Alineamiento y centrado

Composición de textos científicos

29 de octubre de 2021

1. Alineamiento de fórmulas

En diversas ocasiones al escribir un texto matemático hay que alinear o justificar diferentes partes de una fórmula. En este aspecto, LaTeX "puro" es un poco deficitario. Afortunadamente en el paquete `amsmath` hay soluciones al respecto en forma de diferentes entornos. La filosofía general de todos ellos es que, al igual que con las matrices, con `&` indicamos alinear y con `\\` cambiar de línea.

Supongamos por ejemplo que queremos escribir la demostración de

$$S = 1 + 2 + \cdots + n \quad \Rightarrow \quad S = \frac{n(n+1)}{2}.$$

Para ello, deseamos mostrar el bello procedimiento habitual de sumar la expresión para S consigo misma invirtiendo el orden de sus términos. Con este fin, escribimos los dos pasos en fórmulas separadas:

```
\[
 2S
 =
 (1+n)+(2+n-1)+\dots+(n+1),
\]
\[
 =n(n+1).
\]
```

El resultado no es estético por la falta de alineamiento:

$$2S = (1 + n) + (2 + n - 1) + \cdots + (n + 1),$$

$$= n(n + 1).$$

A esto se unen problemas más sutiles acerca del espacio vertical entre las fórmulas y la posibilidad de un salto de página entre ellas.

El paquete `amsmath` incorpora el entorno `align` con el que si precedemos los símbolos "=" con un `&` indicaremos el alineamiento. De alguna forma,

la sintaxis es la de una una matriz $N \times 2$ donde N es el número de ecuación y el 2 se debe a que hay dos miembros en cada ecuación, el primero se justificará a la derecha y el segundo a la izquierda. El entorno `align` es como si incluyera dentro de su definición un entorno `equation`[1] por tanto las fórmulas aparecerán numeradas y no hay que usar `\[...\]` ni tampoco `\begin{equation}...\end{equation}`, que producirían errores. En definitiva, con

```
\begin{align}
 2S
 &= (1+n)+(2+n-1)+\dots+(n+1),
 \\
 &= n(n+1).
\end{align}
```

conseguimos

$$\begin{aligned} 2S &= (1+n)+(2+n-1)+\cdots+(n+1), && (1)\\ &= n(n+1). && (2) \end{aligned}$$

El alineamiento no está limitado a dos líneas, por ejemplo

$$\begin{aligned} 2S &= (1+n)+(2+n-1)+\cdots+(n+1), && (3)\\ &= (n+1)+(n+1)+\cdots+(n+1), && (4)\\ &= n(n+1). && (5) \end{aligned}$$

se obtiene insertando

```
 \\
 &= (n+1)+(n+1)+\dots+(n+1),
```

justo antes del `\\` en el ejemplo anterior. La justificación a la derecha de los primeros miembros se observa en la siguiente variante que proviene del código obvio:

$$\begin{aligned} S+S &= (1+n)+(2+n-1)+\cdots+(n+1), && (6)\\ 2S &= (n+1)+(n+1)+\cdots+(n+1), && (7)\\ 2S &= n(n+1). && (8) \end{aligned}$$

La forma original de LaTeX de resolver el alineamiento es el entorno `eqnarray` que funciona de manera similar salvo que hay que incluir entre

[1] Hay una versión llamada `aligned`, que no veremos aquí, la cual no incluye el modo matemático. Se utiliza para expresiones de cierta complejidad en las que hay varios alineamientos independientes. Sí veremos `split` que está relacionado. Si tienes curiosidad, puedes consultar la documentación del paquete `amsmath`.

dos `&` el término por el que queremos alinear, en el ejemplo anterior sería el signo igual. Este entorno es fuente de muchas críticas pero es tan antiguo que todavía se ve con cierta frecuencia. Esta es la única razón para verlo aquí. En muchos manuales, comenzando por la documentación de `amsmath`, leerás que debes abstenerte de usarlo. El análogo con `eqnarray` del primer ejemplo que hemos visto sería:

```
\begin{eqnarray}
 2S
 &=& (1+n)+(2+n-1)+\dots+(n+1),
 \\
 &=& n(n+1).
\end{eqnarray}
```

que da lugar a

$$\begin{aligned} 2S &= (1+n)+(2+n-1)+\cdots+(n+1), && (9)\\ &= n(n+1). && (10) \end{aligned}$$

