Una breve guía para redactar un trabajo de fin de grado con contenido matemático utilizando LaTeX

Alberto Arenas, Edgar Labarga y Juan Luis Varona

Departamento de Matemáticas y Computación,
Universidad de La Rioja, Logroño
{alberto.arenas,edgar.labarga,jvarona}@unirioja.es

Resumen

En esta guía damos una serie de indicaciones y consejos para alguien que quiera escribir un documento en LaTeX en español con contenido matemático, en particular para escribir un trabajo de fin de grado de un grado de Matemáticas. No es un manual de LaTeX —se supone que quien vaya a usar esta guía ya tiene ciertos conocimientos, o que los está obteniendo por otros medios—, sino que se centra en analizar y dar solución a los principales problemas que pueden surgir, tanto en el manejo de LaTeX como en la redacción de textos con contenido matemático en español.

1. Introducción

Este documento no pretende ser, ni mucho menos, un manual de LaTeX. Si lo que buscas es un manual, los dos libros «oficiales» son *LaTeX: A document preparation system* [14], escrito por Leslie Lamport, el desarrollador original de LaTeX, y *The LaTeX Companion* [17], escrito por los desarrolladores actuales de LaTeX, liderados por Frank Mittelbach, y que incluye información sobre bastantes de los paquetes que acompañan a LaTeX. Por supuesto, hay otros libros muy buenos, entre ellos algunos en español como [4, 5]. También hay libros y/o manuales gratuitos en internet (como `https://en.wikibooks.org/wiki/LaTeX`, `https://www.learnlatex.org/es/` o `https://es.overleaf.com/learn`), e incluso que se incluyen en las distribuciones habituales de TeX/LaTeX (como *La introducción no-tan-corta a LaTeX 2ε*, originalmente en alemán y ahora disponible, con distinto grado de actualización, en bastantes idiomas [19]). También es recomendable el tríptico [7] (una página por las dos caras, suficiente para tener a mano toda la sintaxis más básica). Lamentablemente, algunos manuales gratuitos escritos poco a poco por la comunidad de usuarios comienzan explicando cosas obsoletas que ahora están totalmente superadas, y a las que ya no hay que dar importancia —muchos son los que colaboran añadiendo material útil, pero nadie elimina partes escritas previamente por otros—; más que ayudar, a veces asustan y confunden. Por otra parte, un lugar muy fiable para encontrar soluciones a problemas relacionados con el uso de LaTeX es `https://tex.stackexchange.com`.

Quizás conviene también aclarar cuál es la diferencia entre TeX y LaTeX. Donald E. Knuth desarrolló el TeX original en la década de 1970 [13]. Los libros que Knuth había publicado hasta entonces —esencialmente, las primeras ediciones de su obra magna *The Art of Computer*

Versión del 30 de mayo de 2022.

Programing, que aún continúa ampliando con nuevos tomos— quedaban —tipográficamente y estéticamente hablando— muy de su agrado. Pero en esos años se habían desarrollado procedimientos informáticos que abarataban mucho la impresión, pero que empeoraban el aspecto de los libros. Knuth se propuso solucionar el problema creando un sistema informático que no supusiera pérdida de calidad tipográfica con respecto a lo que se conseguía con los métodos manuales ya obsoletos y muy caros. Pensaba emplear un año en ello, pero al final le costó una década (su lanzamiento oficial fue en 1978). Además de TEX, tuvo que desarrollar METAFONT, el programa (realmente, un lenguaje de programación) destinado a crear la tipografía que se iba a usar en TEX (en particular, todo tipo de símbolos matemáticos); en él las letras se definían a partir de esplines cúbicos de sus contornos (en [12] se pueden ver detalles matemáticos al respecto). TEX se desarrolló en una época en la que la potencia de los ordenadores era bastantes órdenes de magnitud menor que la de los actuales, y algunas decisiones sobre su funcionamiento provienen de eso. Pocos años más tarde, era factible construir, por encima de TEX, lenguajes «de más alto nivel» que facilitasen su uso. Así nació LATEX (su primera versión es de 1984), que es lo que más se usa actualmente. Realmente, esta ha sido una historia muy resumida y desde entonces hasta ahora han pasado muchas cosas importantes que nos hemos dejado en el tintero; por ejemplo, hay que tener en cuenta que, en esa época, el formato pdf no existía, luego la salida que proporcionaban TEX o LATEX no podía ser un archivo `.pdf`, como se hace actualmente.

Para mantener la compatibilidad con los documentos antiguos, TEX está oficialmente «congelado» desde 1989 (salvo para corrección de errores), pero LATEX continúa su desarrollo (siempre procurando mantener la compatibilidad con el pasado, aspecto en el que se hace mucho hincapié), a menudo en aspectos muy alejados de las matemáticas. A decir verdad, los TEX actuales no son estrictamente los de Knuth, pero casi (resumiendo mucho, la principal diferencia, ya mencionada, es que los TEX actuales generan archivos `.pdf`); Knuth sigue siendo el «jefe honorífico» de todo lo relacionado con TEX, pero no desempeña labores ejecutivas. Con vista al futuro —aún no es la distribución oficial por defecto, pero se supone que lo será en unos años—, a TEX se le ha añadido el lenguaje de programación Lua; este es un lenguaje de propósito general (véase [11]), no como TEX que está orientado a cuestiones tipográficas, y permite programar tareas difíciles y/o lentas en TEX o LATEX puro (en [18] puede verse un ejemplo de uso relacionado con las matemáticas). Pero todo esto no importa ahora, así que volvamos a nuestra senda.

Ya ha quedado claro que esto **no** es un manual de LATEX, sino que está destinado a personas a las que ya se les presupone cierto conocimiento en el uso de LATEX. Además, se da por hecho que ya tienes instalada una distribución de TEX en tu ordenador (si no es así, puedes descargarla desde las páginas oficiales del TEX Users Group, `https://tug.org`), o que lo estás usando online en alguna página web (como `https://www.overleaf.com`). En principio, todo el software relacionado con TEX y LATEX es libre (con distintas licencias, siendo la *LaTeX Project Public License* la más usada), así que deberías podértelo instalar sin ningún impedimento, y sin pagar nada. (Existen editores y tipografías que son de pago, pero ten claro que no lo necesitas salvo que, por algún motivo, te apetezca usarlo.)

Lo que aquí se pretende es dar algunas pautas y consejos para escribir un documento de contenido matemático de cierta extensión y redactado en español. En particular, puede servir de guía para escribir un trabajo de fin de grado de cualquier grado de Matemáticas en una universidad española —o una tesis doctoral—, aunque la mayor parte de lo que aquí se explicará se puede aplicar a cualquier otro documento matemático; algunas universidades pueden tener requisitos específicos sobre el formato, claro. Dado que la idea es escribir un documento de unas 50 páginas (un tamaño típico para un trabajo de fin de grado) y con capítulos, usaremos la clase

book. Asimismo podríamos utilizar amsbook, que también es muy común, pero realmente esta clase está diseñada, primordialmente, para los autores que van a escribir libros (en inglés) que vayan a ser publicados por la American Mathematical Society (AMS), y tiene algunos ajustes prefijados en inglés que no siempre son fáciles de adaptar. Por lo demás, lo que se escribe con amsbook tiene un aspecto mucho más compacto (con menos espacios, cabe más texto en el mismo número de páginas) que lo que se escribe con book, y por eso hay bastantes personas que prefieren usar amsbook. Por supuesto, como ambas clases están construidas sobre LaTeX, casi todo funciona igual, más allá de que amsbook ya carga automáticamente algunos paquetes (los de la AMS). Si en vez de un «libro» se quisiera escribir un «artículo», las correspondientes clases que habría que usar serían article o amsart; la principal diferencia entre los libros y los artículos es que los artículos no tienen capítulos (se subdividen en secciones), y los libros sí. Pero dejemos ya esta introducción y vayamos con nuestro objetivo.

2. Un primer documento

Para escribir en español en LaTeX, lo que nos conviene es tener un archivo de texto con extensión `.tex` (algunos usan `.ltx`, da igual) que contenga, al menos, lo siguiente:

```
\documentclass[10pt, a4paper]{article}

\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage{lmodern}

\usepackage[spanish]{babel}
\decimalpoint

\begin{document}

¡Hola, mundo!

\end{document}
```

Esto es un documento LaTeX basado en la clase `article` (podríamos haber usado `book`, claro), cuya letra de tamaño ordinario será de «10 puntos», está destinado a un papel de tamaño A4 (el habitual en casi todo el mundo menos en Norteamérica), carga varios «paquetes» en el preámbulo, y cuyo contenido —lo que aparecerá posteriormente en el correspondiente archivo `.pdf`— será, exclusivamente, el texto «¡Hola, mundo!».

Vamos a comentar los paquetes que aparecen ahí:

- inputenc sirve para decirle a LaTeX qué codificación de caracteres vamos a usar en el archivo `.tex` de nuestro documento. Lo recomendable actualmente es usar codificación utf8. Todas las demás están obsoletas y se mantienen solo por compatibilidad (pero realmente hay bastantes personas que aún usan codificación latin1). Esto permite que en el archivo `.tex` podamos escribir directamente textos como como «¡hola!, cigüeña: ¿qué hora es?», sin tener que teclearlo como `!`hola!, cig\"ue\~na: ?`qu\'e hora es?`. Si fuésemos a escribir en inglés, que no es el caso que nos ocupa, podría ser razonable no cargar ninguna codificación

y escribir de esa forma los pocos caracteres no ingleses que fueran a salir (nombres de personas, habitualmente); pero, en otro caso, no hay ninguna justificación. De hecho, desde 2018, las distribuciones de LaTeX ya usan automáticamente codificación utf8 sin necesidad de cargar el paquete inputenc, aunque es conveniente cargarlo por si compartes el archivo con otra persona que use una distribución de LaTeX no muy moderna. Más aún, LuaLaTeX —la extensión de LaTeX que incorpora el lenguaje de programación Lua— solo funciona con codificación utf8. Un detalle importante: `\usepackage[utf8]{inputenc}` solo afecta a LaTeX, no al editor que estés usando para escribir el archivo `.tex`. En algún menú o alguna preferencia tendrás que configurar que tu editor use la misma codificación que vas a usar en LaTeX; de lo contrario, tendrás un lío morrocotudo con los acentos y otros caracteres no ingleses (afortunadamente, los editores también vienen por defecto configurados en modo utf8).

- Así como inputenc se refiere a la entrada de caracteres en el archivo `.tex`, fontenc se refiere a la salida en el archivo `.pdf`. Tradicionalmente, y por ejemplo, LaTeX habría «dibujado» una «a acentuada» en el `.pdf` como un carácter «a» con un acento encima; es decir, no habría usado el carácter «á». Esto, que proviene de épocas en las que cada bit era importante, y que visualmente no se nota, tiene actualmente un efecto colateral relevante: no permite buscar palabras acentuadas (o con eñes) en el `.pdf`, ni permite copiarlas bien desde el `.pdf` para pegarlas en otro archivo. Con `\usepackage[T1]{fontenc}`, en el `.pdf` se usan verdaderos caracteres acentuados y desaparece este problema.

- Las letras originales de TeX/LaTeX (cuando Donald Knuth diseñó TeX también tuvo que diseñar todas las tipografías que iba a usar, que denominó Computer Modern Roman) no estaban adaptadas para, entre otras cosas, usar `\usepackage[T1]{fontenc}`. Las distribuciones actuales ya no emplean las tipografías originales, pero, aún así, suele ser conveniente cargar `\usepackage{lmodern}` si se va a usar la opción `T1` de fontenc, pues las tipografías lmodern (denominadas Latin Modern) —que por otra parte tienen el mismo aspecto que las originales de TeX— tienen más calidad que las que LaTeX emplea por defecto con `T1`. Aunque para bastantes matemáticos usuarios habituales de LaTeX resultaría ciencia ficción, se pueden utilizar muchas otras tipografías si uno lo desea (simplemente cargando otro paquete en lugar de lmodern); eso está fuera de lo que aquí se pretende explicar pero, por ejemplo, prueba poniendo `\usepackage{kpfonts}` o `\usepackage{fourier}` en cualquier archivo `.tex` que tengas y verás cómo cambia el aspecto tipográfico de todo el documento.

