fixdif 宏包

AlphaZTX

2022/7/19　Version 1.3a*

摘　要

fixdif 宏包重定义了 LaTeX 中的 \d 命令，使之在数学模式下可以得到微分算符 d，在正文中仍然是原来的重音符号（例如 \d{o} 得到 o̩）。

本宏包支持不同的编译方式，在 pdfTeX、XeTeX 和 LuaTeX 均可使用。本宏包在 XeTeX 和 LuaTeX 下兼容 unicode-math 宏包。

目　录

1　背景 2

2　宏包简介 3

2.1　对 unicode-math 宏包的兼容 3

2.2　对 hyperref 宏包的兼容 4

2.3　基本命令及宏包选项 4

3　通过 fixdif 定义微分算符命令 5

3.1　通过命令名称定义微分算符命令 5

3.2　定义包含多个命令或字符的微分算符命令 6

4　正文中临时的微分算符命令 7

5　举例 7

*https://github.com/AlphaZTX/fixdif

1 背景

我们一般更推荐在微分算符与前面的内容之间留出一个比较小的间距[1]，比如在下面展示的两种情况中，右边的比左边的更好：

$$f(x)\mathrm{d}x \qquad \& \qquad f(x)\,\mathrm{d}x.$$

微分算符 d 与其前面的表达式之间的间距可认为是乘积的一种表示。

有些用户习惯使用下面的方法定义一个宏：

```
\renewcommand\d{\mathop{\mathrm{d}}\!}
```

这个命令在很多情况下不会产生什么大问题，但是在下面给出的 3 个使用场景下就会出现不容忽略的问题：

1. 当 d 出现在用斜杠表示的分式的分母中时，d 前面会留出多余的间距。例如 `$\d y/\d x$` 得到 $\d y/\d x$；
2. 上面定义的 `\d` 会覆盖原来的 `\d` 命令。在 LaTeX 的原始定义下，`\d` 是一个重音符号命令，用来产生字母下方的点。如果使用了上面的定义，则在正文中 `\d{o}` 将不会得到“o̩”而是得到一条报错。
3. 前面说到，d 前面的间距可以认为是乘积的一种表示，因此这个间距可被视为一个二元运算符。在上下标中时，二元运算符在两侧的间距应当消失，例如 $a+b$ 放到上标中就会变成 $^{a+b}$。但是通过上面的方法定义的 `\d` 在上下标中，其前面的间距不会消失，例如 `$a^{f(x)\d x}$` 会得到 $a^{f(x)\,\mathrm{d}x}$ 而不是 $a^{f(x)\mathrm{d}x}$。

面对以上问题，解决方案是使用本宏包。

[1] 见 TeX.SE。

2 宏包简介

在导言区添加

```
\usepackage{fixdif}
```

以使用本宏包。使用了本宏包后，在正文中输入

```
\[ f(x)\d x,          \quad    \frac{\d y}{\d x},
   \quad\d y/\d x,    \quad    a^{y\d x}.          \]
```

会得到

$$f(x)\,\mathrm{d}x,\quad \frac{\mathrm{d}y}{\mathrm{d}x},\quad \mathrm{d}y/\mathrm{d}x,\quad a^{y\mathrm{d}x}.$$

2.1 对 unicode-math 宏包的兼容

如果你在文档中使用了 unicode-math 宏包（编译方式为 XeTeX 或 LuaTeX），需注意以下几点：

- 如需使用 amsmath 宏包，请在 unicode-math 之前载入 amsmath。
- 在载入 unicode-math 后通过 `\setmathfont` 指定数学字体。若使用默认的数学字体，建议显式指明 `\setmathfont{Latin Modern Math}`，这样可以避免在用斜杠表示的分式中产生不好的间距；
- fixdif 宏包需要在 unicode-math 宏包后载入。

从而，引入宏包的正确顺序是：

```
\usepackage{amsmath}
\usepackage{unicode-math}
\setmathfont{...}[...]
\usepackage{fixdif}
```

2.2 对 hyperref 宏包的兼容

如果你使用了 hyperref 宏包，需注意 fixdif 宏包需要在 hyperref 之后载入，否则会引起冲突。

2.3 基本命令及宏包选项

`\d` fixdif 宏包提供了 `\d` 命令用于在数学模式中得到微分算符“d”。在正文中，`\d` 会保留其原有的重音符号的功能。例如：

```
$\d x$ 和 \d x
```

以上代码会得到“$\mathrm{d}x$ 和 $\dot{x}$”。

设置 \d 的字体 本宏包提供了两个选项来控制 `\d` 的字体，分别为 `rm` 和 `normal`。其中，`rm` 为默认值，表示 `\d` 的字体为罗马体（`\mathrm`）；`normal` 表示 `\d` 的字体为数学模式下的常规体（`\mathnormal`），效果类似于意大利体。例如在载入宏包时开启 `normal` 选项：

```
\usepackage[normal]{fixdif} % 导言区
```

正文中的 `$f(x)\d x$` 会得到 $f(x)\,dx$。

`\resetdfont` 除了上面的两种字体，fixdif 也支持通过在导言区使用 `\resetdfont` 命令自定义 `\d` 的字体。例如：

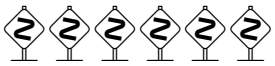
```
\resetdfont{\mathsf}        % 导言区
```

在此设置下，正文中的 `\d x` 会得到 $\mathsf{d}x$。

`\partial` **`\partial` 的行为** fixdif 宏包将 `\partial` 归为微分算符，正文中的 `\partial` 前面也会有自动的间距。例如 `$\partial x\partial y$` 会得到 $\partial x\,\partial y$。若不需要将 `\partial` 视为微分算符，可以开启 `nopartial` 选项：

```
\usepackage[nopartial]{fixdif}
```

