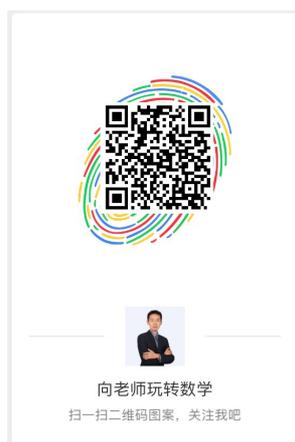

User's Guide

For the $\mathcal{A}\mathcal{M}\mathcal{S}$ math Package

amsmath 包使用手册



学好 \LaTeX , 一定要多读包。

译者: 向禹
Email: 739049687@qq.com
微博: 向老师玩转数学
微信公众号: 向老师讲数学

Version: 2.1

目 录



1	简介	1
2	amsmath 包的可选项	3
3	行间公式	5
3.1	简介	5
3.2	单个公式	7
3.3	不带对齐的换行公式	7
3.4	对齐的换行公式	8
3.5	不带对齐的公式组	8
3.6	带有多列对齐的公式组	9
3.7	对齐构建区块	10
3.8	调整标签的位置	11
3.9	垂直间距与多行公式的换行	12
3.10	中断行间公式	12
3.11	公式编号	13
3.11.1	编号秩序	13
3.11.2	编号公式的交叉引用	13
3.11.3	附属编号序列	13
3.11.4	编号风格	14
4	各种各样的数学特性	15
4.1	矩阵	15
4.2	数学空格命令	16
4.3	Dots	16
4.4	不间断的破折号	17
4.5	数学中的音符	17
4.6	根号	18
4.7	带盒子的公式	18
4.8	上下箭头	18
4.9	扩展的箭头	18
4.10	将一个符号附加给另一个符号	19

4.11 分式及相关结构	19
4.11.1 <code>\frac</code> , <code>\dfrac</code> 和 <code>\tfrac</code> 命令	19
4.11.2 <code>\binom</code> , <code>\dbinom</code> 和 <code>\tbinom</code> 命令	19
4.11.3 <code>\genfrac</code> 命令	19
4.12 连分数	20
4.13 Smash 选项	20
4.14 定界符	21
4.14.1 定界符的大小	21
4.14.2 竖线符号	22
5 算符名称	23
5.1 定义新算符名称	23
5.2 <code>mod</code> 及其相关符号	24
6 <code>\text</code> 命令	25
7 积分与求和	26
7.1 多行上下标	26
7.2 <code>\sideset</code> 命令	26
7.3 上下标的放置与 <code>limits</code> 选项	27
7.4 多重积分负号	27
8 交换图	28
9 使用数学字体	29
9.1 简介	29
9.2 推荐使用的数学字体命令	29
9.3 粗体数学符号	30
9.4 斜体希腊字母	30
参考文献	31





第 1 章 简介



amsmath 包是一个 \LaTeX 软件包，它为改进包含数学公式的文档的信息结构和打印输出提供了各种增强功能。不熟悉 \LaTeX 的读者可知参考 [3]。如果您安装了最新版本的 \LaTeX ，**amsmath** 包是已经提供好了。当新版本的 **amsmath** 包发布的时候，可以通过<http://mirror.ctan.org/macros/latex/required/amsmath.zip>进行更新。

本文档描述了 **amsmath** 包的特性，并讨论了他们的用途。**amsmath** 包还包含了一些辅助的包：

amsbsy **amsopn** **amsxtra** **amscd** **amstext**

这些都和数学公式的内容有一定关系。想要获取更多关于数学符号与数学字体的信息，参考 [8] 和<https://www.ams.org/tex/amsfonts.html>。

如果您是一个长期的 \LaTeX 用户，并且在您所写的内容中有大量的数学知识，那么您可以在这个 **amsmath** 特性列表中识别一些常见问题的解决方案：

- 类似于 $\backslash\sin$ 和 $\backslash\lim$ 的一种简单方法来定义一个新的运算符，包括合适的边间距和自动选择正确的字体样式和大小（即使在下标或上标的使用中也一样）。
- **eqnarray** 环境的多个替代项，使各种类型的公式排列更易于编写。
- 公式编号自动向上或向下调整以避免在公式内容上套印（与 **eqnarray** 不同）。
- 等号周围的间距与等号环境中的正常间距相匹配（与 **eqnarray** 不同）。
- 生成多行下标的方法，正如常用于求和或求积符号一样。
- 用可变的公式编号代替手动编号的一种简单方法。
- 一种对选定的公式生成类似于形式 (1.3a)(1.3b)(1.3c) 的子公式编号。

amsmath 主包提供了各种行间公式和其他数学结构。

amstext 提供了在公式中输入文本的 $\backslash\text{text}$ 命令。

amsopn 提供了定义类似与 $\backslash\sin$ 和 $\backslash\lim$ 的新运算符的 $\backslash\text{DeclareMathOperator}$ 命令。

amsbsy 为了向后的兼容性，这个包仍然保留着，但是更推荐使用 \LaTeX 附带的 **bm** 包。

amscd 提供了画简单交换图的的 **CD** 环境（不支持对角线的箭头）。

amsxtra 提供了一些零碎的东西，比如`\fracwithdelims` 和`\accentedsymbol`，以便与使用版本 1.1 创作的文档兼容。

amsmath 包包含了 **amstext**、**amsopn** 和 **amsbsy**。而 **amscd** 和 **amsxtra** 的特性只有分别加载了这些包才能使用。

独立的 **mathtools** 包 [10] 提供了对 **amsmath** 的一些增强：**mathtools** 会自动加载 **amsmath**，因此如果使用 **mathtools**，则无需再加载 **amsmath**。一些 **mathtools** 的设置会在以下适当位置标注。



第 2 章 amsmath 包的可选项



amsmath 包包含如下可选项：

`centertags`（缺省）对于包含`split`环境的公式，将公式编号垂直放置在公式总高度的中间。

`tbtags` 顶部或底部标签：对于包含`split`环境的公式，如果编号在右边（或左边），则将公式编号与最后一个（或第一行）放在同一水平。

`sumlimits`（缺省）将行间公式的求和符号的上标和下标分别放在上方和下方。这个选项也影响了其他类型的符号—— $\prod, \coprod, \otimes, \oplus$ 等，但是不包含积分（见下）。

`nosumlimits` 总是将求和符号的上标和下标放在求和符号的旁边，包括行间公式。

`intlimits` 类似于`sumlimits`，但是针对积分符号的。

`nointlimits`（缺省）是`intlimits`的反义。

`namelimits`（缺省）类似于`sumlimits`，但是对某些运算符例如 `det, inf, max, min` 等出现在行间公式中时，通常在下面放置下标。

`nonamelimits` 是`namelimits`的反义。

`alignedleftspaceyes`

`alignedleftspaceno`

`alignedleftspaceyesifneg`

要使用这些包选项之一，将选项名称放在`\usepackage`命令的可选参数中——例如`\usepackage[intlimits]{amsmath}`。 $\mathcal{A}\mathcal{M}\mathcal{S}$ 文档类和其他预加载想要的选项的文档类都必须通过`\documentclass`来指定——例如`\documentclass[intlimits,tbtags,reqno]{amsart}`。

amsmath 包还识别通常通过`\documentclass`命令选择（隐式或显式）的以下选项，因此不需要在`\usepackage{amsmath}`的选项列表中重复声明。

`leqno` 将公式编号放在左边。

`reqno` 将公式编号放在右边。

`fleqn` 将公式放在距离左边缘固定缩进的位置而不是放在中间。

有三个选项被添加来控制`aligned`和`gathered`环境左侧的空格。在 2017 发布之前，这些结构的左侧而不是右侧添加了一个较小的空格。这似乎是操作的一个意外特征，一般是通过在环境前面加前缀`\!`来修正。

