Apuntes de LaTeX

Capítulo 9-2: Dibujando con LaTeX

(Más cosas sobre **PSTricks**)

1 Nodos y conexiones

Cargando el paquete `pst-node`, podemos conectar información y poner etiquetas dentro de una documento, sin conocer la posición exacta de los objetos a conectar. Se pueden definir tres tipos de objetos:

Definiciones de nodos Para asignar un nombre y una forma a un objeto

Conexiones entre nodos Para conectar dos nodos, previamente identificados por sus nombres

Etiquetas de conexiones Para fijar etiquetas a las conexiones

1.1 Definición de nodos

Para declarar que un objeto LaTeX es un nodo se puede utilizar:

`\rnode[Posición]{NombreNodo}{Objeto}`

donde el argumento optativo `Posición` tiene la misma sintaxis que para el comando `\rput`; se utiliza esta información para determinar, más adelante, el punto desde donde saldrán las conexiones. `NombreNodo` sirve para referirnos al nodo (el nombre debe contener sólo letras y numeros, comenzando por una letra) y `Objeto` es el elemento LaTeX correspondiente a ése nodo. Para utilizar nodos en un entorno `pspicture`, basta con declararlos dentro del argumento de un comando `\rput`

Existen otras variantes para crear nodos con formas predefinidas:

`\pnode(x,y){NombreNodo}` → Crea un nodo 0-dimensional (punto) en las coordenadas (x,y).

`\cnode[Parámetros](x,y){R}{NombreNodo}` → Dibuja un círculo en las coordenadas (x,y), de radio `R`, y le asigna un nombre de nodo.

`\circlenode[Parámetros]{NombreNodo}{Objeto}` → Encierra `Objeto` en un círculo y le asigna un nombre de nodo.

`\ovalnode[Parámetros]{NombreNodo}{Objeto}` → Encierra `Objeto` en un óvalo y le asigna un nombre de nodo

1.2 Conexiones

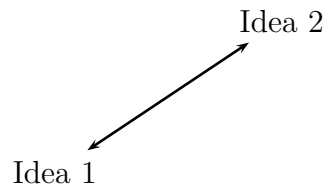
Para conectar dos nodos, existe una variedad de comandos, que se diferencia en el modo en el que se establece la conexión (recta, curva, zig-zag, etc..). Todos ellos comienzan por "nc" y tienen la misma sintaxis:

`\ncTipodeConexion[Parámetros]{TipoDeFlecha}{NodoA}{NodoB}`

donde en `Parámetros` podemos especificar diversas propiedades de la línea de conexión (grosor, color, tipo, etc...), y y en `TipoDeFlecha` si queremos puntas de flecha. `NodoA` y `NodoB` son los nombres de los nodos que queremos conectar. Veamos ahora con ejemplos gráficos sencillos los distintos tipos de conexión:

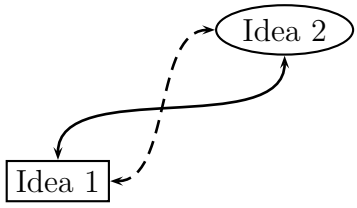
- `\ncline`

```
\begin{pspicture}(4,2)
\rput(1,0){\rnode{A}{Idea 1}}
\rput(4,2){\rnode{B}{Idea 2}}
\ncline[nodesep=4pt,%
linewidth=1pt]{<->}{A}{B}
% nodesep es la separación entre
% un nodo y el final de la línea
% de conexión
\end{pspicture}
```



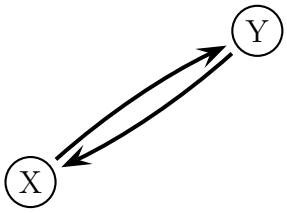
- `\nccurve` Los parámetros `angleA` y `angleB` determinan la orientación con la que se establecen las conexiones

```
\begin{pspicture}(4,2)
\rput(1,0){\rnode{A}%
{\psframebox{Idea 1}}}
\rput(4,2){\ovalnode{B}{Idea 2}}
\nccurve[linewidth=1pt,angleA=90,%
angleB=-90]{<->}{A}{B}
\nccurve[linewidth=1pt,%
linestyle=dashed,
angleA=0,angleB=180]{<->}{A}{B}
\end{pspicture}
```



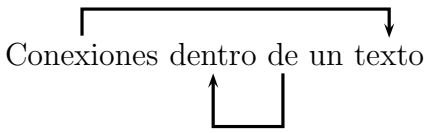
- \ncarc

```
\begin{pspicture}(4,2)
\rput(1,0){\circlenode{A}{X}}
\rput(4,2){\circlenode{B}{Y}}
\psset{nodesep=3pt,%
linewidth=1.5pt,arrowsize=8pt}
\ncarc{->}{A}{B}
\ncarc{->}{B}{A}
\end{pspicture}
```



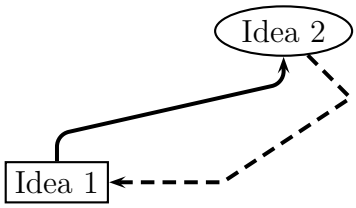
- \ncbar angle controla los ángulos a los que salen las conexiones, y arm la longitud del brazo de la conexión

```
\rnode{A}{Conexiones}
\rnode{B}{dentro}
\rnode{C}{de} un \rnode{D}{texto}
\ncbar[linewidth=1.2pt,nodesep=3pt,
angle=90]{->}{A}{D}
\ncbar[linewidth=1.2pt,nodesep=3pt,
angle=-90,arm=20pt]{<-}{B}{C}
```



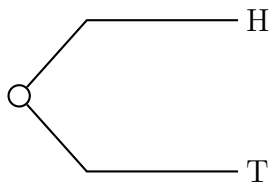
- \ncdiag (nótese en en ejemplo como se pueden modificar los parámetros angleA, angleB, armA y armB

```
\begin{pspicture}(4,2)
\rput(1,0){\rnode{A}%
{\psframebox{Idea 1}}}
\rput(4,2){\ovalnode{B}{Idea 2}}
\ncdiag[linewidth=1.5pt,angleA=90,
angleB=-90,linearc=2mm]{->}{A}{B}
\ncdiag[linewidth=1.5pt,%
linestyle=dashed,
angleA=0,angleB=-45,
armA=1.5,armB=0.8]{<-}{A}{B}
\end{pspicture}
```



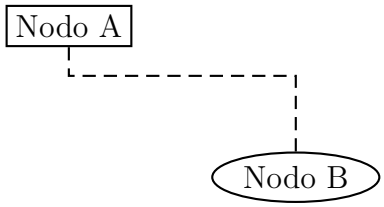
- `\ncdiagg` Similar al anterior, pero con armB=0 (sólo dos brazos)

```
\columnbreak \vspace*{0.1cm}
\cnode(0,0){4pt}{a}
\rput[l](3,1){\rnode{b}{H}}
\rput[l](3,-1){\rnode{c}{T}}
\ncdiagg[angleA=180,armA=2cm,
nodesepA=3pt]{b}{a}
\ncdiagg[angleA=180,armA=2cm,
nodesepA=3pt]{c}{a}
```



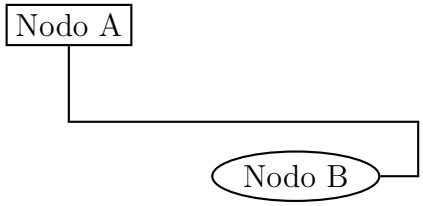
- `\ncangle` Tres segmentos en ángulo recto; se pueden especificar los ángulos de unión `angleA` y `angleB`, así como el brazo `armB`

```
\rput(1,0){\rnode{A}%
{\psframebox{Nodo A}}}
\rput(4,-2){\ovalnode{B}{Nodo B}}
\ncangle[angleA=-90,angleB=90,%
armB=1cm,linestyle=dashed]{A}{B}
```