- Al usar `\usepackage[spanish]{babel}` le estamos diciendo a LaTeX que el documento está en español. Eso hace que, por ejemplo, el comando `\today` saque la fecha en español y no en inglés (que es lo que asume por defecto si no se carga babel), y que en la bibliografía ponga automáticamente «Bibliografía» en vez de «Bibliography». Por otra parte, `\decimalpoint` o `\decimalcomma` se encargan de decirle a spanish si, para escribir números decimales en la salida `.pdf`, tiene que usar punto o coma como separador decimal. Independientemente de si uno quiere conseguir «3.14» o «3,14», en el archivo `.tex` hay que escribir `$3.14$`; al generar el `.pdf`, la opción spanish de babel te lo sacará como «3.14» o «3,14» según tengas `\decimalpoint` o `\decimalcomma` en el preámbulo (es decir, antes del `\begin{document}`). Actualmente, la Real Academia Española recomienda usar el punto como separador decimal, y además en matemáticas es muy conveniente pues es lo que usan todos los lenguajes de programación y paquetes de cálculo simbólico. spanish

tiene muchas otras opciones que puedes usar leyendo el manual (además de en tu disco duro, lo tienes en [3]). El responsable actual de babel (y de spanish) es el español Javier Bezos; es de formación humanística, y es, sin duda, una de las personas que más sabe de LaTeX del mundo (sirva este comentario para quitarte de la cabeza que LaTeX solo se usa para escribir matemáticas).

En documentos de contenido matemático, es muy conveniente incorporar también los paquetes `amsmath` (entre otras cosas, proporciona muchas opciones para escribir fórmulas que ocupan varias líneas, y facilita la definición de operadores matemáticos), `amsthm` (permite definir entornos de tipo teorema) y `amssymb` (proporciona muchos símbolos adicionales: en particular, permite escribir $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$ o $\mathbb{C}$ mediante `\mathbb{N}` o similar). Así se cargan los tres a la vez:

```
\usepackage{amsmath, amsthm, amssymb}
```

Más adelante en este escrito hablaremos un poco más de alguno de estos paquetes. Para ver su sintaxis, consulta el manual; o, también, el documento [8].

3. Algunos paquetes útiles

LaTeX es modular, no carga todo de golpe, solo lo que necesita para su funcionamiento básico; si quieres cargar cosas extras —«paquetes», en la nomenclatura de LaTeX— hay que decírselo. Eso era muy importante en el pasado, cuando los ordenadores eran mucho menos potentes que ahora, y esta parquedad se mantiene, entre otras cosas, por compatibilidad. Pero, aunque a los ordenadores actuales les sobre tanto potencia como memoria, no hay que cargar paquetes a lo loco, pues puede haber conflictos entre unos y otros que actúen sobre las mismas órdenes internas de LaTeX; sobre todo, si se cargan paquetes «raros». Es muy mala idea acostumbrarse a cargar un montón de paquetes que no se sabe para qué sirven, y copiarlos de unos archivos `.tex` a otros.

Hay algunos paquetes muy útiles, que te pueden venir bien. Comentaremos unos cuantos. De todos ellos hay un manual, y todas las distribuciones de LaTeX incluyen la documentación de todos sus paquetes; la tendrás en tu disco duro, o la puedes buscar en internet.

- fancyhdr: Digamos que, en opinión de la mayoría, las cabeceras de los documentos escritos con la clase article quedan bastante feas. Con el paquete fancyhdr (o con titleps que comentaremos algo más adelante), las cabeceras y pies se pueden configurar para que queden como uno prefiera.

- tocbibind: Cuando en LaTeX generamos un índice con `\tableofcontents`, en dicho índice solo aparecen los capítulos o secciones que tienen numeración (por ejemplo, no estarán los que estén generados con `\chapter*` o `\section*`), y tampoco aparece la bibliografía, que tampoco se numera. La manera más sencilla de conseguir que todo esto aparezca en el índice es cargar el paquete tocbibind. Si no quieres que el propio índice quede reflejado en el índice (puede parecer redundante), basta indicárselo al paquete con una opción: `\usepackage[nottoc]{tocbibind}`.

- graphicx: Permite cargar fotos o gráficos hechos con otros programas. Por ejemplo, si queremos cargar el archivo `foto.jpg` de forma que tenga 7 cm de alto (el ancho se escalará solo para no perder la proporcionalidad), bastará poner `\includegraphics[heigth=7cm]`

`{foto.jpg}`. En principio, graphicx admite archivos con formato `.jpg`, `.png` o `.pdf`. (Nota: graphicx es una versión actualizada del paquete graphics, cuya existencia se mantiene por compatibilidad, dado que su sintaxis difiere en algunos puntos.)

- caption: Se usa para configurar el aspecto de las notas que se ponen (mediante el comando `\caption`) en los pies de figuras y cuadros. Por ejemplo, en este documento se ha usado el paquete caption configurado con `\captionsetup{margin=15pt, font=small, labelfont=bf, labelsep=colon}`.

- geometry: Usar este paquete es la manera más cómoda de diseñar la página (ancho y alto del papel y del texto, espacios para cabecera y pie, etcétera) si no queremos usar el diseño que se obtiene con los ajustes por defecto (ni con las opciones ya previstas). Si solo se desea que el texto ocupe casi todo el papel para, al imprimir, tener menos páginas, no hace falta cargar geometry y pelearse con sus opciones, es suficiente usar `\usepackage{fullpage}`.

- hyperref: Sencillamente impresionante, muy útil y fácil de usar. Sin más que cargarlo, hace que todas las llamadas internas con `\label`/`\ref`, o las de la bibliografía, etcétera, funcionen como enlaces en el archivo `.pdf`. Además, con `\url{...}` se pueden escribir direcciones de páginas web que no solo serán hiperenlaces `.pdf`, sino que automáticamente se partirán en varias líneas si a LaTeX le resulta conveniente hacerlo para formar párrafos; también permite incluir en las URL caracteres reservados de LaTeX pero que resultan habituales en las páginas web, como `$`, `%` o `~`. Realmente, salvo el hecho de que en el `.pdf` funcionen los hiperenlaces, lo mismo se puede conseguir, y con la misma sintaxis, con el paquete url. Si lo que estás escribiendo no está destinado exclusivamente a ser impreso, deberías cargar el paquete hyperref siempre.

- enumitem: Proporciona mucha flexibilidad para adaptar y personalizar entornos `enumerate`, `itemize` y `description`, así como para definir otro tipo de listas.

- titlesec: Permite rediseñar por completo el aspecto de los títulos de capítulos y secciones. Viene acompañado de titleps, que sirve, como fancyhdr, para configurar el aspecto de cabeceras y pies de páginas; y de titletoc, que está destinado a configurar el aspecto del índice. Esta guía usa los paquetes titlesec y titleps.

- pdfpages: Permite incluir, en un documento elaborado con LaTeX, documentos `.pdf` ya disponibles previamente, de una o varias páginas. En particular, para añadir a nuestro documento una portada generada con otros medios, bastaría cargar el paquete y usar `\includepdf{portada.pdf}`.

- longtable: Se usa para escribir tablas que ocupen varias páginas. Un paquete similar es supertabular.

- bookstabs: Mejora la calidad de las tablas si las comparamos con las que genera LaTeX por defecto, y añade muchas posibilidades. Es compatible con longtable.

- color: Permite escribir texto en color; por ejemplo, usando `\textcolor{blue}{...}`, el texto que haya en el lugar de `...` aparecería azul en el `.pdf`. Mucho más potente es el paquete xcolor.

- tikz: Para hacer todo tipo de gráficos utilizando exclusivamente código LaTeX. Desarrollado por Till Tantau, su nombre Ti*k*Z es una autorreferencia de siglas de las que tanto gustan en informática, y significa algo así como «Ti*k*Z no es un programa de dibujo» (en alemán). Internamente, está basado en un lenguaje de nivel más bajo denominado PGF (por *Portable Graphics Format*), por lo que a menudo se le conoce como PGF/Ti*k*Z. Su potencia es impresionante, pero el manual [20] de más de mil páginas atemoriza; afortunadamente, los primeros capítulos sirven como resumen de todo el manual, y el resto se puede usar como consulta (tiene cientos de ejemplos). Para aprender a usarlo, merece la pena comenzar leyendo algún resumen aún más básico, como los de [15, 16]. También puede resultar útil el resumen visual recogido en [6]. En la página `https://texample.net/tikz/examples/` hay varios centenares de preciosos gráficos elaborados con Ti*k*Z, junto con su código, pero la mayoría son demasiado complicados: son una muestra del tipo de gráficos que se pueden hacer, pero no son útiles para aprender. Por otra parte, hay decenas de paquetes especializados que se apoyan en Ti*k*Z y lo extienden para usos gráficos muy diversos (circuitos eléctricos, dibujos de animales, ornamentos, juegos...). Si se usa tikz a la vez que babel, puede ser útil añadir `\usetikzlibrary{babel}` tras cargar tikz; eso evita algunas incompatibilidades.

- pgfplots: Está basado en PGF/Ti*k*Z y sirve para representar gráficamente funciones matemáticas usando sus ecuaciones (o tablas de datos), en dos y tres dimensiones, y con todo tipo de opciones. Su desarrollador es Christian Feuersänger, y el manual [9] tiene más de 500 páginas, con cientos de ejemplos. En la página web `https://pgfplots.net` se pueden encontrar muchos más, con el correspondiente código para generarlos. El paquete pgfplots viene acompañado de otro denominado pgfplotstable, que puede resultar útil para elaborar tablas de datos.

- algorithm2e: Un paquete destinado a escribir algoritmos con aspecto de lenguajes de programación o pseudocódigo. (Hay otros paquetes con parecidas intenciones, como algpseudocode o la pareja formada por algorithmic y algorithm.)

- listings: Permite escribir listados de código de numerosos lenguajes de programación.

Hay algunos paquetes que no están destinados a ser usados en la versión final de lo que estamos escribiendo, sino que, fundamentalmente, sirven de ayuda en el proceso de escritura (al final, se pueden quitar o comentarlos con `%` para que no actúen). En este sentido, podemos señalar estos tres:

- showkeys: Por ejemplo, con `\usepackage[notcite,notref]{showkeys}`, en el `.pdf` se muestran, en un recuadro, las `\label` que tengamos puestas en el archivo `.tex`. Esta ayuda es útil para tener a la vista cuál es la `\label` que hemos puesto en cada sitio cuando estamos escribiendo y queremos citarla.

- backref: En cada `\bibitem` que tengamos en la bibliografía, se muestra en qué página del documento se cita. Sirve para darse cuenta de si tenemos bibliografía no citada (por supuesto, podemos mantener ese tipo de información en la versión final de lo que estemos escribiendo si así lo deseamos, pero no es lo habitual). Si también se usa hyperref, el paquete backref hay que cargarlo después.

- refcheck: En el `.pdf` se muestran todas las `\label` y `\bibitem` que hayamos incluido en el archivo `.tex`; las que no se citan aparecen señaladas con interrogaciones. Esta información es útil ya que lo habitual es numerar solo las ecuaciones que se citan en el texto.

