que se accede mediante el comando **help** al que bastará con dar cualquier nombre de comando para que nos explique su funcionamiento y opciones. Para salir del modo interactivo tenemos dos comandos equivalentes **exit** y **quit**.

En general, cualquier expresión matemática aceptada por los lenguajes de programación más comunes como C, FORTRAN, PASCAL o BASIC es válida. Los espacios en blanco son ignorados y la precedencia de los operadores es como en C o FORTRAN. Acepta números complejos que deben escribirse en la forma {3,2} = 3 + 2i, en esta forma {0,1} representa al propio i. Las funciones de `gnuplot` son las mismas de la librería matemática UNIX (las mismas que utiliza el FORTRAN). Para mayor información sobre las funciones y operadores que admite el `gnuplot` ejecutar:

```
gnuplot> help expressions functions
```

pudiéndose pedir información adicional sobre las funciones disponibles.

Los comandos primarios del programa son **plot** y **splot**. Ellos dibujan funciones y datos de muchísimas formas posibles. **plot** se utiliza para gráficos en 2-d (variables `x, y`) y **splot** se utiliza para gráficos en 3-d (variables `x, y, z`). Una explicación detallada de estos comandos puede obtenerse mediante:

```
gnuplot> help plot
gnuplot> help splot
```

Los comandos set y unset

Dos comandos importantes son **set** y **unset** ya que con ellos se pueden modificar los gráficos de **plot** y **splot**. El comando **set** permite establecer gran cantidad de opciones, por lo que resulta también conveniente estudiarlo a fondo mediante:

```
gnuplot> help set
```

Con el comando **set terminal** puede incluso cambiarse el dispositivo de salida del gráfico (x11 para ventanas Xwindow, postscript para salida a fichero .ps o .eps, png para salida a un fichero gráfico, table para salida a fichero en forma de tabla ASCII con los valores de x, y, z en columnas, latex para salidas a LaTeX, ...), normalmente se utiliza con **set output** para dirigir la salida a un fichero. Ambos comandos (**set terminal** y **set output**) nos serán muy útiles para obtener gráficas en papel y veremos un ejemplo al final de esta práctica.

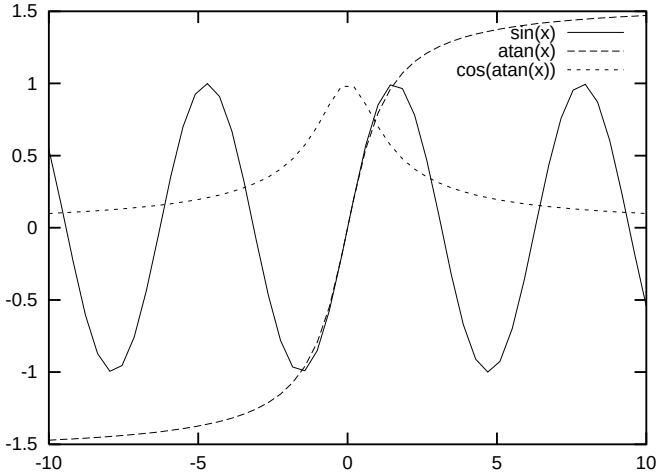
Por último, se pueden restaurar los valores iniciales con el comando **reset**.

El comando plot

A continuación veremos unos ejemplos de uso de **plot** y **splot** para generar gráficos diversos en pantalla (que es la opción por defecto, equivalente a **set terminal x11**).

Representando funciones

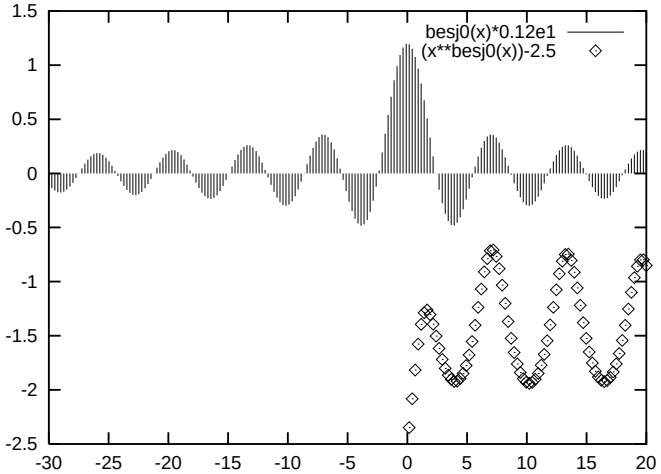
Empezaremos por gráficos simples en 2-d de funciones analíticas. Por ejemplo, nuestro primer gráfico será:



para obtenerlo ejecutar:

```
gnuplot> set samples 50                          #Calcula la función en 50 puntos
gnuplot> plot [-10:10] sin(x),atan(x),cos(atan(x))  #Representa las funciones entre -10 y 10
```