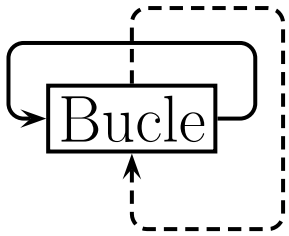
- `\ncangles` Cuatro segmentos en ángulo recto; se pueden especificar los ángulos de unión `angleA` y `angleB`, así como los brazos `armA` y `armB`

```
\rput(1,0){\rnode{A}%
{\psframebox{Nodo A}}}
\rput(4,-2){\ovalnode{B}{Nodo B}}
\ncangles[angleA=-90,angleB=0,%
armA=1cm,armB=0.5cm]{A}{B}
```



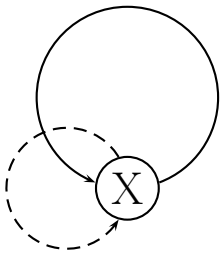
- `\ncloop` Se pueden especificar los ángulos angleA, angleB (con una diferencia de ± 180 grados) los brazos `armA`, `armB`, y la distancia del bucle `loopsize`; nótese como también se puede especificar `arm`, que equivale a hacer `armA`=`armB`

```
\psset{linewidth=1.5pt,
arrowsize=7pt}
\rnode{a}{\psframebox{\huge Bucle}}
\ncloop[angleA=0,
angleB=180,loopsize=1,
arm=.5cm,linearc=.2]{->}{a}{a}
\ncloop[angleA=90,angleB=-90,
loopsize=2,arm=1cm,linearc=.2,
linestyle=dashed]{->}{a}{a}
```



- `\nccircle` Coloca un circulo alrededor del nodo, se puede cambiar el ángulo de inicio `angleA`; su sintasis es: `\nccircle{NodoA}{Radio}`

```
\circlenode{A}{\Large X}
\nccircle{->}{A}{1.2cm}
\nccircle[angleA=90,%
linestyle=dashed]{->}{A}{0.8cm}
```



1.3 Etiquetas

Para poner etiquetas a las conexiones, existen varias posibilidades básicas:

`\naput[Parámetros]{Objeto}` ⟶ Coloca `Objeto` **encima** de la conexión

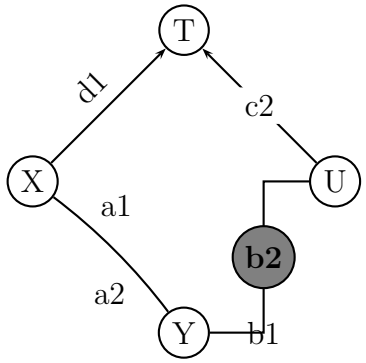
`\nbput[Parámetros]{Objeto}` ⟶ Coloca `Objeto` **debajo** de la conexión

`\ncput[Parámetros]{Objeto}` ⟶ Coloca `Objeto` **en medio** de la conexión

`\ncput*[Parámetros]{Objeto}` ⟶ Coloca `Objeto` **en medio** de la conexión, partiendo la línea de conexión

Todos estos comandos deben escribirse *inmediatamente a continuación* del que define la conexión (`\ncline`, `\nccurve`, etc...). Podemos ajustar mediante parámetros la rotación de la etiqueta (`nrot`=Ángulo) y la posición de la misma a lo largo de la lista de conexión (`npos`=Número); ésta se determina a través de un número que ha de ser una fracción *del número total de segmentos que posea la conexión*; así, para cuatro segmentos (`\ncangles`) puede variar entre 0 y 4; para 3 segmentos (`\ncangle`, por ejemplo) entre 0 y 3, etc...

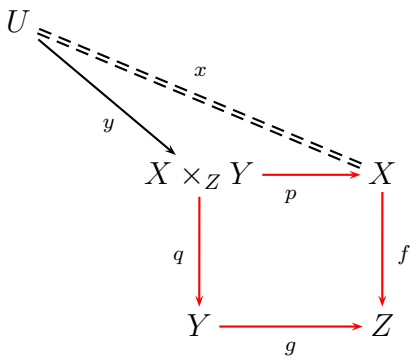
```
\begin{pspicture}(5,4)
\rput(1,2){\circlenode{A}{X}}
\rput(3,0){\circlenode{B}{Y}}
\rput(3,4){\circlenode{C}{T}}
\rput(5,2){\circlenode{D}{U}}
\ncarc{A}{B}\naput[npos=0.3]{a1}
\nbput[npos=0.7]{a2}
\ncangle[armB=6mm,angleA=0,
angleB=180]{B}{D}
\ncput[npos=1]{b1}\ncput[npos=1.5]
{\pscirclebox[fillstyle=solid,
fillcolor=gray]{\bfseries b2}}
\ncline{->}{D}{C}\ncput*{c2}
\ncline{->}{A}{C}\naput[nrot=45]{d1}
\end{pspicture}
```



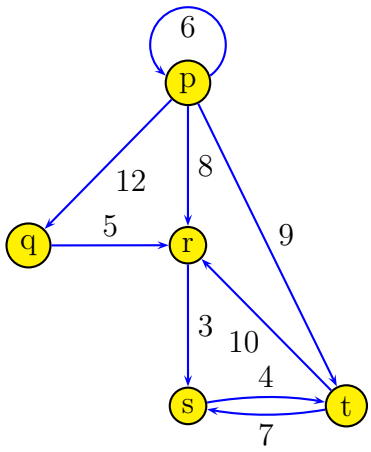
1.4 El entorno psmatrix

El paquete pst-node proporciona, además, el entorno `psmatrix`, el cual posee la misma estructura que el entorno matemático `matrix`. En éste entorno, cada celda define automáticamente un nodo etiquetado por sus coordenadas de posición en la matrix. Podemos utilizar comandos de conexiones empleando tales coordenadas de posición, así como añadir etiquetas a las conexiones, con la sintaxis ^ / _ / > / < para etiquetas arriba/abajo/derecha/izquierda, respectivamente. La separación entre filas y entre columnas se cambia con los parámetros `rowsep` y `colsep`; también pueden enmarcarse los nodos con el parámetro `mnode` (que puede tomar los valores circle, oval, tri, dia, ...). Veánse los siguientes ejemplos (notando cómo, en el primer caso, al tener formato matemático algunos de los elementos de matriz, se debe abrir y el modo matemático antes de comenzar `psmatrix`):

```
$
\begin{psmatrix}
[colsep=1.5cm,rowsep=1.5cm]
U & & \\ & X\times_Z Y & X \\
 & Y & Z % (Definición de la matriz)
 \psset{arrows=->,nodesep=1mm}
\everypsbox{\scriptstyle}
% (para hacer etiquetas pequeñas)
\ncline{1,1}{2,2}_{y}
\ncline[linestyle=dashed,
doubleline=true]{-}{1,1}{2,3}^{x}
\ncline[linecolor=red]{2,2}{3,2}<{q}
\ncline[linecolor=red]{2,2}{2,3}_{p}
\ncline[linecolor=red]{2,3}{3,3}>{f}
\ncline[linecolor=red]{3,2}{3,3}_{g}
\end{psmatrix}
$
```




```
\begin{psmatrix}[fillstyle=solid,
fillcolor=yellow,mnode=circle,
colsep=1.5] & p \\ q & r \\ & s & t
\end{psmatrix}
\psset{linecolor=blue,arrows=->,
labelsep=1mm,shortput=nab}
\nccircle{1,2}{0.5cm}^{6}
\ncline{1,2}{2,1}^{12}
\ncline{1,2}{2,2}^{8}
\ncline{1,2}{3,3}^{9}
\ncline{2,1}{2,2}^{5}
\ncline{2,2}{3,2}^{3}
\ncline{3,3}{2,2}^{10}
\ncarc[arcangle=10]{3,3}{3,2}^{7}
\ncarc[arcangle=10]{3,2}{3,3}^{4}
```