新的缺省行为旨在确保在大多数情况下，环境中没有添加很小的空格，并且使用`\!\begin{aligned}`的现有文档继续像以前一样有效。

`alignedleftspaceyes` 通常在`aligned`和`gathered`的左侧添加`\,`。

`alignedleftspaceno` 通常在`aligned` 和`gathered` 的左侧不加`\,`。

`alignedleftspaceyesifneg` 只在环境前面添加了负距离以后才添加`\,`，（新的缺省行为）。



第 3 章 行间公式



3.1 简介

`amsmath` 提供了很多除 \LaTeX 自带之外的行间公式结构。增加的集合包括

<code>equation</code>	<code>equation*</code>	<code>align</code>	<code>align*</code>
<code>gather</code>	<code>gather*</code>	<code>alignat</code>	<code>alignat*</code>
<code>multline</code>	<code>multline*</code>	<code>flalign</code>	<code>flalign*</code>
<code>split</code>			

(虽然标准的`eqnarray`环境仍然可用，但是最好还是用`align`或者`equation+split`。在`eqnarray`环境中，关系符号周围的间距不是首选的数学间距，并且与其他环境中出现的间距不一致。此环境中的长行可能会导致错误放置或套印的公式编号。此环境也不支持使用定理包提供的`\qed`或`\qedhere`。)

除了`split`外，每个环境都至少有加星号和不加星号两种形式，其中未加星号的使用 \LaTeX 的公式计数器自动编号，您可以将`\notag`放在任一行结束之前的位置来抑制该编号。`\notag`不应该出现在行间公式外，因为它会打乱编号。您还可以使用您自己的`\tag{label}`来覆盖数字，这里的`label`表示任意字段，比如`$$`或者`ii`来给公式“编号”。一个`tag`可以通过`\tag{\ref{<label>}<modifier>}`来引用一个不同的已经标签过的行间公式，其中`<modifier>`为可选项。如果您使用了`hyperref`包，则使用`\ref*`；使用带星号的形式`\ref`可防止引用包含嵌套链接的已修改标签，使其无法链接到行间公式。

还有一个`\tag*`命令，它可以使您提供的文本按字面排版，而不在其周围添加括号。在所有`amsmath`对齐结构的未编号版本中，也可以使用`\tag`和`\tag*`。在示例文件`testmath.tex`和`subeqn.tex`中可以找到使用`\tag`标记的一些示例。

`split`环境是一种特殊的从属形式，只在其中某个环境内部使用。但不能在`multline`内使用。`split`支持仅一个对齐(`&`)列；如果需要更多，则应使用`aligned`或`alignedat`。`split`结构的宽度是整行宽度。

在进行对齐的结构中(`split`, `align`及其变体)，关系符号前面有一个`&`，但后面没有，不同于`eqnarray`。将`&`放在关系符号后面会干扰正常间距；它必须放在前面。

在所有的多行环境中，行之间用`\\`分割，`\\`不应用于结束行，否则放在最后一行会导致不想要的额外的垂直间距。

在所有的数学环境中(行内或行间)，不允许使用空行(相当于`\par`)，这会导致错误。

不同行间公式的比较（垂直线表示标准边线）

$\begin{equation*} a=b \end{equation*}$	$a = b$
$\begin{equation} a=b \end{equation}$	$(3.1) \quad a = b$

$\begin{equation}\label{xx} \begin{split} a&=b+c-d\\ &\quad+e-f\\ &=g+h\\ &=i \end{split} \end{equation}$	$(3.2) \quad \begin{aligned} a &= b + c - d \\ &+ e - f \\ &= g + h \\ &= i \end{aligned}$
---	--

$\begin{multline} a+b+c+d+e+f\\ +i+j+k+l+m+n \end{multline}$	$(3.3) \quad \begin{aligned} a + b + c + d + e + f \\ + i + j + k + l + m + n \end{aligned}$
--	--

$\begin{gather} a_1=b_1+c_1\\ a_2=b_2+c_2-d_2+e_2 \end{gather}$	$(3.4) \quad a_1 = b_1 + c_1$
\end{gather}	$(3.5) \quad a_2 = b_2 + c_2 - d_2 + e_2$

$\begin{align} a_1=b_1+c_1\\ a_2=b_2+c_2-d_2+e_2 \end{align}$	$(3.6) \quad a_1 = b_1 + c_1$
\end{align}	$(3.7) \quad a_2 = b_2 + c_2 - d_2 + e_2$

$\begin{align} a_{11}&=b_{11}&a_{12}&=b_{12}\\ a_{21}&=b_{21}&a_{22}&=b_{22}+c_{22} \end{align}$	$(3.8) \quad a_{11} = b_{11} \quad a_{12} = b_{12}$
\end{align}	$(3.9) \quad a_{21} = b_{21} \quad a_{22} = b_{22} + c_{22}$



3.4 对齐的换行公式

类似于`multiline`，`split`环境是针对单个长公式，超出一行而不得分成多行公式。与`\multiline`不同的是，`split`环境提供了行之间的对齐，使用`&`标记对齐点。与其他`amsmath`的公式结构不同，`split`环境不提供编号，因为它只能在其他行间公式结构中使用，一般是在`equation`，`align`或者`gather`等提供编号的环境。例如

$$(3.11) \quad H_c = \frac{1}{2n} \sum_{l=0}^n (-1)^l (n-l)^{p-2} \sum_{l_1+\dots+l_p=1} \prod_{i=1}^p \binom{n_i}{l_i} \cdot [(n-1) - (n_i - l_i)]^{n_i - l_i} \cdot \left[(n-l)^2 - \sum_{j=1}^p (n_i - l_i)^2 \right].$$

```
\begin{equation}\label{e:barwq}\begin{split}
H_c&=\frac{1}{2n} \sum^n_{l=0}(-1)^l(n-l)^{p-2}
\sum_{l_1+\dots+l_p=1}\prod^p_{i=1} \binom{n_i}{l_i}\backslash
&\quad\cdot[(n-1)-(n_i-l_i)]^{n_i-l_i}\cdot
\Bigl[(n-l)^2-\sum^p_{j=1}(n_i-l_i)^2\Bigr].
\end{split}\end{equation}
```

除了像`\label`这样生成不可见实体的命令，`split`结构应该构成整个封闭结构的整个主体。

3.5 不带对齐的公式组

`gather`环境用于接连的不需要对齐的公式组，每一个公式分别在整行宽度居中。`gather`中的公式被命令`\`分开，`gather`中的每一个公式都可以包含一个`\begin{split}...\end{split}`结构，例如

```
\begin{gather}
first equation\backslash
\begin{split}
second & equation\backslash
& on two lines
\end{split}
\backslash
third equation
\end{gather}
```



$$\begin{aligned}
 (3.18) \quad & x = y_1 - y_2 + y_3 - y_5 + y_8 - \dots \quad \text{by (??)} \\
 (3.19) \quad & = y' \circ y^* \quad \text{by (??)} \\
 (3.20) \quad & = y(0)y' \quad \text{by Axiom 1.}
 \end{aligned}$$

`flalign` 环境 (“full length alignment”) 将公式列之间的空间拉伸到可能的最大宽度，只在公式编号的空白处留下足够的空间。

$$\begin{aligned}
 (3.21) \quad & x = y & X = Y \\
 (3.22) \quad & x' = y' & X' = Y' \\
 (3.23) \quad & x + x' = y + y' & X + X' = Y + Y'
 \end{aligned}$$

```

\begin{flalign}
x&=y && & X&=Y\\
x'&=y' && & X'&=Y'\\
x+x'&=y+y' && & X+X'&=Y+Y'
\end{flalign}