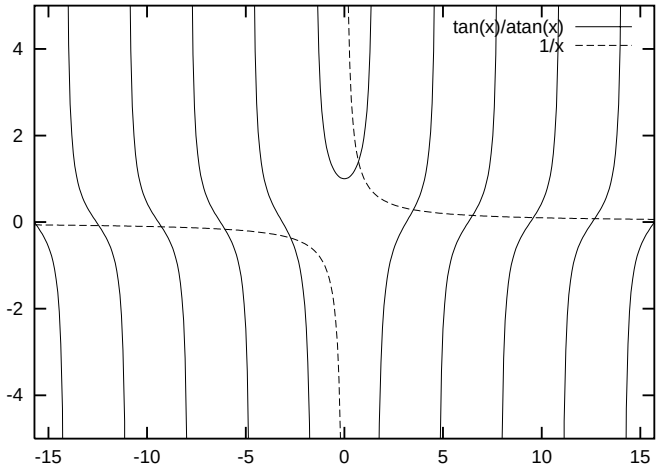
resultará conveniente practicar modificando el comando **set samples** a valores como 20 o 500 y ejecutando posteriormente **replot** que permite redibujar el gráfico anterior con las nuevas especificaciones. También podéis probar a modificar el intervalo en `x` (de [-10:10] a otros valores). El siguiente gráfico es demostrativo de que podemos utilizar puntos o 'impulsos' (líneas verticales de `y = 0` a `y = f(x)`) en lugar de líneas continuas:



para obtenerlo ejecutar:

```
gnuplot> set samples 200
gnuplot> plot [-30:20] besj0(x)*0.12e1 with impulses, (x**besj0(x))-2.5 with points
```

podemos dibujar otras funciones tan complicadas como se desee, por ejemplo:



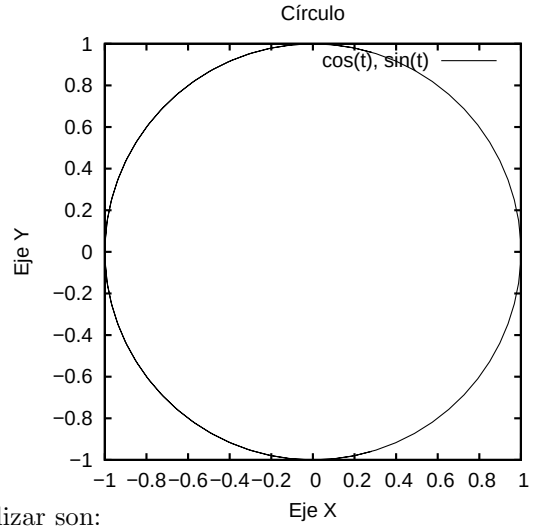
para obtenerlas ejecutar:

```
gnuplot> set samples 400
gnuplot> plot [-5*pi:5*pi] [-5:5] tan(x)/atan(x), 1/x
```

donde puede apreciarse que hemos puesto límites al intervalo en `y` [-5:5], además aquí el comando **set samples** es muy importante en el resultado para la función `tan(x)/atan(x)`.

Representación de funciones en modo paramétrico

La opción **set parametric** permite dibujar gráficas de funciones expresadas en modo paramétrico. Así para representar un círculo

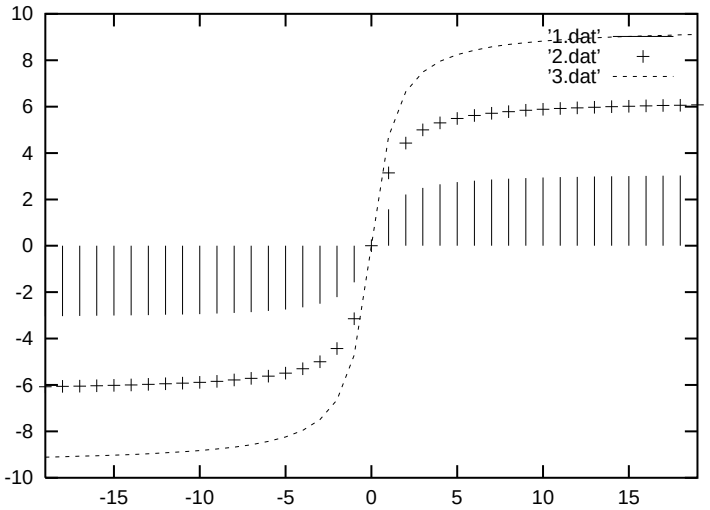


los comandos que hay que utilizar son:

```
gnuplot> reset
gnuplot> set parametric                 #Modo paramétrico
gnuplot> set encoding iso_8859_1        #Permite poner acentos en el título
gnuplot> set title "Círculo"
gnuplot> set xlabel "Eje X"
gnuplot> set ylabel "Eje Y"
gnuplot> set size square                #Permite hacer cuadrada la imagen
gnuplot> plot cos(t),sin(t)
gnuplot> reset
```

