Texto y tipos de letra

Composición de textos científicos

6 de octubre de 2023

1. Texto y espaciamiento en fórmulas

A pesar de que algunos defiendan que las matemáticas son un lenguaje universal, lo cierto es que en los razonamientos matemáticos aparece una cantidad nada desdeñable de lenguaje natural. Dicho más llanamente, por mucho que lo parezca en la cultura popular, no todo en matemáticas son fórmulas, lo típico es que estén combinadas con palabras. Una primera idea ingenua para poner en práctica esta combinación en LaTeX es teclear el texto en modo matemático. Si, por ejemplo, escribimos las hipótesis del teorema fundamental del cálculo como

```
\[
 F(x)=\int_0^x f
 donde f es continua.
\]
```

Esto no puede funcionar porque LaTeX es incapaz de adivinar que "donde" no es el nombre de una función sino texto y que la segunda f es una expresión matemática. El resultado sería:

$$F(x) = \int_0^x fdondefescontinua.$$

Hay entonces dos cuestiones que resolver, la primera indicar qué cosas son texto y la segunda guardar el espaciamiento adecuado. Por si no estuviera claro ya, el modo matemático hace caso omiso de los espacios y si tratamos de hacer párrafos, lanza un mensaje de error.

La forma primitiva de solucionar lo primero es emplear `\mbox{...}`, pero en el paquete `amsmath` está definido `\text{...}` con el mismo propósito que tiene un nombre más sugestivo y por ello está más extendido entre los matemáticos. De esta forma una solución es

```
\[
 F(x)=\int_0^x f
 \text{ donde }f\text{ es continua}.
\]
```

que produce

$$F(x) = \int_0^x f \text{ donde } f \text{ es continua.}$$

Un comentario al margen es que `\text` y `\mbox` admiten encadenar modo matemático *inline* y de texto, entonces con el mismo fin también podríamos escribir `\text{ donde $f$ es continua}`.

Seguramente para el gusto de la mayoría, el "donde" está demasiado cerca de la integral y no hemos solucionado el problema del espaciamiento. Añadir espacios a mano por ejemplo con `\text{ donde }` no tiene ningún efecto porque una de las joyas de LaTeX en lo relativo al texto es que distribuye los espacios entre palabras automáticamente, independientemente del espaciamiento en el fichero fuente: produce prosa aunque escribamos poesía. Una solución de urgencia es utilizar la barra seguida de un espacio `\ ` que fuerza a que aparezca un espacio, pero claramente depender de cosas como `\text{\ \ \ \ \ donde }` para obtener un resultado aceptable no parece una solución seria. Dos comandos muy habituales para aumentar el espaciamiento especialmente, y casi exclusivamente, en las fórmulas son

`\quad` y `\qquad`

dando el primero cierto espaciamiento y el segundo su doble. Si miramos un manual[1] leeremos que `\quad` equivale al ancho de una "M" (mayúscula) con las fuentes que estemos usando para matemáticas. La realidad es que esta información tiene un interés relativo porque casi nadie tiene intuición en esos términos y porque en LaTeX las longitudes tienen habitualmente cierta holgura. Volviendo a nuestra fórmula, parece más aceptable

$$F(x) = \int_0^x f \qquad \text{donde } f \text{ es continua.}$$

que se consigue con `\qquad\text{donde }` etc.

Al hilo de los espacios, una línea con un solo espacio aparecerá naturalmente como una línea en blanco. Así a menudo en los documentos vemos cosas como

```
Una línea

\

Otra línea
```

para crear un doble espacio entre párrafos. De hecho, no es necesario teclear el espacio tras la barra, la barra sola tiene el mismo efecto. Más adelante en el curso, veremos cómo tener más control con el espaciamiento vertical.

[1] Por ejemplo en la magnífica ayuda de Overleaf `https://www.overleaf.com/learn/latex/Spacing_in_math_mode`.