2 Dibujando funciones

Hemos visto anteriormente el comando `\dataplot`, útil para representar conjuntos de datos. Por otra parte, el paquete pst-plot proporciona dos comandos para representar gráficamente funciones matemáticas dependientes de una variable:

`\psplot[Parámetros]{xmin}{xmax}{función}` $\longrightarrow$ Para representar funciones de la forma $y = f(x)$

`\parametricplot[Parámetros]{tmin}{tmax}{función}` $\longrightarrow$ Para representar funciones en forma paramétrica: $(x(t), y(t))$

En ambos casos, se deben especificar los valores máximo y mínimo para los que se calcula el valor de la función. Podemos cambiar el número de puntos en los que se evaluará la función a través del parámetro `plotpoints` (por defecto, igual a 50).

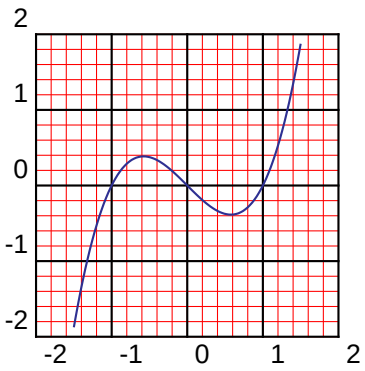
La forma de la función debe expresarse en lo que se conoce como "Reverse Polish Notation" (RPN). Así, por ejemplo, para representar la función $y = x^3 - x$ el valor del argumento `función` debe ser igual a: "x 3 exp x sub" (siempre la incógnita debe escribirse como "x", en el caso de funciones explícitas, y "t" para funciones en forma paramétrica). Afortunadamente, dado que esta notación es bastante compleja, LaTeX proporciona una herramienta de traducción; cargando el paquete pst-infixplot podemos cambiar de la notación RPN a la notación normal (infix). El procedimiento es el siguiente:

1. Se utiliza la instrucción `\infixtoRPN{función}` para traducir la forma de la función en notación normal a la notación RPN. La sintaxis RPN queda almacenada en la variable "`\RPN`"

2. Se introduce la variable `\RPN` en el argumento del comando `\psplot` (respectivamente `\parametricplot`)

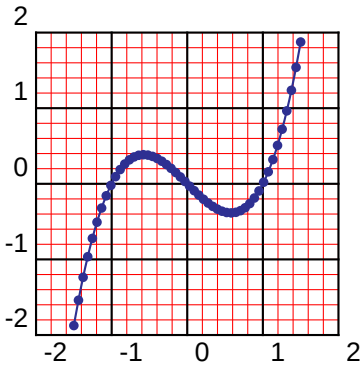
Veamos un ejemplo; para representar la función $y = x^3 - x$ en el intervalo $-1.5 \leq x \leq 1.5$, habría que, en principio, escribir:

```
\begin{pspicture}(-2,-2)(2,2)
\psgrid[gridcolor=red,%
subgridcolor=red]
\psplot[plotstyle=curve,%
linecolor=Blue]%
{-1.5}{1.5}{x 3 exp x sub}
\end{pspicture}
```



ó, alternativamente, usando `\infixtoRPN{función}`, obtenemos el mismo resultado con:

```
\begin{pspicture}(-2,-2)(2,2)
\psgrid[gridcolor=black,%
subgridcolor=red]
\infixtoRPN{(x^3)-x}
\psplot[plotstyle=curve,%
linecolor=Blue,
showpoints=true]%
{-1.5}{1.5}{\RPN}
\end{pspicture}
```

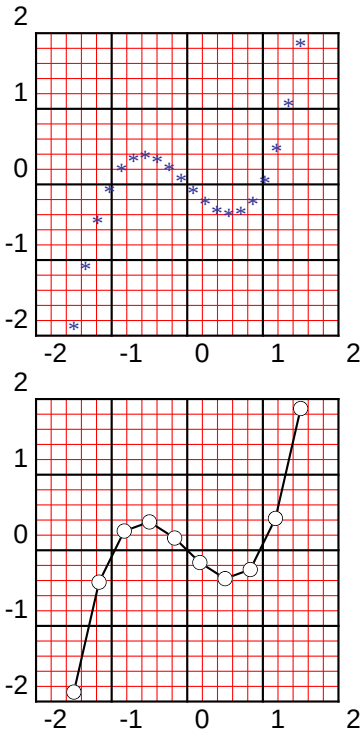


(nótese cómo en este último caso, hemos incluido el parámetro `showpoints`, que resalta los puntos utilizados para calcular la función).

El parámetro `plotstyle` controla la forma de representar la curva; podemos elegir entre `dots`, `line`, `curve`; veamos dos ejemplos para `dots` (donde se cambia el tipo de punto con `dotstyle=asterisk`) y `line`:

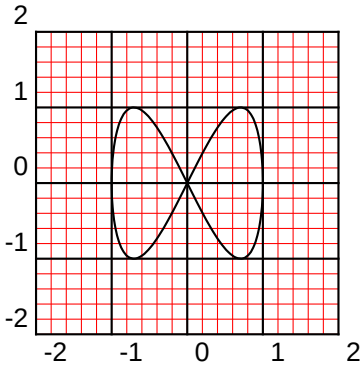
```
\begin{pspicture}(-2,-2)(2,2)
\psgrid[gridcolor=black,%
subgridcolor=red]
\infixtoRPN{(x^3)-x}
\psplot[plotstyle=dots,%
linecolor=Blue, plotpoints=20,
dotstyle=asterisk]%
{-1.5}{1.5}{\RPN}
\end{pspicture}

\begin{pspicture}(-2,-2)(2,2)
\psgrid[gridcolor=black,%
subgridcolor=red]
\infixtoRPN{(x^3)-x}
\psplot[plotstyle=line,%
plotpoints=10, showpoints=true,
dotstyle=o, dotsize=2mm]%
{-1.5}{1.5}{\RPN}
\end{pspicture}
```



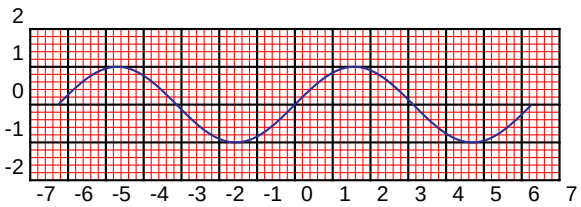
El siguiente ejemplo ilustra la forma de representar una función paramétrica:

```
\begin{pspicture}(-2,-2)(2,2)
\psgrid[gridcolor=black,%
subgridcolor=red]
\infixtoRPN{sin(t),sin(2*t)}
\parametricplot[plotstyle=curve]%
{0}{360}{\RPN}
\end{pspicture}
```



Nótese como las funciones trigonométricas trabajan por defecto **en grados**; si queremos representarlas en radianes, se debe escalar la coordenada x por un factor de $\frac{\pi}{180} = 0.0174$:

```
{\psset{unit=0.5}
\begin{pspicture}(-7,-2)(7,2)
\psgrid[gridcolor=black,%
subgridcolor=red,%
gridlabels=8pt]
\infixtoRPN{sin(x)}
\psplot[xunit=0.0174,%
plotstyle=curve,%
linecolor=Blue]%
{-360}{360}{\RPN}
\end{pspicture}}
```



