Soluciones Gráficas en LaTeX

Edgar Fuentes
e-mail: fuentesej@gmail.com

10 de junio de 2009

Resumen

Este artículo forma parte de las investigaciones que lleva a cabo el Club de LaTeX con el apoyo del *Instituto de Matemática y Cálculo Aplicado* de la Universidad de Carabobo.

Aquí, se expondrán brevemente, algunas cuestiones elementales y ejemplos de aplicación con respecto a la creación de gráficos vectoriales empleando PSTricks y Ti*k*Z, como dos paquetes de referencia.

Ante todo, no se pretende sustituir la extensa documentación que explica con detalle conceptos de mayor profundidad, sino solo dar una muestra de las capacidades.

Índice

Introducción **2**

1. PSTricks **2**
Principio 3
Serie 3

2. Ti*k*Z **5**
Principio 6
Incidencia Oblicua 7

Conclusiones **8**

Referencias **9**

Introducción

LaTeX por naturaleza posee su propio entorno para describir gráficos (`picture`) que, en general, resulta tedioso, extenso y con capacidades muy limitada. La utilización de paquetes como PSTricks y Ti*k*Z, por el contrario, facilitan esta tarea.[1] Por otra parte, al generar un gráfico insertado sobre código LaTeX, se obtiene como principal ventaja la aplicación de todo su potencial, en cuanto a manejo de texto y ecuaciones.

Ambos sistemas, tienen metas similares: proporcionar un acercamiento al lenguaje, hasta permitir la creación de gráficos compatibles con documentos LaTeX y TeX. Este acercamiento está en contraste con el empleo de programas externos de dibujo, cuya salida es incluida en el documento usando la técnica convencional.

Este documento consta de dos secciones en las que se abordarán los paquetes ya mencionados, acompañados de sus principios de operación y de un ejemplo ilustrativo.

1. PSTricks

PSTriks está basado en PostScript que es un lenguaje de descripción de página con un amplio repertorio de instrucciones para graficar. El paquete está formado por un conjunto de macros que permiten incluir gráficos PostScript directamente sobre el código LaTeX gracias a la existencia de programas, tales como dvips, que traducen una salida dvi a PostScript.

A continuación se muestra un ejemplo de documento que contiene un gráfico generado con PSTricks. En el preámbulo se encuentra la declaración del paquete y de otros macros requeridos para aplicaciones especiales. En el ejemplo, `pst-func` da la posibilidad de graficar cierto tipo de funciones matemáticas. El entorno `pspicture` se utiliza para especificar el código del gráfico, este va acompañado de dos coordenadas rectangulares `(x1,y1)` y `(x2,y2)`; esquinas inferior izquierda y superior derecha respectivamente del rectángulo en el que quedará inscrita la figura.

[1] En la UK TeX FAQ [1] figuran una media docena de sistemas que responden a "Drawing with TeX".

```
\documentclass[12pt]{book}
...
\usepackage{pstricks}
% Macros de PSTricks
\usepackage{pst-func}
...
\begin{document}
...
    \begin{pspicture}(x1,y1)(x2,y2)
    ...
    \end{pspicture}
...
\end{document}
```

Principio

Las salidas PostScript y de los procedimientos PostScript pueden ser incluidos en un documento en virtud de que la cabecera de un archivo PostScript es análoga a la de un archivo macro de TeX y esto viene siendo el principio para que las salidas PostScript y de los procedimientos PostScript puedan ser incluidos en un documento. Esto se debe, a que TeX posee una forma de expresar las tareas que otros interpretes deben realizar y de las cuales el mismo ignora. Para ello existe la instrucción `\special` que solo el interprete tiene la capacidad procesar.

En cuanto a las instrucciones, se parte de la idea de que cualquier imagen se puede generar uniendo los puntos que la conforman siguiendo una cierta trayectoria, se emplean instrucciones como, por ejemplo: insertar un trazo entre dos de ellos o un objeto ya definido en una posición específica. En este caso todos los puntos yacen sobre un plano conformando "nodos" y sus posiciones quedan definidas por un sistema de coordenadas rectangulares.[2]

Serie

Supongamos que se desea graficar la siguiente expresión:

$$F_{\mathrm{MM}}(\theta) = \sum_{n=1}^{N} \frac{2}{n\pi} \left(\sin\left(n\frac{3\pi}{4}\right) + 2\sin\left(n\frac{\pi}{2}\right) + \sin\left(n\frac{\pi}{4}\right) \right) \cos(n\theta),$$

con $\theta \in [0, 4\pi]$ y $N \in \mathbb{N}^+$.