```

$$\begin{aligned}
 & x = y & X = Y \\
 & x' = y' & X' = Y' \\
 & x + x' = y + y' & X + X' = Y + Y'
 \end{aligned}$$

```

\begin{flalign*}
x&=y && & X&=Y\\
x'&=y' && & X'&=Y'\\
x+x'&=y+y' && & X+X'&=Y+Y'
\end{flalign*}

```

3.7 对齐构建区块

类似于`equation`，多公式环境`gather`，`align`和`alignat`旨在构建一个全长的结构。这意味着，例如，我们无法在整个结构旁边添加括号。但是变体环境`gathered`，`aligned`和`alignedat`使得内容的长度就是它的实际长度，因此可以用作包含表达式的组件。例如

$$\left. \begin{aligned}
 B' &= -\partial \times E, \\
 E' &= \partial \times B - 4\pi j,
 \end{aligned} \right\} \text{Maxwell 方程组}$$



```

\begin{equation*}
\left.\begin{aligned}
B'&=-\partial\times E,\\
E'&=\partial\times B-4\pi j,
\end{aligned}\right\}
\quad \text{Maxwell 方程组}
\end{equation*}

```

类似于array环境，这些带有-ed的变体也接受可选参数[t]，[b]或者缺省的[c]来指定垂直对齐方式。为协调起见，不要在可选项前添加空格或断行。

像下面的“Cases”结构在数学中时很常见的：

$$(3.24) \quad P_{r-j} = \begin{cases} 0, & \text{如果 } r-j \text{ 为奇数} \\ r!(-1)^{(r-j)/2}, & \text{如果 } r-j \text{ 为偶数} \end{cases}$$

而在 **amsmath** 包中有一个cases环境可以很简单地实现：

```

P_{r-j}=\begin{cases}
0,&\text{如果}r-j\text{为奇数}\\
r!(-1)^{(r-j)/2},&\text{如果}r-j\text{为偶数}
\end{cases}

```

这里注意\text{blabla}的使用，以及嵌套的数学公式。cases的字体设置为\textstyle，如果需要\displaystyle，一定要直接声明；在 **mathtools** 包中提供的dcases环境可以直接实现。

-ed和cases环境只能放在一个封闭的数学环境中，可以是行内的 \dots ，也可以是任意的行间环境。

3.8 调整标签的位置

在多行公式中，放置公式编号可能是一个相当复杂的问题。**amsmath**软件包的环境尽量避免在公式内容上套印公式编号，如有必要，可将该编号向下或向上移动到单独的行。

精确计算一个公式的轮廓的困难有时会导致数字的移动看起来不正确。`\raisetag`命令来调整当前公式编号的垂直位置，如果它已从其正常位置移开，要将特定编号向上移动6pt，使用`\raisetag{6pt}`。（在行间公式结束时，也向上移动此公式后的文本。）这种调整就像换行符和分页符一样是很好的调整，因此在您的文档接近定稿之前，应该先撤消。或者，您最终可能会重复多次微调以跟上文档内容的变化。



3.9 垂直间距与多行公式的换行

您可以使用`\[距离]`命令在所有 **amsmath** 的行间公式环境中获取行与行之间的额外垂直空间，这在 **LaTeX** 中很常见。不要在`\[`和`[`之间键入空格；仅对于 **amsmath** 定义的行间环境，该空格被解释为表示带括号的内容是文档内容的一部分。

当使用 **amsmath** 包时，公式行之间的分页符通常是不允许的；其原理是，此类内容中的分页符应引起作者的个人注意。要在特定的行间公式中获取单独的分页符，可以使用`\displaybreak`。最好将`\displaybreak`命令放在要生效的位置`\[`之前。像 **LaTeX** 的`\pagebreak`一样，`\displaybreak`采用 0 到 4 之间的可选参数来表示分页的可取性。`\displaybreak[0]`表示“允许在不鼓励中断的情况下中断此处”：不带可选参数的`\displaybreak`与`\displaybreak[4]`一样，表示强制分页。

如果您想随处使用分页符，即使是在多行公式的中间，那么您可以将`\allowdisplaybreaks[1]`放在文档的导言区。可选参数 1-4 可用于更精细的控制：`[1]`表示允许分页符，但尽可能避免此选项。参数 2、3、4 表示允许性增加。当使用`\allowdisplaybreaks`启用行间公式分页时，可以像往常一样，使用`\[`*命令禁止给定行后的分页中断。

注：一些公式环境把内容包装在了一个不可断行的盒子中，那么此时`\displaybreak`和`\allowdisplaybreaks`对它们都不起作用，这些包括`split`，`aligned`，`gathered`和`alignedat`。

3.10 中断行间公式

`\intertext`命令用于在多行公式中间插入一或两行文字（参见第六章的`\text`命令）。它的显著特点是保持对齐，如果您只是简单地结束行间公式然后再重新使用行间公式，那么这种对齐就不会出现。`\intertext`只能出现在`\[`或`\[`*命令之后。注意下面例子中的“和”的位置。

```
\begin{align}
A_1&=N_0(\lambda;\Omega')-\phi(\lambda;\Omega),\\
A_2&=\phi(\lambda;\Omega')-\phi(\lambda;\Omega),\\
\intertext{和}
A_3&=\mathcal{N}(\lambda;\omega).
\end{align}
```

$$(3.25) \quad A_1 = N_0(\lambda; \Omega') - \phi(\lambda; \Omega),$$

$$(3.26) \quad A_2 = \phi(\lambda; \Omega') - \phi(\lambda; \Omega),$$

和

$$(3.27) \quad A_3 = \mathcal{N}(\lambda; \omega).$$



mathtools 包提供了 `\shortintertext` 命令用于插入少量的文字；它比 `\intertext` 使用更少的垂直距离，当公式编号在右边时尤其有效。

3.11 公式编号

3.11.1 编号秩序

在 \LaTeX 中，如果您想让公式随着 section 进行编号——即在第一节和第二节中分别让公式编号形式为 (1.1), (1.2), ..., (2.1), (2.2), 等等。——您可以重新定义 `\theequation` 即可：

```
\renewcommand{\theequation}{\thesection.\arabic{equation}}
```

这是非常有效的方法，但是公式编号计数器不会随着新的章节的开始而清零，除非您使用 `\setcounter` 进行重置。此时可以在导言区加入 `\makeatletter\@addtoreset{equation}{section}\makeatother`，但是 **amsmath** 包提供了更好的命令 `\numberwithin`，使得公式编号与章节编号绑定，且随着新一节的开始，公式编号自动归零，输入

```
\numberwithin{equation}{section}
```

正如此命令的名字所示，`\numberwithin` 可以适用于任何计数器，不仅仅是 `equation` 计数器。

3.11.2 编号公式的交叉引用

`\eqref` 使得公式的交叉引用更加简单，它还在公式旁边自动增加了括号。即如果 `\ref{abc}` 得到 3.2，则 `\eqref{abc}` 会得到 (3.2)。由 `eqref` 引用的公式不论内容如何，它旁边的括号都是直立形状的。

3.11.3 附属编号序列

amsmath 还提供了一个包装环境，`subequations`，以便在特定的对齐环境或类似环境中使用从属编号方案对公式进行编号。例如

```
\begin{subequations}
...
\end{subequations}
```

使得在此部分的所有公式编号形式变为 (4.9a)(4.9b)(4.9c)...，如果它的上一个公式编号为 (4.8)。在 `\begin{subequations}` 后紧跟的 `\label` 命令会生成一个带括号的 4.9 的引用标签，而不是 (4.9a)。`subequations` 使用的计数器组是 `parentequation` 和 `equation`，