No queremos abandonar esta sección sin mencionar BEAMER. No es un paquete en sí, sino una clase —al mismo nivel que article o book, por ejemplo—, y está destinada a hacer presentaciones con un cañón de vídeo. En el pasado, había otros paquetes con la misma finalidad (y se pueden seguir usando ahora). Pero, desde que Till Tantau (el mismo que desarrolló Ti*k*Z) presentó la primera versión de BEAMER en 2003, rápidamente se convirtió en el método más usado (y casi único) para hacer presentaciones con LaTeX. El manual [21] —que comienza imaginando que Euclides va a presentar, en un congreso, la demostración de la existencia de infinitos números primos— contiene numerosos ejemplos con personalizaciones para el aspecto de la presentación.

Basado en BEAMER, también existe el paquete beamerposter, destinado a diseñar pósteres para presentarlos en congresos científicos.

4. Referencias cruzadas, bibliografía e hiperenlaces

Para que todo funcione bien y sin errores —sobre todo, cuando se van haciendo cambios en el documento—, es importante no numerar nada a mano. Hay que numerar los teoremas (o entornos similares), las ecuaciones que se van a citar y las referencias de la bibliografía con los mecanismos automáticos de LaTeX, y citarlos también con los métodos previstos. De lo contrario, si algo es, por ejemplo, el «teorema 13», y lo citamos como tal escribiendo `teorema 13` en el archivo `.tex`, pero en una versión posterior añadimos otro teorema antes del 13, ese teorema 13 pasará a ser el 14, y la referencia `teorema 13` ya no aludirá al teorema correcto.

Una vez que tenemos cargado el paquete amsthm, los entornos de tipo teorema deberían ser algo similar a esto:

```
\theoremstyle{plain}
\newtheorem{teo}{Teorema}[chapter]
\newtheorem{cor}[teo]{Corolario}
\newtheorem{lema}[teo]{Lema}
\newtheorem{prop}[teo]{Proposición}

\theoremstyle{definition}
\newtheorem{defi}[teo]{Definición}
\newtheorem{ejemplo}[teo]{Ejemplo}
\newtheorem*{conjetura}{Conjetura}

\theoremstyle{remark}
\newtheorem{nota}[teo]{Nota}
```

Aquí, el `\theoremstyle` establece con qué estilo tipográfico van a aparecer (en el `.pdf`) los entornos de tipo teorema. Por ejemplo, con `plain`, el texto «Teorema número» aparecerá en negrita, y el cuerpo del teorema en itálica (además de `plain`, `definition` y `remark` que están predefinidos, se pueden definir otros estilos, pero no es habitual hacerlo). Lo que aparece entre corchetes es opcional, y se puede usar o no dependiendo de lo que prefiramos. El `[chapter]` indica que la

numeración de los teoremas «cuelga» de la de los capítulos; así, el primer teorema del capítulo 4 aparecerá como «Teorema 4.1». Y el `[teo]` que hay en casi todos los `\newtheorem` indica que deben tener el mismo contador de numeración que el de `\newtheorem{teo}`; esto te puede gustar más o menos, pero, si cada entorno lleva su propia numeración independiente, es mucho más complicado ubicar dónde se encuentran cuando se citan. Finalmente, el `\newtheorem*` que se usa para las conjeturas indica que estas deben aparecer sin numerar.

Así las cosas, para enunciar un teorema utilizaríamos esto:

```
\begin{teo}[Euclides, ca. 300 a. C.]
\label{t:euclides}
No existe ningún número primo que sea el mayor.
\end{teo}
```

(el argumento entre corchetes es opcional, y aparecerá en el `.pdf` entre paréntesis). Si en el texto se quiere aludir a este teorema se hará mediante `teorema~\ref{t:euclides}` (el `~` es un espacio en el que no se permite un cambio de línea), y LaTeX se encargará de asignarle el número correspondiente. Por cierto, en inglés lo habitual sería escribir `Theorem~\ref{t:euclides}` (con mayúsculas), pero en español lo recomendado es no usar mayúsculas; si prefieres usarlas para este tipo de cosas no hay demasiado problema, pero ¡sé consistente!, no lo hagas unas veces sí y otras no en el mismo escrito.

Las demostraciones de los teoremas, proposiciones, etc., se escriben con `\begin{proof}`/`\end{proof}`. LaTeX se encarga de poner la palabra «Demostración» (si estamos usando spanish) y de añadir un símbolo de final de demostración cuando esta acaba (si la demostración acaba con una fórmula centrada se recomienda usar el comando `\qedhere` para conseguir que dicho símbolo aparezca en la línea de la ecuación, no en la siguiente).

Para numerar una ecuación y citarla más adelante, se hace así:

```
\begin{equation}
\label{eq:euler}
e^{i\theta} = \cos(\theta) + i \sen(\theta), \quad \theta \in \mathbb{R}.
\end{equation}
```

Y luego se alude a ella usando `(\ref{eq:euler})` o, mejor, `\eqref{eq:euler}` (en el `.pdf`, esto genera el número de la ecuación entre paréntesis). Si las ecuaciones son mucho más complicadas y no caben en una línea, el paquete amsmath dispone de muchos entornos distintos para escribir este tipo de ecuaciones, numeradas o no, entre ellos: `align`, `align*`, `aligned`, `split`, `multline`, `multline*`, etcétera. LaTeX sin amsmath solo tiene un comando previsto para eso —`eqnarray`—, pero se considera obsoleto y no conviene usarlo (en particular porque, en cuanto al espaciado, el aspecto que da a las ecuaciones construidas con él no es consistente con el de las ecuaciones que no lo usan). Si tienes ecuaciones de bastantes líneas, es una buena idea decirle a LaTeX que, si le viene bien, puede cortar página entre dos líneas de estas ecuaciones; se consigue incluyendo el comando `\allowdisplaybreaks` en el preámbulo de tu documento (si lo haces y quieres evitar un corte entre dos líneas concretas de una fórmula, usa `\\*` en vez de `\\`).

En cualquier trabajo científico son muy importantes las referencias (la bibliografía). Lo normal es que sean referencias a artículos o libros científicos que el lector pueda consultar. Se pueden citar páginas web de manera complementaria, pero si casi todo lo que citas son páginas web (incluida la *Wikipedia*), piensa que no lo estás haciendo bien; aparte de que la fiabilidad de una página web como fuente científica puede dejar mucho que desear, las páginas web pueden

cambiar su contenido, o su URL de acceso, e incluso pueden desaparecer completamente. Intenta reducir al mínimo las citas que sean, exclusivamente, una página web (eso no quiere decir que, si se cita un artículo de una revista científica, no se puede proporcionar la URL en el que se accede a la revista o al artículo).

En matemáticas no hay ningún formato «oficial» para las referencias. Cada revista y/o cada editorial usa el que considera oportuno (más allá de que, casi siempre, las referencias se ordenan alfabéticamente según los autores). Así que, por ejemplo en un trabajo de fin de grado de matemáticas, no hay ninguna norma preestablecida a la que atenerse, salvo una: ¡hay que ser coherente! Si los títulos de los libros se ponen en itálica, todos tienen que estar en itálica; si los autores los escribimos como «N. Apellido», hay que hacerlo siempre así, no unas veces de esa forma y otras «Apellido, N.»; si entre el autor y el título se pone una coma, siempre hay que poner una coma; etcétera.

Por otra parte, los nombres de las revistas no suelen escribirse enteros, sino abreviados. Y las abreviaturas de las revistas científicas sí que son —más o menos— estándar. En matemáticas, lo más habitual es seguir las abreviaturas de las bases de datos *MathSciNet* o *zbMath*; en `https://mathscinet.ams.org/msnhtml/serials.pdf` se puede acceder a las abreviaturas de varios miles de revistas.

Básicamente, hay cuatro tipos de documentos que se pueden citar: artículos, libros, capítulos de libros (o de actas de congresos), y «otros». Al menos para los tres primeros hay que tener claro qué estilo se quiere seguir; en las referencias de este escrito, [1, 4, 5, 11, 13, 14, 17] son libros (todos tienen el título en itálica, la editorial y el año de publicación), [12, 16] son artículos en revistas (con el nombre de la revista en itálica, el volumen en negrita, el año de publicación entre paréntesis, y todos los datos escritos en el mismo orden y con mismo formato), y [10, 22] son capítulos de libros (muy habitualmente, artículos procedentes de congresos, aunque eso no siempre aparece reflejado explícitamente en la referencia); estos dos no tienen nada que ver con LaTeX, están puestos solo para que tengas algún ejemplo. Con «otros» puede haber mucha variedad, y se hace lo que se puede, intentando ser coherente. El formato mostrado aquí para las referencias es el que se usa en *La Gaceta de la Real Sociedad Matemática Española*; pero no tienes por qué seguirlo si no te apetece: ¡hazlo a tu gusto!

Veamos ahora cómo está escrita la bibliografía, y cómo se cita; hay otras maneras de hacerlo, pero explicaremos solo la más habitual en nuestro campo. Lo vamos a hacer con las dos referencias a publicaciones de Donald E. Knuth. Lo que tenemos es lo siguiente (los `...` aluden a todo lo que falta aquí, claro):

```
\begin{thebibliography}{99}

...

\bibitem{Kn-MT}
\textsc{D. E. Knuth},
Mathematical typography,
\textit{Bull. Amer. Math. Soc. (N.S.)}
\textbf{1} (1979), 337--372.

\bibitem{Kn-book}
\textsc{D. E. Knuth},
\textit{The \TeX book},
```

```
Addison Wesley, 1984.

...

\end{thebibliography}
```

Ya hemos comentado antes aspectos sobre el formato, y no incidiremos más en ello. La bibliografía se comienza con `\begin{thebibliography}` y se acaba con `\end{thebibliography}`. El `{99}` tras el `\begin{thebibliography}` sirve para indicarle a LaTeX cuánto espacio tiene que reservar para las etiquetas (de [1] a [22], en nuestro caso), de modo que la bibliografía en sí quede alineada: como tenemos entre 10 y 99 referencias, LaTeX tiene que reservar espacio para dos dígitos, y eso es lo que indica el `{99}` (daría lo mismo haber puesto, por ejemplo, `{00}`, pues en LaTeX con las tipografías habituales todos los dígitos ocupan el mismo espacio, no así las letras). Aparte de eso, lo único que merece la pena señalar es que las etiquetas de esas dos referencias las hemos puesto como `\bibitem{Kn-MT}` y `\bibitem{Kn-book}`, y que para citarlas se usa `\cite{Kn-MT}` y `\cite{Kn-book}`. Como los libros son mucho más largos que los artículos, al citar un libro suele ser conveniente aludir a una parte de él; esto se consigue, por ejemplo, con `\cite[capítulo~18]{Kn-book}`, que en el `.pdf` aparecerá como «[13, capítulo 18]».

LaTeX viene acompañado de algunos programas externos destinados a algunas tareas específicas (su uso no solo requiere cargar algún paquete, sino manejar algún programa que no es LaTeX, aunque esto puede estar más o menos camuflado). Uno es BibTeX, cuya función es el tratamiento de bibliografías. A muchos les encanta; otros, no lo solemos usar. Si quieres usarlo, magnífico; pero, sobre todo si estás empezando con LaTeX, seguramente no merece la pena hacer este esfuerzo adicional justo ahora.

Otro programa de los que acompaña a LaTeX es *MakeIndex*, destinado a generar índices alfabéticos, que se colocan al final del texto y en los que se puede buscar en qué páginas del libro aparecen los términos allí recogidos. Prácticamente cualquier libro científico debería tener uno (no los artículos, que son mucho más cortos); si no, no es un buen libro útil para el estudio y/o la consulta. Seguramente, en un trabajo de fin de grado no hay ninguna necesidad de que incluyas un índice alfabético pero, si quieres hacerlo, adelante, no es difícil. Únicamente una recomendación: el paquete esindex facilita muchísimo la generación de índices alfabéticos en español, pues hace que funcionen, automáticamente, las reglas de ordenación alfabética del español (no se distingue entre vocales acentuadas y sin acentuar, y la «ñ» va después de la «n»).