Volviendo al espaciamiento horizontal, a veces se necesita introducir espacios mucho más pequeños. Por orden creciente de tamaño se indican con

$$\texttt{\textbackslash,} \qquad \texttt{\textbackslash:} \quad \text{y} \quad \texttt{\textbackslash;}$$

En la práctica las diferencias son tan poco apreciables que uno (al menos es mi caso) tiende a usar siempre lo mismo. Con estos miniespacios podemos resolver el problema ya mencionado con el diferencial en las integrales. Por ejemplo,

$$\int_0^1 xdx = \frac{1}{2}$$

se obtiene con `\int_0^1 x dx= \frac 12`. Ciertamente queda feo que dx siga inmediatamente a x y se arregla introduciendo un `\,` o alguno de los otros espacios pequeños antes de dx. Con cada uno de ellos los resultados serían respectivamente:

$$\int_0^1 x\,dx = \frac{1}{2}, \qquad \int_0^1 x\,dx = \frac{1}{2} \quad \text{y} \quad \int_0^1 x\;dx = \frac{1}{2}.$$

Los más puristas escriben `\text{d}x` en lugar de `dx` para que el diferencial tenga tipo de texto y no se confunda con una variable.

Aunque es mucho menos necesario, también se suelen utilizar estos espacios pequeños para separar los dos puntos que aparecen en la definición de conjuntos o de funciones. Este sería un ejemplo sin `\,`

$$A = \{n : \text{hay números con } n \text{ divisores}\}.$$

Lo mismo con `\,:\,` se vería así:

$$A = \{n \,:\, \text{hay números con } n \text{ divisores}\}.$$

Para dar un ejemplo más de espacios pequeños supongamos que queremos indicar la divergencia en una fórmula con div, como se suele hacer en Cálculo II. Lo ortodoxo, sobre todo si lo vamos a emplear a menudo, es definir un nuevo operador para que el espaciamiento sea automático, pero a nuestro nivel la solución rápida es ponerlo como texto. Así la regla del producto para la divergencia sería

```
\text{div}(f\vec{F})=f \text{div}\vec{F}+\vec{F}\cdot \nabla f
```

donde `\nabla` produce el símbolo del gradiente. El resultado es

$$\text{div}(f\vec{F}) = f\text{div}\vec{F} + \vec{F}\cdot\nabla f.$$

Si comparamos esto con $f\cos F$, obtenido mediante `f\cos F`, vemos que el espaciamiento de la segunda divergencia no es coherente con el de otras operaciones matemáticas. La solución es utilizar `\,\text{div}\,` que da un resultado visualmente más atractivo.

Existe también un `\,` negativo que se indica con `\!` y a veces se emplea para evitar *overfulls* mínimos o porque el espacio tras integrales o sumatorios, sobre todo si los límites son extensos, se nos hacen poco estéticos. Por ejemplo, en

$$\int_0^{\cos x} t\,dt \qquad \int_0^{\cos x} t\,dt \qquad \int_0^{\cos x} t\,dt \qquad \int_0^{\cos x} t\,dt$$

la primera fórmula es `\int_0^{\cos x}t\,dt` y en las siguiente se han introducido respectivamente uno, dos y tres `\!` antes de `t\,dt`.

Como seguramente intuirás, en LaTeX es también posible salirse de estos comandos predefinidos y lograr un ajuste manual fino de los espacios. El comando universal es `\hspace{...}` donde el argumento indica el espaciamiento y debe ir acompañado de cierta unidad. Quizá `mm` (milímetros) es la que te resulte más útil y familiar. En el mundo de la tipografía es muy común `pt` (puntos) que constituye una unidad menor (cerca de 1/3 de milímetro). Por ejemplo,

```
\[ f=g\circ h\hspace{40pt}\text{con $h$ continua}. \]
```

y

```
\[ f=g\circ h\hspace{80pt}\text{con $h$ continua}. \]
```

producen

$$f = g \circ h \qquad \text{con } h \text{ continua.}$$

y

$$f = g \circ h \qquad\qquad \text{con } h \text{ continua.}$$

El argumento de `\hspace{...}` puede ser negativo. Así, volviendo al ejemplo de la integral con límites anchos, con

```
\[
 \int_0^{\cos x}\hspace{-10pt}t\,dt
\]
```

conseguimos una invasión exagerada de la zona reservada al límite superior:

$$\int_0^{\cos x} t\,dt$$

Es muy importante no abusar de `\hspace` porque rellenar nuestro código de ajustes manuales lo hará poco legible y poco ajustable (por ejemplo si cambiamos el tamaño de letra). Además es una renuncia a las buenas capacidades automáticas de LaTeX.

