

## RESUMEN DEL PROGRAMA GNUPLOT

Gnuplot es un programa de representación grafica de funciones y superficies, tanto definidas a través de sus expresiones analíticas, como de un conjunto de datos o puntos del plano o del espacio. Permite dibujar contornos de funciones, usar diversos sistemas de coordenadas y manipular ficheros de datos con estructuras diversas. Permite, así mismo visualizar los gráficos en la pantalla del ordenador, imprimirlos en la impresora, o crear ficheros especiales para volcarlos después a una impresora o un plotter o para usarlos desde un procesador de textos como el  $\text{T}_{\text{E}}\text{X}$  o  $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ .

Este documento es una breve introducción al programa gnuplot y describe los comandos más útiles y frecuentemente utilizados. En ningún caso pretende ser un manual de referencia que describa todas las opciones y al que deberá consultar el usuario cuando necesite más información.

Cuando el usuario ejecuta el comando **gnuplot** desde una ventana de comandos aparece el prompt

```
gnuplot>
```

podemos entonces empezar a ejecutar comandos propios del gnuplot de forma interactiva. Tambien se puede escribir **gnuplot** seguido del nombre de un fichero, por ejemplo:

```
gnuplot prueba.dem
```

donde el fichero **prueba.dem** contiene comandos propios del gnuplot. El mismo efecto se consigue si, despues de haber ejecutado **gnuplot**, se usa el comando **load** para cargar el fichero de comandos **prueba.dem**, es decir,

```
gnuplot> load 'prueba.dem'
```

La sintaxis general para representar graficas 2-d de funciones de una variable es:

```
plot {rangos} {<funcion> | {"<fichero-de-datos>" {using ...}}}
      {titulo} {estilo} {, <funcion> {titulo} {estilo}...}
```

donde se pone el nombre de la < función> que queremos representar o el nombre del < fichero-de-datos> , entre comillas (simples o dobles), con los datos numéricos de las coordenadas a representar.