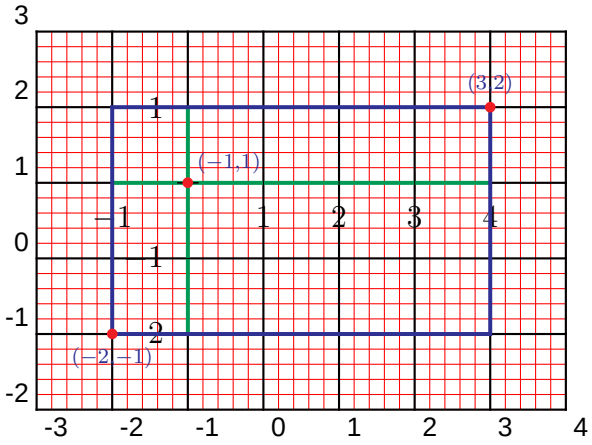
2.1 Ejes de coordenadas

El comando[1]

`\psaxes[Parámetros]{TipoDeFlecha}(x0,y0)(x1,y1)(x2,y2)`

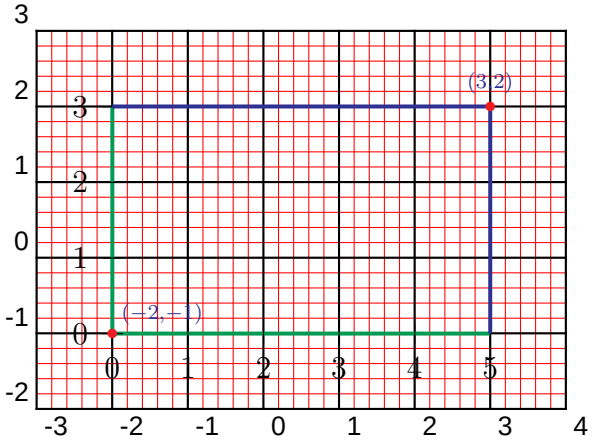
dibuja ejes x e y con el eje x extendiéndose desde x1 a x2, y el eje y desde y1 a y2; los ejes tienen su origen en el punto (x0,y0); por ejemplo:

[1]Nota: Para poder utilizar algunos de los comandos descritos, es importante tener cargado el paquete pstricks-add, que extiende las funcionalidades de PSTricks




```
\psaxes(-1,1)(-2,-1)(3,2)
```

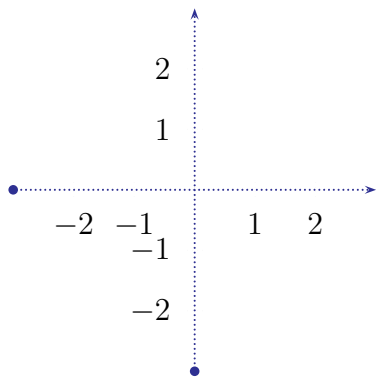
Especificando sólo las coordenadas (x1,y1) y (x2,y2), se obtienen los ejes con origen en la esquina inferior izquierda del rectángulo:




```
\psaxes(-2,-1)(3,2)
```

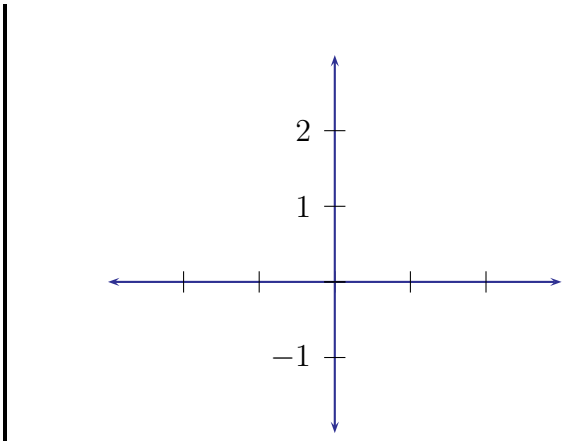
En `Parámetros` y `TipoDeFlecha`, podemos cambiar las características de los ejes coordenados:

```
\psset{unit=0.8cm}
\begin{pspicture}(-3,-3)(3,3)
\psaxes[linecolor=Blue,%
linestyle=dotted,
dotsep=1pt]{*->}%
(0,0)(-3,-3)(3,3)
\end{pspicture}
```



Se puede elegir donde colocar marcas (`ticks`) y etiquetas (`labels`) dando valores a cada uno de estos dos parámetros; los valores posibles son `all` (ambos ejes, opción por defecto), `x`, `y`, `none` (ningún eje); por ejemplo:

```
\begin{pspicture}(-3,-2)(3,3)
\psaxes[linecolor=Blue,%
ticks=all,labels=y]%
{<->}%
(0,0)(-3,-2)(3,3)
\end{pspicture}
```



La posición y tamaño de las marcas se puede especificar a través de los parámetros `tickstyle` y `ticksize`. `tickstyle` puede tomar los valores `full` (marcas a ambos lados del eje), `top` (marcas en el lado opuesto al de las etiquetas) y `bottom` (marcas en el mismo lado que etiquetas).

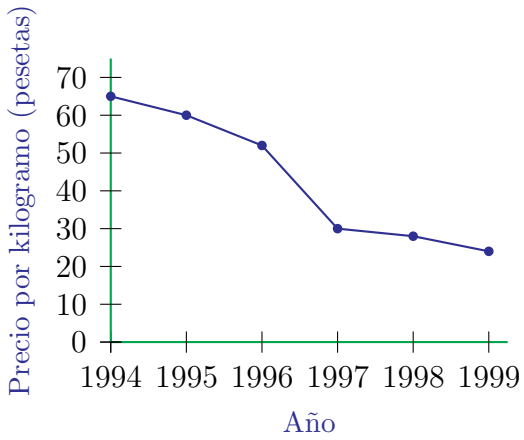
El comando `\psaxes` produce, por defecto, marcas equiespaciadas etiquetadas con enteros consecutivos; ésto puede cambiarse a través de los parámetros mostrados en la tabla siguiente:

Parámetro	Significado	Valor por defecto
`Ox`	Etiqueta en el origen del eje x	0
`Oy`	Etiqueta en el origen del eje y	0
`Dx`	Incremento para etiquetas en el eje x	1
`Dy`	Incremento para etiquetas en el eje y	1
dx	Distancia entre marcas para el eje x	"Dx" × `\psxunit`
dy	Distancia entre marcas para el eje y	"Dy" × `\psyunit`

(`\psxunit` y `\psxunit` son las unidades de longitud
a lo largo de los ejes x e y, respectivamente)

El ejemplo siguiente ilustra cómo modificar estos parámetros para representar gráficamente unos datos:

```
\begin{pspicture}(0,0)(10,7)
\psaxes[linecolor=Green,%
Ox=1994,Oy=0,%
Dx=1,Dy=10,%
dx=2,dy=1]%
(10.5,7.5)
\psline[linecolor=Blue,
showpoints=true]%
(0,6.5)(2,6)(4,5.2)%
(6,3)(8,2.8)(10,2.4)
\uput[d](5.25,-1.5){%
\color{Blue} Año}
\uput[l](-1.5,3.75){%
\rput{90}{%
\color{Blue}%
Precio por kilogramo (pesetas)}}
\end{pspicture}
```