Contradiciendo este último consejo, para finalizar, añadiré una forma un poco chapucera de añadir espacios con el comando `\phantom{...}` que sustituye lo que escribamos dentro por el espacio correspondiente a su longitud. A veces se usa en problemas relacionados con el alineado. Solo a modo de ilustración, con

```
\begin{pmatrix}
 123 & \cos \pi
 \\
 \phantom{12}3 & \phantom{\cos} \pi
\end{pmatrix}
```

lograremos romper la regla de que los elementos de las matrices siempre aparecen centrados por columnas, obteniendo:

$$\begin{pmatrix} 123 & \cos \pi \\ 3 & \pi \end{pmatrix}.$$

Por supuesto esto es una manera muy heterodoxa (y poco generalizable) de conseguir el resultado. Ya veremos qué es lo ortodoxo cuando estudiemos las tablas.

2. Fuentes de texto

En breve y sin todo rigor las fuentes de que disponemos sin instalar ni cargar nada son las siguientes:

Comando	Nombre	Ejemplo
`\textrm{...}`	`roman`	Normal y corriente
`\textbf{...}`	`boldface`	**Negrita**
`\textit{...}`	`italic`	*Cursiva*
`\textsl{...}`	`slanted`	*Inclinada*
`\textsf{...}`	`sans-serif`	Palo seco (no es broma)
`\texttt{...}`	`typewriter`	`Courier`
`\textsc{...}`	`small caps`	VERSALITA

El nombre en inglés de la fuente se ha incluido para ver que el comando proviene de una abreviatura.

En TEX no se usaba `\text`*ab* sino `\`*ab* con las llaves necesarias para indicar el ámbito. Esto también funciona en LATEX y así `\textit{Cursiva}` equivale a `{\it Cursiva}`. En realidad hay alguna diferencia de comportamiento en situaciones raras (conflicto entre fuentes) que exceden este nivel. Si se olvidan las llaves entonces el efecto de la versión TEX del comando se extiende desde el momento que se usa en adelante.

Para resaltar algo se utiliza `\emph` (de *emphasize*) que cambia el tipo de letra de forma automática. Así con `Es \emph{importante} recordar` y `\textit{Es \emph{importante} recordar}` obtenemos

Es *importante* recordar y *Es* importante *recordar*.

Aunque no sea una fuente de texto distinta, hablando de resaltar, el comando `\underbar{...}`, como su nombre indica, subraya. El análogo del ejemplo anterior es que con `Es \underbar{importante} recordar` y `\textit{Es \underbar{importante} recordar}` obtenemos

Es importante recordar y *Es importante recordar.*

El control del tamaño de la fuente se lleva a cabo con `{\`*tamaño* `...}` (o el entorno correspondiente) donde las posibilidades para *tamaño* están recogidas en las siguientes dos tablas:

Comando	`\tiny`	`\scriptsize`	`\footnotesize`	`\small`
Ejemplo	Hola	Hola	Hola	Hola

(hay también un `\normalsize`, pero, lógicamente, es raro usarlo)

Comando	`\large`	`\Large`	`\LARGE`	`\huge`	`\Huge`
Ejemplo	Hola	Hola	Hola	Hola	Hola

Terminemos con algunas maneras de importancia menor para un principiante de cambiar el comportamiento de la fuente de texto.

El entorno `verbatim` ya apareció en una anterior entrega y muestra de manera literal en la fuente `typewriter` todo lo que aparece dentro de él, respetando incluso espacios y renglones. El siguiente texto se escribió tal cual entre dos líneas con `\begin{verbatim}` y `\end{verbatim}`.

```
Seguro que alguien se mofa
si invocando al magisterio
medio en broma medio en serio
digo que esto es una estrofa.
```

Por supuesto, calidad aparte, esta no es la forma buena de escribir poesía, por la fuente y porque hay paquetes especiales para tal fin. Lo mismo se aplica a listados de programas. Posiblemente la principal utilidad de este entorno es escribir manuales de LaTeX. Si queremos usar algo similar, pero dentro de una línea, la alternativa es `\verb!...!`. En realidad podemos usar otros caracteres en vez de `!`, por ejemplo `\verb+...+`, lo que hay que tener cuidado es que ese carácter no aparezca en el interior para no hacer un lío al comando. Así escribiríamos `\verb!calcular \int!` y `\verb+¡calcular \int!+` para obtener `calcular \int` y `¡calcular \int!`

El entorno `verbatim*` hace lo mismo que `verbatim`, pero destacando los espacios con ␣ para que se vean mejor. Hay también un paquete `verbatim` que amplía las posibilidades.