Aunque TeX es un lenguaje de programación completo (una sistema Turing completo, que tiene el poder equivalente a una máquina de Turing universal, si queremos decirlo de manera más precisa), la existencia de estos programas externos está motivada por la dificultad de programar algunas tareas con el lenguaje propio de TeX, o por lo lentas que resultarían (sobre todo, hace años). Con el tiempo, posiblemente muchos de estos programas externos tan íntimamente relacionados con LaTeX se reescribirán en Lua —ya ha ocurrido con alguno—, y entonces, gracias a LuaTeX, ya no hará falta recurrir a ellos.

5. Objetos flotantes

En el mundo de TeX/LaTeX, en el que lo ideal es que el propio LaTeX se encargue de generar todas las páginas sin que el autor fuerce ningún tipo de corte de página, por ejemplo, los objetos flotantes juegan un papel crucial. Y muy sorprendente, si no te has topado antes con ello. Supón

Figura 1: Un borrón para mostrar un ejemplo.

que estás en una página en la que todavía caben 5 cm (en vertical) de texto, pero que quieres colocar una imagen o una tabla que ocupa 8 cm: ¡está claro que no cabe! ¿Qué hacer, dejar 5 centímetros en blanco y pasar a la siguiente página? Queda horroroso. ¿Reestructurar el texto y/o «enrollarse» más para rellenar el hueco, o resumir para conseguir más hueco? Es una idea nefasta, y el cuento de nunca acabar (aparte de que el contenido no se debe supeditar al aspecto, las «trampas» que hayamos hecho para conseguir meter en algún sitio algo que no cabía habrá que deshacerlas o rehacerlas si más tarde hacemos cambios en el documento).

La solución es fácil: basta permitir a LaTeX que «deje flotar» la imagen o la tabla que queremos colocar, y que la ponga en algún sitio (cercano) donde quepa. Esto requiere un cambio de mentalidad al redactar. Ya no se puede decir «Mira esta foto:», pues quizás la foto no va a estar ahí porque no cabía y ha «flotado». En su lugar, hay que redactar diciendo «Mira la figura 7.» (o el número que tenga). Y la foto aparecerá en un lugar cercano con una breve explicación al pie. Esto se maneja como sigue. En lugar de una foto —un archivo externo que se cargaría con `\includegraphics`, pero que en este escrito no necesitamos para nada—, vamos a colocar un rectángulo negro generado con la orden `\rule`:

```
\begin{figure} % Se puede añadir [h], [!h], [tb] [p], etc.
\centering
% \includegraphics[width=7cm]{foto.jpg} % Ejemplo que podría ser real
\rule{6cm}{3.2cm}
\caption{Un borrón para mostrar un ejemplo.}
\label{fig:borron}
\end{figure}
```

Y, en el archivo `.tex`, aludiremos a esta figura llamándola `figura~\ref{fig:borron}` (LaTeX la colocará donde considere oportuno y la numerará automáticamente, claro). En este escrito, podemos ver esta preciosa imagen en la figura 1 (observa además, si estás leyendo en el `.pdf`, que esto es un hiperenlace, pues tenemos cargado el paquete hyperref).

La imagen «flota» gracias al `\begin{figure}`/`\end{figure}`; si, por algún motivo, lo que quieres es impedir que flote (ya hemos comentado que, en general, no es buena idea), pon la imagen dentro de un `\begin{center}`/`\end{center}`, sin utilizar el entorno `figure`.

A LaTeX le podemos indicar dónde preferimos que coloque la figura, usando los argumentos opcionales `h` (*here*, aquí), `t` (*top*, en la parte de arriba de una página), `b` (*bottom*, en la parte de abajo de una página) y `p` (*page*, en una página suelta, quizás juntándola con otras figuras), que se colocan entre corchetes tras el `\begin{figure}`; pero, por mucho que le insistas, no te va a hacer caso si no cabe. Además, añadiendo `!`, le podemos pedir a LaTeX que haga todo lo posible para colocar la figura donde le estamos diciendo, incluso saltándose alguno de sus parámetros estéticos preestablecidos.

A decir verdad, las clases article y book provienen de una época en la que el material gráfico se usaba menos que ahora, y son demasiado estrictas en cuanto a sus consideraciones estéticas para colocar objetos flotantes (cuántos puede poner en una página, qué tanto por ciento de página puede dedicar a objetos flotantes, etc.). Esto es un problema grave si tenemos muchas figuras, pues LaTeX no encuentra dónde ponerlas y las coloca todas al final; queda horroroso. Así pues, es muy conveniente modificar los parámetros que usan article y book para colocar objetos flotantes. Como las clases amsart y amsbook no tienen ese problema, lo más sencillo es añadir, en nuestro archivo `.tex`, los parámetros que se usan allí; basta copiarlos y pegarlos desde `amsart.cls`. Son estos, que te aconsejamos poner en tu documento si usas article o book:

```
\setcounter{topnumber}{4}
\setcounter{bottomnumber}{4}
\setcounter{totalnumber}{4}
\setcounter{dbltopnumber}{4}
\renewcommand{\topfraction}{.97}
\renewcommand{\bottomfraction}{.97}
\renewcommand{\textfraction}{.03}
\renewcommand{\floatpagefraction}{.9}
\renewcommand{\dbltopfraction}{.97}
\renewcommand{\dblfloatpagefraction}{.9}
```

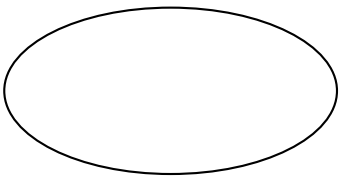
Lo que hemos explicado hasta ahora para figuras se aplica también a las tablas (salvo que usemos tablas de las que se pueden cortar entre varias páginas, para las que podemos usar los paquetes longtable o supertabular). La única diferencia es que, en lugar del entorno `figure`, se usa el entorno `table` (eso es lo que las hace flotar; si no quieres una tabla que flote basta colocarla dentro de un entorno `center`). Además, cada uno de esos dos entornos lleva su propia numeración, con dos contadores distintos, que generan las denominaciones «figura número» y «cuadro número». Para construir la tabla en sí se usa el entorno `tabular`. Observa una, que hace uso del paquete booktabs y tiene líneas horizontales de distinto grosor (sin este paquete, LaTeX solo dispone, para generar líneas horizontales, de los comandos `\hline` y `\cline`):

```
\begin{table}
\centering
\begin{tabular}{llr} \toprule
\multicolumn{2}{c}{Artículo} \\ \cmidrule(r){1-2}
Animal   & Descripción & Precio (\$) \\ \midrule
Mosquito & por gramo & $13.65$ \\
         & cada uno  & $0.01$ \\
Tigre    & huesos triturados & $92.50$ \\
Murciélago & vivo    & $33.33$ \\
Armadillo  & seco (para desalar) & $8.99$ \\ \bottomrule
\end{tabular}
\caption{Precios del mercado de animales de Wuhan en diciembre de 2019.}
\label{tab:Wuhan}
\end{table}
```

Ya hemos advertido que esto no es un manual de LaTeX, así que no aclararemos qué es cada cosa, pero seguramente puedes imaginártelo (el resultado es lo que aparece en el cuadro 1).

Artículo		
Animal	Descripción	Precio ($)
Mosquito	por gramo	13.65
	cada uno	0.01
Tigre	huesos triturados	92.50
Murciélago	vivo	33.33
Armadillo	seco (para desalar)	8.99

Cuadro 1: Precios del mercado de animales de Wuhan en diciembre de 2019.



Por último en este apartado, vamos a mencionar el paquete wrapfig (o wrapfig2, que tiene alguna posibilidad más), que dispone de un entorno `wrapfigure` que sirve para colocar pequeñas figuras en el lateral de un párrafo, como hemos hecho, aquí a la derecha, con una pequeña elipse. Es un paquete «peligroso», pues puede dar lugar a sorpresas desagradables; en particular, si el párrafo en el que se usa comienza muy abajo en la página y la figura completa no cabe antes del final de esa página. El paquete tiene opciones que permiten que la pequeña figura «flote» hasta el lateral de los siguientes párrafos (de hecho, sigue funcionando el mecanismo de `\caption`, `\label` y `\ref`), pero aun así hay que cerciorarse siempre —en la versión final de lo que estemos escribiendo— de que la figura ha quedado bien colocada. ¡Ah!, y esto no se puede usar delante de bastantes entornos habituales de LaTeX, como listas o teoremas. Las rutinas de generación de páginas de LaTeX no están bien adaptadas para poder manejar este tipo de cosas de manera completamente automática.

6. Miscelánea

En esta última sección daremos algunas recomendaciones generales sobre redacción, así como consejos de todo tipo, y señalaremos algunos errores habituales para intentar evitarlos.

6.1. Redacción correcta

Igual que si estuviésemos haciendo una redacción o escribiendo una novela, las matemáticas se redactan. Y hay que usar las reglas gramaticales del castellano (o del idioma en el que estemos escribiendo, claro). No se pueden escribir frases sin verbo, sin símbolos de puntuación adecuados, etc. Repasa las frases que escribas prestando atención a ver si se entienden. Redactar bien requiere tener cuidado y emplear tiempo. A todos nos cuesta hacerlo. Una de las mayores aportaciones de los trabajos de fin de grado de matemáticas es que quien lo hace tiene que aprender a escribir matemáticas, algo que posiblemente no haya practicado en toda la carrera; ¡aprovéchalo si estás leyendo esto porque quieres escribir el tuyo!

Muchas frases típicas de las matemáticas se escriben, a menudo, mal. Por ejemplo, la construcción «Sea $a \in \mathbb{R} \setminus \{0\}$, entonces $a^2 > 0$.» es incorrecta, deberían ser dos frases distintas, separadas por un punto o un punto y coma. En cambio, sí que es correcto escribir «Si $a \in \mathbb{R}\setminus\{0\}$, entonces $a^2 > 0$.».

Otro error frecuente es hablar de «resolver» una raíz cuadrada, un polinomio o una fórmula; se resuelven las ecuaciones o los problemas. Resolver no es lo mismo que calcular o evaluar, por ejemplo. En un trabajo serio, emplea el lenguaje con propiedad.

Asimismo, y aunque en la pizarra o escribiendo apuntes de manera rápida se usen a menudo los símbolos lógicos $\exists$ y $\forall$ («existe» y «para todo»), lo adecuado es no usarlos en un texto matemático bien redactado (salvo que sea un texto de lógica, claro). Y, en principio, tampoco hay que emplear símbolos $\Rightarrow$ (o similares) con el significado de «implica», sobre todo dentro de texto ordinario (a veces sí es aceptable en una fórmula centrada). Así, una redacción correcta de la definición de función continua en un punto sería esta:

> *Decimos que* $f : \mathbb{R} \to \mathbb{R}$ *es continua en* x_0 *si, para todo* $\varepsilon > 0$*, existe* $\delta > 0$ *tal que* $|f(x) - f(x_0)| < \varepsilon$ *cuando* $|x - x_0| < \delta$*.*

(Desde luego, habría que precisar algo más si la función no está definida en todos los reales, pero este escrito no es ni un manual de LaTeX ni un manual de matemáticas.)

Conviene asimismo mencionar que hay que evitar que las frases comiencen por una expresión matemática. Habitualmente, esto se soluciona con un pequeño cambio de redacción: en vez de escribir «f es una función continua.» podríamos poner «La función f es continua.».

Comentemos finalmente que no hay que abusar de las mayúsculas. Es imposible dar una normativa detallada para esto, pero en español no hay por qué escribirlas en los tratamientos (doctor, profesor, presidente), en los nombres de los meses, en los de los teoremas (teorema de la función implícita, teorema de los ceros de Hilbert, lema de Fatou), en jacobiano, laplaciano o noetheriano, cada vez que aparezcan las palabras «matemáticas», «geometría» o «física» —por dar unos pocos ejemplos—, ni en «universidad» salvo que se use en el nombre propio de una universidad concreta.