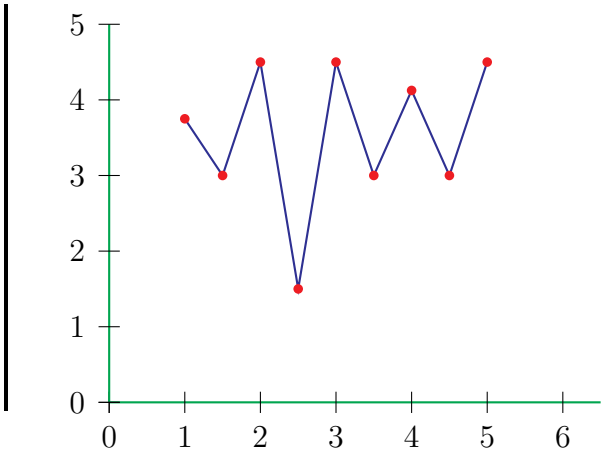
2.2 Manejo de datos externos

Podemos también representar datos guardados en un fichero externo (de forma totalmente análoga al comando `\dataplot`) con el comando `\fileplot[Parámetros]{fichero}`. Los datos pueden ser leídos en forma de pares $x_n y_n$ separados por espacios; por ejemplo, si tenemos el fichero data1.txt, de contenido:

1 3.75
1.5 3.0
2 4.5
2.5 1.5
3 4.5
3.5 3.0
4 4.125
4.5 3.0
5 4.5

podemos representarlos gráficamente de la siguiente forma:

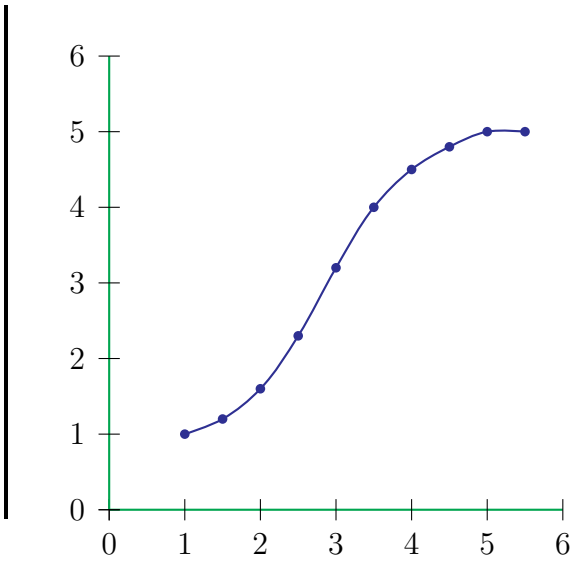
```
\begin{pspicture}(0,0)(6.5,5)
\psaxes[linecolor=Green]%
(6.5,5)
\fileplot[linecolor=Blue]%
{data1.txt}
\fileplot[plotstyle=dots,
linecolor=Red]{data1.txt}
\end{pspicture}
```



En el ejemplo anterior, nótese cómo hemos representado dos veces los datos, la segunda con "`plotstyle=dots`"; la razón de esto está en que el comando `\fileplot` ignora la opción `showpoints`. Este comando también admite menos estilos que `\dataplot`, al no reconocer la opción `curve`. Por tanto, en ciertas situaciones es más útil emplear el comando `\dataplot`. Para este comando, ya comentamos que se deben leer anteriormente los datos con `\readdata`; debemos destacar que, al igual que con `\fileplot`, podemos leer los datos en el mismo formato de parejas de datos separadas por espacios.

Por último, otra alternativa (para listas cortas de datos), es utilizar el comando `\listplot` (que admite las mismas opciones que `\dataplot`). Por ejemplo:

```
\begin{pspicture}(0,0)(5,6)
\psaxes[linecolor=Green]%
(6,6)
\listplot[plotstyle=curve,%
showpoints=true,%
linecolor=Blue]%
{1 1 1.5 1.2 2 1.6 2.5 2.3
3 3.2 3.5 4 4 4.5 4.5 4.8
5 5 5.5 5}
\end{pspicture}
```



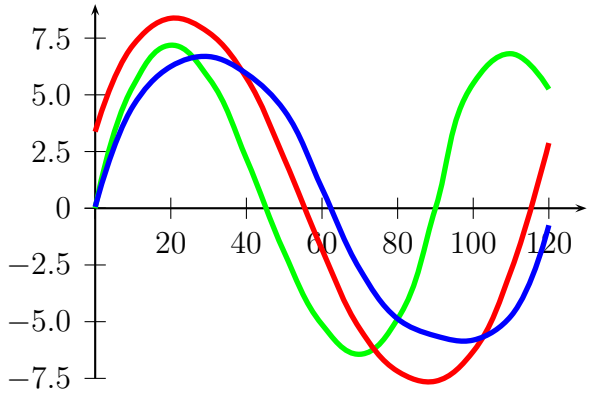
Imaginemos ahora que queremos representar datos de un fichero con varias entradas de datos, de la forma (`x y1 y2 y3`); por ejemplo, tenemos el fichero data2.txt:

0 0 3.375 0.0625
10 5.375 7.1875 4.5
20 7.1875 8.375 6.25

30 5.75 7.75 6.6875
40 2.1875 5.75 5.9375
50 -1.9375 2.1875 4.3125
60 -5.125 -1.8125 0.875
70 -6.4375 -5.3125 -2.6875
80 -4.875 -7.1875 -4.875
90 0 -7.625 -5.625
100 5.5 -6.3125 -5.8125
110 6.8125 -2.75 -4.75
120 5.25 2.875 -0.75

Para procesarlo, pasamos las opciones `plotNo` (número de gráfica) y `plotNoMax` (número total de gráficas, esto es, 3) al comando `listplot`:

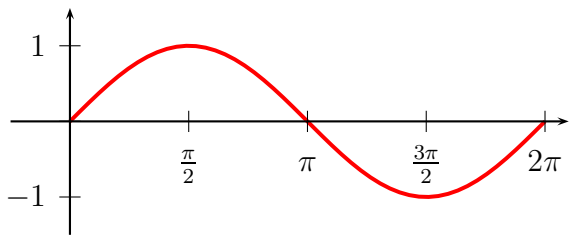
```
\readdata{\Data}{data2.txt}
\psset{xunit=0.05cm,yunit=0.3cm}
\begin{pspicture}(0,-7.5)(130,10)
\psaxes[Dx=20,Dy=2.5]{->}%
(0,0)(0,-7.5)(130,9)
\psset{linewidth=2pt,plotstyle=curve}
\listplot[linecolor=green,%
plotNo=1,plotNoMax=3]{\Data}
\listplot[linecolor=red,%
plotNo=2,plotNoMax=3]{\Data}
\listplot[linecolor=blue,%
plotNo=3,plotNoMax=3]{\Data}
\end{pspicture}
```



2.3 Ejes con unidades trigonométricas

Con la opción `trigLabels=true` en `\psaxes`, podemos poner etiquetas adecuadas a la representación de funciones trigonométricas (π, 2π, etc...). Por ejemplo:

```
\begin{pspicture}(-0.5,-1.25)(7,1.25)
\infixtoRPN{sin(x*180/3.141592654)}
% Fijarse en como cambiamos la
% variable x a radianes!!!
\psplot[linecolor=red,linewidth=%
1.5pt]{0}{6.283185308}{\RPN}
\psaxes[xunit=1.570796327,trigLabels=%
true]{->}(0,0)(-0.5,-1.5)(4.2,1.5)
\end{pspicture}
```



2.4 Funciones en coordenadas polares

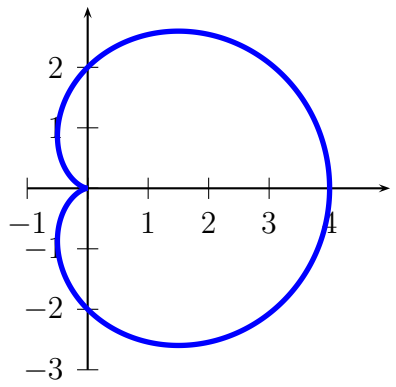
Con la opción `polarplot=true` podemos utilizar `\psplot` para representar funciones en coordenadas polares: [2]