Representando datos

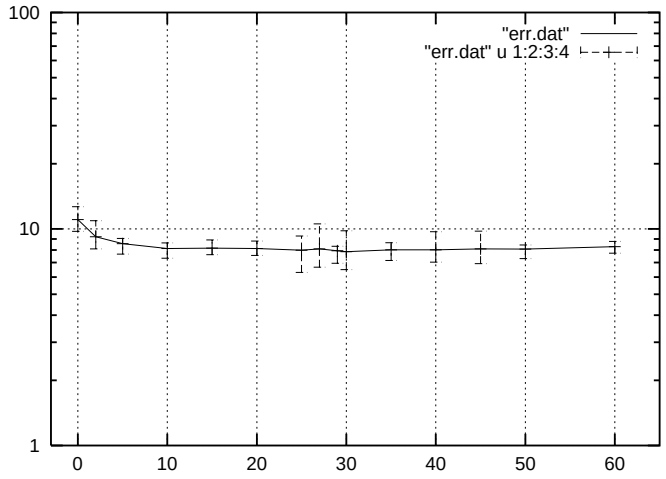
También podemos hacer gráficos con ficheros de datos 2-d, para ello basta que el fichero contenga los valores `x` con su correspondiente `y` como dos columnas del fichero. Un ejemplo de tres ficheros de datos (1.dat, 2.dat y 3.dat, que tenéis que copiar a vuestro directorio de trabajo desde el directorio **/datos/p/paniagu**[1]) dibujados es:

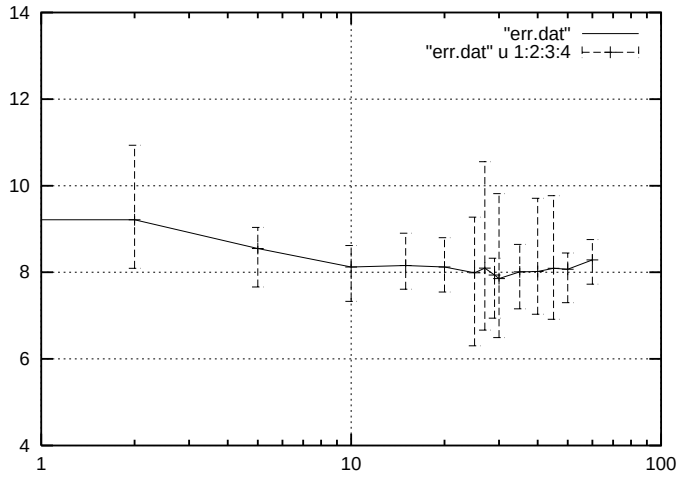


para obtener un gráfico similar bastará con editar los datos (con el editor **kate** por ejemplo), esto puede ejecutarse sin salir de gnuplot utilizando el comando `!` (admiración) tal y como se explica a continuación:

```
gnuplot> !kate 1.dat
gnuplot> !kate 2.dat
gnuplot> !kate 3.dat
gnuplot> set autoscale
gnuplot> plot[-19:19] '1.dat' with impulses, '2.dat', '3.dat' with lines
```

Otros gráficos con datos experimentales requieren el uso de barras de error que indiquen el intervalo en que sería válido el resultado. Además podemos utilizar escalas logarítmicas si fuese necesario, dos ejemplos del mismo fichero de datos 'err.dat' son:





gráficos que pueden obtenerse mediante la siguiente secuencia de comandos:

```
gnuplot> !kate err.dat
gnuplot> set grid
gnuplot> set logscale y
gnuplot> plot [-3:65] [1:100] "err.dat" with lines, "err.dat" u 1:2:3:4 with errorbar
gnuplot> unset logscale y
gnuplot> set logscale x
gnuplot> plot [1:100] [4:14] "err.dat" with lines, "err.dat" u 1:2:3:4 with errorbar
gnuplot> unset logscale x
gnuplot> unset grid
```

[1] Para ello ejecutad el comando: cp /datos/p/paniagu/*.dat .

aquí el fichero 'err.dat' deberá contener 4 columnas: x, y, ybajo, yalto, donde ybajo e yalto corresponden al límite inferior y superior, respectivamente, en el valor de y. Las escalas podrán ajustarse en función de los datos del fichero. Aparte de puntos, líneas o impulsos, pueden construirse gráficos en los que los puntos quedan unidos mediante escalones como en el ejemplo siguiente:

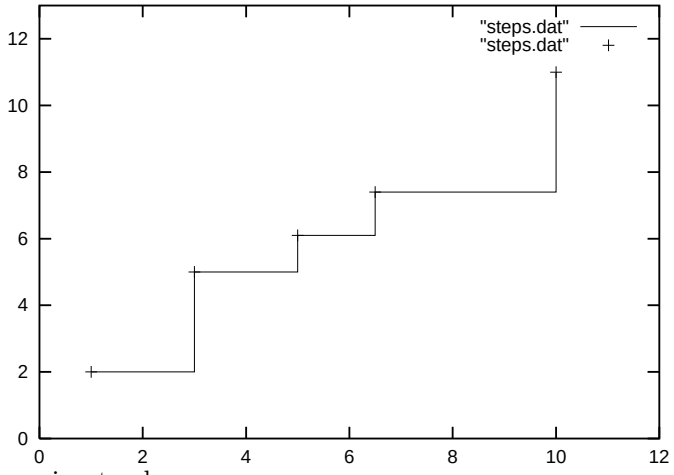


gráfico que puede obtenerse ejecutando:

```
gnuplot> !kate steps.dat
gnuplot> plot [0:12][0:13] "steps.dat" wi steps, "steps.dat" wi point
```

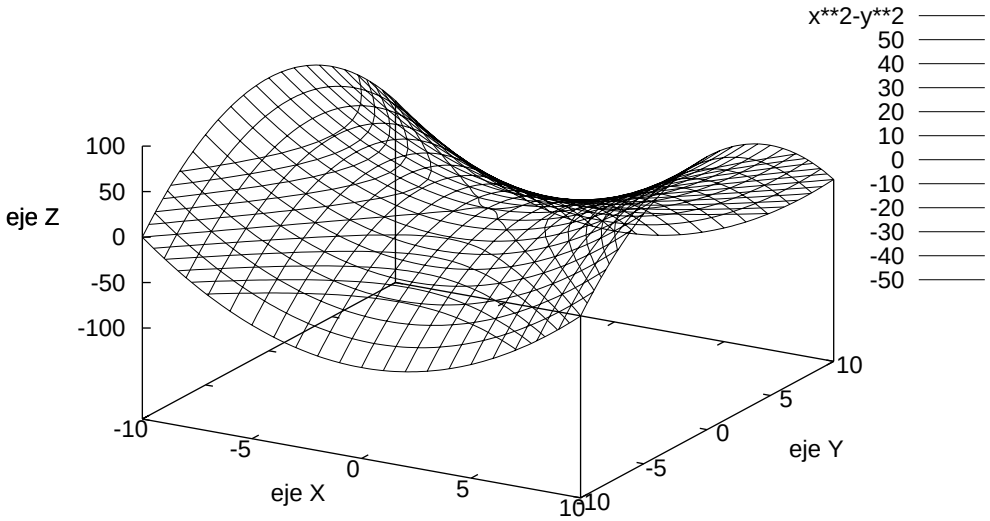
donde 'steps.dat' es un fichero que contiene 5 líneas con parejas (x,y).

El comando splot

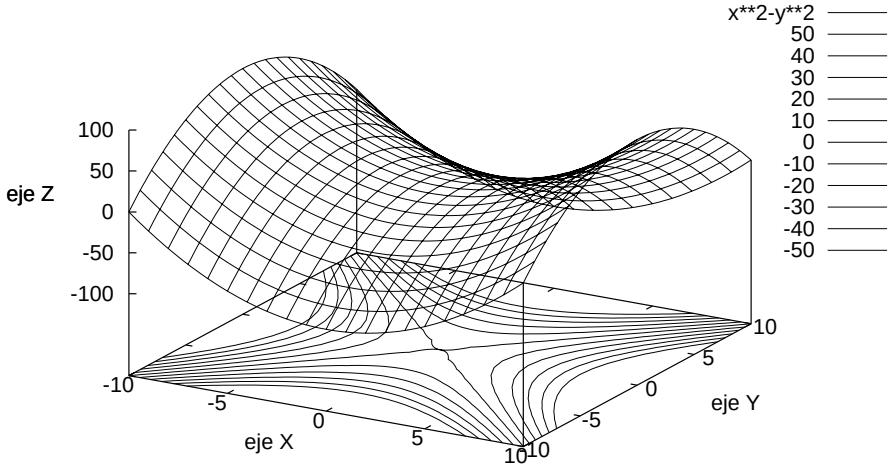
Representando funciones

Los gráficos 3-d son también sencillos de obtener como veremos a continuación para el siguiente ejemplo, que puede obtenerse ejecutando las instrucciones siguientes:

```
gnuplot> unset parametric
gnuplot> set key default
gnuplot> set isosamples 21
gnuplot> set ylabel "eje X"
gnuplot> set ylabel "eje Y"
gnuplot> set zlabel "eje Z"
gnuplot> set cntrparam levels discrete -50,-40,-30,-20,-10,0,10,20,30,40,50
gnuplot> set contour surface              # Permite representar las líneas de nivel en la superficie
gnuplot> splot [-10:10] [-10:10] x**2-y**2
```