6.2. Final de párrafo y espacios verticales

En un documento `.tex`, un retorno de carro es lo mismo que un espacio, y dejar varios espacios seguidos es lo mismo que dejar solo uno. Realmente, esto es lo más habitual en cualquier lenguaje de programación.

Un fin de párrafo (un punto y aparte), se consigue dejando una línea vacía en el archivo `.tex`; es decir, poniendo dos «retornos de carro» (da igual si hay más de una línea vacía seguida). También se puede usar `\par`, pero no es lo habitual.

Un error común es usar `\\` para acabar un párrafo. ¡Nunca hay que hacer eso! `\\` está pensado para cambios de línea que no son cambios de párrafo, dentro de tablas, en las filas de una matriz, en una fórmula de varias líneas, etc. (También al escribir poesía.)

Hay quien acaba los párrafos escribiendo `\\` y dejando, a continuación, una línea vacía porque eso tiene, como efecto colateral en el sitio donde se hace, un aumento de espaciado entre párrafo y párrafo. Pero cambiar el espacio entre párrafos es muy sencillo sin más que cambiar, en el preámbulo, el valor asignado a la variable `\parskip`. Eso sí, hay que hacerlo permitiéndole a LaTeX que pueda ensanchar o estrechar un poco los espacios verticales para conseguir páginas bonitas. Por ejemplo, con

```
\parskip = 2pt plus 1.5pt minus 1.0pt
```

le estaríamos diciendo a LaTeX que, normalmente, deje `2pt` de espacio adicional entre párrafos, y que, si le viene bien, deje algo más (`1.5pt`) o algo menos (`1.0pt`). Lo mismo que aquí hemos

empleado el «punto» (`pt`) como unidad de medida, se pueden emplear muchas otras, como centímetros (`cm`), milímetros (`mm`), la anchura de una «m» (`em`), o la altura de una «x» (`ex`); estas dos últimas dependen del tipo y del tamaño de letra que estemos utilizando, claro.

En general, no es buena idea añadir espacios verticales en un documento LaTeX. El espaciado vertical debería venir dado por la estructura (es decir, por comandos de tipo `\section`, `\subsection`, `\subsubsection`, listas, teoremas, entornos de todo tipo, etc.), no por espacios añadidos «a mano». Si por algún motivo lo haces, al menos no uses órdenes como `\vspace{1cm}`, pues de ese modo estás impidiendo que LaTeX use la flexibilidad que necesita para ensanchar o estrechar un poco los espacios verticales. En su lugar, usa comandos como `\bigskip`, `\medskip` o `\smallskip`, que tienen la flexibilidad incorporada.

6.3. Espacios que no cortan línea

No es visualmente agradable que un párrafo acabe con una línea con un solo carácter. Por ejemplo, imaginemos que el final de un párrafo es «Y esto prueba la continuidad de f.». Podría ocurrir que, cuando LaTeX construye el `.pdf`, la última línea del párrafo sea «f.». Esto es fácil de evitar, basta usar un «espacio que no rompe línea» (el carácter `~`); así pues, lo conveniente sería acabar la frase anterior así: `de~$f$.` Es una buena idea acostumbrarse a hacerlo siempre, no solo si vemos, en el `.pdf`, que realmente se ha producido ese corte de línea indeseado.

Hay otras situaciones en las que no se desea que pueda producirse un corte de línea, y de nuevo conviene incluir un `~`. Por ejemplo, en `N.~Apellido`. O entre un teorema o una figura (u otros entornos numerados) y su número cuando se cita: `teorema~\ref{t:euclides}`.

6.4. Guiones

En LaTeX disponemos de guiones de tres tamaños diferentes, que se escriben mediante `-`, `--` y `---`. Salvo en los casos que vamos a comentar a continuación, lo normal es usar `-`; por ejemplo, en «austro-húngaro». También se usa `-` para el signo menos, pero dentro de entornos matemáticos, como en «-1» (escrito `$-1$`), que nunca debe aparecer como «-1», que es lo que ocurriría si escribiésemos `-1` sin estar en modo matemático.

Se suele usar `--` (que da lugar a «–» y se denomina «raya corta») para indicar intervalos de páginas («21–30») o años, o en otras situaciones en las que hay un principio y un final («tren Logroño–Zaragoza»).

El guion de mayor tamaño se consigue con `---` (genera «—» y se denomina «raya»). Se usa en diálogos, también para marcar cada uno de los elementos de una lista si estos se van escribiendo en líneas independientes (con este cometido también se emplean otros símbolos, como topos •), y como símbolo delimitador en incisos. Con este último uso es con el que hay que tener más cuidado.

Cuando LaTeX construye párrafos, tiene que decidir dónde cortar líneas. Y, para él, un sitio muy bueno para cortar una línea es justo después de un guion (de cualquier tamaño). Pero claro, si escribimos «Hoy —que es lunes— es un mal día», no sería correcto un corte de línea entre «—» y «que»; lo mismo que no se puede cortar línea entre «(» y «que» en «Hoy (que es lunes) es un mal día». Pero LaTeX no sabe que no puede cortar línea entre «—» y «que», y lo hará a menudo (la razón de que LaTeX no tenga previsto esto es que el uso de las rayas en inglés es distinto que en español). Afortunadamente, es muy fácil evitar que se pueda producir ese corte indeseado: basta escribir

```
Hoy ~---que es lunes--- es un mal día.
```

Con `~---` (que está definido en spanish de babel), LaTeX nunca cortará línea después de la raya. No hay que usar `~---` en la raya que cierra el inciso, salvo en caso de que después haya un signo de puntuación (para evitar que salga — al final de una línea y el signo de puntuación al principio de la línea siguiente).

Hay a quien la raya — que se genera con `---` le parece demasiado larga, y en su lugar prefiere usar – (la que se genera con `--`). Si te incluyes entre ellos, no hay ningún problema; pero, en ese caso, aplica a `--` y `~--` todo lo explicado anteriormente sobre `---` y `~---`.

Para concluir, conviene mencionar que, en algo como «cazador-recolector», LaTeX, por defecto, solo verá como corte de línea válido el guion. Si queremos permitir cortes de línea en las sílabas de las dos palabras, hay métodos previstos para ello, pero una solución sencilla es indicárselo así: `ca\-za\-dor-re\-co\-lec\-tor`.

6.5. Comillas y letras voladitas

En inglés, las comillas se escriben como en “comillas inglesas”, y es lo que LaTeX sabe hacer por defecto: lo anterior se consigue con ``` ``comillas inglesas'' ```. En otros idiomas (incluso refiriéndonos solo a los que utilizan caracteres latinos), sus comillas se escriben de otras formas; por ejemplo, en alemán es „así”, en checo y rumano „así“, en finés ”así”, y un largo etcétera. Lo tradicional en español, francés, italiano y portugués es usarlas como en «comillas latinas» (que es el nombre que reciben). Debido al auge de los ordenadores (y a que por defecto se diseñan pensando principalmente en las costumbres del inglés), las comillas inglesas están muy extendidas en muchos idiomas; por cierto, no se escriben como en `"comillas rectas"`, sino que son curvas. Pero, en los teclados, la tecla `"` siempre tiene fácil acceso, y los programas de escritura al uso se encargan de transformar `"` al vuelo en “ y ” según la posición sea de abrir o cerrar.

Si queremos utilizar comillas latinas, es muy fácil: al margen de que, si estamos usando codificación utf8 de inputenc podemos utilizar las caracteres « y » tal como salen del teclado con algún tipo de combinación de teclas —¡no me digas que no sabes cómo sacar eso en tu teclado!—, con spanish de babel se consiguen sin más que usar `<<comillas latinas>>` (o `"<comillas latinas">`). En español, si dentro de unas comillas queremos poner otras, podemos llegar hasta tres niveles de anidamiento: el nivel externo con «», el intermedio con “” y el interno con ‘’.

Antes de seguir, hagamos una precisión. Cuando hemos escrito ``` ``comillas inglesas'' ```, el `''` no es una comilla doble `"`, sino dos comillas simples. En TeX tradicional, es equivalente usar dos comillas simples o una doble (en ambos casos se obtiene ”). Pero babel se queda con la comilla doble `"` como carácter reservado para sus propios intereses; eso permite, por ejemplo, que con `"<` y `">` se obtenga « y ».

Ese carácter comilla doble `"` lo aprovecha babel para bastantes más cosas, muchas veces de maneras dependientes del idioma (cada idioma tiene sus propias necesidades). En particular, en spanish se usa, también, para obtener algunas letras voladitas, De hecho, esta es la razón por la que, en este apartado, se hable a la vez de comillas y de letras voladitas, dos cosas que, en principio, podían parecer lejanas.

Una letra voladita es una letra que está desplazada hacia arriba y suele ser más pequeña que las del texto donde está, como ^{a}. Pero no es un superíndice de matemáticas, pues no tiene que aparecer con tipografía matemática, como ocurriría con ^{a}. Las letras voladitas se usan, sobre todo, en abreviaturas: se pone un punto y luego va la letra voladita, como en adm.^{ón}, que es

la abreviatura de «administración», o en M.ª Dolores. A muchos, cuando les comentan que ese punto existe, les suena raro, pero es lo que se ha hecho tradicionalmente, y la presencia del punto pasaba desapercibida (el caso es que mucha gente ahora no pone el punto).

La manera más fácil de escribir letras voladitas en LaTeX es con `\textsuperscript`; por ejemplo, en `adm.\textsuperscript{ón}`; en general, esto suele ser lo más recomendable, pues no depende de ningún paquete. Con spanish, también se puede hacer `adm\sptext{ón}`, pero la letra voladita queda un poco más baja (adm.ón), luego no conviene mezclar ambos métodos (la orden `\sptext` coloca automáticamente el punto que debe preceder a las letras voladitas).

También es muy común utilizar letras voladitas para abreviar los ordinales: 1.ª, 2.º, 3.er... Para abreviaturas asociadas a ordinales podemos usar `"a` y `"o` y `"er`, que equivale a `\sptext{a}`, `\sptext{o}` y `\sptext{er}`. Obsérvese que aquí está actuando el carácter `"` que se ha reservado babel para su propio uso. De hecho, esto hace que, si inadvertidamente usamos `"` como comilla (inglesa) en vez de las correctas, podamos obtener resultados esotéricos, dependiendo de qué carácter vaya después.

Algunas tipografías incluyen caracteres que son, ya de por sí, letras voladitas, en particular la º y la ª que están presentes en los teclados. Pero añaden un pequeño subrayado que debe evitarse, y por tanto no se deben escribir los ordinales —o la abreviatura de María— con inputenc (a menos que se esté seguro de que el resultado es el que se busca).

Por cierto, si se quiere añadir una cita textual, se pone entre comillas si está dentro de texto, pero no hace falta usar comillas si está destacada dentro de un entorno apropiado y, quizás, con alguna tipografía distinta. Por ejemplo, con

```
\begin{quote}
\emph{Un matemático es una máquina que transforma café en teoremas.}
\end{quote}
```

obtendríamos lo siguiente (frase atribuida tanto a como Alfréd Rényi como a Pál Erdős, ambos matemáticos húngaros):

> *Un matemático es una máquina que transforma café en teoremas.*

6.6. Los capítulos siempre empiezan a la derecha, en página impar

Los libros se imprimen por las dos páginas y, con el libro abierto, a la derecha siempre hay una página impar. Además, como regla general, los capítulos (o partes del libro similares, como los índices y la bibliografía) siempre empiezan a la derecha (en página impar, por tanto). Así pues, si un capítulo acaba en una página impar, el libro deberá dejar una página par vacía (la página par que estará por detrás de la del final del capítulo), y el capítulo siguiente empezará en la siguiente página, que ya será impar y estará a la derecha.