por ejemplo,

```
plot sin(x)
```

producirá el gráfico de la Figura 1, mientras que

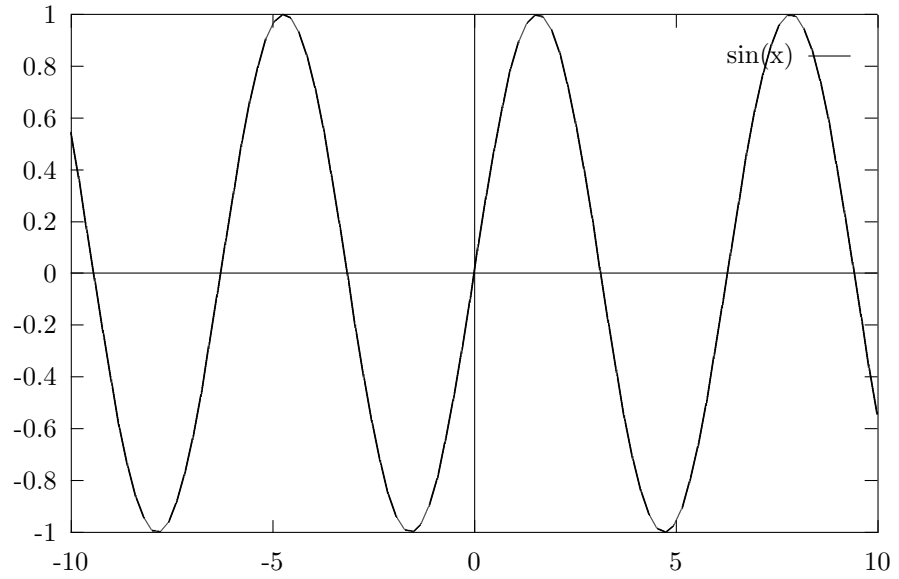


Figure 1: Un ejemplo sencillo:  $y = \sin(x)$

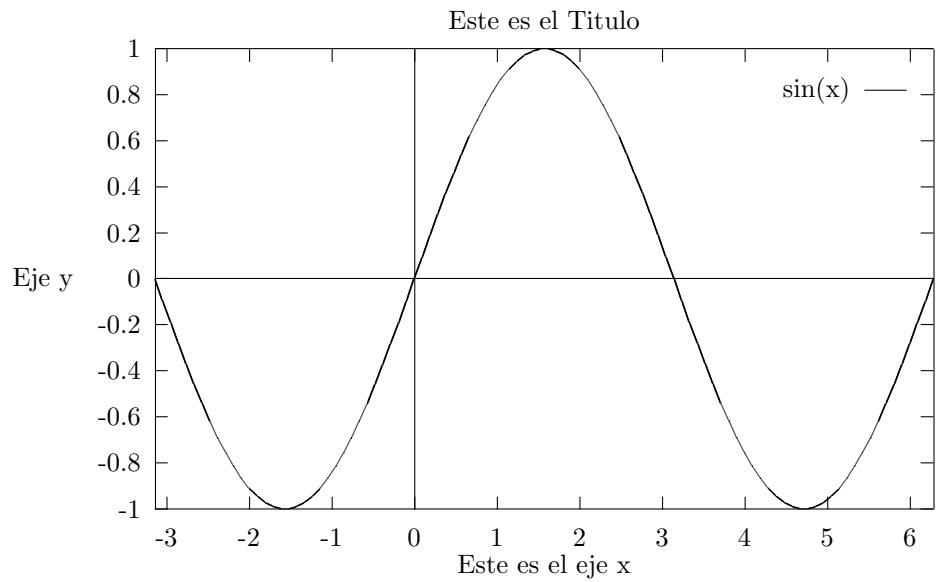


Figure 2: Un ejemplo mas completo:  $y = \sin(x)$

```

set xlabel "Este es el eje x"
set ylabel "Eje y"
set title "Este es el Titulo"
plot [-pi:2*pi] sin(x)

```

produce un gráfico más completo como el de la Figura 2 con título, texto en los ejes, y con un dominio definido para la variable x.

Análogamente para superficies 3-D la sintaxis general del comando **splot** es:

```

splot {ranges} {<function> | {"<datafile>" {index i} {using ...}}}
      {title} {style} {, <function> {title} {style}...}

```

con lo que

```
splot x * y
```

pruducirá el gráfico de la Figura 3.

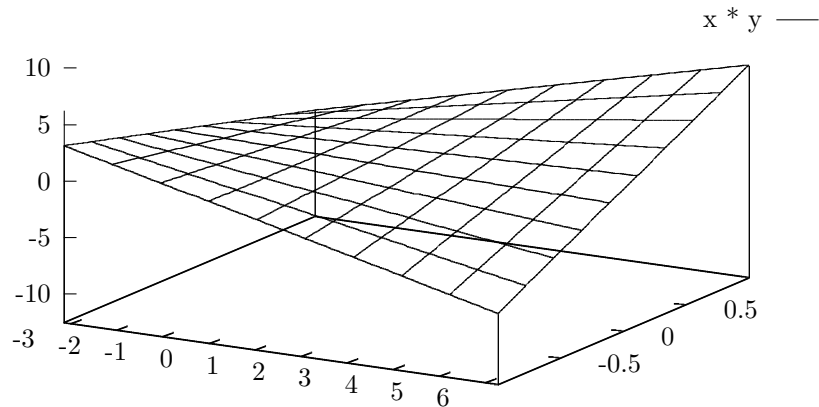


Figure 3: Un ejemplo de superficie:  $z = x * y$

Cuando se quiere representar datos numéricos de un fichero de datos, tenemos varias posibilidades. Supongamos que tenemos un fichero **data1.dat** con datos numéricos de manera que en cada línea aparecen 3 valores que representan la **x**, la **y1** y la **y2** de dos gráficas 2-D de dos funciones. Y supongamos que quiero representar sólo la grafica de los puntos (**x,y1**) entonces basta que ponga

```
plot 'data1.dat'
```

Si quiero que la gráfica no represente los puntos aislados, sino unidos con líneas añadiré *with lines*. Si deseamos representar la segunda función  $(x,y_2)$  pondremos *using 1:3* para indicar las columnas 1 y 3 como valores de  $x$  y de  $y$ . Así podemos obtener la gráfica de la Figura 4 con los siguientes comandos:

```
plot 'data1.dat' using 1:2 with lines, 'data1.dat' using 1:3 with lines
```