Para llevar a cabo esta tarea, haciendo una revisión de los paquetes basados PSTricks disponibles, se presentan dos alternativas, la primera; `pst-func` que contiene el comando `\psFourier` para hacer representaciones en serie de fourier a partir de

[2] La orden `\SpecialCoor` habilita el uso de un sistema de coordenadas polares.

sus coeficientes de la forma trigonométrica, la segunda; `pstricks-add` a través de la instrucción `Sum` para graficar series. Esta última es más sencilla porque conocemos la expresión analítica y será la que trataremos.

En la figura 1 se muestra el resultado de incluir el código subsiguiente:

```
1  \def\getColor#1{\ifcase#1 blue\or green\fi}
2  \psset{trigLabels=true,xunit=\pstRadUnit,algebraic,plotpoints=600}
3  \begin{pspicture}(0,-3)(13.4,3.6)
4      \footnotesize
5      \psaxes[trigLabelBase=2,ticklinestyle=dotted,yticksize=0
6              13cm,xticksize=-3 3cm,dx=\psPiH]{->}(0,0)(0,-3)(13.4,3.6)
7              [$\theta$,-90][$F_\mathrm{MM}$,180]
8      \multido{\N=1+9,\nColor=0+1,\Real=-1.9+-0.4}{2}{
9          \psplot[linecolor=\getColor{\nColor}]{0}{\psPiFour}{
10             Sum(n,1,1,\N,(2/(n*Pi))*(sin(n*3*Pi/4)+2*sin(n*Pi/2)
11                 +sin(n*Pi/4))*cos(n*x))}
12         \psline[linecolor=\getColor{\nColor}](11,\Real)(11.5,\Real)
13         \rput*[l](11.5,\Real){$N=\N$}
14     }
15     \psplot[linecolor=red]{0}{\psPiFour}{
16         Sum(n,1,1,100,(2/(n*Pi))*(sin(n*3*Pi/4)+2*sin(n*Pi/2)
17             +sin(n*Pi/4))*cos(n*x))}
18     \psline[linecolor=red](11,-2.7)(11.5,-2.7)
19     \rput*[l](11.5,-2.7){$N=100$}
20 \end{pspicture}
```

Para hacer uso del paquete, se declara en el preámbulo:

```
\usepackage{pstricks-add}
```

En la línea 2 se encuentra la instrucción `\psset` donde se establece la configuraciones globales necesarias, en su argumento se encuentran `trigLabels` definido como `true` para etiquetar los ejes en múltiplos o fracciones de π, `xunit` en `\pstRadUnit` para colocar el eje x como una medida angular en radianes, `algebraic` para habilitar el uso de expresiones en notación algebraica y `plotpoints`. Seguidamente, se encuentra el entorno `pspicture` (de la linea 3 a la 20), en este se hallan las instrucciones `\psaxes`, `\psplot` y `\rput*`. A continuación se explican sus funciones:

En la linea 5 `\psaxes` posee la siguiente estructura:

```
\psaxes[opciones]{estilo}(CoorOrig)(CoorInf)(CoorSup)[labelx][labely]
```

Donde las opciones indican que base de la enumeración es de $\frac{\pi}{2}$, el estilo de las rejillas es punteado, las rejillas en el eje x van de 0 a 13cm, en el eje y van de -3 a 3cm y los pasos sobre el eje x son de $\frac{\pi}{2}$. El `estilo` posee la forma de los ejes (una línea terminada en flecha `->`). `CoorOrig` es la coordenada del origen, `CoorInf` y `CoorSup` son las coordenadas inferior y superior respectivamente de la región definida por los ejes. Finalmente, los ejes se etiquetan con `labelx` y `labely`.