`\psplot[polarplot=true,...]{AnguloIni}{AnguloFin}{r(alpha)}`

Dibujemos por ejemplo una cardioide:

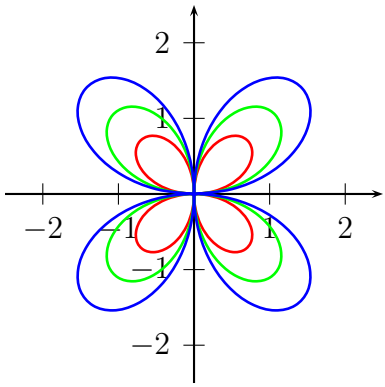
$\rho = 2(1 + \cos(\theta))$

```
\begin{pspicture}(-1,-3)(5,3)
\psaxes{->}(0,0)(-1,-3)(5,3)
\infixtoRPN{2*(1+cos(x))}
\psplot[polarplot=true,%
linewidth=2pt,linecolor=blue,%
plotpoints=500]{0}{360}{\RPN}
\end{pspicture}
```



Otro ejemplo:

```
\begin{pspicture}(-2.5,-2.5)(2.5,2.5)
\psaxes{->}(0,0)(-2.5,-2.5)(2.5,2.5)
\psset{linewidth=.35mm,%
plotstyle=curve,polarplot=true}
\infixtoRPN{2*sin(x)*cos(x)}
\psplot[linecolor=red]{0}{360}{\RPN}
\infixtoRPN{3*sin(x)*cos(x)}
\psplot[linecolor=green]{0}{360}{\RPN}
\infixtoRPN{4*sin(x)*cos(x)}
\psplot[linecolor=blue]{0}{360}{\RPN}
\end{pspicture}
```



2.5 La opción `algebraic`

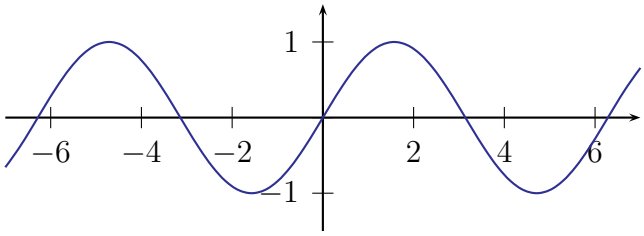
Es útil saber que, aparte del modo de transformar notaciones mediante el comando `\infixtoRPN`, podemos también utilizar la opción `algebraic=true` en `\psplot` para cambiar la forma en que introducimos la forma de `función` en

[2] Al comienzo del capítulo, olvidamos mencionar que, con carácter general, las coordenadas polares se escriben en la forma (r; θ) (esto es, separadas por punto y coma, a diferencia de las cartesianas, que se separan por una coma)

`\psplot[Parámetros]{xmin}{xmax}{función};`

Por ejemplo:

```
\begin{pspicture}(-7,-1.5)(7,1.5)
\psaxes[Dx=2]{->}(0,0)(-7,-1.5)(7,1.5)
\psplot[algebraic=true,%
plotstyle=curve,%
linecolor=Blue]%
{-7}{7}{sin(x)}
\end{pspicture}
```



ADVERTENCIA!!! En el ejemplo anterior puede verse como la opción `algebraic` cambia el argumento de las funciones trigonométricas a RADIANES (que, por defecto, trabajan en grados).

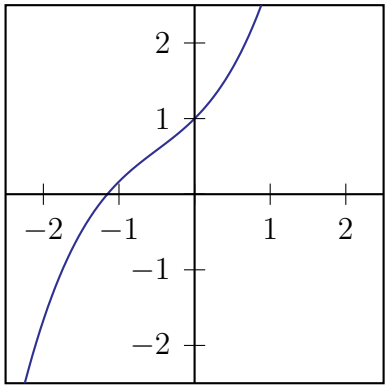
2.6 Funciones `SUM` y `IFTE`

La función SUM se utiliza para sumar iterativamente funciones; tiene la siguiente sintaxis:

`SUM(NombreIndice, indice-comienzo, paso, indice-final, función)`

que significa que `función`, dependiente de NombreIndice, es sumada desde `indice-comienzo` hasta `indice-final`, con incremento `paso`. Esto es útil, por ejemplo, para representar desarrollos en series funcionales; veamos en el ejemplo siguiente cómo representar $f(x) = 1 + x + x^2/2 + x^3/3$

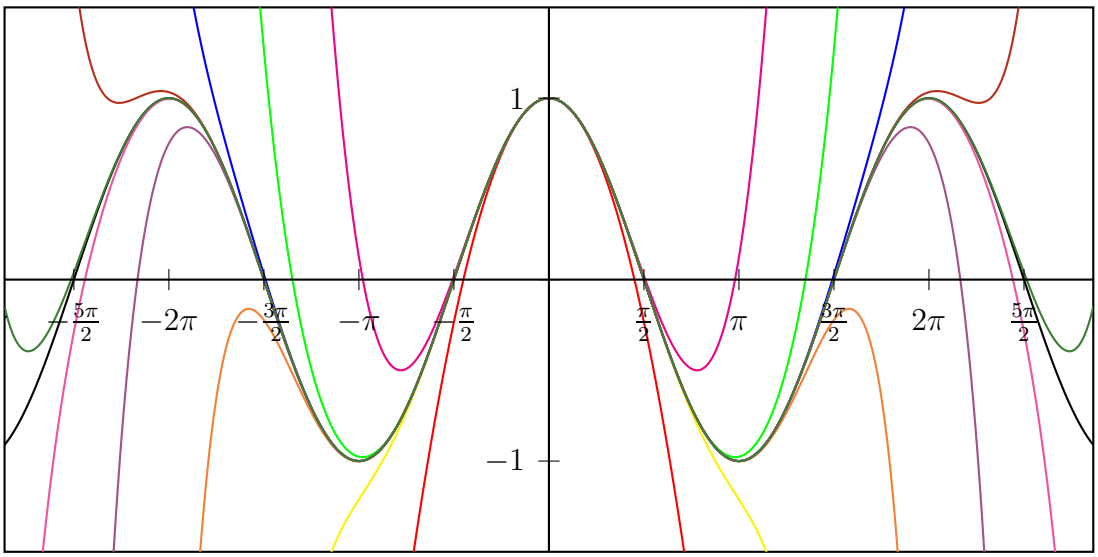
```
\begin{pspicture}(-2.5,-2.5)(2.5,2.5)
\psclip{\psframe(-2.5,-2.5)(2.5,2.5)}
\psplot[algebraic=true,%
plotstyle=curve,%
linecolor=Blue]{-2.5}{2.5}%
{1 + SUM(ind,1,1,3,(x^ind)/ind)}
\psaxes(0,0)(-2.5,-2.5)(2.5,2.5)
\endpsclip
\end{pspicture}
```



(obsérvese cómo se emplea el comando `\psclip{Objeto} ... \endpsclip` para recortar el gráfico; pruébese a eliminarlo y véase qué ocurre con la gráfica de la función)