而`\addtocounter`, `\setcounter`, `\value`等, 可以像往常一样应用于这些计数器。要获得任何除小写字母之外的从属编号, 使用标准的 L^AT_EX 改变编号形式 [3], §6.3, §C.8.4。例如像下面一样重定义`\theequation`来得到罗马数字。

```
\begin{subequations}
\renewcommand{\theequation}{\theparentequation \roman{equation}}
...
\end{subequations}
```

3.11.4 编号风格

默认公式编号设置为`\normalfont`, 这意味着在粗体部分标题中, 粗体是被抑制的。解决方法是使用`(\ref{...})`而不用`\eqref{...}`。

如果行间公式编号被指定了较小的类型, 则公式编号的大小也会变小。默认的大小可以通过在导言区加入以下补丁得到确保:

```
\makeatletter
\renewcommand{\maketag@@}[1]{\hbox{\m@th\normalsize\normalfont#1}}%
\makeatother
```

(这个修正或许会放在 **amsmath** 以后的版本中。)



第 4 章 各种各样的数学特性



4.1 矩阵

除了 L^AT_EX 的标准的 `array` 环境外, **amsmath** 提供了一些其他的矩阵环境。`pmatrix`, `bmatrix`, `Bmatrix`, `vmatrix` 和 `Vmatrix` 分别带有定界符 $()$, $[\]$, $\{ \}$, $\| \|$ 和 $\| \| \|$ 。为了命名的连贯性, 还有一个 `matrix` 环境不带定界符。这在 `array` 环境中并不是完全多余的; `matrix` 环境比 `array` 还使用了更加节俭的水平空格; 而且与 `array` 环境不同的是, 您不需要给出矩阵环境的列规范。默认情形下, 至多可以有 10 个居中的列¹。(如果您的列有左对齐或右对齐或者其他格式需求, 您可以使用 `array`, 或者 **mathtools** 包提供了这些环境的 * 变体环境, 带有可选参数来指定左对齐或右对齐。)

要生产一个适合文本的小矩阵, 我们有 `smallmatrix` 环境 (例如 $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$) 比普通矩阵更适合于单行文本。这里的定界符一定要指明。(**mathtools** 包提供了 `smallmatrix` 的 `p,b,B,v,V` 版本, 以及上面讨论过的 * 变体) 上述例子生成是通过

```
\bigl(\begin{smallmatrix}
a&b\\ c&d
\end{smallmatrix}\bigr)$
```

`\hdotsfor{<number>}` 在矩阵的给定列数中生成一行点, 例如

```

a b c d
e .....
\begin{matrix}
a&b&c&d\\
e&\hdotsfor{3}
\end{matrix}$
```

点之间的距离可以通过方括号选项修改, 例如, `\hdotsfor[1.5]{3}`。方括号里面的数字会被作为一个倍数 (例如默认的倍数是 1.0)

$$(4.1) \quad \begin{pmatrix} D_1 t & -a_{12} t_2 & \dots & -a_n t_n \\ -a_{21} t_1 & D_2 t & \dots & -a_{2n} t_n \\ \dots & \dots & \dots & \dots \\ -a_{n1} t_1 & -a_{n2} t_2 & \dots & D_n t \end{pmatrix}$$

```
\begin{equation}
\begin{pmatrix}
\end{pmatrix}
```

¹确切地说, 矩阵的最大列数由计数器 **MaxMatrixCols**(默认值 =10) 控制, 如果有必要的话, 可以用 L^AT_EX 的 `\setcounter` 或 `\addtocounter` 进行更改。

```

D_1t&-a_{12}t_2&\dots&-a_{n}t_n\\
-a_{21}t_1&D_2t&\dots&-a_{2n}t_n\\
\hdotsfor[2]{4}\\
-a_{n1}t_1&-a_{n2}t_2&\dots&D_nt
\end{pmatrix}
\end{equation}

```

4.2 数学空格命令

`amsmath` 包稍微扩展了一组数学间距命令，如下所示。这些命令的全称和缩写形式都是健壮的，它们也可以在数学之外使用。

缩写	全称	例子	缩写	全称	例子
	无空格	$\Rightarrow \Leftarrow$		无空格	$\Rightarrow \Leftarrow$
<code>\,</code>	<code>\thinspace</code>	$\Rightarrow \Leftarrow$	<code>\!</code>	<code>\negthinspace</code>	$\Rightarrow \Leftarrow$
<code>\:</code>	<code>\medspace</code>	$\Rightarrow \Leftarrow$		<code>\negmedspace</code>	$\Rightarrow \Leftarrow$
<code>\;</code>	<code>\thickspace</code>	$\Rightarrow \Leftarrow$		<code>\negthickspace</code>	$\Rightarrow \Leftarrow$
		$\Rightarrow \quad \Leftarrow$			$\Rightarrow \quad \Leftarrow$
		$\Rightarrow \quad \Leftarrow$			$\Rightarrow \quad \Leftarrow$

为了最大限度地控制数学间距，请使用`\mspace`和数学单位。一个数学单位，或者说`\mu`，等于 1/18 em。因此，要得到一个负的`\quad`，您可以输入`\mspace{-18.0\mu}`。

4.3 Dots

对于各种上下文中省略号点（抬升的或者与基线持平的）的首选位置，没有普遍的共识，因此这可以看做是个人的喜好问题。使用下面的语义导向的命令

- `\dotsc` 意味着“dots with commas”
- `\dotsb` 意味着“dots with binary operators/relations”
- `\dotsm` 意味着“multiplication dots”
- `\dotsi` 意味着“dots with integrals”
- `\dotso` 意味着“other dots”（除上面之外的）

而不用`\ldots`或`\cdots`，您可以让您的文档随时随地适应不同的惯例，例如，万一您必须把它交给一个在这方面坚持遵循家庭传统的出版商，各种类型的默认处理遵循美



国数学社会惯例：

Then we have the series A_1, A_2, \dots ,

the regional sum $A_1 + A_2 + \dots$,

the orthogonal product $A_1 A_2 \dots$, and

the infinite integral $\int_{A_1} \int_{A_2} \dots$.

Then we have the series A_1, A_2, \dots , the regional sum $A_1 + A_2 + \dots$, the orthogonal product $A_1 A_2 \dots$, and the infinite integral

$$\int_{A_1} \int_{A_2} \dots$$

在大多数情况下。可以使用未区分的`\dots`，**amsmath** 将根据即时上下文输出最合适的形式：如果不适当的形式出现了，则可以在检查输出后更正。

4.4 不间断的破折号

`\nobreakdash` 命令用来抑制在连字符或破折号后换行的可能性。例如您把page1-9 写为page 1\nobreakdash--9，那么在破折号与9 之间绝不会出现换行。您还可以使用`\nobreakdash` 来防止不想要的连字符，比如`p-adic`。如果频繁使用的话，您可以使用缩写：

```
\newcommand{\p}{$p$\nobreakdash}% for “\p-adic”
\nobreakdash--}% for “pages 1\nobreakdash 9”
% for “\n dimensional” (“n-dimensional”):
\nobreakdash-\hspace{0pt}}
```

最后一个例子展示了如何在下面的单词中禁止连字符后换行。（在连字符后添加零宽度的空格即可）。

4.5 数学中的音符

在一般的 \LaTeX 中，在数学中添加二重音符是很差的效果，但是在 **amsmath** 包里，您可以改良二重音符： $\hat{\hat{A}}$ (`\hat{\hat{A}}`)。

除了 \LaTeX 里面已有的`\dot`和`\ddot`命令之外，**amsmath** 还提供了`\dddots`和`\ddddot`命令来生成三重和四重点音符。