Aquí se ve la razón de una de las críticas hacia `eqnarray` que motivan no emplearlo: el espaciamiento alrededor del símbolo "=", el que hemos alineado, no es coherente con lo que observamos en otras fórmulas. En principio no hay ninguna razón para cambiar el espaciamiento en una fórmula solo por el hecho de que la pongamos encima o debajo de otra. Hay además algún otro problema con la situación de los números de las fórmulas en situaciones extremas y con la manera de asignar etiquetas. En definitiva, `eqnarray` te debería servir solo para leer código de otras personas y si lo copias, es preferible que lo cambies por `align` con la correspondiente modificación en la sintaxis.

En ambos entornos quitaremos la numeración añadiendo al nombre un asterisco, es decir, empleando `align*` y `eqnarray*`, esto es similar a lo visto con `equation`. Si lo que queremos es omitir solo algunos números, emplearemos `\notag` o `\nonumber` en la línea correspondiente, cualquiera de los dos vale siempre que tengamos cargado `amsmath`. Por ejemplo,

$$\begin{aligned} 2S &= (1+n)+(2+n-1)+\cdots+(n+1),\\ &= n(n+1). && (11) \end{aligned}$$

se obtiene con

```
\begin{align}
 2S &= (1+n)+(2+n-1)+\dots+(n+1), \nonumber
 \\
 &= n(n+1).
\end{align}
```

Cada una de las subfórmulas alineadas con `align` admite sus etiquetas y sus referencias independientes. Si en el caso anterior añadimos la etiqueta `\label{eq:fin}` tras `&= n(n+1).`, al escribir `Por \eqref{eq:fin}` obtendremos "Por (11)".

Una ventaja más de `align` sobre el original `eqnarray` es que permite alineamiento múltiple. Antes de dar un ejemplo, pasemos por otras dos formas de alineamiento simple. Una es el entorno `split` que es similar a `align` pero hay que escribirlo dentro de un entorno matemático `\[...\]` o `equation` y cuando se usa este último, para la numeración se considera la fórmula como un todo, apareciendo un solo número. Por ejemplo

```
\begin{equation}
  \begin{split}
  2S &= (1+n)+(2+n-1)+\dots+(n+1),
  \\
  &= n(n+1).
 \end{split}
\end{equation}
```

genera

$$\begin{aligned} 2S &= (1+n)+(2+n-1)+\cdots+(n+1), \\ &= n(n+1). \end{aligned} \tag{12}$$

El número queda entre las dos líneas.

El otro tipo de alineamiento es el que utilizaríamos para definir una función a trozos. Con este fin podríamos combinar los delimitadores `\left\{` y `\right.` con el entorno `array` que sirve para hacer tablas matemáticas o con `matrix`, que daría un resultado menos correcto, pero hay una solución más directa que es el entorno `cases`. Por ejemplo la función $f(x) = |x| + |x-1|$ la escribiríamos a trozos como

$$f(x) = \begin{cases} -2x+1 & \text{si } x \le 0, \\ 1 & \text{si } 0 < x \le 1, \\ 2x-1 & \text{si } x > 1. \end{cases}$$

Lo cual se consigue con la fuente:

```
f(x)=
\begin{cases}
 -2x+1 &\text{si }x\le 0,
 \\
 1 &\text{si }0<x\le 1,
 \\
 2x-1 &\text{si }x> 1.
\end{cases}
```

En breve, el entorno `cases` funciona como una matriz de dos columnas con justificación a la izquierda.