3. Fuentes matemáticas

A pesar de que ya hemos aprendido suficiente para crear fórmulas muy complicadas, al copiar un fragmento de un texto de matemáticas pasaríamos un apuro con algo bastante básico: no sabemos cómo escribir las letras que representan los principales conjuntos de números como los reales o los enteros. Tampoco sabríamos reproducir la letra caligráfica que se usa con cierta frecuencia. En algunos temas un poco avanzados también hay tradición de utilizar letra gótica, a pesar de su legibilidad cuestionable. Estos tres tipos de letra se indican en LaTeX con

`\mathbb{...}`, `\mathcal{...}`, y `\mathfrak{...}`.

En caso de que te lo estés preguntando el `bb` es de *blackboard*, las letras tal como se escriben en la pizarra, y `frak` abrevia *fraktur*, el nombre inglés, calco del alemán, de un tipo común de letra gótica. Una observación importante es que `\mathbb` y `\mathcal` están reservados para letras mayúsculas, con minúsculas o números obtendremos símbolos extraños.

Por ejemplo

$$\mathbb{R}\times\mathbb{Q}\longrightarrow\mathcal{A}\longrightarrow\mathfrak{so}(3)$$

se obtiene con

```
\mathbb{R}\times \mathbb{Q}
\longrightarrow
\mathcal{A}
\longrightarrow
\mathfrak{so}(3)
```

Una tentación de los que comienzan a escribir textos matemáticos cuando se tienen todas las posibilidades de LaTeX es utilizar notaciones barrocas. Esto no es nada deseable si se quiere facilitar la claridad y legibilidad. Volviendo a las letras góticas, la "A" obtenida con `$\mathfrak{A}$` se ve como $\mathfrak{A}$ que cualquiera sin experiencia confundiría con una "U". ¿Sabrías por ejemplo qué letra es $\mathfrak{S}$ usada para indicar ciertas sumas?

Si haces un trabajo de fin de grado de matemáticas es muy probable que no utilices ni una sola letra gótica y es posible que pocas letras caligráficas, pero es muy raro que no aparezcan $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$ o $\mathbb{C}$. En algunos de estos trabajos aparecerá por doquier uno de estos conjuntos de números y se vuelve un poco pesado teclear todo el comando cada vez. Vamos a ver un avance de lo más básico de la definición de nuevos comandos. Si incluimos en la cabecera:

```
\newcommand{\R}{\mathbb{R}}
```

estaremos indicando a LaTeX que existe un nuevo comando llamado `\R` que tiene el mismo efecto que `\mathbb{R}` con ello ahorraremos pulsaciones tecleando por ejemplo $\mathbb{R}^n$ como `\R^n` en vez de como `\mathbb{R}^n`.

Lo mismo sirve con otras letras y seguro que muchos matemáticos, como yo, copian de una cabecera a otra la lista:

```
\newcommand{\N}{\mathbb N}
\newcommand{\Z}{\mathbb Z}
\newcommand{\Q}{\mathbb Q}
\newcommand{\R}{\mathbb R}
\newcommand{\C}{\mathbb C}
```

Nótese la pequeña variante con respecto a lo anterior omitiendo las llaves más interiores. Esto se debe a que, como ocurre con otros comandos, si no hay llaves `\mathbb`, `\mathcal` y `\mathfrak` actúan sobre el siguiente carácter distinto de un espacio.

En la tipografía habitual matemática las funciones y los números van en letra normal mientras que las variables van en cursiva. Por ejemplo, con `\tan(3x+y)` obtenemos $\tan(3x+y)$. En ocasiones muy especiales uno quiere variar este comportamiento haciendo que una fórmula o una parte de ella tenga un tipo de letra de texto distinto al predeterminado. Ciertamente esto no ocurre muy a menudo y, en general, es mejor evitarlo en la medida de lo posible. Por si sirve de ilustración, yo a veces lo hago en las fórmulas a pie de una imagen cambiándolas a `sans-serif` para distinguirlas del resto. La forma de modificar el tipo es emplear `\math`*ab* de forma totalmente similar a `\text`*ab* excepto que *ab* no pueden ser ni `sc` ni `sl`.