要想得到上标的`hat`或者`tilde`符号，加载 **amsxtra** 包，使用`\sphat`或者`\sptilde`。使用方法为`A\sphat` (A^\frown)。

要在数学音符的位置放置任意一个符号，或者要得到下音符，可以参考 Javier Bezos 制作的 **accents** 包。（**amsmath** 包一定要在 **accents** 包之前加载。）



4.6 根号

在一般的 \LaTeX 中，根号指标的放置有时候并不好： $\sqrt[k]{\beta}$ (`\sqrt[\beta]{k}`)。在 `amsmath` 包中，`\leftroot` 和 `\uproot` 可以让您调整根号的位置：

```
\sqrt[\leftroot{-2}\uproot{2}\beta]{k}
```

将 β 向上和向右平移： $\sqrt[k]{\beta}$ 。在 `\leftroot` 中的负的参数使得 β 向右移。这里的距离单位是非常小的数，适用于这里的调整。

4.7 带盒子的公式

`\boxed` 命令给它的参数放置一个盒子，类似于 `\fbox`，但不同的是这里的内容必需是在数学模式中：

$$(4.2) \quad \boxed{\eta \leq C(\delta(\eta)) + \Lambda_M(0, \delta)}$$

```
\boxed{\eta \leq C(\delta(\eta)) + \Lambda_M(0, \delta)}
```

4.8 上下箭头

基本的 \LaTeX 提供了 `\overrightarrow` 和 `\overleftarrow` 命令。而 `amsmath` 提供了一些其他的扩展的上下箭头：

```
\overleftarrow \underleftarrow \overrightarrow
\underrightarrow \overleftrightharpoon \underleftrightharpoon
```

4.9 扩展的箭头

`\xleftarrow` 和 `\xrightarrow` 产生的箭头可以自适应于它的上标或下标。这两个命令带一个可选参数（上标）和一个必选参数（下标，可以为空）：

$$(4.3) \quad A \xleftarrow{n+\mu-1} B \xrightarrow[T]{n\pm i-1} C$$

```
A\xleftarrow{n+\mu-1}B\xrightarrow[T]{n\pm i-1}C
```



4.10 将一个符号附加给另一个符号

\LaTeX 提供了 `\stackrel` 命令可以在一个二元算符上放置一个上标，而在 `amsmath` 中有更多的命令，`\overset`，`\underset`，可以把一个符号放在另一个符号，关系或者其他东西的上方或下方。输入 `\overset{*}{X}` 会在 X 上方放置一个上标 $*$ ，即 X^* ；`\underset` 是类似地放置一个下标。

也参见 §7.2 中关于 `\sideset` 的描述。

4.11 分式及相关结构

4.11.1 `\frac`，`\dfrac` 和 `\tfrac` 命令

`\frac` 命令是 \LaTeX 的一个基本命令，带有两个参数——分子和分母——然后用普通的分式打出来。`amsmath` 还提供了 `\dfrac` 和 `\tfrac` 命令分别作为 `\displaystyle\frac...` 和 `\textstyle\frac...` 的缩写

$$(4.4) \quad \frac{1}{k} \log_2 c(f) \quad \frac{1}{k} \log_2 c(f) \quad \sqrt{\frac{1}{k} \log_2 c(f)} \quad \sqrt{\frac{1}{k} \log_2 c(f)}$$

```
\begin{equation}
\frac{1}{k}\log_2 c(f)\;\tfrac{1}{k}\log_2 c(f)\;
\sqrt{\frac{1}{k}\log_2 c(f)}\;\sqrt{\dfrac{1}{k}\log_2 c(f)}
\end{equation}
```

4.11.2 `\binom`，`\dbinom` 和 `\tbinom` 命令

对于类似于 $\binom{n}{k}$ 的二项式表达式，`amsmath` 提供了 `\binom`，`\dbinom` 和 `\tbinom` 命令：

$$(4.5) \quad 2^k - \binom{k}{1} 2^{k-1} + \binom{k}{2} 2^{k-2}$$

```
2^k-\binom{k}{1}2^{k-1}+\binom{k}{2}2^{k-2}
```

4.11.3 `\genfrac` 命令

`\frac`，`\binom` 以及他们的变体的功能都是由 `\genfrac` 命令来生成的，`\genfrac` 带有六个参数，最后两个参数对应于 `\frac` 的分子与分母，前两个参数是可选的定界符（比如在 `\binom` 中看到的）第三个参数是一条有厚度的线（`\binom` 把线的厚度设置为 0，也就是不可见）；第四个参数是数学格式：整数值 0-3 分布对应



于`\displaystyle`, `\textstyle`, `\scriptstyle` 和 `\scriptscriptstyle`, 如果第三个参数置空, 则线条的厚度默认为“正常”。

`\genfrac{左定界符}{右定界符}{厚度}{数学格式}{分子}{分母}`

这里说明一下`\frac`, `\tfrac` 和 `\binom` 是如何定义的:

`\newcommand{\frac}[2]{\genfrac{}{}{}{}{#1}{#2}}`

`\newcommand{\tfrac}[2]{\genfrac{}{}{}{1}{#1}{#2}}`

`\newcommand{\binom}[2]{\genfrac{({}{)}{0pt}{}{}{#1}{#2}}`

如果您发现自己要在一篇文档中重复地用`\genfrac` 定义一个特殊的记号, 为了您自己以及出版社的方便, 您可以用类似于`\frac`, `\binom` 的形式定义一个名字有意义的简写记号。

最原始的生成分数的命令`\over`, `\overwithdelims`, `\atop`, `\atopwithdelims`, `\above`, `\abovewithdelims` 和 **amsmath** 一起使用的时候会发出警告信息, 这个原因在 `technote.tex` 中讨论了。

4.12 连分数

$$(4.6) \quad \frac{1}{\sqrt{2} + \frac{1}{\sqrt{2} + \frac{1}{\sqrt{2} + \dots}}}$$

```
\begin{equation}
\cfrac{1}{\sqrt{2}+
\cfrac{1}{\sqrt{2}+
\cfrac{1}{\sqrt{2}+\dotsb
}}}
\end{equation}
```

这样比直接用`\frac` 产生的结果更好看。要把其中任何一个分式的分子放在左边或右边是靠`\cfrac[l]` 或 `\cfrac[r]` 来实现的, 而不是`\frac`。

4.13 Smash 选项

`\smash` 命令用来输出一个高度和深度为 0 的子公式, 有时候在调整子公式与旁边符号的位置是很有用的。在 **amsmath** 中, `\smash` 命令带有一个可选参数 `[t]` 或 `[b]`,



因为有时候在保留自然深度和高度时，`smash` 只用于 `top` 或 `bottom` 会更好。例如，当旁边的根号因为内容的高度和深度的差异在位置或大小不均匀时，`\smash` 命令可以使其更加一致。试比较 $\sqrt{x} + \sqrt{y} + \sqrt{z}$ 和 $\sqrt{x} + \sqrt{y} + \sqrt{z}$ 的区别，其中后者是由 `\sqrt{x}+\sqrt{\smash[b]{y}}+\sqrt{z}` 生成的。

4.14 定界符

4.14.1 定界符的大小

由 `\left` 和 `\right` 自适应的定界符大小有两点限制：首先它是机械式地产生足够大的定界符能够包围定界符中间的最大内容，其二就是大小的范围甚至不是近似连续的，但有相当大的量级跳跃。这意味着，对于给定的分隔符大小而言，太大的数学片段无穷大，将得到更大的下一个大小，在正常大小的文本中，将跳转 3pt 左右。有两种或三种情况通常会调整定界符大小。使用一组名字中包含 ‘big’ 的命令。