Al hilo de las fórmulas que ocupan varios renglones, aunque no es un alineamiento (no contiene `&`), el entorno `multline`, como su nombre indica, permite dividir una fórmula en múltiples líneas. También admite la variante `multline*` para no numerar. Solo hay que indicar el lugar o lugares donde queremos dividir la fórmula. Por ejemplo,

```
\begin{multline}
 210 = 1+2+3+4+5+6+7+8+9+10\\
 +11+12+13+14+15+16+17+18+19+20
\end{multline}
```

produce

$$\begin{multline} 210 = 1+2+3+4+5+6+7+8+9+10 \\ +11+12+13+14+15+16+17+18+19+20 \end{multline} \tag{13}$$

Añadiendo otro `\\` tras `+15` y cambiando `multline` por `multline*` el resultado es:

$$\begin{multline*} 210 = 1+2+3+4+5+6+7+8+9+10 \\ +11+12+13+14+15 \\ +16+17+18+19+20 \end{multline*}$$

Después de este receso volvamos a `align`. También sirve para obtener varias columnas de fórmulas alineadas. Su uso es como antes con el añadido de que las diferentes columnas deben estar separadas por un nuevo `&`. Lo mejor para entenderlo es ver un ejemplo. Consideremos

$$\begin{aligned} 3 &= 1+1+1, & 3 &= 2+1, & x &= 2020, \\ 2 &= 1+1, & 2 &= 2, & y &= 2021, \\ 1 &= 1, & 1 &= 1, & x+y &= 4041. \end{aligned}$$

Lo podemos obtener con

```
\begin{align*}
 3 &= 1+1+1, &  3 &= 2+1, &  x &= 2020,
 \\
 2 &= 1+1,   &  2 &= 2,   &  y &= 2021,
 \\
 1 &= 1,     &  1 &= 1,   &  x+y &= 4041.
\end{align*}
```

2. Justificación y párrafos en texto

La regla general es que debemos dejar hacer a LaTeX en lo relativo a la justificación de los párrafos. El comando `\break` fuerza la ruptura de una línea en un punto y también se pueden hacer chapucillas introduciendo espacios pequeños para que cierta palabra pase a otra línea pero lo habitual es que esto produzca resultados muy poco estéticos y la única libertad que deberíamos permitirnos es el uso de algunos `\-` para sugerir posibles divisiones de palabras y de algunos `~` para ligar palabras, sin abusar de ello. La justificación de LaTeX tiende a que los espacios sean homogéneos y las líneas tengan la misma longitud. La única excepción habitual a esta justificación es el centrado que resulta de utilidad para poner un título o resaltar una porción de texto. Con este fin, basta incluir en un entorno `center` el párrafo que queramos que aparezca con justificación centrada. Por ejemplo,

```
\begin{center}
 \Huge
 Este es el título de mi magnífico trabajo
\end{center}
```

genera

Este es el título de mi magnífico trabajo

Seguramente nos resulte fea la división en líneas: la palabra "trabajo" no ha cabido en la primera línea y queda aislada en la siguiente. La forma de arreglarlo es introducir tras la palabra "de" un `\\` que indica un cambio de línea, como ya sugería el formato de las matrices. En este contexto `\\` es equivalente al `\break` antes mencionado pero no en general[2]. Entonces escribiremos:

```
\begin{center}
 \Huge
 Este es el título de
 \\
 mi magnífico trabajo
\end{center}
```

[2]Si usamos la doble barra en un párrafo normal no centrado se cambiará de línea como si hubiera un punto y aparte pero sin introducir sangría (el espacio de separación inicial a la derecha) mientras que con `break` solo indicamos que hay que cambiar de línea por ahí pero las palabras de antes y de después estarán justificadas tratando de llenar todo el renglón.

que da lugar a

Este es el título de mi magnífico trabajo

Separar `Este es el título de\\ mi magnífico trabajo` ha sido solo para facilitar la legibilidad de la fuente (esto es, manías mías).

Los entornos `flushleft` y `flushright` aplican justificación a la izquierda y a la derecha. Son de mucha menor relevancia práctica que `center` y apostaría a que hay muchos usuarios medianamente avanzados que ni los conocen. Por ejemplo, con el paquete `lipsum`[3],

```
\begin{flushright}
\it\lipsum[4]
\end{flushright}
```

da lugar a:

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Una posible aplicación es incluir una firma al final de un documento. Con

```
\begin{flushright}
 Madrid 23 de octubre
 \\
 Perico el de los Palotes
\end{flushright}
```

se obtiene

Madrid 23 de octubre
Perico el de los Palotes

[3]Este paquete tiene cierta incompatibilidad con `babel`, Recuérdalo si te salen errores raros al emplearlo. En internet puedes ver un pequeño código para evitarlo que he copiado en la cabecera de la fuente de este documento.

