The fixdif Package

Zhang Tingxuan

2022/7/19　Version 1.3a*

简介

fixDif 宏包在 LaTeX 中重定义了\d命令，并提供来定义微分算子命令的接口。

本宏包不仅可用 pdfTeX, XeTeX, LuaTeX编译，还兼容 XeTeX and LuaTeX下的 unicode-math 宏包。

目录

第 1 节　背景

为求美观，我们通常会在微分算子和它前面的表达式之间保留一定的空白[1]. 比如以下情况：

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

我们通常会认为左边比右边好看，在 $f(x)$ 和 $\mathrm{d}x$ 之间的小空白可以视为 $f(x)$ 和 $\mathrm{d}x$ 的乘积符号。

因此，有些用户会喜欢定义这样的命令:

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

虽然这个命令在“行间公式”和“行内公式”都很有效，但是依然存在以下三个问题:

1. d 前面的空白在行内分式中依旧出现. 比如, `$\d y/\d x$` 会呈现为 $\mathrm{d}y/\mathrm{d}x$;
2. \d 不能用于数学模式以外的地方. 即 \d{o} 不能用于在文本模式下产生类似 “o̩” 的效果;
3. d 和它前面表达式之间的空白被视作乘积符号，而一个算符通常应该是二元的。拿 \cdot (·) 来举例. 当在“行间公式”或“行内公式”下 $x\cdot y$ 间的算符会保留，但在“角标公式”或“脚本脚本公式”则该消失，例如 $a^{x\cdot y}$。因此，该空白也应该在角标下消失，但是 `$a^{f(x)\d x}$` 出来的 $a^{f(x)\,\mathrm{d}x}$ 依旧会存在空白，而不是期望中的 $a^{f(x)\mathrm{d}x}$。

如果想解决以上问题，你可以试试本宏包。

第 2 节　引言

在导言区使用以下命令即可加载本宏包

```
\usepackage{fixdif}
```

在文档区使用以下命令

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

将会出现

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

2.1 兼容 unicode-math

如果你已经在文档里使用 XeTeX/LuaTeX 下的 unicode-math 宏包，那你得注意下面的问题:

- 如果要使用 amsmath 宏包, 请确保 unicode-math 在 amsmath 之后 被加载。
- 最好使用 unicode-math 提供的 \setmathfont 命令指定数学字体以避免在行内分式情况下出现多余空白的问题, 如 $\mathrm{d}y/\mathrm{d}x$.
- fixdif 宏包一定要在 unicode-math 之后 加载。

因此，正确的顺序应该是

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

2.2 兼容 hyperref

If you want to use the hyperref package simultaneously, remember to load hyperref *before* the fixdif package, otherwise the hyperref package will cause conflicts.

2.3 基础命令以及宏包参数

\d The fixdif package provides a \d command for the differential operator "d" in math mode. When in text, \d behaves just like the old \d command in LaTeX or plain TeX as an accent command. For example,

```
$\d x$ and \d x
```

will produce "$\mathrm{d}x$ and x̣".

2.3.0.1 Set the font of \d

There are two basic package options to control the \d's style in math mode — rm and normal. The default option is rm, in which case `$f(x)\d x$` produces $f(x)\,\mathrm{d}x$. If you chose the normal option, for example

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

`$f(x)\d x$` would produces $f(x)\,dx$.

\resetdfont Besides the previous two optional fonts, you can reset the font of differential operator "d" through \resetdfont command in preamble:

```
\resetdfont{\mathsf}
```

then `\d x` will produce $\mathsf{d}x$.

2.3.0.2 Control the behavior of \partial

\partial In default, \partial will also be regarded as a differential operator in this package. If you don't like this default setting, you can use the nopartial option:

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

第 3 节　微分算子的定义

Attention! The commands in this section can be used in preamble only!

3.1 单命令定义

\letdif `\letdif{⟨cmd⟩}{⟨csname⟩}` (preamble only)

The \letdif command has two arguments — the first is the newly-defined command and the second is the control sequence *name* of a math character, that is, a command without its backslash. For example,

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

[1]See https://tex.stackexchange.com/questions/14821/whats-the-proper-way-to-typeset-a-differential-operator.

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

then \vr will produce a δ (\delta) with automatic skip before it.

Through the \letdif command, we can redefine a math character command by its name. For example,

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

then \delta itself will be a differential operator.

The second argument ⟨*csname*⟩ of \letdif command can be used repeatedly.

\letdif* \letdif*{⟨*cmd*⟩}{⟨*csname*⟩}　(preamble only)

This command is basically the same as \letdif, but this command will patch a correction after the differential operator. This is very useful when a math font is setted through unicode-math package. For example,

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

this will cause bad negative skip after \vr, but if you change the last line into

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

you will get the result correct.