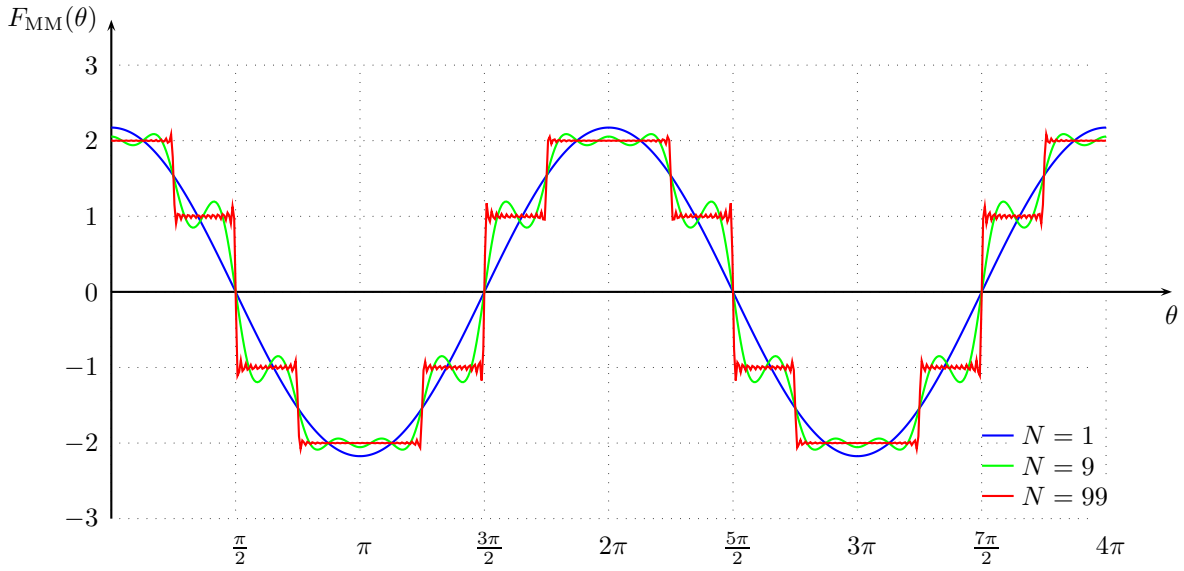

Figura 1: Representación para distintos valores de N

En las líneas 9 y 15 `\psplot` es una de los comandos del paquete `pstricks-add` para trazar funciones matemáticas de una variable, en conjunto con `multido` [2] que posee los acumuladores `\N` y `\nColor`, dan lugar a las curvas en azul, en verde y en rojo respectivamente. Esta posee la siguiente estructura:

```
\psplot[config]{xIni}{xFin}{f(x)}
```

contiene la expresión de $f(x)$ que se muestra para valores de x que van desde `xIni` hasta `xFin` con una determinada configuración `config`. En este caso `Sum` se utiliza para incluir la serie y es de la forma:

```
Sum(index,start,step,end,f(x,index))
```

y es equivalente a

$$\sum_{\text{index=start}}^{\text{end}} f(x, \text{index}),$$

en pasos `step`.

En las lineas 13 y 19, la instrucción `\rput*` coloca el texto de su argumento en la posición indicada con un fondo en blanco que produce la leyenda.

2. Ti*k*Z

Otra solución se halla en el paquete Ti*k*Z. Este nos permite escribir código PGF (Portable Graphics Format) lo más natural posible, es decir, PGF nos proporciona una interface de usuario primitiva de bajo nivel, mientras que la de Ti*k*Z es de más alto nivel.

La palabra Ti*k*Z es un acrónimo recurrente que significa "Ti*k*Z ist kein Zeichenprogramm" (Ti*k*Z no es un programa de dibujo en alemán), donde se recuerda que no se trata de un programa interactivo para dibujar.