Por supuesto en la práctica querremos dejar un espacio para la firma. Vamos a ver dos maneras de hacerlo. La primera se basa en que `\\` admite un argumento opcional que indica la altura a añadir al salto de línea. Hay muchas unidades posibles, entre las que te resultan familiares están `cm` y `mm`. Así podríamos escribir `\\[1.5cm]` o `\\[15mm]` para obtener

Madrid 23 de octubre

Perico el de los Palotes

En general yo prefiero la unidad más pequeña `pt` (punto de impresión) porque es más precisa con números enteros y me he acostumbrado a ella. Equivale a unos 0.35 *mm* por tanto algo casi equivalente a lo anterior sería emplear `\\[43pt]`. El argumento de `\\` puede ser una longitud negativa aunque, por supuesto, en nuestro ejemplo no tiene sentido.

Otra manera de ajustar el espacio a nuestras necesidades es emplear el comando `\vspace{...}` que es el análogo vertical de `\hspace{...}` que ya habíamos visto. Es importante no abusar de estos comandos. Los puntos suspensivos indican una longitud, positiva o negativa, como en el argumento de `\\`. Un ejemplo de este control universal del espacio es:

```
\hspace{15pt}a\hspace{10pt} b
\vspace{7pt}

c\hspace{-13pt} d
```

que produce:

a b

dc

Nótese la inversión de "c" y "d". Si no hubiéramos dejado una línea en blanco tras `\vspace` (o antes) no funcionaría. Sin entrar en detalles, eso se debe a que hay que estar en el modo vertical de TEX para que funcione y este modo se activa con cambios de línea. En el ejemplo de la firma, podríamos poner `\vspace{1.5cm}` tras de `\\` o dejar como aquí una línea en blanco.

En relación con el espacio vertical en texto, recordemos que dejar una línea en blanco produce un cambio de línea lo que incorporará una sangría de comienzo de párrafo con nuestro documento `article`, por cierto para impedirlo en un párrafo en particular basta preceder la nueva línea con `\noindent`. Si queremos un espacio vertical mayor, lo que se llama doble espacio, podemos crear una línea falsa con un solo espacio representado

mediante `\ ` (con la barra sola sirve). Así a menudo en los documentos vemos cosas como

```
Una línea

\

Otra línea
```

Existen también los comandos

`\smallskip`, `\medskip` y `\bigskip`

que añaden 3, 6 y 12 puntos al salto de línea básico (esto es alrededor de 1, 2 y 4 milímetros). Una cosa importante a tener en cuenta es que estas longitudes, como otras en LaTeX, tienen holgura. Esto significa por ejemplo en el caso de `\bigskip` que si el algoritmo se ve muy apurado en la distribución de párrafos en un página (por ejemplo para no dividir una matriz entre dos páginas) entonces puede reducir su tamaño hasta 8 puntos o aumentarlo hasta 16. Lejos de ser un defecto, la holgura permite que el resultado sea más equilibrado de lo que se obtendría con algunos procesadores de texto al uso. Al igual que con `\vspace`, estos comandos deben seguirse de una línea en blanco, es decir, debemos escribir

```
Una línea
\bigskip

Otra línea
```

Si no la dejásemos, lo que veríamos es "Una línea Otra línea".

A caballo entre el alineamiento y el espaciamiento está `\hfill` que es un espacio horizontal que tiene una holgura positiva arbitrariamente grande. Lo podemos usar por tanto para completar una línea con espacios. Imaginemos que queremos poner dos firmas en nuestro documento. Una posibilidad es utilizar

```
\noindent
Madrid 23 de octubre\hfill Madrid 23 de octubre
\\[15mm]
Perico el de los Palotes \hfill Fulanito de Tal
```

El `\noindent` es para impedir la sangría inicial al cambiar de párrafo. El resultado sería:

Madrid 23 de octubre Madrid 23 de octubre

Posiblemente el alineamiento de la segunda firma no nos agrade mucho. Una solución chapucera es introducir un `\hspace{...}` tras "Tal" y la otra más ortodoxa (aunque fuera de nuestros conocimientos actuales) es meter los bloques de firma en tablas separadas con un solo `\hfill`.