3.2　多命令或字符串定义

\newdif \newdif{⟨*cmd*⟩}{⟨*multi-cmd*⟩}　(without correction, preamble only)
\newdif* \newdif*{⟨*cmd*⟩}{⟨*multi-cmd*⟩}　(with correction, preamble only)

The first argument of these commands is the newly-defined command; and the second argument should contain *more than one* tokens. For example, if you have loaded the xcolor package, you can use the following line:

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

Then you get the \redsfd as a differential operator. Take another example,

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

Then you get \D for an uppercase upright "D" as a differential operator.

If your second argument contains only one command like \Delta, it's recommended to use \letdif or \letdif* instead.

\newdif and \newdif* will check whether ⟨*cmd*⟩ has been defined already. If so, an error message will be given.

\renewdif \renewdif{⟨*cmd*⟩}{⟨*multi-cmd*⟩}　(without correction, preamble only)
\renewdif* \renewdif*{⟨*cmd*⟩}{⟨*multi-cmd*⟩}　(with correction, preamble only)

These two commands are basically the same as \newdif and \newdif*. The only difference is that \renewdif and \renewdif* will check whether ⟨*cmd*⟩ has *not* been defined yet. If so, an error message will be given.

第 4 节　暂时使用微分算子

\mathdif \mathdif{⟨*symbol*⟩}　(without correction, in math mode only)
\mathdif* \mathdif*{⟨*symbol*⟩}　(with correction, in math mode only)

These two commands can be used in math mode only, more specifically, after \begin{document}. For example, $x\mathdif{\Delta}\psi$ will get $x\,\Delta\psi$.

第 5 节　参考示例

This section shows how to use this package properly in your document.

Take the two examples below:

```
\letdif{\Delta}{Delta}       % Example 1, in preamble
\letdif{\nabla}{nabla}       % Example 2, in preamble
```

Actually, the second example is more reasonable. Sometimes, we take "Δ" as laplacian (equivalent to ∇^2), while "Δ" can also be regarded as a variable or function at some other times. Consequently, it's better to save a different command for "Δ" as laplacian while reserve `\Delta` as a command for an ordinary math symbol "Δ". However, in the vast majority of cases, "∇" is regarded as nabla operator so there is no need to save a different command for "∇". Then we can correct the code above:

```
\letdif{\laplacian}{Delta}   % Example 1, corrected, in preamble
```

With the xparse package, we can define the command in another method:

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

Then `\laplacian` produces ∇^2 and `\laplacian*` produces Δ.

5.0.0.1 Dealing with "+" and "−"

If you input `$-\d x$`, you'll get "$-\,\mathrm{d}x$" in your document. However, if you think "$-\mathrm{d}x$" is better, you can input `-{\d x}`. The "`\d x`" in a *group* will be regarded *ordinary* but not *inner* so that the small skip will disappear. Maybe "$-\,\mathrm{d}x$" is just okay.

第 6 节　源代码

```
1 ⟨*package⟩
```

Check the TEX format and provides the package name.

```
2 \NeedsTeXFormat{LaTeX2e}
3 \ProvidesPackage{fixdif}[2022/7/19 Interface for defining differential operators.]
```

6.1 Control the skip between slashes and differential operator

Change the math code of slash (/) and backslash (\) so that the skip between slashes and differential operators can be ignored.

```
4 \@ifpackageloaded{unicode-math}{
```

If the unicode-math package has been loaded, use the XƎTEX/LuaTEX primitive `\Umathcode` to change the type of slashes. The numeral "4" stands for "open".

```
5   \Umathcode`\/="4 "0 "002F
6   \Umathcode"2044="4 "0 "2044
7   \Umathcode"2215="4 "0 "2215
8   \Umathcode"2F98="4 "0 "2F98
9   \Umathcode`\\="4 "0 "005C
10  \Umathcode"2216="4 "0 "2216
11  \Umathcode"29F5="4 "0 "29F5
12  \Umathcode"29F9="4 "0 "29F9
13 }{
```

If the unicode-math package has not been loaded, use the TEX primitive `\mathcode` to change the type of slashes. The `\backslash` needs to be redefined through `\delimiter` primitive too.

```
14  \mathcode`\/="413D
15  \mathcode`\\="426E % \backslash
16  \def\backslash{\delimiter"426E30F\relax}
17 }
```

6.2 Patch the skips around the differential operator

\mup@tch The following \mup@tch patches the skip after the differential operator.

```
\def\mup@tch{\mathchoice{\mskip-\thinmuskip}{\mskip-\thinmuskip}{}{}}
```

The \s@beforep@tch patches the commands with star (\letdif*, etc).

```
\def\s@beforep@tch{\mathchoice{}{}{\mbox{}}{\mbox{}}}
```