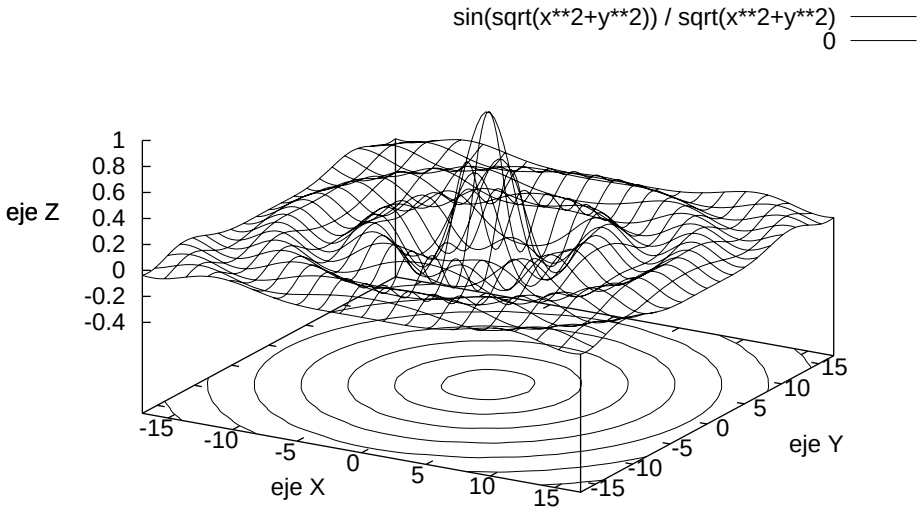
El comando **set contour surface** dibuja también curvas de nivel sobre la propia superficie, pero el resultado sobre papel es difícil de visualizar, de ahí que en el siguiente ejemplo pedimos que se dibujen los contornos sobre la base (el plano XY):



que se obtiene ejecutando:

```
gnuplot> set contour base                    # Permite representar las líneas de nivel en la base
gnuplot> replot
```

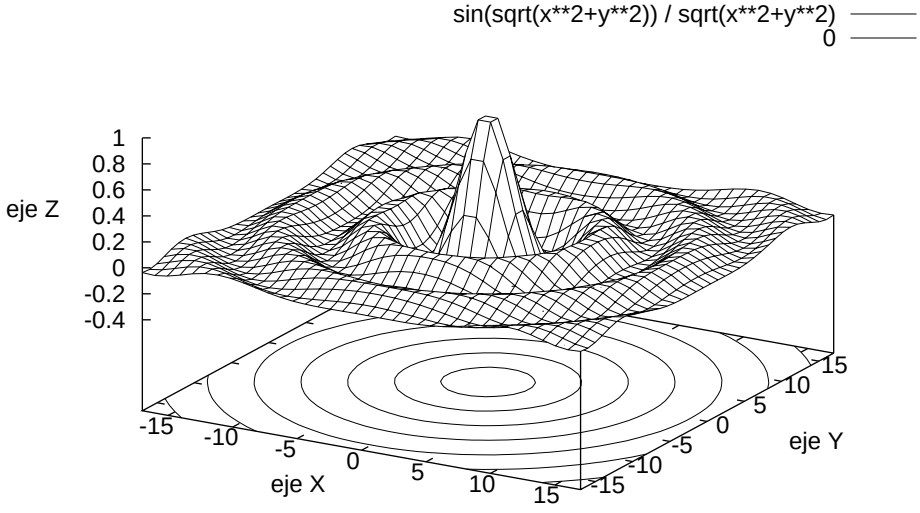
Practicaremos ahora con el comando **set isosamples** que es el equivalente al **set samples** pero para el comando **splot**. Para ello realizamos un gráfico 3-d algo más complicado:



podemos ver que la función dibujada ahora tiene una variación mayor y las ondulaciones, así como las curvas de nivel en la base, tienen peor definición. El comando utilizado ha sido:

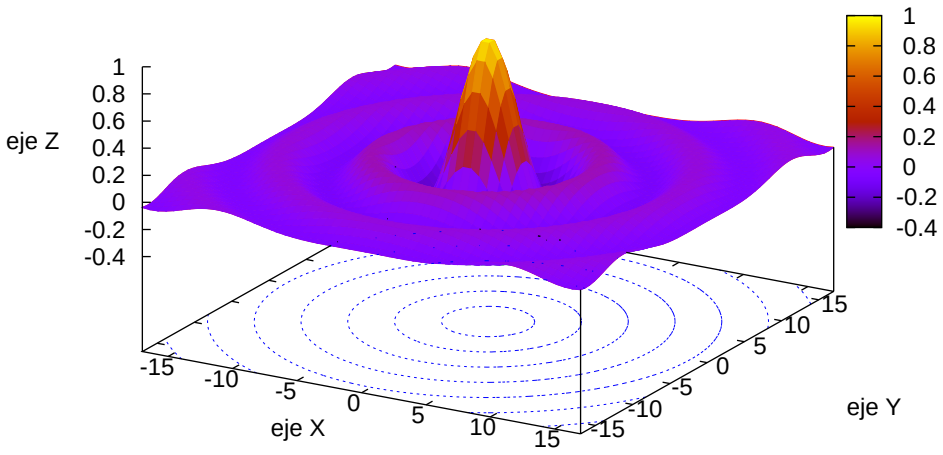
```
gnuplot> set isosamples 21
gnuplot> splot [-17:17] [-17:17] sin(sqrt(x**2+y**2)) / sqrt(x**2+y**2)
```