A continuación se muestra la estructura de un documento que contiene un gráfico generado con Ti*k*Z. En el preámbulo, se encuentra la declaración del paquete y de las bibliotecas adicionales que dependerán de la aplicación que se esté desarrollando, en este caso `decorations.pathmorphing` posee líneas con forma de serpiente. El entorno `tikzpicture` se utiliza para especificar el código que generará el gráfico, en él se encuentran los comandos que, similar a como ocurre con PSTricks, poseen la información gráfica. Opcionalmente, para hacer el código más flexible [3], se agrupan características comunes entre objetos definiendo estilos que se declaran localmente como opciones del entorno `tikzpicture` o globalmente con `\tikzset`.

```
\documentclass[11pt]{article}
...
\usepackage{tikz}
% Bibliotecas adicionales
\usetikzlibrary{
    decorations.pathmorphing
}
...
\begin{document}
...
   \tikzset{estilos}
    \begin{tikzpicture}[estilos]
    ...
    \end{tikzpicture}
...
\end{document}
```

Principio

PGF provee comandos en LaTeX que utilizan las capacidades gráficas, por igual, tanto de PostScript como de PDF [4]. Dependiendo de como sea procesado el documento, de entre dvips, dvi2pdf o pdfLaTeX, el sistema de comandos elige diferentes instrucciones `\special`.

Su sintaxis es una mezcla entre METAFONT[3] y PSTricks, por lo tanto, sus instrucciones poseen el mismo principio y se incluye la utilización directa de sistema de coordenadas rectangular o polar.

En el caso más simple, estas instrucciones describen segmentos rectos que unen puntos sobre un plano. Para gráficas más complejas, otros objetos gráficos primitivos

[3]Lenguaje de programación utilizado para definir fuentes tipográficas vectoriales.

puede insertarse; como por ejemplo rectángulos, círculos, arcos, texto, cuadrículas entre otros.

Incidencia Oblicua

En la figura 2 se muestra el gráfico generado a partir del siguiente código:

```
\begin{tikzpicture}[
    media/.style={font={\footnotesize\sffamily}},
    wave/.style={
        decorate,decoration={snake,post length=2mm,amplitude=2mm,
        segment length=2mm},thick},
    interface/.style={
        postaction={draw,decorate,decoration={border,angle=-45,
                    amplitude=0.3cm,segment length=3mm}}},
    ]
    \fill[gray!10,rounded corners] (-4,-3) rectangle (4,0);
    \draw[blue,line width=.5pt,interface](-4,0)--(4,0);
    \draw[dashed,gray](0,-3)--(0,3);
    \draw(0,0.15)node[above]{$x$};
    \draw[<->,line width=1pt] (1,0) node[above]{$y$}-|(0,-1)
          node[left]{$z$};
    \draw[->,wave]
         (135:3.2cm)--(135:2.5cm)node[right]{$f^0$};
    \draw[gray](0:0cm)--(135:2cm);
    \path (0,0)++(113:1cm)node{$\phi$};
    \draw[->](0,0.75)arc(90:135:.75cm);
    \draw[->,wave]
         (-30:2.5cm)--(-30:3.2cm)node[right]{$f^+$};
    \draw[gray](0:0cm)--(-30:2cm);
    \path (0,0)++(-60:1cm)node{$\theta$};
    \draw[->] (0,-0.75) arc (-90:-30:.75cm);
    \draw[->,wave]
         (45:2.5cm)--(45:3.2cm)node[right]{$f^-$};
    \path (0,0)++(-22:1.75cm) node{$\psi$};
    \draw[gray](0:0cm)--(45:2cm);
    \draw[->] (0,-1.5)arc(-90:45:1.5cm);
    \path[media] (-3,.6)  node {medio 1}
                 (-3,-.6) node {medio 2};
    \filldraw[fill=white,line width=1pt](0,0)circle(.12cm);
    \draw[line width=.6pt] (0,0)
                           +(-135:.12cm) -- +(45:.12cm)
                           +(-45:.12cm) -- +(135:.12cm);
    \draw[-latex,thick](3.2,0.5)node[right]{$\mathsf{S_{1,2}}$}
          to[out=180,in=90] (3,0);
```

```
39 \end{tikzpicture}
```

El preámbulo debe contener:

```
\usepackage{tikz}
\usetikzlibrary{
    decorations.pathreplacing,
    decorations.pathmorphing
}
```

En las primeras líneas se establecen los estilos, por ejemplo `wave` se refiere a los segmentos en forma serpiente con todas sus características que serán utilizados posteriormente. A continuación sigue el cuerpo del gráfico (de la línea 10 a la 38) en donde se aprecian las instrucciones `\fill`, `\draw` y `\path`.