3 通过 fixdif 定义微分算符命令



注意！本节中的命令只能在导言区中使用！

3.1 通过命令名称定义微分算符命令

\letdif `\letdif{⟨cmd⟩}{⟨csname⟩}` （仅限在导言区使用）

`\letdif` 命令有两个参数：第一个参数 ⟨*cmd*⟩ 是新定义的微分算符命令，以反斜杠开头；第二个参数 ⟨*csname*⟩ 是需要设置为微分算符的数学符号的命令名称，也就是一个完整的命令去掉最前面的反斜杠。举个例子：

```
\letdif{\vr}{delta}
```

这样定义的 `\vr` 在数学模式中就相当于一个 `\delta`，但是它的前面会加上自动的间距。

通过 `\letdif` 命令，我们可以原封不动地把一个数学符号变成微分算符。例如：

```
\letdif{\delta}{delta}
```

这样 `\delta` 本身就会具备微分算符的属性。

`\letdif` 的第二个参数 ⟨*csname*⟩ 可以重复使用，也就是说，你可以把同一个符号定义为两个不同的微分算符命令。例如：

```
\letdif{\nabla}{nabla}
\letdif{\grad}{nabla}
```

这样 `\nabla` 和 `\grad` 都会得到微分算符 ∇。

\letdif* `\letdif*{⟨cmd⟩}{⟨csname⟩}` （仅限在导言区使用）

`\letdif*` 与 `\letdif` 基本相同，唯一的区别在于 `\letdif*` 会在微分算符后面添加额外的矫正。建议在使用了 unicode-math 的文档中使用 `\letdif*`。例如：

```
\usepackage{unicode-math}
\setmathfont{TeX Gyre Termes Math}
\usepackage{fixdif}
\letdif{\vr}{updelta}
```

这样定义的 \vr 后面就会有负的间距，如果不希望得到这一负间距，可以把上面的最后一行改为：

```
\letdif*{\vr}{updelta}
```

3.2 定义包含多个命令或字符的微分算符命令

\newdif \newdif{⟨*cmd*⟩}{⟨*multi-cmd*⟩} （仅限在导言区使用，后无矫正）
\newdif* \newdif*{⟨*cmd*⟩}{⟨*multi-cmd*⟩} （仅限在导言区使用，后有矫正）

上面两个命令的第一个参数 ⟨*cmd*⟩ 是新定义的微分算符命令，第二个参数 ⟨*multi-cmd*⟩ 是超过一个字（token）的命令组合或字符串。举个例子，在 xcolor 宏包下可以这样使用：

```
\newdif{\redsfd}{\textsf{\color{red}d}}
```

这样就定义了一个 \redsfd 命令。再举一个例子：

```
\newdif{\D}{\mathrm{D}}
```

这样，在正文中输入 $y\D x$ 就会得到 $y\D x$。

如果 \newdif(*) 后面的第二个参数 ⟨*multi-cmd*⟩ 仅包含一个字（token），也就是一个命令或字符，请使用 \letdif(*) 命令而不是 \newdif(*)。

\newdif(*) 会检查第一个参数 ⟨*cmd*⟩ 是否已经被定义，若 ⟨*cmd*⟩ 已被定义，则会报错。

\renewdif \renewdif{⟨*cmd*⟩}{⟨*multi-cmd*⟩} （仅限在导言区使用，后无矫正）
\renewdif* \renewdif*{⟨*cmd*⟩}{⟨*multi-cmd*⟩} （仅限在导言区使用，后有矫正）

这两个命令用于将 ⟨*cmd*⟩ 重定义为微分算符命令。若 ⟨*cmd*⟩ 还未被定义，则会报错。

4 正文中临时的微分算符命令

\mathdif `\mathdif{⟨symbol⟩}` （仅限在数学模式使用，后无矫正）

\mathdif* `\mathdif*{⟨symbol⟩}` （仅限在数学模式使用，后有矫正）

如需在正文中少量使用某一数学符号作为微分算符，可以使用上面的两个命令。例如，`$x\mathdif{\Delta}\psi$` 会得到 $x\,\Delta\psi$。

5 举例

本节将告诉你如何在文档中正确使用 fixdif 宏包。

看以下两行代码：

```
\letdif{\Delta}{Delta}
\letdif{\nabla}{nabla}
```

哪一行更合理一些？答案是第二行。

有时，大写的“Δ”会被用作拉普拉斯算符（与 ∇^2 意义相同）；另一些时候，大写的“Δ”会被当作变量或函数。所以对 Δ 作为拉普拉斯算符和变量或函数的情况分别进行定义是更好的做法。我们建议将 `$\Delta$` 作为 Δ 的原始定义进行保留，对表示拉普拉斯算符的 Δ 另加定义。从而我们这样修改上面的第一行：

```
\letdif{\laplacian}{Delta}
```

如果你比较擅长使用 xparse 宏包的接口，你也可以采用下面的方法：

```
\letdif{\nabla}{nabla}
\DeclareDocumentCommand{ \laplacian }{ s }{
  \IfBooleanTF { #1 } { \mathdif{\Delta} } { \nabla^2 }
}
```

这样，`\laplacian` 会得到 ∇^2，`\laplacian*` 会得到 Δ。

处理正负号与微分算符之间的间距 输入 `$-\d x$` 会得到 $-\mathrm{d}x$。如果你不希望在正负号与微分算符之间保留一定的间距，可以输入 `$-{\d x}$`，这样就会得到 $-\mathrm{d}x$。不过话说回来，$-\mathrm{d}x$ 的效果也并非不可接受。