Por ejemplo, `f^3(x)=f_2(x)` daría el formato habitual $f^3(x)=f_2(x)$ y conseguiríamos

$$\mathsf{f}^3(\mathsf{x})=\mathsf{f}_2(\mathsf{x}), \qquad f^{\mathit 3}(x)=f_{\mathit 2}(x) \qquad \text{y} \qquad f^3(\mathtt{x})=\mathtt{f}_2(x)$$

respectivamente con `\mathsf{f^3(x)=f_2(x)}`, `\mathit{f^3(x)=f_2(x)}` y `f^3\mathtt{(x)=f}_2(x)`.

4. Caracteres especiales

Hay una serie de caracteres que en LaTeX tienen un significado diferente al que vemos en el teclado, son:

```
#   $   %   &   ~   _   ^   \   {   }
```

Ya ha aparecido el significado de todos ellos excepto de `#` que se utiliza en la programación y de `~` que se utiliza para forzar que no se separen dos palabras consecutivas en líneas diferentes. A modo de recordatorio, esta es una tabla con los usos del resto:

`$`	Modo matemático
`%`	Comentarios
`&`	Matrices
`_`	Subíndices

`^`	Superíndices
`\`	Comienzo de comando
`{`	Comienzo de bloque
`}`	Fin de bloque

Como regla, si necesitamos escribir estos caracteres los precederemos con \ con la excepción de la propia barra \ que se obtiene en modo matemático mediante `\backslash` ya que la secuencia \\ está ocupada para su uso por ejemplo para indicar el cambio de fila en matrices. También `\^` tiene un comportamiento excepcional que veremos en breve. A veces algunos editores se hacen un pequeño lío con los colores cuando se usan estos comandos especiales, por ejemplo porque no distinguen bien `\%` de una línea de comentario. En cierto modo el espacio es también un carácter especial porque LaTeX no los respeta fielmente, da igual que pongamos uno o veinte. Por ello, como ya hemos visto, también admite la secuencia de escape `\␣` (barra seguida de espacio) similar a la de los otros caracteres especiales para forzar a que aparezca un espacio.

Un ejemplo es que escribiríamos `El 10\% de 20\$ son 2\$` para obtener "El 10 % de 20$ son 2$". Hablando de divisas, el uso de LaTeX se extendió en los años 90 antes de que se introdujera el euro y por tanto es natural que no incluya de serie un comando para mostrar su símbolo. El paquete más empleado para subsanarlo es `eurosym` que define el comando `\euro` y otros relacionados. En definitiva, si en nuestra cabecera incluimos

```
\usepackage{eurosym}
```

entonces `El 10\% de 20{\euro} son 2\euro` produce "El 10 % de 20€ son 2€". Si el símbolo te parece poco estético busca información sobre este paquete, por ejemplo con `\usepackage[gen]{eurosym}` obtendrás una tipografía algo distinta. El comando `\euro`, como es habitual en LaTeX, no distingue los espacios que vienen detrás y es por ello que en el primer `\euro` son necesarias las llaves, también con la variante `\euro{}`.

Para terminar esta sección de caracteres especiales y aunque no tenga demasiado que ver con lo anterior, hay unos "dígrafos" que se usan con cierta frecuencia. Con dos guiones se consigue un guion algo más largo que se utiliza por ejemplo para indicar un intervalo de páginas. Así con `véase pp.7--14` se obtiene "véase pp.7–14" mientras que con el guion simple el resultado sería "véase pp.7-14". El otro ejemplo a destacar aquí son las comillas. Si usamos las del teclado el resultado es muy feo, esencialmente porque LaTeX no sabe cuándo las estamos abriendo o cerrando. Hay editores adaptados (como Kile) que traducen automáticamente esas comillas en los símbolos correctos, siendo las impares de apertura y las pares de clausura. Si no disponemos de esta traducción automática, las primeras se indican con dos acentos graves (los contrarios a los usados en español que están en la tecla a la derecha de la `p`) y las segundas con dos apóstrofos (tecla a la derecha del `0`). Otra posibilidad más complicada, y menos común, es `\lq\lq` y `\rq\rq` (*left quotes* y *right quotes*). En la mayoría de los casos, el último `\rq` vendrá seguido de `{}`.