La resolución puede mejorarse cambiando el número de puntos (cuadrículas) utilizadas en la realización del gráfico, por ejemplo, si hacemos **set isosamples 36** y luego **replot** obtenemos un nuevo gráfico con mayor definición. También hemos ocultado la parte de la gráfica que se encuentra detrás, para ello hemos utilizado el comando **set hidden3d**:



es conveniente practicar con diferentes **set isosamples N** seguidos de **replot** y observar los resultados en pantalla.

También es posible representar las superficies en mapas de colores. Para ello se puede utilizar el algoritmo pm3d, tal y como se muestra en el siguiente ejemplo.

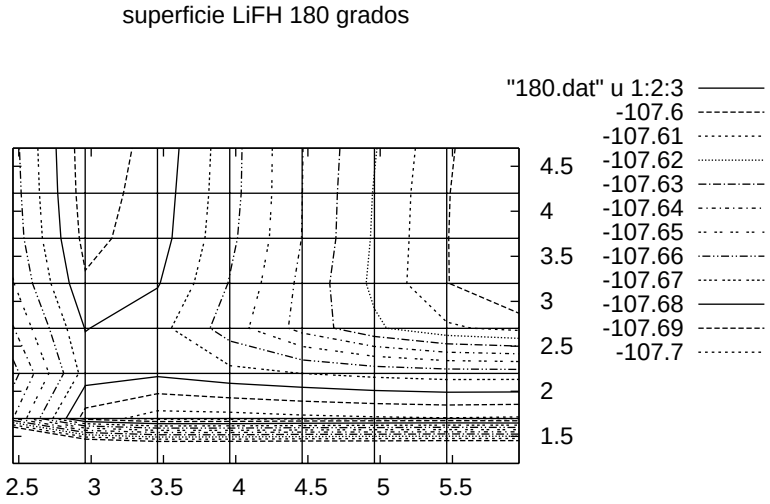


que se obtiene con

```
gnuplot> set pm3d
gnuplot> set palette
gnuplot> replot
```

Representando datos

En ocasiones resulta interesante obtener gráficos 2-d con curvas de nivel. Para esto podemos hacer el gráfico usando el comando 3-d **splot** y estableciendo varias opciones que permitan ver sólo la base y desde arriba (de forma que el resultado sea un plano 2-d). El siguiente gráfico es un ejemplo de esto con un fichero de datos correspondientes a cálculos de energías solución de la ecuación de Schrödinger para la reacción colineal (los tres átomos en línea) Li + FH → LiF + H:

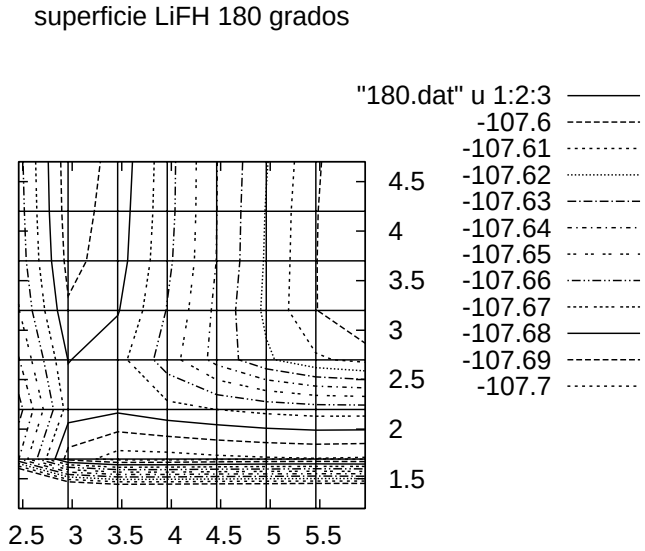


Podemos observar que el resultado no es muy espectacular por varias razones, pero si tenemos en cuenta que el fichero de datos contiene tan sólo 8×8 puntos (representados en la malla del fondo) lo parecerá menos. Los comandos usados para generar el gráfico son:

```
gnuplot> !kate 180.dat                          #edita los datos (8x8 puntos)
gnuplot> reset                                  #anula los anteriores comandos del set
gnuplot> set clabel '%11.5g'                    #Permite poner 5 cifras significativas en la leyenda
gnuplot> unset ztics                            #elimina la escala en el "eje Z"
gnuplot> set xrange [2.46:5.96]                 #selecciona el intervalo en el "eje X"
gnuplot> set yrange [1.20:4.70]                 #selecciona el intervalo en el "eje Y"
gnuplot> set parametric
gnuplot> set view 0, 0, 1, 1                    #vista desde arriba del plano XY
```

```
gnuplot> set contour base
gnuplot> set style data lines                        #dibuja lineas
gnuplot> set title "superficie LiFH 180 grados"
gnuplot> set cntrparam levels incremen -107.702,0.010,-107.600
gnuplot> splot "180.dat" u 1:2:3
```