定界符	文本	<code>\left</code>	<code>\bigl</code>	<code>\Bigl</code>	<code>\biggl</code>	<code>\Biggl</code>
大小	大小	<code>\right</code>	<code>\bigr</code>	<code>\Bigr</code>	<code>\biggr</code>	<code>\Biggr</code>
结果		$(b)(\frac{c}{d})$	$(b)(\frac{c}{d})$	$(b)(\frac{c}{d})$	$(b)(\frac{c}{d})$	$(b)(\frac{c}{d})$

第一种情形是求和算符中带有上下标的，用 `\left` 和 `\right` 产生的定界符一般比需要的大，因此用 `Big` 或 `bigg` 类的大小能得到更好的结果：

$$\left[\sum_i a_i \left| \sum_j x_{ij} \right|^p \right]^{1/p} \quad \left[\sum_i a_i \left| \sum_j x_{ij} \right|^p \right]^{1/p}$$

`\left[\sum_i a_i\left|\sum_j x_{ij}\right|^p\right]^{1/p}\quad\biggl[\sum_i a_i\Bigl|\sum_j x_{ij}\Bigr|^p\biggr]^{1/p}`

第二种情形是有多个定界符的时候，`\left` 和 `\right` 会使得所有定界符大小相同（因为这已经足以覆盖其中最大的内容了），但事实上您可能希望使得外面的定界符稍微大于内嵌的定界符：

```
\left(a_1 b_1)-(a_2 b_2)\right)
\left(a_2 b_1)+(a_1 b_2)\right)
\quad\text{versus}\quad
\bigl(a_1 b_1)-(a_2 b_2)\bigr)
\bigl(a_2 b_1)+(a_1 b_2)\bigr)
```

第三种情况是在运行文本时有一种稍微过大的对象，例如 $\left|\frac{b'}{a'}\right|$ ，这里由 `\left` 和 `\right` 产生的定界符会导致过多的行伸展。在这种情况下，可以使用 `\bigl` 和 `\bigr` 生成比基本大小稍大但仍能适应正常行距的定界符： $\bigl|\frac{b'}{a'}\bigr|$



在一般的 L^AT_EX 中，`\big`，`\bigg`，`\Big` 和 `\Bigg` 定界符无法适当地伸展，但是加载了 `amsmath` 包以后就可以做到了。

4.14.2 竖线符号

`amsmath` 包提供了 `\lvert`，`\rvert`，`\lVert`，`\rVert` 命令（相比于 `\langle`，`\rangle`）以解决垂直字符 `|` 的过载问题。这个符号在数学中表示非常广泛的数学关系：比如数论中的整除关系： $p|q$ ，或者绝对值符号 $|z|$ ，或者集合符号中的“使得”，或者“取值于” $f_{\zeta}(t)|_{t=0}$ 。

使用的多样性本身并不是那么糟糕：然而，糟糕的是，并非所有的使用都采用相同的排版方式，而且知识渊博的读者的完全辨别能力在数学文档的计算机处理中是无法复制的。因此，建议在任何给定的文档中，`vert` 或者 `|` 与一个选定的数学记号之间应该一一对应，这对双竖线命令 `\|` 也是一样的。这将立即排除使用定界符 `|` 和 `\|` 的可能性。因为 `\left` 和 `\right` 定界符在相邻符号中不相关，因此，在文档前言中定义适当的命令，以便任意成对的竖条定界符的使用。

```
\providecommand{\abs}[1]{\lvert#1\rvert}
\providecommand{\norm}[1]{\lVert#1\rVert}
```

于是此文档就可以用 `\abs{z}` 生成 $|z|$ ，用 `\norm{v}` 生成 $\|v\|$ 。`mathtools` 包提供了 `\DeclarePairedDelimiter` 命令用来定义类似于 `\abs` 的命令，但是定界符大小是可变的。



第 5 章 算符名称



5.1 定义新算符名称

数学函数，例如 \log ， \sin 和 \lim 用罗马形式输出使得它们与一般的单个字母的斜体数学变量相区别。比较常用的函数都有预定义的名称，`\log`，`\sin`，`\lim` 等，但是在数学文档中经常会出现一些新的命令，所以 `amsmath` 包提供了一个定义新算符名称的命令。要定义一个类似于 `\sin` 的命令 `\xxx`，您可以在导言区输入

```
\DeclareMathOperator{\xxx}{xxx}
```

于是在输入 `\xxx` 时就会以适当的字体输出 `xxx`，并且如果有必要的话会在两边自动加上适当的间距，所以您得到的是 $A \ xxx \ B$ 而不是 $Axxx B$ 。在 `\DeclareMathOperator` 的第二个参数（名称部分）中，第二个连字符是文本连字符而不是负号，星号 `*` 将会以抬升的文本输出而不是以一个居中的数学星号（比较 $a-b*c$ 和 $a - b * c$ ），否则名称文本将以数学模式打印，以便您可以使用，例如那里的上标和下标。

如果新的算符需要带有 ‘limits’ 上标或下标，例如 \lim , \sup , \max ，使用带 `*` 形式的 `\DeclareMathOperator` 命令：

```
\DeclareMathOperator*{\Lim}{Lim}
```

也可以参看 §7.3 的关于上下标的设置。

下面的算符名称是预定义的：

<code>\arccos</code>	<code>arccos</code>	<code>\deg</code>	<code>deg</code>	<code>\lg</code>	<code>lg</code>	<code>\projlim</code>	<code>projlim</code>
<code>\arcsin</code>	<code>arcsin</code>	<code>\det</code>	<code>det</code>	<code>\lim</code>	<code>lim</code>	<code>\sec</code>	<code>sec</code>
<code>\arctan</code>	<code>arctan</code>	<code>\dim</code>	<code>dim</code>	<code>\liminf</code>	<code>lim inf</code>	<code>\sin</code>	<code>sin</code>
<code>\arg</code>	<code>arg</code>	<code>\exp</code>	<code>exp</code>	<code>\limsup</code>	<code>lim sup</code>	<code>\sinh</code>	<code>sinh</code>
<code>\cos</code>	<code>cos</code>	<code>\gcd</code>	<code>gcd</code>	<code>\ln</code>	<code>ln</code>	<code>\sup</code>	<code>sup</code>
<code>\cosh</code>	<code>cosh</code>	<code>\hom</code>	<code>hom</code>	<code>\log</code>	<code>log</code>	<code>\tan</code>	<code>tan</code>
<code>\cot</code>	<code>cot</code>	<code>\inf</code>	<code>inf</code>	<code>\max</code>	<code>max</code>	<code>\tanh</code>	<code>tanh</code>
<code>\coth</code>	<code>coth</code>	<code>\injlim</code>	<code>injlim</code>	<code>\min</code>	<code>min</code>		
<code>\csc</code>	<code>csc</code>	<code>\ker</code>	<code>ker</code>	<code>\Pr</code>	<code>Pr</code>		
<code>\varinjlim</code>	\varinjlim	<code>\varliminf</code>	\varliminf				
<code>\varprojlim</code>	\varprojlim	<code>\varlimsup</code>	\varlimsup				

5.2 mod 及其相关符号

命令`\mod`、`\bmod`、`\pmod`、`\pod`用于处理“mod”符号的特殊间距约定。`\bmod`和`\pmod`可以在 \LaTeX 中使用，但使用`amsmath`包时，`\pmod`用于非行间模式时，它的间距将调整为较小的值。`\mod`和`\pod`是某些作者首选的`\pmod`变体：`\mod`省略了括号，而`\pod`省略“mod”并保留括号。

$$(5.1) \quad \gcd(n, m \bmod n); \quad x \equiv y \pmod{b}; \quad x \equiv y \bmod c; \quad x \equiv y \pmod{d}$$

```
\begin{equation}
\gcd(n, m \bmod n); \quad x \equiv y \pmod{b};
\quad x \equiv y \bmod c; \quad x \equiv y \pmod{d}
\end{equation}
```