En la línea 10, la instrucción `\fill` crea un rectángulo relleno de color gris con las esquinas redondeadas que va desde `(-4,-3)` hasta `(4,0)`.

`\draw` es una instrucción muy utilizada que en términos generales y para efectos de describir el código, posee el siguiente aspecto:

```
\draw[opc](coor1)node[opc]{arg}--(coor2)++(coor3)-|(coor4)obj(coor5)...
```

`opc` son las opciones del dibujo, incluyendo los estilos, `coor1`, `coor2`, `coor3`... son las coordenadas requeridas para describir el objeto y se crea secuencialmente a partir de `coor1`, estas se pueden expresar en coordenadas rectangulares (x, y), polares $(\phi : \rho)$ o como combinación de ambas. Entre cada par de coordenadas se coloca el tratamiento que se desea, `--` une con una linea recta los puntos, `++` coloca la última coordenada como origen partiendo de la última referencia, [4] `-|` une los puntos con un ángulo recto y `obj` crea un objeto de `coor4` a `coor5` (en el ejemplo se hallan circulos "`circle`", rectángulos "`rectangle`" y arcos "`arc`"). Si `node` precede a una coordenada, esta es etiquetada con su argumento "`arg`" y con las opciones "`opc`".

La instrucción `\path` contiene información de líneas y curvas sin especificar la manera en que van a ser aplicadas, por ejemplo: `\path[draw]` es equivalente a `\draw` y `\path[fill]` a `\fill`, en la línea 19 se observa como se utiliza para llegar a una posición y colocar `$\phi$` sin haber efectuado trazo alguno.

Conclusiones

Estas soluciones proveen robustez y uniformidad al acabado final del documento. Sin embargo, también se heredan desventajas como: una curva de aprendizaje pronunciada y pequeños cambios requieren tiempo para efectuarse.

A pesar de haber más sistemas, estos son los más recurridos por su amplio desarrollo, por su capacidades de programación y porque poseen extensiones que facilitan la producción de gráficos típicos en la ciencia y la ingeniería como: mapas mentales, diagramas circuitales, diagramas de bloques, entre otros.

[4] Coordenadas específicas.

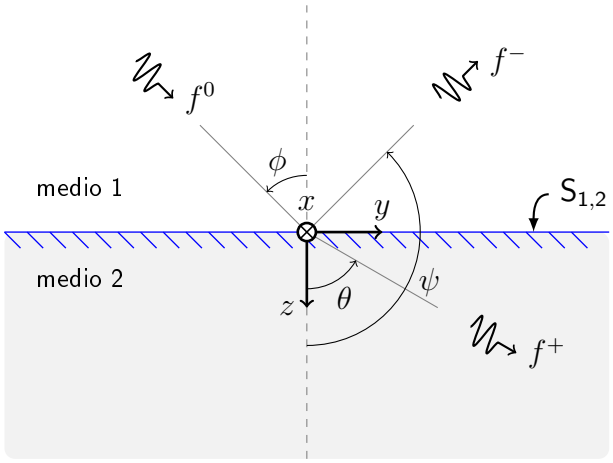

Figura 2: Incidencia oblicua

Los criterios para elegir entre uno de ellos recaen especialmente sobre el factor tiempo, Ti*k*Z posee instrucciones simples, bibliotecas de gran utilidad y portabilidad, mientras que PSTricks posee instrucciones más complejas pero una amplia gama de macros útiles que facilitan la composición.

Referencias

[1] UKTUG, *The UK TeX FAQ*. UKTUG, December 2006. ftp://cam.ctan.org/tex-archive/help/uk-tex-faq/letterfaq.pdf.

[2] T. V. Zandt, *Documentation for multido.tex: A loop macro for Generic TEX*, 1.41 ed., 2004.

[3] T. Tantau, *TikZ and pgf Manual for version 2.00*. Institut für Theoretische Informatik - Universität zu Lübeck, February 2007.

[4] A. Mertz and W. Slough, "Graphics with PGF and Ti*k*Z," in *Practical TEX*, TUGboat, 2007.

[5] T. V. Zandt and D. Girou, "Inside pstricks," TUGboat, 1994.

[6] Dominique Rodriguez and Herbert Voß, *pstricks-add additionals Macros for pstricks v.3.08*, August 2008.