Existen también las variantes `\hrulefill` y `\dotfill` que completan con una línea horizontal o con puntos. Por ejemplo, con

```
\label{thispag}Esta página \dotfill \pageref{thispag}

Esta página \hrulefill \pageref{thispag}
```

Se obtiene:

Esta página 10
Esta página _____ 10

Más realista es introducir `\hrulefill` en una línea en solitario para separar dos partes de un documento. Sin hacer nada especial la barra es baja por tanto es posible que deseemos emplear

```
Una línea

\hrulefill

\

Otra línea
```

para conseguir dicho separador o quizá cambiando el `\ ` por `\bigskip`.

Existe también un `\vfill` para jugar con espaciamientos verticales. Solo diré que si nuestro propósito es únicamente dejar el resto de la página en blanco y pasar a otra, tenemos `\newpage`.

3. Más control vertical en fórmulas

Es posible también controlar espacios verticales en fórmulas. Por cuestiones relacionadas con el modo vertical, `\vspace{...}` no se vuelve muy útil. Para algún ajuste fino se puede usar `\raisebox{...}{...}` donde el primer argumento es la medida y el segundo sobre la que se aplica. En realidad opera sobre cajas de texto y por tanto debemos utilizar el modo matemático si lo que queremos mover es una fórmula. No parece que sea muy realista tener que jugar con espacios verticales en fórmulas por tanto el

uso de `\raisebox{...}{...}` es anecdótico y, muchas veces, no demasiado aconsejable. Aquí va un ejemplo:

$$x = a\raisebox{2pt}{bc}\raisebox{-1pt}{d} \qquad y = a\raisebox{2pt}{bc}\raisebox{-1pt}{d}.$$

que se obtiene con

```
\[
 x =  a\raisebox{2pt}{bc}\raisebox{-1pt}{d}
 \qquad
 y =  a\raisebox{2pt}{$bc$}\raisebox{-1pt}{$d$}.
\]
```

En el primer caso la tipografía de las letras movidas es la de texto y en la segunda la de matemáticas.

Por lo dicho, está claro que `\raisebox` se puede usar en modo texto. Obtendremos el efecto especial l\raisebox{1pt}{í\raisebox{1pt}{n\raisebox{1pt}{e\raisebox{1pt}{a}}}} con

```
\raisebox{1pt}{l\raisebox{1pt}{í\raisebox{1pt}{n%
\raisebox{1pt}{e\raisebox{1pt}{a}}}}}
```

El % es irrelevante, es solo porque no cabía en la línea en este documento. Es para evitar que la `n` se "contamine" con algún posible espacio posterior.

Fuera de estos efectos de fantasía, a veces para dar una explicación nos gustaría que algo apareciera en un tipo menor sobre una expresión matemática o por debajo de ella, como vimos con `\xrightarrow`. Con este fin están los comandos `\overset` y `\underset` que requieren dos argumentos: el primero, la expresión que queremos poner por encima o por debajo y el segundo, la que tomamos como base. Por ejemplo, si $x + y = 4$ y queremos recordar $y = 3$ para deducir $x = 1$ lo podremos escribir como

$$x + y = 4 \underset{y=3}{\Longrightarrow} x = 1$$

empleando

```
\[
 x+y=4 \underset{y=3}{\Longrightarrow} x=1
\]
```

Otro ejemplo plausible es

```
\[
 1+\overset{n\text{ veces}}{\cdots\cdots}+1=n.
\]
```

que produce

$$1 + \overset{n \text{ veces}}{\cdots\cdots} + 1 = n.$$

Con `\cdots` conseguiremos un centrado vertical más adecuado que con `\dots`. Hay ciertas reglas por las que el paquete `amsmath` decide cuándo `\dots` es lo mismo `\cdots` y cuándo no de modo que muchas veces basta escribir `\dots` sin preocuparse. Normalmente acierta pero aquí las llaves de `\overset` le confunden.