5. Internacionalización

Los acentos y letras propias de un idioma, como la eñe en nuestro caso, en los primeros tiempos del TEX había que teclearlas con secuencias especiales lo cual era bastante farragoso. Enseguida los editores adaptados incorporaron traducciones automáticas, de forma que cuando uno tecleaba "ñ" internamente en la fuente escribía `\~n` que es la secuencia correspondiente.

Afortunadamente, con el desarrollo de LATEX apareció un paquete que resolvía de manera bastante fiable los problemas de internacionalización. Es el paquete `babel` que podemos cargar en la cabecera con un idioma particular, en nuestro caso

```
\usepackage[spanish]{babel}
```

También podemos omitir `[spanish]` e indicar `spanish` en los parámetros de `\documentclass`. Este paquete redefinirá algunos operadores en modo matemático, por ejemplo el símbolo "lím" pasa a tener acento y define otros nuevos como `\sen` y `\tg` aunque `\sin` y `\tan` seguirán funcionando. Estos cambios a veces dan lugar a algún dolor de cabeza por incompatibilidades con otros paquetes. Por ejemplo, aunque casi nadie hagamos caso, la RAE recomienda que usemos las comillas españolas que son como pares de paréntesis angulares (*guillemets*) y consecuentemente `babel` permite que `<<hola>>` dé el resultado «hola», pero resulta que en el paquete más básico de diagramas conmutativos `<<` y `>>` están definidos de otra forma y este paquete dejará de funcionar con `babel`. Una manera un poco drástica de desactivar algunas de estas peculiaridades conflictivas es:

```
\usepackage[spanish,es-noquoting,es-noshorthands]{babel}
```

Otro tema es la coma decimal. Cargando `babel` con español la fuente `$3.14$` produce 3,14 lo cual nos puede incomodar (pensemos por ejemplo en un vector con coordenadas decimales separadas por comas). Una posibilidad para desactivar este comportamiento es poner `\decimalpoint` en alguna línea de la cabecera después de cargar el paquete.

Parece que la norma en español es que las abreviaturas de operadores se acentúan cuando el nombre completo lo está, por ejemplo lím. Si eso no te convence, existe la posibilidad de quitar los acentos globalmente con el comando `\unaccentedoperators` ubicado, de nuevo, en la cabecera después de la carga de `babel`. Si lo pusiéramos antes, haría saltar un error porque está definido en el propio paquete.

En `babel` hay también información acerca de cómo separar las palabras en sílabas por si acaso es necesario dividir alguna con un guion al final de una línea. Si nos hemos inventado una palabra nueva y queremos dar una indicación acerca de cómo dividirla lo podemos hacer introduciendo `\-` en los posibles lugares de separación. Eso no tendrá efecto en cómo vemos la

palabra si no se divide. Por ejemplo, con `svet\-loru\-menim` obtendremos "svetlorumenim" y en caso de tener que dividirla al final de una línea LaTeX tratará dejar en ella "svet-" o "svetloru-". El paquete `hyphenat` da más posibilidades para controlar la división de las palabras. De alguna forma el opuesto de `\-`, pero con pares de palabras es `~`. Si escribimos `tú~y~yo` estaremos indicando que la palabra `tú` vaya en la misma línea que `y` y en la misma que `yo`. Por supuesto si abusamos mucho de ello tendremos todas las papeletas para provocar *overfulls*. A menudo se usa esta virgulilla para evitar que un número o un carácter no alfabético quede al principio de una línea, lo cual resultaría feo. Por ejemplo, uno escribiría `Carlos~III`.

Incluso si solo tenemos pensado en escribir en inglés y en español, es fácil que tengamos ocasionalmente que mencionar nombres extranjeros que tengan una acentuación peculiar o letras especiales. Aquí hay algunos ejemplos que no es posible teclear directamente porque no aparecen en nuestro teclado o no son reconocidos por la opción en español. Ten en cuenta que, recíprocamente, si envías una fuente LaTeX a alguien y quiere usarla sin cargar el paquete de español y la codificación adecuada, debería reemplazar los acentos, etc. por unas secuencias de escape con no aparecen aquí.

