

The `fixdif` Package

Zhang Tingxuan

2022/7/27 Version 1.3b*

Abstract

The `fixdif` package redefines the `\d` command in \LaTeX and provides an interface to define commands for differential operators.

The package is compatible with `pdf \TeX` , `X \TeX` and `Lua \TeX` . Furthermore, the package is compatible with `unicode-math` package in `X \TeX` and `Lua \TeX` .

Contents

1	The background	1
2	Introduction	2
2.1	When using <code>unicode-math</code>	2
2.2	When using <code>hyperref</code>	2
2.3	Basic commands and package options	3
3	Define commands for differential operators	3
3.1	Define commands with a single command name	3
3.2	Define commands with multi commands or a string	4
4	Using differential operators temporarily	5
5	Examples	5

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

6 The source code	5
6.1 Control the skip between slashes and differential operator	6
6.2 Patch the skips around the differential operator	6
6.3 Declare the package options	6
6.4 Deal with the <code>\d</code> command	7
6.5 User’s interface for defining new differential operators	7
6.6 In-document commands: <code>\mathdif</code> and <code>\mathdif*</code>	8

1 The background

It’s usually recommended that one should reserve a small skip between the differential operator and the expression before it¹. Take the following cases as an example:

$$f(x)dx \quad \text{and} \quad f(x) \, dx.$$

We usually consider that the example on the right side is better than the one on the left side. The little skip between $f(x)$ and dx can be regarded as a symbol of the product of $f(x)$ and dx .

So some users prefer to define a macro like this:

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

This macro works well in “display math” and “text math”, but we still face the following three problems:

1. The skip before “d” would still be reserved in “text fraction”, which is regarded bad. For example, `\d y/\d x` produces dy/dx ;
2. This `\d` command cannot be used out of math mode. In another word, `\d{o}` would not produce “o” in text;
3. The skip between “d” and the expression before it can be regarded as a product operator. A product operator is definitely a binary operator.

Take `\cdot` (\cdot) as an example. A binary operator reserves small skips before and after itself when in “display math” or “text math” such as $x \cdot y$, but the skips will disappear in “script math” or “script script math” such as $a^{x \cdot y}$. Thus the small skip should also disappear in script, but `\a^{f(x)\d x}` still produces $a^{f(x) \, dx}$ but not $a^{f(x)dx}$.

To solve these problems, you can try this package.

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

2 Introduction

To load this package, write

```
\usepackage{fixdif}
```

in the preamble. In your document,

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

will produce

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

2.1 When using unicode-math

If you are using `unicode-math` package with XeTeX/LuaTeX in your document, you must pay attention to the following items:

- If you want to use `amsmath` package, make sure that the `unicode-math` package is loaded *after* `amsmath`.
- You had better specify the math font through the `\setmathfont` command provided by `unicode-math` in order to avoid bad skip in text fraction like $\mathrm{d}y/\mathrm{d}x$.
- Load the `fixdif` package *after* `unicode-math`.

Therefore the correct order is

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

2.2 When using 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 Basic commands and package options

`\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 L^AT_EX or plain T_EX as an accent command. For example,

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

will produce “ $\mathrm{d}x$ and \dot{x} ”.

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 $f(x)dx$. 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 dx .

`\partial` Control the behavior of `\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}
```

3 Define commands for differential operators

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

3.1 Define commands with a single command name

`\letdif` `\letdif{<cmd>}{<cname>}` (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,

```
\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 `<cname>` of `\letdif` command can be used repeatedly.

`\letdif*` `\letdif*{<cmd>}{<cname>}` (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 Define commands with multi commands or a string

```

\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{\redsf}{\textsf{\color{red}d}}
```

Then you get the `\redsf` 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.

4 Using differential operators temporarily

```

\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 $x\Delta\psi$.

5 Examples

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 .

Dealing with “+” and “−” If you input `-$\d x$`, you’ll get “ $-dx$ ” in your document. However, if you think “ $-dx$ ” 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 “ $-dx$ ” is just okay.

6 The source code

```
1 <*package>
```

Check the \TeX format and provides the package name.