Eso es lo que hace LaTeX por defecto tanto con la clase book como con amsbook. Si por algún motivo queremos conseguir un libro escrito solo por una cara —no es lo normal, pero seguro que la administración española lo pide para algo en alguna normativa de obligado cumplimiento—, se puede conseguir con la opción `oneside`. En este caso, los capítulos empezarán donde toque y LaTeX no tendrá que añadir páginas en blanco. Pero dejemos los esoterismos y volvamos a lo habitual.

Cuando LaTeX deja una de esas páginas en blanco, la clase book mantiene en ellas la cabecera que venía usando (amsbook no lo hace). A muchos, esto nos parece feo. Es fácil automatizar que no ocurra, basta redefinir el comando `\cleardoublepage`. Esto se consigue así:

```
\let\cleardoublepageOriginal\cleardoublepage
\renewcommand{\cleardoublepage}%
   {\newpage{\pagestyle{empty}\cleardoublepageOriginal}}
```

Aunque no entiendas muy bien qué quiere decir este código, las páginas pares vacías no tendrán cabeceras si lo pones en el preámbulo de tu documento `.tex`.

Por otra parte, ya hemos comentado en otro apartado que a menudo conviene añadir, al principio de un trabajo de fin de grado o documento similar, una portada generada por otros medios (proporcionada por la universidad, por ejemplo). Si haces esto, deberá haber una página en blanco (o con cualquier otra cosa que quieras poner) después de la portada.

Según como hayas añadido esa portada (por ejemplo, si la has cargado desde LaTeX con el comando `\includepdf` del paquete `pdfpages`, como se ha explicado antes), lo normal es que esa página en blanco esté (puede depender de alguna cosa más, no entraremos en detalles). Pero, si tras generar el `.pdf` final con LaTeX añades esa portada con un programa externo para manipular archivos `.pdf`, esa página que debería estar detrás de la portada por detrás no estará, y deberás añadirla también. De lo contrario romperás toda la estructura de páginas pares e impares y, al imprimir, todo quedará al revés de como debería haber quedado. Lo hagas como lo hagas, cerciórate de que esto te queda bien. (Por si te sirve de algo, en LaTeX tienes los comandos `\clearpage` y `\cleardoublepage` que te pueden resultar útiles; también dispones de la orden `\newpage`, que usada después de un `\null` puede generarte una página en blanco.)

6.7. Puntuación en las ecuaciones

Ya hemos dicho que las matemáticas se redactan, y que hay que hacer frases gramaticalmente bien construidas. Esto incluye a las fórmulas, tanto si están dentro de texto como centradas (bien sea porque son demasiado grandes para ponerlas en texto ordinario o si las queremos destacar). Las fórmulas se integran en las frases y —también si son fórmulas centradas— deben incluir signos de puntuación si la frase lo requiere (por ejemplo, un punto final, si la frase acaba en una fórmula).

Por otra parte, no hay que dejar, en el archivo `.tex`, una línea vacía antes o después de las fórmulas centradas. Hay algunos que lo hacen, inadvertidamente, para que en el código `.tex` sea fácil identificar rápidamente las fórmulas centradas, pero es muy mala idea pues tiene efectos colaterales no deseados. Una línea vacía es un «punto y aparte» en un párrafo; puede dejar espacio vertical extra y, sobre todo, para LaTeX un punto y aparte es un lugar ideal para cortar página, justo lo contrario de lo que debería ocurrir delante de una fórmula centrada, pues una página que comience con una fórmula centrada es «feo» (da la impresión de ser un título).

Después de una fórmula centrada solo hay que dejar una línea vacía si la fórmula es el final del párrafo; de hecho, si ahí se deja una línea vacía que no debería estar, las consecuencias en el `.pdf` serán evidentes (salvo que hayamos desactivado la sangría), pues el texto después de la fórmula aparecerá sangrado.

Finalmente, comentemos una costumbre errónea muy extendida. Hay quien, siempre o a menudo, coloca «:» delante de cada fórmula matemática centrada. Esos dos puntos tienen que estar si y solo si la gramática de la frase lo exige. Por ejemplo, sería correcto escribir «Euler probó la siguiente fórmula:» —y entonces se coloca la fórmula centrada—, pero habría que escribir «Euler probó que» —y entonces se coloca la fórmula centrada—. Si, con la fórmula colocada dentro de texto ordinario, los dos puntos no tienen que estar, con una fórmula centrada tampoco.

6.8. Operadores matemáticos

LaTeX tiene muchas «funciones matemáticas de varias letras» predefinidas con comandos, como `\sin`, `\cos` y `\tan`, que siempre deben aparecer como letra «recta»; por ejemplo, esas tres funciones quedarán así: sin, cos, tan. Si estamos escribiendo en español, spanish de babel define algunas otras, como `\sen`, `\tg` (en español es más común usar tg que tan). También redefine silenciosamente lo que se obtiene con algunas funciones, como con `\lim`, `\max` y `\min`, de modo que salga lím, máx y mín (con acento). Todo esto se puede configurar siguiendo lo que indica el manual de spanish.

Pero hay veces que queremos utilizar funciones «de varias letras» que no están predefinidas, y hay que hacerlo de modo que sean un operador matemático. Esto es muy fácil, pues amsmath tiene mecanismos previstos para ello. Por ejemplo, si, como es habitual, queremos denotar el máximo común divisor con mcd, bastaría usar `\operatorname{mcd}`. Sin el `\operatorname`, lo que obtendríamos sería *mcd*, que es como quedarían tres variables m, c y d multiplicadas (con funciones de nombres más complicados, y que combinan mayúsculas y minúsculas, lo cual es común en informática, esto puede quedar horroroso si no se hace bien, y ser muy confuso).

Para no tener que escribir `\operatorname{mcd}` cada vez, y que podamos usar `\mcd` como en el caso de los ejemplos anteriores de funciones trigonométricas, hay que definir el comando `\mcd`, y eso también es muy sencillo. Se conseguiría de cualquiera de estas dos maneras:

```
\newcommand{\mcd}{\operatorname{mcd}}
\DeclareMathOperator{\mcd}{mcd}
```

La orden `\newcommand` de LaTeX es muy útil para definir nuevos comandos que faciliten la escritura de cualquier cosa que vaya a aparecer de manera repetida en tu documento. Por ejemplo, si estás redactando un texto sobre números complejos y defines `\newcommand{\C}{\mathbb{C}}` podrás usar `\C` cada vez que quieras conseguir $\mathbb{C}$. Tiene muchas más posibilidades (no solo con matemáticas); te aconsejamos que aprendas a usarla. Quizás observar este ejemplo con dos parámetros (a los que se alude como `#1` y `#2`) te sirva de ayuda. Definamos

```
\newcommand{\suc}[2]{\{#1_{#2}\}_{#2=1}^{\infty}}
```

Al usar `$\suc{a}{n}$` y `$\suc{x}{j}$` obtendremos, respectivamente, $\{a_n\}_{n=1}^{\infty}$ y $\{x_j\}_{j=1}^{\infty}$.

6.9. Espaciado en matemáticas

Al escribir matemáticas, no solo tienen importancia los símbolos sino cómo se distribuyen los espacios a su alrededor; por ejemplo, una relación binaria requiere dejar un poco de espacio antes y después del símbolo, como ocurre aquí con el «pertenece»: $x \in A$ (compara con el incorrecto $x\in A$); pero los delimitadores (paréntesis o similares) se comportan de manera diferente. LaTeX tiene unas cuantas parejas de comandos que generan el «mismo dibujo» pero distribuyendo espacios de distinta forma. La mejor manera de ver esto es con un ejemplo:

```
\mid x+y \mid \le \mid x \mid + \mid y \mid,
\qquad
\vert x+y \vert \le \vert x \vert + \vert y \vert.
```

Lo que se obtendría así es

$$\mid x+y \mid \le \mid x \mid + \mid y \mid, \qquad |x+y| \le |x| + |y|.$$

Está claro que la manera correcta de representar la desigualdad triangular es la de la derecha; a la izquierda, tal como están distribuidos los espacios, parece que estamos tomando el valor absoluto de $\leq$. Lo mismo ocurriría para normas con `\parallel` y `\Vert` (en ambos casos obtendríamos el dibujo $\|$ pero con distinta distribución de los espacios); por cierto, `\vert` y `\Vert` son equivalentes, respectivamente, a `|` y `\|`, y escribirlo así suele ser más cómodo.

Ya que ha surgido `\mid`, ¿para qué se emplea en lugar de `\vert`? Pues, por ejemplo, al definir un conjunto. Con

```
\Omega = \{(x,y) \in \mathbb{R}^2 \mid 9x^2 + 16y^2 \le 25\}
```

obtendríamos $\Omega = \{(x, y) \in \mathbb{R}^2 \mid 9x^2 + 16y^2 \leq 25\}$ (en lugar de |, también se usa : con el mismo significado). Otro uso habitual de `\mid` es para indicar «divide a»: al escribir `$2 \mid 8$` obtenemos $2 \mid 8$, y su negación `$2 \nmid 7$` da $2 \nmid 7$ (hay a quien le gusta el espaciado de 2|8, pero es inconsistente con el de la versión negada «no divide a»).

Otro ejemplo que merece la pena destacar es el del producto escalar. El código

```
<x,y> = |x|\cdot |y| \cos(\theta),
\qquad
\langle x,y \rangle = |x|\cdot |y| \cos(\theta).
```

generaría esto:

$$< x, y >= |x| \cdot |y| \cos(\theta), \qquad \langle x, y \rangle = |x| \cdot |y| \cos(\theta).$$

De nuevo lo de la derecha es lo correcto; lo realmente importante no es que la forma de los «paréntesis angulares» ⟨ y ⟩ sea distinta de < y >, sino el espaciado, que, colocado de manera correcta, permite identificar el grupo $\langle x, y \rangle$ de un solo vistazo. Sin duda, la tipografía facilita la lectura.

Normalmente, no hay que enmendar la plana a LaTeX en cuanto a cómo distribuye los espacios en matemáticas. Lo mejor (que además coincide con lo más cómodo para el que escribe) es dejarle hacer, y que él se encargue. Pero hay unas pocas situaciones en las que lo recomendable es introducir un pequeño espacio horizontal, que se consigue con `\,` (tampoco es difícil, por tanto). Una es después de un factorial (si luego hay más símbolos matemáticos): queda mejor $7!\, n$ (hecho con `7!\,n`) que $7!n$. Otra, con el «d» con significado de «diferencial» en las integrales. De nuevo lo mejor es comparar. Con

```
\int_0^1 \int_0^1 x y dx dy = \frac{1}{4},
\qquad
\int_0^1 \int_0^1 x y \,dx \,dy = \frac{1}{4}
```

obtendríamos

$$\int_0^1 \int_0^1 xydxdy = \frac{1}{4}, \qquad \int_0^1 \int_0^1 xy\, dx\, dy = \frac{1}{4},$$

y es claro que en el ejemplo de la derecha se identifican mucho mejor los diferenciales dx y dy.

Por cierto, en algunos sitios acostumbran a escribir la «d» de «diferencial» como letra recta: $\mathrm{d}x$ en vez de dx. Pero esta nunca ha sido la costumbre entre los matemáticos (ni en español ni en inglés); no parece más bonito, no aporta ninguna ventaja en cuanto a la legibilidad de las fórmulas y además es más complicado de escribir, así que no recomendamos hacerlo.

También en algunos ámbitos abogan por escribir i (la unidad imaginaria) y e (la base de los logaritmos neperianos) como «i» y «e», en lugar del i y e común entre los matemáticos. Pero,

además de que cuesta más trabajo, no parece que escribir $\mathrm{e}^{\mathrm{i}\pi} = -1$ tenga ninguna ventaja —en cuanto a claridad del significado, que es lo que se pretende cuando uno intenta tener cuidado con la tipografía matemática— sobre $e^{i\pi} = -1$, luego tampoco recomendamos hacerlo.