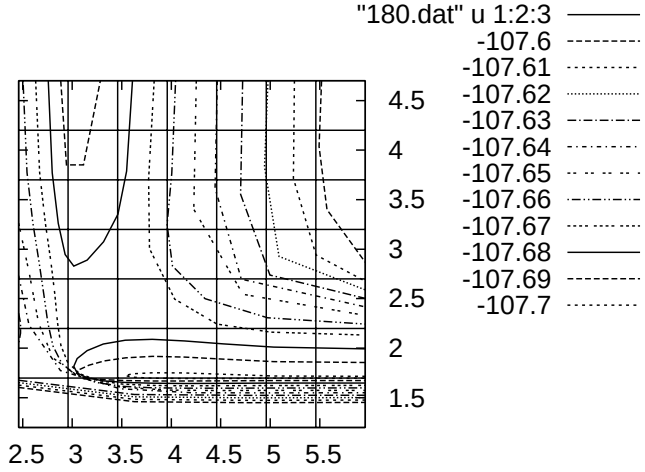
Una de las razones que hacen poco agradable el gráfico anterior es que teniendo una malla cuadrada (8×8) el resultado es rectangular con separaciones muy diferentes en `x` e `y` cuando debían ser iguales. Esto es debido al dispositivo (la pantalla gráfica suele ser de 1024×768 o 800×600 lo que da una relación de 1 a 0.75). Deberá tenerse en cuenta que este efecto será diferente sobre papel. Este efecto puede arreglarse mediante el comando **set size 0.75,1.** o, alternativamente, con **set size square**:



que se ha obtenido haciendo:

```
gnuplot> set size square
gnuplot> replot
```

Otro de los motivos que hacen desagradable el gráfico anterior es el de la falta de datos que hacen que las curvas de nivel sean muy agudas según zonas. Esto puede arreglarse suavizando con splines:



que se ha obtenido haciendo:

```
gnuplot> set cntrparam bspline
gnuplot> set cntrparam points 3
gnuplot> set cntrparam order 10
gnuplot> splot "180.dat" u 1:2:3
```

Por último para conseguir una salida a un fichero gráfico, de tipo “png” o “postscript” (que posteriormente puede incluirse en un texto LyX) de cualquiera de los gráficos anteriores deberá hacerse lo siguiente **después de haber dibujado el gráfico en pantalla**:

```
gnuplot>                                  #... (se supone realizado el gráfico en pantalla)
gnuplot> set terminal png 22              #establece el tipo de terminal a tipo png font 22
gnuplot> set output "fichero.png"         #establece la salida a un fichero
gnuplot> replot                           #redibuja el gráfico (en formato eps y sobre fichero)
gnuplot> set terminal x11                 #vuelve a establecer el tipo de terminal a pantalla X11
```

donde ‘fichero.png’ es el fichero al que ha ido a parar el gráfico (en formato png) y que podremos usar posteriormente (desde LyX, ...). La última instrucción restablece la salida a pantalla, para poder seguir realizando nuevos gráficos. Para ficheros en postscript, en vez de **set terminal png** utilizaremos **set terminal postscript eps 22** y la salida en ese caso será **set output “fichero.eps”**.

El comando fit

A continuación vamos a utilizar el comando **fit**, que permite ajustar un conjunto de puntos (x,y) o (x,y,z) a una función real definida por el usuario, utilizando para ello una implementación del algoritmo de Marquardt-Levenberg para ajustes con parámetros no lineales. Cualquier variable de las definidas en la función se puede utilizar como parámetro en el ajuste.

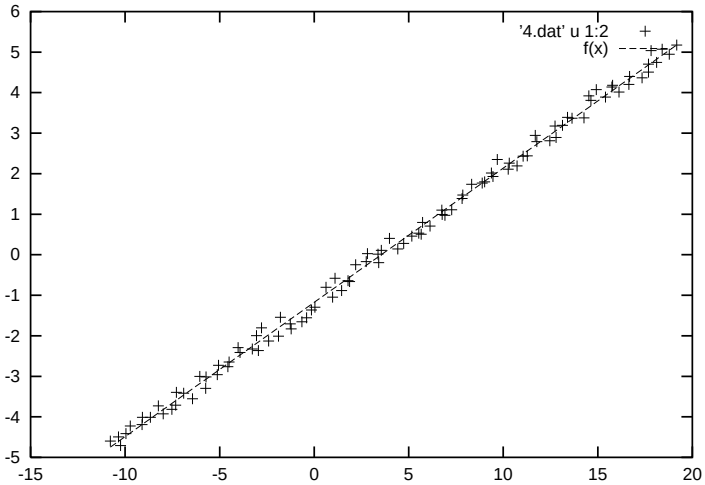
A continuación veremos algunos ejemplos de uso de **fit** para el caso de ajustes en una dimensión. Empezaremos realizando el ajuste de los datos del fichero 4.dat a una recta. Para ello, es conveniente realizar primero la representación gráfica de los datos que queremos ajustar, con el fin de analizar la forma de la dependencia entre las dos variables.

```
gnuplot>
gnuplot> plot ’4.dat’ u 1:2 w p           # representamos los datos
gnuplot> f(x) = a*x + b                   # definimos la función f(x)
gnuplot> fit f(x) ’4.dat’ via a,b         # realizamos el ajuste
gnuplot> replot f(x)                      # dibujamos el ajuste
```

