

The `fixdif` Package

Zhang Tingxuan

2023/02/21 Version 2.0b*

Abstract

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

The package does well with $\text{pdf}\text{\TeX}$, $\text{X}\text{\LaTeX}$ and $\text{Lua}\text{\TeX}$, only works with \LaTeX format. Furthermore, this package is compatible with `unicode-math` package in $\text{X}\text{\LaTeX}$ and $\text{Lua}\text{\TeX}$.

Contents

1	The background	1
2	Introduction	2
2.1	Basic commands and package options	2
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	4
5	Examples	4
6	The source code	5
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 a small skip should be reserved between the differential operator and the expression before it¹. Take the following line 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 left side. The small skip between $f(x)$ and dx can be regarded as a binary operator.

Some users prefer to define a macro like this:

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

This macro works well in display math and text math, but still appears with the following three problems:

1. The skip before “d” still exists before the denominator in “text fraction”. This is what we do not hope to see. For example, $\mathop{d} y/\mathop{d} x$ produces dy/dx .
2. \mathop{d} is defined as a text accent command in L^AT_EX 2_ε kernel. If we defined like this, \mathop{o} could not produce “o” in text.
3. The skip before “d” should behave like skips around a binary operator. It should disappear in script math and script script math. For example, $\mathop{a+b}$ yields $a + b$ while $\mathop{a+b}$ yields a^{+b} , the skips around “+” disappear in superscript. But in the definition above, $\mathop{f(x)d} x$ yields $f(x)^{dx}$ but not $f(x)dx$.

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,

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

will produce

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

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

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

ields “ dx and \grave{x} ”.

Set the font of `\d` There are two package options to control the style of `\d` 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, that is

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

```
$f(x)\d x$ yields  $f(x)dx$ .
```

`\resetdfont` Regardless of the two options above, you can reset the font of `\d` through `\resetdfont` command in preamble:

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

then `$\d x$` yields dx . Notice that the argument of `\resetdfont` should be a command with *one* argument.

`\partial` **Control the behavior of `\partial`** In default, `\partial` will be regarded as a differential operator after you load `fixdif`. If you don’t like this default setting, you can use the `nopartial` option:

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

If you choose to use the default settings, `\partialnondif` yields the ordinary symbol “ ∂ ”.

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 takes 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,

¹See <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 $\langle csname \rangle$ 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 \rangle nondif` in math mode. For example, in default, `\partialnondif` yields the old partial symbol “ ∂ ”.

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

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

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

Then you get $\backslash D$ for an uppercase upright “D” as a differential operator.

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

$\backslash newdif$ and $\backslash newdif*$ will check whether $\langle cmd \rangle$ has been defined already. If so, an error message will be given.

```
 $\backslash renewdif$   $\backslash renewdif\{\langle cmd \rangle\}\{\langle multi-cmd \rangle\}$           (without correction, preamble only)
 $\backslash renewdif*$   $\backslash renewdif*\{\langle cmd \rangle\}\{\langle multi-cmd \rangle\}$       (with correction, preamble only)
```

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

4 Using differential operators temporarily

```
 $\backslash mathdif$   $\backslash mathdif\{\langle symbol \rangle\}$           (without correction, in math mode only)
 $\backslash mathdif*$   $\backslash mathdif*\{\langle symbol \rangle\}$       (with correction, in math mode only)
```

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

5 Examples

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

Take the two examples below:

```
 $\backslash letdif\{\backslash Delta\}\{Delta\}$           % Example 1, in preamble
 $\backslash letdif\{\backslash nabla\}\{nabla\}$       % 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 $\backslash Delta$ 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:

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

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

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

Then $\backslash laplacian$ produces ∇^2 and $\backslash laplacian*$ produces Δ .

Dealing with “+” and “−” If you input $-\mathrm{d} x$, you’ll get “ $-dx$ ” in your document. However, if you think “ $-dx$ ” is better, you can input $-\mathrm{d} x$. The “ $\mathrm{d} x$ ” 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 $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ format and provides the package name.

```
2 \NeedsTeXFormat{LaTeX2e}
3 \ProvidesPackage{fixdif}[2023/02/21 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 $\mathrm{X}_{\mathrm{E}}\mathrm{T}_{\mathrm{E}}\mathrm{X}$ / $\mathrm{L}^{\mathrm{u}}\mathrm{a}\mathrm{T}_{\mathrm{E}}\mathrm{X}$ 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}{\@tempswatruel}
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>

```