

# The `fixdif` Package

Zhang Tingxuan

2023/02/01 Version 2.0a\*

## Abstract

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

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

## Contents

<b>1</b>	<b>The background</b>	<b>1</b>
<b>2</b>	<b>Introduction</b>	<b>2</b>
2.1	Basic commands and package options . . . . .	2
<b>3</b>	<b>Define commands for differential operators</b>	<b>3</b>
3.1	Define commands with a single command name . . . . .	3
3.2	Define commands with multi commands or a string . . . . .	4
<b>4</b>	<b>Using differential operators temporarily</b>	<b>4</b>
<b>5</b>	<b>Examples</b>	<b>4</b>
<b>6</b>	<b>The source code</b>	<b>5</b>
6.1	Control the skip between slashes and differential operator . . . . .	5
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 . . . . .	6
6.5	User's interface for defining new differential operators . . . . .	7
6.6	In-document commands: <code>\mathdif</code> . . . . .	8

---

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

## 1 The background

It's usually recommended that one should reserve a small skip between the differential operator and the expression before it<sup>1</sup>. 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\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,  $\mathop{d} y/\mathop{x}$  produces  $dy/dx$ ;
2. This  $\mathop{d}$  command cannot be used out of math mode. In other words,  $\mathop{o}$  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  $\mathop{\cdot}$  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  $\mathop{a}^{\mathop{f(x)} \, dx}$  still produces  $a^{f(x) \, dx}$  but not  $a^{f(x)dx}$ .

To solve these problems, you can try this package.

## 2 Introduction

To load this package, write

```
\usepackage{fixdif}
```

in the preamble (fixdif allows you to write this line anywhere in the preamble since version 2.0). In your document,

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

will produce

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

---

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

## 2.1 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<sup>A</sup>T<sub>E</sub>X or plain T<sub>E</sub>X as an accent command. For example,

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

will produce “ $dx$  and  $\grave{x}$ ”.

**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 produce  $f(x)dx$ .

`\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}
```

If you use the default setting, `\partialnondif` yields the ordinary symbol  $\partial$ .

## 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  $\delta$  (`\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  $\langle csname \rangle$  of `\letdif` command can be used repeatedly. If you want to get the ordinary symbol of  $\langle csname \rangle$ , you can input `\partialnondif \langle csname \ranglenondif` in math mode. For example, in default, `\partialnondif` yields the old partial symbol “ $\partial$ ”.

```
\letdif*{\langle cmd \rangle}{\langle csname \rangle} (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{\langle cmd \rangle}{\langle multi-cmd \rangle} (without correction, preamble only)
\newdif*{\langle cmd \rangle}{\langle multi-cmd \rangle} (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  $\langle cmd \rangle$  has been defined already. If so, an error message will be given.

```
\renewdif \renewdif{\langle cmd \rangle}{\langle multi-cmd \rangle} (without correction, preamble only)
\renewdif*{\langle cmd \rangle}{\langle multi-cmd \rangle} (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  $\langle cmd \rangle$  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*{<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 “ $\Delta$ ” as laplacian (equivalent to  $\nabla^2$ ), while “ $\Delta$ ” can also be regarded as a variable or function at some other times. Consequently, it’s better to save a different command for “ $\Delta$ ” as laplacian while reserve `\Delta` as a command for an ordinary math symbol “ $\Delta$ ”. However, in the vast majority of cases, “ $\nabla$ ” is regarded as nabla operator so there is no need to save a different command for “ $\nabla$ ”. 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  $\nabla^2$  and `\laplacian*` produces  $\Delta$ .

**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  $\text{\TeX}$  format and provides the package name.

```
2 \NeedsTeXFormat{LaTeX2e}
3 \ProvidesPackage{fixdif}[2023/02/01 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.

If the `unicode-math` package was loaded, use the  $X_{\text{TeX}}/\text{LuaTeX}$  primitive `\Umathcode` to change the type of slashes. The numeral “4” stands for “open”. If `unicode-math` was not loaded but `fontspec` loaded, check if `fontspec` had reset math fonts, that is to say, the `no-math` option.

```
4 \AtBeginDocument{%
5 \ifcsname symbf\endcsname%
6   \csname bool_if:cF\endcsname{g__um_main_font_defined_bool}%
7   {\csname __um_load_lm:\endcsname}%
8   \Umathcode`\/="4 \symoperators "002F%
9   \Umathcode"2044="4 \symoperators "2044%
10  \Umathcode"2215="4 \symoperators "2215%
11  \Umathcode"2F98="4 \symoperators "2F98%
12  \Umathcode`\\="4 \symoperators "005C%
13  \Umathcode"2216="4 \symoperators "2216%
14  \Umathcode"29F5="4 \symoperators "29F5%
15  \Umathcode"29F9="4 \symoperators "29F9%
16 \else\ifcsname fontspec\endcsname
17   \csname bool_if:cT\endcsname{g__fontspec_math_bool}%
18   {%
19     \everymath{\mathcode`\/="413D\relax}%
20     \PackageWarning{fixdif}{Requires `no-math' option of fontspec!\MessageBreak}%
21     }% fontspec only influences "/"
22 \fi\fi}
```