El siguiente ejemplo enseña cómo integrar este comando con el comando `\multido` para representar el desarrollo de la función $\cos(x) = \sum_{n=0}^{+\infty} \frac{(-1)^n x^{2n}}{n!}$, con n variable. Obsérvese como se define un nuevo comando `\getColor`, en función de un argumento variable, para dar un color distinto a cada gráfica (la definición de este comando se hace utilizando el comando básico de LaTeX `\ifcase ... \fi`):

```
\psset{unit=0.8cm} %% reescalamos globalmente el dibujo
\psset{algebraic=true,plotpoints=500,yunit=3}
\def\getColor#1{\ifcase #1 black\or
red\or magenta\or yellow\or green\or Orange\or blue\or DarkOrchid\or
BrickRed\or Rhodamine\or OliveGreen\fi}
\begin{pspicture}(-9,-1.5)(9,1.5)
 \psclip{\psframe(-9,-1.5)(9,1.5)}
  \psplot{-9}{9}{cos(x)}
   \multido{\n=1+1}{10}{%
   \psplot[linecolor=\getColor{\n}]{-9}{9}{%
    SUM(ijk,0,1,\n,(-1)^ijk*x^(2*ijk)/fact(2*ijk))}}
\psaxes[xunit=1.570796327,trigLabels=true](0,0)(-6,-1.5)(6,1.5)
\endpsclip
\end{pspicture}
```

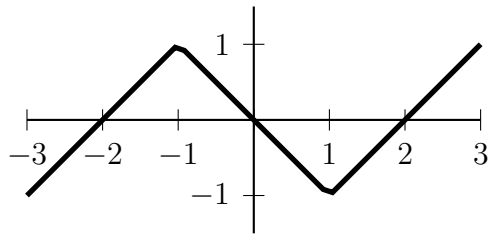


Otra herramienta que merece la pena conocer es la función `IFTE`, que puede utilizarse para definir funciones a trozos:

`IFTE(Condición,Parte-Verdadero,Parte-Falso)`

donde, si se cumple `Condición` para la variable x, se utiliza la función definida en `Parte-Verdadero`, y en caso contrario, la definida en `Parte-Falso`. Véase como encadenándolas, se puede representar una función a trozos:

```
\begin{pspicture}(-3,-1.5)(3,1.5)
\psaxes(0,0)(-3,-1.5)(3,1.5)
\psset{algebraic=true,linewidth=2pt}
\psplot{-3}{3}{IFTE(x<-1,x+2,%
IFTE(x<1,-x,x-2))}
\end{pspicture}
```



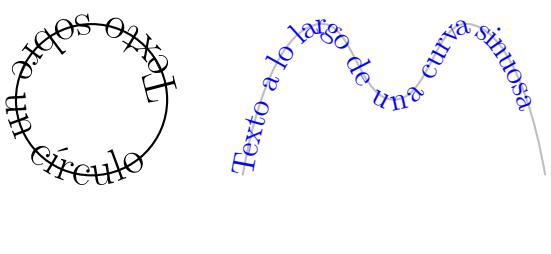
3 Trucos con texto

El comando

`\pstextpath[Posición](x,y){ObjetoGráfico}{Texto}`

permite escribir `Texto` a lo largo de un camino determinado por `ObjetoGráfico`; ambos se superpondrán, aunque el objeto gráfico (recta, curva, etc...) puede eliminarse utilizándolo con la opción `linestyle=none`. Para el argumento optativo `Posición` puede elegirse entre `l` (justificar texto al comienzo del camino), `c` (centrar texto sobre el camino) ó `r` (justificar texto al final del camino). `(x,y)` denota el desplazamiento aplicado al texto respecto al camino, siendo `x` un desplazamiento a lo largo del camino, e `y` un desplazamiento perpendicular. Por ejemplo:

```
\begin{pspicture}(7,2)
\pstextpath{\pscircle(1,1){1cm}}
{\Large Texto sobre un círculo}
\psset{linecolor=lightgray}
\pstextpath{\pscurve(3,0)(4,2)%
(5,1)(6,2)(7,0)}{\blue Texto a lo
largo de una curva sinuosa}
\end{pspicture}
```



Empleando los comandos `\psarc` y `\psarcn` (que dibujan arcos de circunferencia en sentido contrario a las agujas del reloj y en sentido de las agujas del reloj, respectivamente), en lugar de `\pscircle`, podemos cambiar la dirección en la que se escribe el texto (por ejemplo, para producir un logotipo):

```
\begin{pspicture}(-2,-2)(2,2)
\psset{linestyle=none}
\rput(0,0){\includegraphics%
[scale=1.3]{escudo.eps}}
\pstextpath[c]{\psarcn(0,0){1.4}%
{180}{0}}{Facultad de Ciencias}
\pstextpath[c]{\psarc(0,0){1.4}{180}%
{0}}{\small Universidad de Valladolid}
\end{pspicture}
```

El comando

`\pscharpath[Parámetros]{Texto}`

nos permite tratar `Texto` como si fuese una curva cerrada, a la que es posible dar contorno ajustando el parámetro `linestyle`, ó rellenarla ajustando el parámetro `fillstyle`. Por ejemplo:

```
\pscharpath[linestyle=solid,%
fillstyle=solid,%
fillcolor=SkyBlue,%
linewidth=1pt,linecolor=Red]%
{\scalebox{4}{\bfseries PSTricks}}

\pscharpath[linecolor=Yellow,%
linestyle=none,fillstyle=gradient,%
gradbegin=Yellow,gradend=Red,%
gradmidpoint=1,gradangle=5]%
{\scalebox{4}{\sffamily\bfseries%
 PSTricks}}
```

PSTricks

PSTricks

4 Objetos PSTricks a medida

EN esta sección describiremos algunos comandos útiles para construir nuevos objetos PsTricks; imaginemos que, en nuestro documento, vamos a utilizar con frecuencia un objeto, con ciertas especificaciones de tamaño y con sus parámetros característicos. Por ejemplo, dibujamos un punto rojo grande como:

`\psdots[dotsize=0.8cm,linecolor=Red](0,0)` $\longrightarrow$ 

Podemos definir ahora un nuevo comando, de nombre `\bigdot`, que dibuje un punto de tamaño 0.8 cm y de color rojo mediante:

`\newpsobject{bigdot}{psdots}{dotsize=0.8cm,linecolor=Red}`

a partir de lo cual, dibujar nuevos puntos con esas especificaciones puede hacerse utilizando el nuevo comando PSTricks `\bigdot`:

```
\begin{pspicture}(-0.5,-0.5)(1.5,1.37)
\bigdot(0,0)(1,0)(0.5,0.866)
\end{pspicture}
```

vemos como `\bigdot` posee la misma sintaxis que `\psdots`, el comando en el que se basa, y que por defecto utiliza los nuevos parámetros con los que lo hemos definido.