```
2 \NeedsTeXFormat{LaTeX2e}
3 \ProvidesPackage{fixdif}[2022/7/27 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 \XeTeX/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.

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

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

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

6.3 Declare the package options

Declare the options of the package and execute them.

```

20 \DeclareOption{rm}{\@ifpackageloaded{unicode-math}
21   {\def\@@dif{\symrm{d}}}{\def\@@dif{\mathrm{d}}}}
22 \DeclareOption{normal}{\def\@@dif{d}}
23 \DeclareOption{partial}{\def\fixdif@partial@bool{1}}
24 \DeclareOption{nopartial}{\def\fixdif@partial@bool{0}}
25 \ExecuteOptions{rm,partial}
26 \ProcessOptions\relax

```

Control the behavior of `\partial`.

```

27 \def\fixdif@partial@true{1}
28 \ifx\fixdif@partial@bool\fixdif@partial@true
29   \AtEndOfPackage{\letdif\partial\partial}
30 \fi

```

`\resetdfont` Define the `\resetdfont` command.

```

31 \gdef\resetdfont#1{\let\@@dif\relax%
32   \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.

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

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

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

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

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

6.5 User's interface for defining new differential operators

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

```
36 \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`.

```
37 \ifcsname #2@old\endcsname\else%
38 \expandafter\let\csname #2@old\expandafter\endcsname
39 \csname #2\endcsname%
40 \fi%
```

Finally let `#1` be the new command.

```
41 \gdef#1{\mathinner{\csname #2@old\endcsname}\mup@tch}%
42 }}
```

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

```
43 \def\s@letdif#1#2{\AtBeginDocument{%
44 \ifcsname #2@old\endcsname\else%
45 \expandafter\let\csname #2@old\expandafter\endcsname
46 \csname #2\endcsname%
47 \fi%
48 \gdef#1{\mathinner{\s@beforep@tch\csname #2@old\endcsname\mbox{}}}\mup@tch}%
49 }}
50 \def\letdif{\@ifstar\s@letdif\@letdif}
```

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

```
51 \long\def\@newdif#1#2{\AtBeginDocument{%
52 \ifdefined#1
53 \PackageError{fixdif}{\string#1 is already defined.}
54 {Try another command instead of \string#1.}%
55 \else
```

```

56   \long\gdef#1{\mathinner{#2}\mup@tch}%
57   \fi%
58 }}
59 \long\def\s@newdif#1#2{\AtBeginDocument{%
60   \ifdefined#1
61     \PackageError{fixdif}{\string#1 is already defined.}
62     {Try another command instead of \string#1.}%
63   \else
64     \long\gdef#1{\s@beforep@tch\mathinner{#2\mbox{}}}\mup@tch}%
65   \fi%
66 }}
67 \def\newdif{\@ifstar\s@newdif\@newdif}

```

`\renewdif` Define the `\renewdif` and `\renewdif*` commands.

```

\renewdif*
68 \long\def\@renewdif#1#2{\AtBeginDocument{%
69   \ifdefined#1
70     \long\gdef#1{\mathinner{#2}\mup@tch}%
71   \else
72     \PackageError{fixdif}{\string#1 has not been defined yet.}
73     {You should use \string\newdif instead of \string\renewdif.}%
74   \fi%
75 }}
76 \long\def\s@renewdif#1#2{\AtBeginDocument{%
77   \ifdefined#1
78     \long\gdef#1{\s@beforep@tch\mathinner{#2\mbox{}}}\mup@tch}%
79   \else
80     \PackageError{fixdif}{\string#1 has not been defined yet.}
81     {You should use \string\newdif instead of \string\renewdif.}%
82   \fi%
83 }}
84 \def\renewdif{\@ifstar\s@renewdif\@renewdif}

```

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

```

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

```

End of the package.

```
88 \end{package}
```