Nombre	Fuente
l'Hôpital	`l'H\^opital`
Erdős	`Erd\H os`
Sebastião	`Sebasti\~ao`
Français	`Fran\c cais`
Łaba	`\L aba`
Krzyżanowski	`Krzy\.zanowski`

Nombre	Fuente
Tụy	`T\d uy`
Gårding	`G\r arding`
Vrănceanu	`Vr\u anceanu`
Vopěnka	`Vop\v enka`
Størmer	`St\o rmer`
Yıldırım	`Y\i ld\i r\i m`

El primer ejemplo ilustra que `\^` es un poco excepcional entre los caracteres especiales porque se superpone al siguiente. Si queremos un circunflejo solo podríamos emplear `\^\ ` para situarlo sobre un espacio. Los acentos graves y agudos se pueden teclear con un teclado español, pero solo sobre vocales, precedidos por la barra lograremos que aparezcan sobre otras letras. Por ejemplo "Sierpiński" se obtiene con `Sierpi\'nski`.

6. Acentos matemáticos

Aparte de los acentos propiamente dichos correspondientes a los diferentes idiomas, también hay lo que se podrían llamar "acentos" matemáticos que modifican a algunos símbolos. Los más usados son lo que normalmente llamamos *tilde* y *gorro*. Así es tradición que la transformada de Fourier de una función se escriba con un circunflejo sobre su nombre. Podríamos emular esto pasando provisionalmente a modo texto con el tipo de letra correspondiente y añadiendo el circunflejo como en la sección anterior. Por ejemplo

para obtener $\hat{f}(\xi)$ escribiríamos `$\textit{\^f}(\xi)$`. Es fácil sospechar que esto es demasiado complicado (aparte del espaciamiento raro). Así es, en LaTeX ya hay comandos que permiten decorar caracteres o expresiones con lo que podría calificarse como acentos. Los más empleados son:

```
\tilde    \hat    \bar    \underline    \dot    \ddot
```

Su efecto sobre la f, la notación típica para representar funciones, es

$$\tilde{f} \quad \hat{f} \quad \bar{f} \quad \underline{f} \quad \dot{f} \quad \ddot{f}$$

En todos los casos podemos indicar el ámbito de comando siguiéndolo de un bloque entre llaves, pero en la práctica eso es solo útil con `\underline`, por lo que se indica en el párrafo siguiente. Si deseamos subrayar más de una letra emplearemos `\underline{...}`.

Si repetimos el mismo ejemplo con una letra más ancha como la m el resultado de los tres primeros no es muy estético:

$$\tilde{m} \quad \hat{m} \quad \bar{m} \quad \underline{m} \quad \dot{m} \quad \ddot{m}$$

Los resultados serían todavía menos aceptables si queremos que el "acento" matemático aparezca sobre más de una letra. Por ejemplo `$\tilde{wm}$` da lugar a $\tilde{wm}$. Para arreglarlo hay versiones ajustables de `\tilde` y `\hat` y para `\bar` se puede usar el hermano de `\underline`. Los comandos correspondientes son:

```
\widetilde    \widehat    \overline
```

Por ejemplo

```
\widetilde{w}(\xi)= \widehat{as}(\xi)=\overline{me}(\xi).
```

produce

$$\widetilde{w}(\xi) = \widehat{as}(\xi) = \overline{me}(\xi).$$

Por si te lo estás preguntando, no es adecuado emplear `\underbar` en modo matemático en vez de `\underline` porque es el método para subrayar textos y por tanto supone que lo de dentro está en este modo cambiando el tipo de letra. Por ejemplo, `$\underbar{f}$` da $\underbar{f}$ y `$\underbar{f^2}$` hace saltar un error porque `f^2` no es válido en modo texto.

No se me ocurre una situación natural en que uno quiera poner un acento normal (agudo) encima de una letra matemática, pero se puede emular fácilmente utilizando `\text` con el tipo de letra que aparece en la fórmula, que es típicamente cursiva.