Podemos cambiar las propiedades del nuevo objeto así definido modificando sus parámetros en el modo usual; por ejemplo:

```
\begin{pspicture}(-0.5,-0.5)(1.5,1.37)
\bigdot(0,0)
\bigdot[linecolor=Green](1,0)
\bigdot[linecolor=Blue](0.5,0.866)
\end{pspicture}
```

En caso de que para un objeto PSTricks utilicemos repetidamente una colección de parámetros determinada, es posible abreviar la escritura definiendo un nuevo parámetro de estilo con:

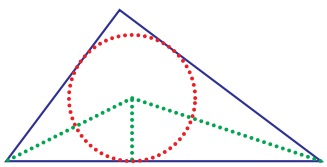
```
\newpsstyle{NombreEstilo}{Param1=Valor1,Param2=Valor2,...}
```

Así, por ejemplo, definiendo:

```
\newpsstyle{newdots}{linestyle=dotted,%
linewidth=1.5pt,dotsep=1pt,linecolor=Green}
```

podemos entonces utilizar el nuevo conjunto de parámetros en cualquier objeto PSTricks, especificando `style=newdots`:

```
\psset{unit=0.5cm}
\begin{pspicture}(0,-0.5)(8.33,4.5)
\pspolygon[linecolor=Blue]%
(0,0)(8.33,0)(3,4)
\psline[style=newdots]%
(0,0)(3.33,1.67)(8.33,0)
\psline[style=newdots]%
(3.33,1.67)(3.33,0)
\pscircle[style=newdots,%
linecolor=Red](3.33,1.67){1.67}
\end{pspicture}
```

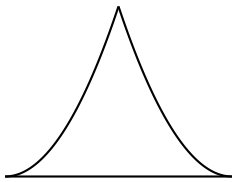


Finalmente, PSTricks permite también colorear regiones de forma arbitraria, mediante el comando:

`\pscustom[Parámetros]{Camino-Cerrado}`

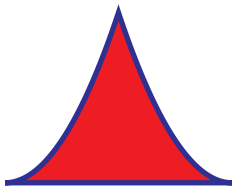
donde, en `Parámetros`, especificamos el tipo de borde y relleno de la región, y `Camino-Cerrado` debe consistir en una curva arbitraria cerrada (en caso de que no lo sea, PSTricks la cerrará y rellenará a su libre albedrío). Veamos un ejemplo; primero delimitamos una región mediante dos trozos de parábola y una recta:

```
\begin{pspicture}(-1.5,0)(1.5,3)
\psplot[algebraic=true]%
{-1.5}{0}{(x+1.5)^2}
\psplot[algebraic=true]%
{0}{1.5}{(x-1.5)^2}
\psline(1.5,0)(-1.5,0)
\end{pspicture}
```



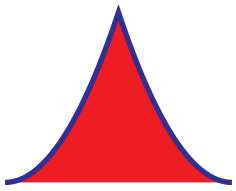
y, a continuación, la rellenamos de rojo utilizando `\pscustom` (poniendo un borde azul, por ejemplo):

```
\pscustom[fillstyle=solid,%
fillcolor=Red,linestyle=solid,%
linecolor=Blue,linewidth=2pt]{%
\psplot[algebraic=true]%
{-1.5}{0}{(x+1.5)^2}
\psplot[algebraic=true]%
{0}{1.5}{(x-1.5)^2}
\psline(1.5,0)(-1.5,0)}
```

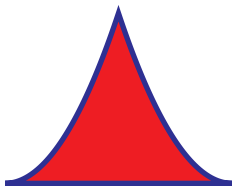


Dentro del argumento `Camino-Cerrado` del comando `\pscustom`, puede sernos de utilidad el comando `\closepath`, que cierra el camino, uniendo los puntos de comienzo y final automáticamente. Compárense los siguientes dos ejemplos:

```
\begin{pspicture}(-1.5,0)(1.5,3)
\pscustom[fillstyle=solid,%
fillcolor=Red,linestyle=solid,%
linecolor=Blue,linewidth=2pt]{%
\psplot[algebraic=true]%
{-1.5}{0}{(x+1.5)^2}
\psplot[algebraic=true]%
{0}{1.5}{(x-1.5)^2}}
\end{pspicture}
```

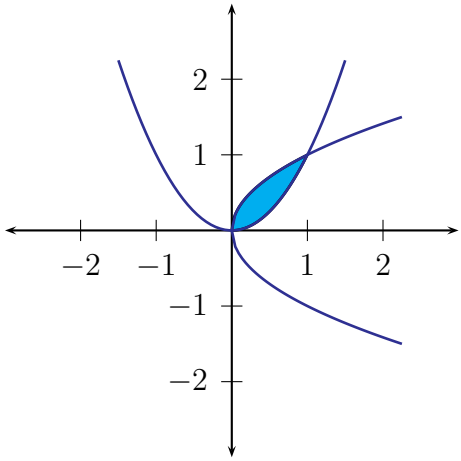


```
\begin{pspicture}(-1.5,0)(1.5,3)
\pscustom[fillstyle=solid,%
fillcolor=Red,linestyle=solid,%
linecolor=Blue,linewidth=2pt]{%
\psplot[algebraic=true]%
{-1.5}{0}{(x+1.5)^2}
\psplot[algebraic=true]%
{0}{1.5}{(x-1.5)^2} \closepath}
\end{pspicture}
```



Finalmente, veamos otro ejemplo de cómo rellenar la región entre dos curvas:

```
\begin{pspicture}(-3,-3)(3,3)
\psaxes{<->}(0,0)(-3,-3)(3,3)
\psset{algebraic=true,linewidth=1pt}
\pscustom[fillstyle=solid,%
fillcolor=Cyan]{%
 \psplot{0}{1}{x^2}
 \psplot{1}{0}{sqrt(x)}}
\psset{linecolor=Blue}
\psplot{-1.5}{1.5}{x^2}
\psplot{0}{2.25}{sqrt(x)}
\psplot{0}{2.25}{-sqrt(x)}
\end{pspicture}
```



5 Efectos 3-D

A lo largo de esta sección describiremos diversas capacidades de los paquetes pst-3d y pst-3dplot, diseñados para construir y visualizar objetos 3D dentro de LaTeX

5.1 Sombras

Podemos sombrear cualquier objeto, se puede utilizar el comando:

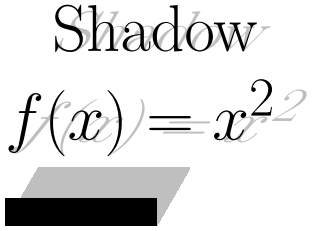
`\psshadow[Parámetros]{Objeto}`

donde la tabla siguiente especifica los parámetros que podemos cambiar:

Parámetro	*Valores*	*Defecto*
`Tshadowangle`	`<angulo>`	`60`
`Tshadowcolor`	`<color>`	`lightgray`
`Tshadowsize`	`<valor real>`	`1`

Veamos unos ejemplos:

```
\psshadow{\huge Shadow}\\[10pt]
\psshadow{\huge $f(x)=x^2$}\\[15pt]
\psshadow[Tshadowsize=2.5]{%
\rule{2cm}{10pt}}
```



5.2 Abatimientos

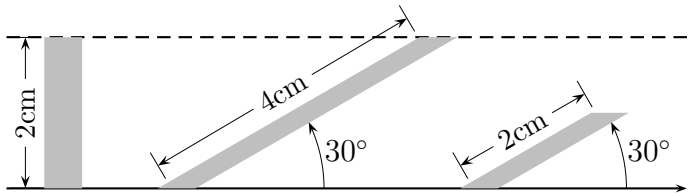
Mediante el abatimiento de objetos, es posible simular vistas perspectivas de objetos tridimensionales. pst-3d define dos comandos análogos para ello:

`\pstilt[Parámetros]{Ángulo}{Objeto}`

`\psTilt[Parámetros]{Ángulo}{Objeto}`

donde `Ángulo` es el ángulo que el objeto será abatido; la diferencia entre ambas versiones del comando se ilustra en la figura siguiente: `\pstilt` abate objetos de forma que las longitudes se conservan, mientras que `\psTilt` preserva la longitud original de los objetos, con lo cual éstos pueden, en principio, ser infinitamente largos. Ángulos de 0 y 180 grados no están permitidos.

`\Bar` `\psTilt{30}{\Bar}` `\pstilt{30}{\Bar}`



(`\Bar` se define como `\psframe*[linecolor=lightgray](0,0)(0.5,2)`)

Ejemplo:

```
\psTilt{60}{
\begin{pspicture}(-0.5,-0.5)(2,2)
\psaxes[axesstyle=frame](2,2)
\pscircle(1,1){0.8cm}
\end{pspicture}}
```