第 6 章 `\text` 命令



`\text` 命令的主要作用是用于一些行间公式的文字。它和 \LaTeX 的 `\mbox` 命令的作用很像，但是有更多的优点。如果您希望一段文字出现在下标，可以输入 `..._{\text{word or phrase}}`，这个要比 `\mod` 的等价形式简单：

`...{\mbox{\rmfamily\scriptsize word or phrase}}`。注意到标准的 `\textrm` 命令会使用 **amsmath** 的 `\text` 命令，但是保证 `\rmfamily` 字体被使用。

$$(6.1) \quad f_{[x_{i-1}, x_i]} \text{ is monotonic, } i = 1, \dots, c + 1$$

`f_{[x_{i-1}, x_i]} \text{ is monotonic, } \quad i=1, \dots, c+1`

`\text` 的字体会和周围的环境字体保持一致，例如在定理环境中，`\text` 的内容会变成斜体字体。

如果在 `\text` 命令中包含数学表达式，一定要用 `$...$` 指明。

$$\partial_s f(x) = \frac{\partial}{\partial x} f(x_0) \quad f(x) \quad \text{for } x = x_0 + I - x.$$

`\partial_s f(x) = \frac{\partial}{\partial x} f(x_0) \quad f(x) \quad \text{for } x = x_0 + I - x.`

函数名不应该用 `\text` 输入，而应该用 `\mathrm` 或 `\DeclareMathOperator` 比较合适。这些是固定实体，不应根据外部内容进行更改（例如出现在用斜体设置的定理中），并且在声明的运算符的情况下，自动应用适当的间距。

第 7 章 积分与求和



7.1 多行上下标

`\substack` 命令可以用于生成多行下标或上标，例如

$$\begin{array}{l} \backslash\text{sum}_{\backslash\text{substack}} \\ \{0 \leq i \leq m \backslash \\ \{0 < j < n\}} \\ P(i, j) \end{array} \quad \sum_{\substack{0 \leq i \leq m \\ 0 < j < n}} P(i, j)$$

更一般的形式是 `subarray` 环境可以指定每一行的左对齐而不限于居中，例如

$$\begin{array}{l} \backslash\text{sum}_{\backslash\text{begin}\{\text{subarray}\}\{1\}} \\ \quad i \backslash\text{in} \backslash\Lambda \backslash \quad 0 < j < n \\ \backslash\text{end}\{\text{subarray}\}} \\ P(i, j) \end{array} \quad \sum_{\substack{i \in \Lambda \\ 0 < j < n}} P(i, j)$$

7.2 `\sideset` 命令

`\sideset` 命令是为了一个特殊用途：将一个符号放在一个巨算符的上标或下标角落上，比如 \sum 或 \prod 。注意这个命令不能用于其他非求和符号类的巨算符。典型的例子是如果您想在一个求和符号上输入一撇 (`\prime`)，如果没有 `'limits'` 选项，您可以使用 `\nolimits`，比如在行间模式中输入 `\sum\nolimits' E_n`：

$$(7.1) \quad \sum' E_n$$

然而，如果您想要的不仅是一撇，还想要在求和符号的上方或下方再输入一些其他的东西，这就不容易了——当然，要是没有 `\sideset` 的话，这是非常困难的。但是利用 `\sideset`，您可以输入

$$\begin{array}{l} \backslash\text{sideset}\{\}' \\ \backslash\text{sum}_{\{n < k, \backslash; \backslash\text{text}\{\$n\$为奇数}\}} n E_n \end{array} \quad \sum'_{n < k, n \text{ 为奇数}} n E_n$$

额外的一对空大括号可以通过这样一个事实来解释：`\sideset` 具有在一个较大

的标识符的每个角都放置一个或多个符号的能力：要在乘积符号的每个角都放置一个星号，您可以键入：

`\sideset{_*^*}{_*^*}\prod` \prod_{*}^{*}

7.3 上下标的放置与 *limits* 选项

默认的上标未知决定于主算符。求和类符号默认为 `displaylimits` 放置：当求和类符号出现在行间公式时，上标或下标出现在求和类符号的正上方或正下方，但是在行内公式中，上下标被放置在旁边，来避免周围的文本行的不美观和浪费地伸展。而对带有上下标的积分类符号则默认是放在旁边，即使是在行间公式也一样。（参看第二章中的 `intlimits` 选项）。

诸如 `sin` 或 `lim` 之类的算符名，可能既有 `displaylimits`，又有 `limits` 放置模式，这取决于它们的定义。标准的算符名是根据其数学用途来定义的。

`\limits` 和 `\nolimits` 命令可以用来修改主算符的正常上下标位置：

$$\sum_X, \quad \iint_X, \quad \lim_{n \rightarrow \infty}$$

要定义一个命令使得其上标与 `\sum` 的 `displaylimits` 行为一致，在其定义的末尾加入 `\displaylimits`。当多种 `\limits`，`\nolimits` 或 `\displaylimits` 选项连续出现时，最后一个选项优先。

7.4 多重积分符号

`\iint`，`\iiint`，`\iiiiint` 输出多重积分符号，且给出自适应的间距，在文本和行间模式都是一样。`\idotsint` 是这种积分号的扩展，只给出两个积分号，但在中间用点来填充。

$$(7.2) \quad \iint_A f(x, y) \, dx dy \quad \iiint_A f(x, y, z) \, dx dy dz$$

$$(7.3) \quad \iiidotsint_A f(w, x, y, z) \, dw dx dy dz \quad \int_A \cdots \int f(x_1, \dots, x_k)$$



第 8 章 交换图



一些交换图，比如 $\mathcal{A}\mathcal{M}\mathcal{S}\text{-}\mathcal{T}\mathcal{E}\mathcal{X}$ 中的，是可以用 **amscd** 画的。对于更复杂的交换图，作者应该用更综合性的包，比如 **TikZ**（特别地，**tikz-cd**）或者 **XY-pic**，但是对于简单的不含对角线箭头的交换图，**amscd** 包的命令应该就足够了。这里是一个例子：

$$\begin{array}{ccc} S^{\mathcal{W}_\Lambda} \otimes T & \xrightarrow{j} & T \\ \downarrow & & \downarrow \text{End } P \\ S \otimes T / I & \xlongequal{\quad} & (Z \otimes T) / J \end{array}$$

```
\begin{CD}
S^{\mathcal{W}_\Lambda} \otimes T @>j>> T \\
@VVV @VV\text{End } P\\
S \otimes T / I @= (Z \otimes T) / J
\end{CD}
```

在CD环境中，@>>>，@<<<，@VVV和@AAA分别给出向右，左，下，上的箭头。对于水平箭头，在第一个与第二个>（或<）之间的内容将会以上标的形式输出。而第二个与第三个>（或<）之间的内容会以下标形式输出。类似地，在前两个或后两个A或V之间的内容分别以左标或右标输出。@=和@|分别给出水平和垂直的双竖线。空箭头命令@.可以用来在需要的地方填充一个不可见的箭头。

第 9 章 使用数学字体



9.1 简介

对于 \LaTeX 的更综合性的信息, 参考 \LaTeX 的字体说明 (`fntguide.tex`) 或者 \LaTeX 伴侣 [4]。在 \LaTeX 的数学字体中的基本命令有 `\mathbf`, `\mathrm`, `\mathcal`, `\mathsf`, `\mathtt`, `\mathit`。额外的数学命令, 比如 `\mathbb` 用于黑板字体, `\mathfrak` 用于 Fraktur, `\mathscr` 用于 Euler 手记, 这些可以分别通过加载 `amsfonts` 和 `euscript` 得到。