Use `\mathcode` to change the type of slashes. The `\backslash` needs to be redefined through `\delimiter` too.

```
23 \mathcode`\/="413D
24 \mathcode`\\="426E% \backslash
25 \protected\def\backslash{\delimiter"426E30F\relax}
```

## 6.2 Patch the skips around the differential operator

`\fd@mu@p` The following `\fd@mu@p` patches the skip after the differential operator.

```
26 \def\fd@mu@p{\mathchoice{\mskip-\thinmuskip}{\mskip-\thinmuskip}{-}{-}}
```

The `\s@fd@mu@p` patches the commands with star (`\letdif*`, etc).

```
27 \def\s@fd@mu@p{\mathchoice{}{-}{\hbox{}}{\hbox{}}}
```

## 6.3 Declare the package options

```
28 \DeclareOption{rm}{%
29   \AtBeginDocument{\ifcsname symbf\endcsname%
30     \gdef\@fd@dif{\symrm{d}}\fi}%
31   \gdef\@fd@dif{\mathrm{d}}}
```

```

32 \DeclareOption{normal}{\gdef\fd@dif{d}}
33 \DeclareOption{partial}{\@tempswatrue}
34 \DeclareOption{nopartial}{\@tempswafalse}
35 \ExecuteOptions{rm,partial}
36 \ProcessOptions\relax
37 \if@tempswa
38   \AtEndOfPackage{\letdif{\partial}{partial}}
39 \fi

```

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

```
40 \gdef\resetdfont#1{\AtBeginDocument{\let\fd@dif\relax\gdef\fd@dif{#1{d}}}}
```

#### 6.4 Deal with the `\d` command

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

```
41 \def\fd@dif{\mathinner{\@fd@dif}\fd@mu@p}
```

`\fd@d@acc` Restore the `\d` command in text by `\fd@d@acc` with `\let`.

```
42 \AtBeginDocument{\let\fd@d@acc\d}
```

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

```
43 \DeclareRobustCommand\d{\ifmmode\fd@dif\else\expandafter\fd@d@acc\fi}}
```

#### 6.5 User's interface for defining new differential operators

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

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

```

44 \def\@letdif#1#2{\AtBeginDocument{%
45   \ifcsname #2nondif\endcsname\else%
46   \expandafter\let\csname #2nondif\expandafter\endcsname
47     \csname #2\endcsname%
48   \fi%
49   \DeclareRobustCommand#1{\mathinner{\csname #2nondif\endcsname}\fd@mu@p}%
50 }}

```

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

```

51 \def\s@letdif#1#2{\AtBeginDocument{%
52   \ifcsname #2nondif\endcsname\else%
53   \expandafter\let\csname #2nondif\expandafter\endcsname
54     \csname #2\endcsname%
55   \fi%
56   \DeclareRobustCommand#1{\mathinner{\s@fd@mu@p\csname #2nondif\endcsname\hbox{}}}\fd@mu@p}%
57 }}

```

```
58 \DeclareRobustCommand\letdif{\@ifstar\s@letdif\@letdif}
59 \@onlypreamble\letdif
```

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

```
60 \long\def\@newdif#1#2{\AtBeginDocument{%
61   \ifdefined#1
62     \PackageError{fixdif}{\string#1 is already defined}
63     {Try another command instead of \string#1.}%
64   \else
65     \DeclareRobustCommand#1{\mathinner{#2}\fd@mu@p}%
66   \fi%
67 }}
68 \long\def\s@newdif#1#2{\AtBeginDocument{%
69   \ifdefined#1
70     \PackageError{fixdif}{\string#1 is already defined}
71     {Try another command instead of \string#1.}%
72   \else
73     \DeclareRobustCommand#1{\s@fd@mu@p\mathinner{#2\hbox{}}}\fd@mu@p}%
74   \fi%
75 }}
76 \DeclareRobustCommand\newdif{\@ifstar\s@newdif\@newdif}
77 \@onlypreamble\newdif
```

`\renewdif` Define the `\renewdif` command.

```
78 \long\def\@renewdif#1#2{\AtBeginDocument{%
79   \ifdefined#1
80     \DeclareRobustCommand#1{\mathinner{#2}\fd@mu@p}%
81   \else
82     \PackageError{fixdif}{\string#1 has not been defined yet}
83     {You should use \string\newdif instead of \string\renewdif.}%
84   \fi%
85 }}
86 \long\def\s@renewdif#1#2{\AtBeginDocument{%
87   \ifdefined#1
88     \DeclareRobustCommand#1{\s@fd@mu@p\mathinner{#2\hbox{}}}\fd@mu@p}%
89   \else
90     \PackageError{fixdif}{\string#1 has not been defined yet}
91     {You should use \string\newdif instead of \string\renewdif.}%
92   \fi%
93 }}
94 \DeclareRobustCommand\renewdif{\@ifstar\s@renewdif\@renewdif}
95 \@onlypreamble\renewdif
```

## 6.6 In-document commands: `\mathdif`

```
96 \def\@mathdif#1{\mathinner{#1}\fd@mu@p}
97 \def\s@mathdif#1{\s@fd@mu@p\mathinner{#1\mbox{}}}\fd@mu@p}
98 \DeclareRobustCommand\mathdif{\@ifstar\s@mathdif\@mathdif}
```

End of the package.

```
99 </package>
```