Otro ajuste al que —en nuestra opinión— posiblemente no merece la pena hacer mucho caso es el uso de `f \colon A \to B` (queda $f\colon A \to B$) en vez de `f : A \to B` (queda $f : A \to B$) para denotar una aplicación. El espacio tras la f es algo menor en el primer caso, pero no parece fácil justificar que eso es realmente mejor o que contribuye a la claridad.

Mencionemos asimismo algo que, a menudo, causa dificultades cuando se escriben listas separadas por comas dentro de texto. Si escribimos «`$a,b$ y $c$`» el espaciado queda distinto a si lo ponemos como «`$a$, $b$ y $c$`»; no es muy grave, pero es mejor la segunda forma, sobre todo si, en vez de a y b, tenemos expresiones más grandes, pues a LaTeX no le gusta cortar una expresión matemática después de una coma (por ejemplo, en `$a+1,b+1$` preferiría cortar línea en los `+`). Esto acarrea un problema de cierta importancia cuando, dentro de texto, se escriben expresiones como `$D = \{0,1,2,3,4,5,6,7,8,9\}$`, o más largas. LaTeX no va a cortar línea después de las comas y no va a poder construir párrafos bien justificados. La solución es fácil: añadir `\allowbreak` después de las comas donde queramos permitir el corte de línea.

Una cosa más. Ya hemos comentado que se puede elegir qué queremos como separador decimal, coma o punto. ¿Y qué usar como separador de miles? La respuesta es rápida y contundente: ¡ninguno de los dos! Está totalmente contraindicado usar comas o puntos como separadores de miles porque, obviamente, puede dar lugar a confusión con los decimales. Como separador de miles se puede usar un espacio fino, que se consigue con `\,`. Así, por ejemplo,

```
2^{2^5} + 1 = 641 \cdot 6\,700\,417
```

quedará $2^{2^5}+1 = 641 \cdot 6\,700\,417$ (factorización del número de Fermat F_5, descubierta por Euler).

6.10. Distintos tipos de puntos suspensivos

En matemáticas hay distintos tipos de puntos suspensivos: $\ldots$, $\cdots$, $\vdots$ y $\ddots$. ¿Cuándo usar $\ldots$ y cuándo $\cdots$? Sin mucho detalle, se puede decir que hay que usar el que se corresponde con la «altura media» de lo que tengan al lado: así, usaríamos $\alpha_1, \ldots, \alpha_n$ pero $\alpha_1 + \cdots + \alpha_n$.

Los puntos suspensivos también se usan en texto. Los que se consiguen con `\dots` (…) tienen más separación que los que se consiguen escribiendo tres puntos ordinarios uno tras otro (...). La verdad es que en español siempre ha sido lo habitual poner los puntos con poca separación. Pero, si así lo hiciéramos, los puntos suspensivos de entornos matemáticos iban a quedar con distinto espaciado que los de texto ordinario, y no parece muy coherente. Por eso suele ser conveniente usar `\dots` para los puntos suspensivos en texto.

Aquí surge un problemita con el que hay que tener cuidado. Supongamos que escribimos

```
diez, nueve, ocho\dots uno, cero.
```

Ahí, el espacio que sigue a `\dots` hace de «final de comando», y no es un espacio efectivo: se pierde (quedará «diez, nueve, ocho…uno, cero.»). Pero ese espacio tiene que estar, luego hay que ponerlo, y se hace así: `ocho\dots\ uno`. Por cierto, si una frase acaba con puntos suspensivos, en español no hay que añadir después el punto final.

6.11. Palabras dentro de un entorno matemático

Si dentro de matemáticas hay palabras en español (o en otro idioma), hay que conseguir que sean texto, no letras matemáticas una detrás de otra, como si fuesen variables multiplicadas. Lo mejor es hacerlo con el comando `\text`. Por ejemplo, para definir la función de Dirichlet, una función que no es continua en ningún punto de su dominio, podríamos hacer esto:

```
f(x) =
\begin{cases}
0, & \text{si } x \in \mathbb{R} \setminus \mathbb{Q}, \\
1, & \text{si } x \in \mathbb{Q}.
\end{cases}
```

Asimismo, compara esto: si escribimos

```
h_{flauta}, h_{trompeta} \text{ y } h_{\text{flauta}}, h_{\text{trompeta}}
```

quedaría $h_{flauta}, h_{trompeta}$ y $h_{\text{flauta}}, h_{\text{trompeta}}$; aquí, en lugar de `\text`, habría sido mejor usar `\textnormal`, para asegurarnos de que el texto va a salir con «letra normal» incluso aunque estemos en un entorno en el que el texto se escribe en itálica (el enunciado de un teorema, por ejemplo).

6.12. Operadores que cambian de tamaño

Unos cuantos operadores matemáticos cambian de tamaño o de comportamiento —en cuanto a dónde y cómo distribuyen los símbolos que se usan— dependiendo si estamos en una fórmula centrada o dentro de texto. Por ejemplo,

```
\sum_{n=1}^{\infty} \frac{1}{n^2} = \frac{\pi^2}{6},
\qquad
\int_{-\infty}^{\infty} e^{-x^2}\,dx = \sqrt{\pi}
\qquad\text{y}\qquad
\lim_{x\to\infty} \frac{\pi(x)}{x/\log(x)} = 1
```

queda

$$\sum_{n=1}^{\infty} \frac{1}{n^2} = \frac{\pi^2}{6}, \qquad \int_{-\infty}^{\infty} e^{-x^2}\,dx = \sqrt{\pi} \qquad \text{y} \qquad \lim_{x\to\infty} \frac{\pi(x)}{x/\log(x)} = 1$$

como fórmula centrada, pero $\sum_{n=1}^{\infty} \frac{1}{n^2} = \frac{\pi^2}{6}$, $\int_{-\infty}^{\infty} e^{-x^2}\,dx = \sqrt{\pi}$ y $\lim_{x\to\infty} \frac{\pi(x)}{x/\log(x)} = 1$ en texto ordinario. Este distinto comportamiento está diseñado así aposta para que, dentro de texto, el espacio entre líneas no se tenga que ensanchar demasiado. Hay distintas formas de indicarle a LaTeX que haga otra cosa, pero, en general, lo aconsejable es dejarlo tal como queda por defecto.

Por cierto, en el ejemplo del sumatorio dentro de texto, posiblemente hubiera quedado mejor escribiendo $1/n^2$ y $\pi^2/6$. No siempre conviene usar `\frac` para escribir fracciones: el aspecto de $2^{1/p}$ es indudablemente más atractivo que el de $2^{\frac{1}{p}}$.

6.13. Paréntesis de distintos tamaños

En LaTeX tienes a tu disposición paréntesis de distintos tamaños. Por ejemplo, con

```
\Bigg( \bigg( \Big( \big( ( \quad ) \big) \Big) \bigg) \Bigg)
```

se consigue

$$\Bigg(\bigg(\Big(\big(\quad\big)\Big)\bigg)\Bigg).$$

Es importante usar los paréntesis del tamaño adecuado de manera que abarquen razonablemente bien la expresión que encierran. También se puede usar `\left(` y `\right)`, que adaptan automáticamente su tamaño (aunque a menudo quedan demasiado grandes, y además dejan un poco de espacio adicional). Lo mismo se puede hacer con corchetes [ ], con llaves { }, con paréntesis angulares ⟨ ⟩, con barras verticales |, o con barras dobles ‖, y con algún símbolo más, como ⌊ ⌋ (que se utiliza a menudo para denotar «parte entera», preferiblemente a [ ]). Y, con la sintaxis adecuada, todos estos símbolos se pueden usar como delimitadores de matrices (o determinantes); se recomienda usar para ello las estructuras que proporciona el paquete amsmath.

Así pues, salvo que por algún motivo decidas hacerlo adrede, no hay razón para mezclar paréntesis, corchetes y llaves con intención de que se distinga dónde empieza y dónde acaba cada cosa en una expresión complicada. Con los cambios de tamaño, hay más que suficiente.

Hablando de paréntesis y símbolos similares, la notación más usual para un intervalo abierto es (a, b), pero hay quien prefiere usar $]a, b[$ (esto es bastante común en francés). Pero no hay que usar simplemente `]a,b[` para escribir un intervalo abierto, pues LaTeX tiene a `[` como símbolo de apertura y a `]` como símbolo de cierre, y lo estamos usando al revés, con lo cual el espaciado no será el adecuado cuando se combina con otros símbolos en la misma expresión: por ejemplo, con `$I = ]a,b[$` obtendríamos $I =]a, b[$, en lugar del $I =]a, b[$ bien espaciado que se obtendría con `$I = \mathopen]a,b\mathclose[$`. Moraleja: usa $I = (a, b)$ para los intervalos abiertos y no te compliques la vida.

6.14. Formatos para los gráficos

Aquí podríamos precisar mucho más, pero necesitaríamos extendernos demasiado y eso está fuera del alcance de esta guía. Así que, conscientemente, diremos algunas pequeñas mentiras, y contaremos lo más habitual como si fuese lo único posible.

El compilador latex clásico (que no generaba archivos `.pdf`, sino unos archivos denominados `.dvi` que para nosotros ya no tienen interés) podía incluir archivos gráficos en formato PostScript (con extensión `.eps`, aunque a veces también se usa `.ps`). El actual, que denominaremos pdflatex pues genera archivos `.pdf`, puede incluir archivos gráficos en formato `.jpg` (o `.jpeg`), `.png` o `.pdf`. Nos estamos refiriendo aquí a archivos gráficos que nunca son de varias páginas.

Los formatos `.eps` y `.pdf` son, en principio, bastante parecidos; al menos en la dirección `.eps` → `.pdf`, la transformación es directa de muchas formas. Ambos son lo que en artes gráficas denominan «formatos vectoriales», lo cual quiere decir que lo que contienen no es una nube de puntos, sino lo que llamaríamos «dibujo lineal», en el que lo representado es fruto de unas ecuaciones matemáticas para curvas o para contornos de figuras, y son esas ecuaciones lo que recoge el archivo. Las ecuaciones matemáticas se escalan sin dificultad, así que en estos gráficos se puede hacer zum (*zoom*, en inglés) de manera indiscriminada, y no hay ninguna pérdida de calidad en lo que se observa. (Una precisión: aparte de lo que acabamos de explicar, tanto los

`.eps` como `.pdf` pueden contener también algo que sí que sea una nube de puntos, y eso sí estaría afectado por la pérdida de calidad al hacer cambios de escala.)

Los archivos `.jpg` y `.png` siempre son nubes de puntos, con determinada resolución. Lo más normal es que una foto sea un archivo `.jpg` o `.png`, y que, por ejemplo, una circunferencia con una recta secante sea un `.pdf` o `.eps`. Si podemos conseguirlo, los archivos con gráficos matemáticos deberían ser `.pdf` (o `.eps`); no solo tendrán más calidad sino que los archivos ocuparán menos. También son vectoriales los archivos `.svg`, pero LaTeX no puede usarlos directamente.

Desde `.pdf` o `.eps` se pasa a `.jpg` o `.png` con facilidad; lo contrario es imposible si lo que se quiere es conseguir archivos vectoriales (hay programas que permiten «vectorizar» imágenes, pero eso es otra historia). Si generas archivos gráficos desde programas externos para incluirlos (con `\includegraphics`), procura que sean `.pdf`. Y, a ser posible, que tengan las «letras incrustadas» (es decir, que la tipografía que se haya usado esté incluida en el archivo, lo cual evita algunos problemas).