Que muestra el modo más simple de ejecución del comando **fit**:

```
fit función ’datos’ via parámetros
```

El comando **fit** devuelve, en cada iteración, información detallada sobre el estado del ajuste. Esta información se guarda también en el fichero ”fit.log”.

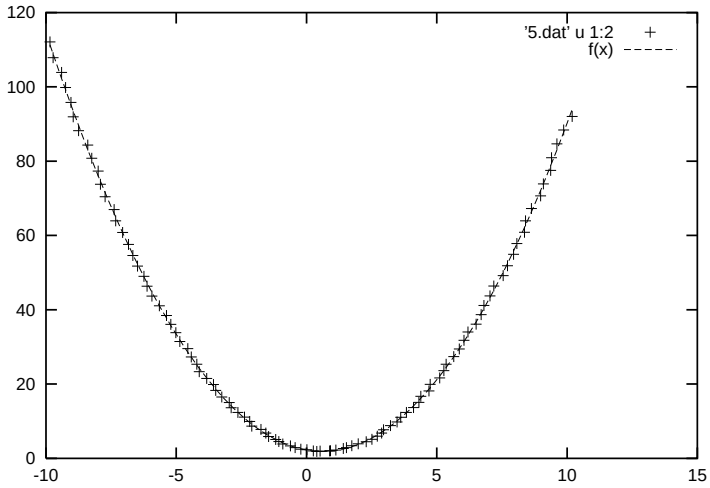


Como se observa en la figura, los datos del fichero 4.dat se ajustan a una línea recta, de ahí que hayamos elegido la función $f(x) = a * x + b$.

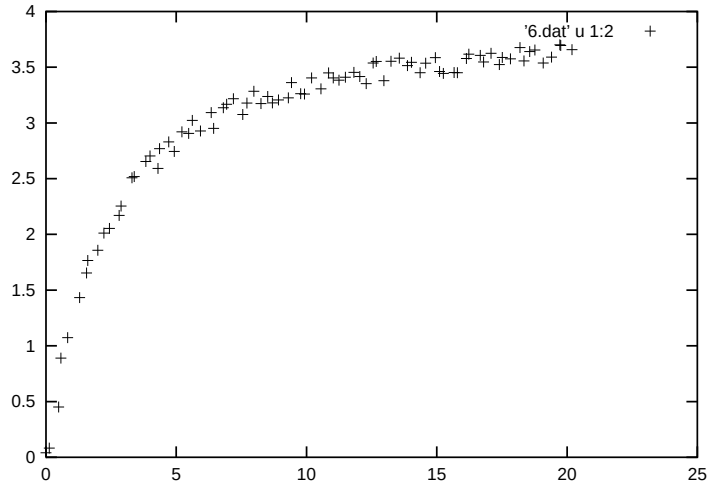
Como se puede ver en el siguiente ejemplo, también se puede ajustar a cualquier otro tipo de función. Así, para el ajuste de los datos del fichero 5.dat, la mejor función es de tipo parabólico:

```
gnuplot>
gnuplot> plot '5.dat' u 1:2 w p            # representamos los datos
gnuplot> g(x) = d*x**2 + e*x + f           # definimos la función g(x)
gnuplot> fit g(x) '5.dat' via d,e,f        # realizamos el ajuste
gnuplot> replot g(x)                       # dibujamos el ajuste
```

tal y como se observa en la siguiente figura:



Como último ejercicio, ajustad los datos del fichero 6.dat (cuya representación se muestra en la siguiente figura), probando diferentes funciones de ajuste.



Los comandos save y load

Por último, se pueden salvar a un fichero los comandos utilizados para realizar una gráfica, con el fin de poder cargarlos en una sesión nueva. Así

```
gnuplot> save 'sesion.gnu'              # salva la sesión en el fichero sesion.gnu
```

salva la sesión, mientras que

```
gnuplot> load 'sesion.gnu'              # carga la sesión del fichero sesion.gnu
```

carga la misma sesión (previamente guardada).