6.3 Declare the package options

Declare the options of the package and execute them.

```
\DeclareOption{rm}{\@ifpackageloaded{unicode-math}
  {\def\@@dif{\symrm{d}}}{\def\@@dif{\mathrm{d}}}}
\DeclareOption{normal}{\def\@@dif{d}}
\DeclareOption{partial}{\def\fixdif@partial@bool{1}}
\DeclareOption{nopartial}{\def\fixdif@partial@bool{0}}
\ExecuteOptions{rm,partial}
\ProcessOptions\relax
```

Control the behavior of \partial.

```
\def\fixdif@partial@true{1}
\ifx\fixdif@partial@bool\fixdif@partial@true
  \AtEndOfPackage{\letdif{\partial}{partial}}
\fi
```

\resetdfont Define the \resetdfont command.

```
\gdef\resetdfont#1{\let\@@dif\relax%
  \def\@@dif{#1{d}}}
```

6.4 Deal with the \d command

\@dif \@dif is the differential operator produced by \d in math mode. Here we prefer \mathinner to \mathbin to make the skip.

```
\def\@dif{\mathinner{\@@dif}\mup@tch}
```

\d@accent Restore the \d command in text by \d@accent with the \let primitive.

```
\let\d@accent\d
```

\d Redefine the \d command. In text, we need to expand the stuffs after \d

```
\DeclareRobustCommand\d{\ifmmode\@dif\else\expandafter\d@accent\fi}
```

6.5 User's interface for defining new differential operators

\letdif \letdif* Define the \letdif and \letdif* command. The internal version of \letdif is \@letdif, of \letdif* is \s@letdif.

```
\def\@letdif#1#2{\AtBeginDocument{%
```

#1 is the final command; #2 is the "control sequence name" of #1's initial definition. Here we create a command (\csname#2@old\endcsname) to restore #2.

```
  \ifcsname #2@old\endcsname\else%
  \expandafter\let\csname #2@old\expandafter\endcsname
    \csname #2\endcsname%
  \fi%
```

Finally let #1 be the new command.

```
  \gdef#1{\mathinner{\csname #2@old\endcsname}\mup@tch}%
}}
```

The definition of \s@letdif is similar, but with the patch for negative skips.

```
\def\s@letdif#1#2{\AtBeginDocument{%
  \ifcsname #2@old\endcsname\else%
  \expandafter\let\csname #2@old\expandafter\endcsname
    \csname #2\endcsname%
  \fi%
  \gdef#1{\mathinner{\s@beforep@tch\csname #2@old\endcsname\mbox{}}\mup@tch}%
}}
\def\letdif{\@ifstar\s@letdif\@letdif}
```

\newdif
\newdif*

Define the \newdif and \newdif* commands. #1 is the final command; #2 is the "long" argument.

```
\long\def\@newdif#1#2{\AtBeginDocument{%
  \ifdefined#1
    \PackageError{fixdif}{\string#1 is already defined.}
      {Try another command instead of \string#1.}%
  \else
    \long\gdef#1{\mathinner{#2}\mup@tch}%
  \fi%
}}
\long\def\s@newdif#1#2{\AtBeginDocument{%
  \ifdefined#1
  \PackageError{fixdif}{\string#1 is already defined.}
    {Try another command instead of \string#1.}%
  \else
    \long\gdef#1{\s@beforep@tch\mathinner{#2\mbox{}}\mup@tch}%
  \fi%
}}
\def\newdif{\@ifstar\s@newdif\@newdif}
```

\renewdif
\renewdif*

Define the \renewdif and \renewdif* commands.

```
\long\def\@renewdif#1#2{\AtBeginDocument{%
  \ifdefined#1
    \long\gdef#1{\mathinner{#2}\mup@tch}%
  \else
    \PackageError{fixdif}{\string#1 has not been defined yet.}
      {You should use \string\newdif instead of \string\renewdif.}%
  \fi%
}}
\long\def\s@renewdif#1#2{\AtBeginDocument{%
  \ifdefined#1
    \long\gdef#1{\s@beforep@tch\mathinner{#2\mbox{}}\mup@tch}%
  \else
    \PackageError{fixdif}{\string#1 has not been defined yet.}
      {You should use \string\newdif instead of \string\renewdif.}%
  \fi%
}}
\def\renewdif{\@ifstar\s@renewdif\@renewdif}
```

6.6 In-document commands: \mathdif and \mathdif*

```
\def\@mathdif#1{\mathinner{#1}\mup@tch}
\def\s@mathdif#1{\s@beforep@tch\mathinner{#1\mbox{}}\mup@tch}
\DeclareRobustCommand\mathdif{\@ifstar\s@mathdif\@mathdif}
```

End of the package.

```
⟨/package⟩
```