Por supuesto, si uno genera gráficos dentro del propio archivo `.tex` usando tikz, pgfplots, u otros paquetes que hay para ello, lo que obtengamos será vectorial, y de magnífica calidad.

A decir verdad —y esto no tiene ninguna importancia si uno está generando su propio documento y no lo tienen que procesar otros—, en muchos sitios que utilizan archivos `.tex` (revistas de investigación matemática) piden los archivos gráficos en `.eps`. En principio, los `.eps` ni siquiera se ven en la pantalla (están pensados para impresoras y se necesita algún programa que los interprete, por ejemplo Ghostscript, que viene en las distribuciones de LaTeX), pero hay muchos programas de los habituales en artes gráficas que están especializados en procesar archivos `.eps`, y los técnicos de artes gráficas se sienten cómodos en ese mundo. Si es el caso, no te resultará complicado pasar de `.jpg`, `.png` o `.pdf` a `.eps` (hay páginas en internet que lo hacen, y las distribuciones de LaTeX incluyen el programa ImageMagick que también sirve para eso). Pero, si hemos diseñado un precioso dibujo con Ti*k*Z y nos lo piden en `.eps`, ¿qué hacemos? Lo más sencillo, sin ninguna pérdida de calidad, es utilizar la clase standalone de LaTeX. Por ejemplo, imaginemos —por hacer algo muy sencillo— que queremos dibujar una circunferencia con una recta secante que va del punto P al Q. Hagámoslo así:

```
\documentclass[tikz,border=1mm]{standalone}
\usepackage{tikz}
\begin{document}
\begin{tikzpicture}
\draw[thick] (0,0) circle(1);
\draw (-1.7,-0.4) node[below]{$P$} -- (1.5,0.9) node[below]{$Q$};
\end{tikzpicture}
\end{document}
```

Procesando ese archivo con pdflatex, obtendremos un archivo `.pdf` que contendrá la circunferencia y la recta secante con 1 mm de borde en blanco alrededor. Si lo procesamos con latex, obtendremos lo mismo en un archivo `.ps` (podemos cambiar la extensión y poner `.eps` a mano si lo preferimos, da igual). Es lo que se muestra en la figura 2; el marco gris no forma parte de la gráfica, sino que quiere indicar que el archivo solo contendrá lo que hay dentro del marco, no será una página A4 (u otro tipo de tamaño de papel) con ese dibujo en ella.

Por cierto, ya que hemos comentado que lo anterior puede ser útil para enviar un artículo a una revista de investigación, si ese es el caso no hay que intentar personalizar el aspecto del documento para que quede estéticamente a tu gusto. La revistas se publican con su propio

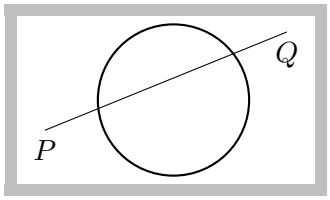


Figura 2: Una circunferencia con una recta secante.

formato, y todo lo que hagas para cambiar el aspecto de tu documento lo van a tener que deshacer.

6.15. Archivos auxiliares

Cuando LaTeX genera el archivo `.pdf` a partir del `.tex`, necesita crear (y utilizar) algunos archivos auxiliares, que tendrán el mismo nombre que el archivo `.tex` pero con distintas extensiones. Si tu ordenador no muestra las extensiones de los archivos, puedes no aclararte bien con esto; afortunadamente, puedes decirle que las muestre.

El archivo auxiliar más importante es el `.aux`. LaTeX, cada vez que procesa el archivo `.tex`, guarda en un `.aux` las `\label` que aparecen, junto con el número (de lo que sea) que les corresponde; y también guarda lo necesario para los `\bibitem`. A su vez, LaTeX lee el archivo `.aux` (el que haya antes de generar el nuevo), y usa la información que hay en él en los `\ref`, `\eqref`, `\pageref` o `\cite`. Lógicamente, la primera vez que procesamos el archivo `.tex`, el archivo `.aux` no existe, luego LaTeX no puede cargarlo previamente en memoria y no puede interpretar los `\ref` o similares. Por este motivo, las referencias aparecen con **??** la primera vez que se procesa el archivo `.tex`, y hay que procesarlo una vez más. Y, cuando estamos redactando, el `.aux` se queda viejo a menudo, y de nuevo hay que procesar el `.tex` dos veces.

Otro archivo auxiliar es `.toc` (recoge la información que se necesita en la `\tableofcontents`), y algunos paquetes utilizan el suyo propio (por ejemplo, hyperref utiliza `.out`). También están `.log` (que guarda la información sobre el proceso de generar el `.pdf` desde el `.tex`, y donde se recogen los posibles errores o advertencias), y `.synctex.gz` (en el que se guarda información que el editor puede usar para sincronizar posiciones entre el `.tex` y el `.pdf`, y poder pasar fácilmente de las posiciones de uno a las posiciones del otro).

Ya hemos dicho que el uso de `.aux` requiere, a veces, que procesemos el archivo `.tex` dos veces. Lo mismo ocurre con bastantes de los demás archivos auxiliares (en raras ocasiones, incluso más de dos veces). Se puede conseguir que LaTeX procese el archivo siempre dos veces, o que se dé cuenta de si necesita hacerlo o no. Pero, actualmente, procesar un archivo LaTeX típico cuesta, como mucho, escasos segundos, así que, en general, no merece la pena.

Todos estos archivos auxiliares los podemos borrar si queremos, y LaTeX los volverá a generar si volvemos a procesar el archivo `.tex`. Es más, en ocasiones es muy buena idea borrar los archivos auxiliares. Al procesar un `.tex` con errores, en los archivos auxiliares se ha podido guardar información a medias o con errores de sintaxis. Aunque corrijamos el `.tex`, los errores de los archivos auxiliares estarán todavía cuando LaTeX los lee la siguiente vez que va a procesar el `.tex`; y de nuevo obtendremos errores. En estas circunstancias, lo más aconsejable es borrar todos los archivos auxiliares (muchos editores tiene alguna forma cómoda de conseguirlo).

6.16. Modo «nonstop»: LaTeX ignora los errores

Cuando compilamos un archivo `.tex` se obtiene un `.pdf`; aunque el propósito es otro, empleamos «compilar» con el mismo sentido de lo que se hace con los lenguajes de programación ordinarios: compilamos un archivo `.c` y obtenemos un ejecutable. (Como el objetivo de TEX/LATEX es componer páginas, en nuestro caso también se usa «componer» en vez de «compilar».)

En la mayoría de los lenguajes de programación, el compilador (o el intérprete, si ese fuera el caso) se detiene cuando encuentra un error (una excepción notable son los navegadores, que, cuando encuentran un error en un archivo `.html`, salen de él como pueden, y no se detienen).

TEX y LATEX, en principio, también se detienen cuando encuentran un error de sintaxis o de otro tipo (solo en los errores, no en las «advertencias»). Pero se pueden configurar para que no lo hagan y que, si aparece un error, intenten solventarlo de alguna forma y continúen con la compilación del documento. Si ejecutáramos LATEX desde una terminal (pdflatex, realmente), esto se haría con

```
pdflatex -interaction=nonstopmode
```

En concreto, esto hace que LATEX intente pasar por alto los primeros 100 errores, salvo que alguno sea tan serio como para no saber qué hacer (por ejemplo, si falta un archivo).

Eso sí, si hay errores no hay ninguna seguridad de que lo que se obtiene en el `.pdf` sea lo que se pretendía. Es muy mala idea utilizar LATEX en modo «nonstop». Pero hay algunos editores y páginas web para usar LATEX en internet que, por defecto, están configurados para compilar en modo «nonstop». Esto estará configurado en algún menú o en alguna preferencia, y seguro que se puede desactivar quitando, del sitio adecuado, el `-interaction=nonstopmode` que hemos señalado antes.

Referencias

[1] J. Bezos, *Tipografía y notaciones científicas*, Ediciones Trea, Gijón, 2008.

[2] J. Bezos, Ortotipografía y notaciones matemáticas, 2016, `http://www.texnia.com/archive/ortomatem.pdf`

[3] J. Bezos, Estilo spanish para el sistema babel, 2021, `http://mirrors.ctan.org/macros/latex/contrib/babel-contrib/spanish/spanish.pdf`

[4] B. Cascales, P. Lucas, J. M. Mira, A. Pallarés y S. Sánchez-Pedreño, *LATEX, una imprenta en tus manos*, Aula Documental de Investigación, Madrid, 2000.

[5] B. Cascales, P. Lucas, J. M. Mira, A. Pallarés y S. Sánchez-Pedreño, *El libro de LATEX*, Prentice Hall, Madrid, 2003.

[6] J. P. Casteleyn, Visual Ti*k*Z, 2018, `https://www.ctan.org/tex-archive/info/visualtikz`

[7] W. Chan (traducido y adaptado a español por J. L. Ribera), Acordeón para LATEX 2_ε, 2015, `http://mirror.ctan.org/info/latexcheat/latexcheat-esmx/latexsheet-esmx.pdf`

[8] M. Downes (con actualizaciones de B. Beeton), Short math guide for LATEX, 2017, `https://www.ctan.org/tex-archive/info/short-math-guide`

[9] C. Feuersänger, Manual for package pgfplots, 2021, `http://mirror.ctan.org/graphics/pgf/contrib/pgfplots/doc/pgfplots.pdf`

[10] J. Guillera, History of the formulas and algorithms for π, *Gems in experimental mathematics*, 173–188, Contemp. Math., 517, Amer. Math. Soc., Providence, RI, 2010.

[11] R. Ierusalimschy, *Programming in Lua*, 4.ª ed., Lua.org, 2016.

[12] D. E. Knuth, Mathematical typography, *Bull. Amer. Math. Soc. (N.S.)* **1** (1979), 337–372.

[13] D. E. Knuth, *The TEXbook*, Addison Wesley, 1984.

[14] L. Lamport, *LATEX: A document preparation system*, 2.ª ed., Addison Wesley, 1994.

[15] A. Mertz y W. Slough, Graphics with PGF and Ti*k*Z, *The PracTEX Journal* 2007, n.º 1, 22 pp. Disponible en `http://www.tug.org/pracjourn/2007-1/mertz/`

[16] A. Mertz y W. Slough, A Ti*k*Z tutorial: Generating graphics in the spirit of TEX, *TUGboat* **30** (2009), 214–226. Disponible en `http://tug.org/TUGboat/tb30-2/tb95mertz.pdf`

[17] F. Mittelbach, M. Goossens, J. Braams, D. Carlisle y C. Rowley, *The LATEX Companion*, 2.ª ed., Addison-Wesley, 2004. (La tercera edición, que tendrá dos tomos, está en marcha, y la previsión es que se publique a principios de 2023: `https://tex.stackexchange.com/questions/612573/the-latex-companion-3rd-edition`)

[18] J. I. Montijano, M. Pérez, L. Rández y J. L. Varona, Numerical methods with LuaLATEX, *TUGboat* **35** (2014), 51–56. Disponible en `https://www.tug.org/TUGboat/tb35-1/tb109montijano.pdf`

[19] T. Oetiker, H. Partl, I. Hyna y E. Schlegl, *La introducción no-tan-corta a LATEX 2ε*, distintos idiomas y fechas, disponibles en `https://www.ctan.org/tex-archive/info/lshort/`

[20] T. Tantau, PGF & Ti*k*Z, 2021, `http://mirror.ctan.org/graphics/pgf/base/doc/pgfmanual.pdf`

[21] T. Tantau, J. Wright y V. Miletić, The beamer class, 2022, `http://mirror.ctan.org/macros/latex/contrib/beamer/doc/beameruserguide.pdf`

[22] T. Tao, The dichotomy between structure and randomness, arithmetic progressions, and the primes, *International Congress of Mathematicians* (Madrid, 2006), Vol. I, 581–608, European Mathematical Society Publishing House, Zurich, 2007.