9.2 推荐使用的数学字体命令

如果您需要在您的文档中频繁使用数学字体命令, 您可能希望它们能有简写名称, 比如 `\mb` 而不是 `\mathbf`。当然, 这并不是难事, 只要您使用适当的 `\newcommand` 命令就可以定义简写了。但是在 \LaTeX 中, 提供短名称实际上对作者是一种伤害, 因为这会模糊一个更好的选择: 定义来自底层数学对象名称的自定义命令名称, 而不是来自用于区分对象的字体名称。例如, 如果使用粗体表示向量。如果您定义一个 `vector` 命令而不是一个 `math-bold` 命令, 那么从长远来看您会得到更好的服务。

```
\newcommand{\vect}[1]{\mathbf{#1}}
```

您可以输入 `\vect{a}+\vert{b}` 来得到 $\mathbf{a} + \mathbf{b}$ 。如果您决定在接下来的几个月内因为其他目的使用粗体, 而向量使用一个小箭头, 您只需要修改 `\vect` 的命令即可, 否则您不得不通过文档来替换所有出现的 `\mathbf`, 甚至需要检查每一个命令看它是否确实是一个向量的例子。

用于为特定字体的不同字母分配不同的命令名也是有用的。

```
\DeclareSymbolFont{AMSb}{U}{msb}{m}{n}% 或者使用amsfonts包  
\DeclareSymbolFont{\C}{\mathalpha}{AMSb}{'43}  
\DeclareSymbolFont{\R}{\mathalpha}{AMSb}{'52}
```

这些声明定义了命令 `\C` 和 `\R`, 是来自 `AMSb` 数学符号字体的黑板字体。如果您在文档中更喜欢实数或复数, 您会发现此方法比定义一个 `\field` 命令, 然后输入 `\field{C}`, `\field{R}` 更方便。但是为了获得最大的灵活性和控制权, 可以定义这样一个 `\field` 命令, 然后根据该命令定义 `\C` 和 `\R`:

```
\usepackage{amsfonts}%获取\mathbb 字母表
\newcommand{\field}[1]{\mathbb{#1}}
\newcommand{\C}{\field{C}}
\newcommand{\R}{\field{R}}
```

9.3 粗体数学符号

`\mathbf` 命令通常用于在数学中输出粗体拉丁字母，但是对于绝大多数其他数学符号都是无效的。或者它的效果依赖于正在使用的一组数学字体。例如，输入

```
\Delta \mathbf{\Delta}\mathbf{+}\delta \mathbf{\delta}
```

将得到 $\Delta\Delta + \delta\delta$ ，`\mathbf` 对加号和小写的 `delta` 无效。因此 `amsmath` 提供了两个额外的命令：`\boldsymbol` 和 `\pmb`，可以用来对其他的数学符号生效。`\boldsymbol` 可以用于 `\mathbf` 不起作用的数学符号当且仅当您当前的数学字体集包括该符号的粗体版本。对于没有由您的数学字体集提供的真正粗体版本的任何数学符号，可以使用 `\pmb` 作为最后的手段：“`pmb`”代表“穷人的粗体”，该命令通过用轻度平板印刷打印符号的多个副本来工作。输出的质量较差，尤其是对于包含细线笔画的符号。当标准的默认 \LaTeX 数学字体集正在使用（`CMR`）时，可能需要 `\pmb` 的唯一符号是大型运算符符号，`\sum`、扩展定界符符号或 `amssymb` 包 [8] 提供的额外数学符号。

下面的公式展示了一些可能的结果：

```
A_{\infty} + \pi A_0
\sim \mathbf{A}_{\boldsymbol{\infty}} \boldsymbol{+}
\boldsymbol{\pi} \mathbf{A}_{\boldsymbol{0}}
\sim \pmb{A}_{\pmb{\infty}} \pmb{+} \pmb{\pi} \pmb{A}_{\pmb{0}}
```

$$A_{\infty} + \pi A_0 \sim \mathbf{A}_{\boldsymbol{\infty}} + \boldsymbol{\pi} \mathbf{A}_{\boldsymbol{0}} \sim \pmb{A}_{\pmb{\infty}} + \pmb{\pi} \pmb{A}_{\pmb{0}}$$

如果您只想使用 `\boldsymbol` 命令而不加载完整的 `amsmath` 包，那么推荐 `bm` 包（这是一个标准的 \LaTeX 包，而不是一个 $\mathcal{A}\mathcal{M}\mathcal{S}$ 包，如果您有 1997 或者更新的 \LaTeX 版本，这个包已经包含在内了。）。

9.4 斜体希腊字母

对于斜体的大写希腊字母，提供了以下命令



<code>\varGamma</code>	Γ	<code>\varSigma</code>	Σ
<code>\varDelta</code>	Δ	<code>\varUpsilon</code>	Υ
<code>\varTheta</code>	Θ	<code>\varPhi</code>	Φ
<code>\varLambda</code>	Λ	<code>\varPsi</code>	Ψ
<code>\varXi</code>	Ξ	<code>\varOmega</code>	Ω
<code>\varPi</code>	Π		



参考文献



- [1] George Grätzer, *More Math into L^AT_EX*, fifth ed., Springer, New York, 2016.
- [2] Donald E. Knuth, *The T_EXbook*, Addison-Wesley, Reading, MA, 1984.
- [3] Leslie Lamport, *L^AT_EX: A document preparation system*, 2nd revised ed., Addison-Wesley, Reading, MA, 1994.
- [4] Frank Mittelbach, Michel Goossens, et al., *The L^AT_EX companion*, second ed., Addison-Wesley, Reading, MA, 2004. This is now also available as an ebook, in both English and German; see [. http://www.latex-project.org/help/books](http://www.latex-project.org/help/books). The front matter, including the full Table of Contents, can be viewed online, from a link on the same page.
- [5] Frank Mittelbach and Rainer Schöpf, *The new font family selection—user interface to standard L^AT_EX*, *TUGboat* **11**, no. 2 (June 1990), pp. 297-305.
- [6] Michael Spivak, *The joy of T_EX*, 2nd revised ed., Amer. Math. Soc., Providence, RI, 1990.
- [7] *AMS author handbook*, separate versions for journal articles, monographs and proceedings articles, Amer. Math. Soc., Providence, RI, 2017; [. https://www.ams.org/publications/authors/tex/author-handbook](https://www.ams.org/publications/authors/tex/author-handbook).
- [8] *AMSF_{onts} version 2.2d—user’s guide*, Amer. Math. Soc., Providence, RI, 2002; distributed with the AMSF_{onts} package [. http://mirror.ctan.org/tex-archive/fonts/amsfonts/doc/amsfndoc.pdf](http://mirror.ctan.org/tex-archive/fonts/amsfonts/doc/amsfndoc.pdf).
- [9] *Using the **amsmath** package*, version 2.20.3, Amer. Math. Soc., Providence, RI, 2017; <http://mirror.ctan.org/tex-archive/macros/latex/required/amscls/doc/amsthdoc.pdf>
- [10] Morten Hogholm, Lars Masden, *The **amsmath** package*, 2018; <http://mirror.ctan.org/tex-archive/macros/latex/contrib/mathtools/mathtools.pdf>.
- [11] The L^AT_EX Project, <https://www.latex-project.org/>.
- [12] nline discussion group comp.text.tex, [https:// groups.google.com/ forum/ #! forum/ comp.text.tex](https://groups.google.com/forum/#!forum/comp.text.tex).
- [13] Online question and answer forum **tex.stackexchange**, <https://tex.stackexchange.com/>.