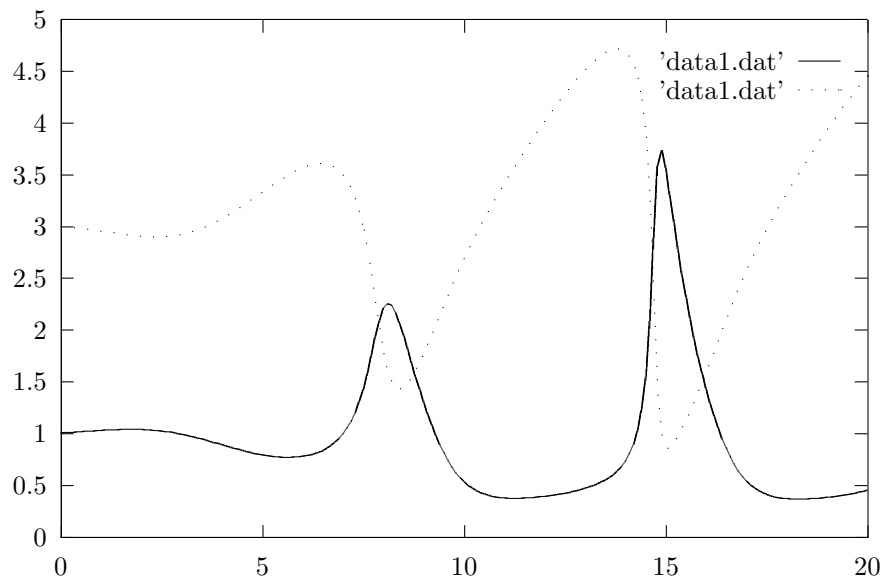


Figure 4: Grafica de los datos  $x,y_1,y_2$  del fichero `data1.dat`

Obsérvese que los datos numéricos deben tener formato entero o real, como 0.25, 2.5 o 0.25E01, pero no se admite el formato "D", como en 0.25D01. Una línea en blanco en un fichero de datos hace que nos movamos al punto siguiente sin dibujar la línea.

Se pueden representar contornos de funciones con el comando

```
set contour{base | surface | both}
```

donde se indica si se quiere representar los contornos de la superficie o de la base o ambos, y con muchas opciones y posibilidades de escalar las variables con escala logarítmica, definir el número de líneas de la malla o de contornos a dibujar, así como el punto de vista desde donde se mira a la superficie o a los contornos. Así por ejemplo con

```
set view 20,340,1,2
set cntrparam levels discrete 1,3,10,30,100
set contour
set nosurface
plot [-1.5:1.5] [-0.5:1.5] (1-x)**2 + 100*(y - x**2)**2
```

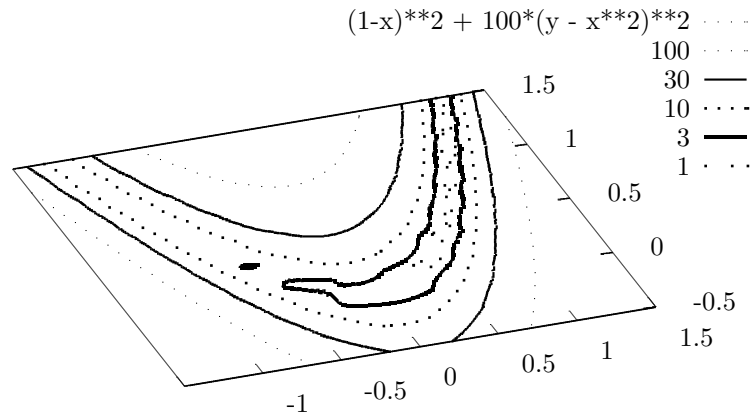


Figure 5: Contornos de la función de Rosenbrock

se obtienen los contornos de la función de Rosenbrock de la Figura 5.

Para imprimir un dibujo generado por gnuplot se debe usar el comando **lasergnu** seguido del comando del gnuplot que genera el gráfico entre comillas, por ejemplo con

```
lasergnu 'plot sin(x)'
```

se imprimiría el gráfico de la Figura 1. La opción **-f**, como en

```
lasergnu -f prueba.dem
```

imprime los dibujos que se generarían con los comandos gnuplot del fichero de datos **prueba.dem**.

Finalmente, volvemos a recordar que se pueden dibujar diagramas de barras, curvas en coordenadas polares y paramétricas, superficies en coordenadas cilíndricas o esféricas, poner etiquetas, marcas en los ejes, modificar el tamaño de la figura, modificar los colores y tipos de líneas o salvar una sesión de trabajo en un fichero para continuar en otro momento.

El comando **help** da ayuda interactiva sobre algunos de los comandos del gnuplot. Y el manual impreso describe con detalle todas las posibilidades y opciones de los comandos. Algunos ejemplos y ficheros de demostración se encuentran en el directorio

```
/usr/local/doc/gnuplot
```