

The `fixdif` Package

Zhang Tingxuan

2022/05/27 Version 1.0*

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

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	The source code	5
4.1	Control the skip between slash and differential operator	5
4.2	Patch the skips around the differential operator	5
4.3	Declare the package options	5
4.4	Deal with the <code>\d</code> command	6
4.5	User's interface for defining new differential operators	6

*<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 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\mathrm{\mathop{\mathrm{d}}\!\}
```

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

1. The skip before “d” would still be reserved in “text fraction”, which is regarded bad. For example, `\mathrm{d} y/\mathrm{d} x` produces dy/dx ;
2. This `\mathrm{d}` command cannot be used out of math mode. In another word, `\mathrm{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 `\mathrm{d}` (`\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 `\mathrm{d}` 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. In your document,

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

will produce

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

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

2.1 When using unicode-math

If you are using `unicode-math` package with X_YTeX/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` provided by `unicode-math` in order to avoid bad skip in text fraction like dy/dx .
- 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 the text, `\d` behaves just like the old `\d` command in L^ATeX or plain TeX as an accent command. For example,

`\d x` and `\d x`

will produce “ dx 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 whose case `\d x` produces $f(x)dx$. If you chose the `normal` option, for example

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

`\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:

```
\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 *name* of a *single* character command (without the backslash on the front). For example,

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

then the `\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

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

The first argument of these commands is the newly-defined command; and the second argument should contain *more than one commands* or *a string*. 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`, you should use `\letdif` or `\letdif*` instead.

These two commands will check whether `⟨cmd⟩` has been defined already. If so, an error message will be given.

<code>\newdif</code>	<code>\renewdif{⟨cmd⟩}{⟨multi-cmd⟩}</code>	(without correction, preamble only)
<code>\newdif*</code>	<code>\renewdif*{⟨cmd⟩}{⟨multi-cmd⟩}</code>	(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 The source code

```
1 ⟨*package⟩
```

Check the `TeX` format and provides the package name.

```
2 \NeedsTeXFormat{LaTeX2e}
3 \ProvidesPackage{fixdif}[2022/05/26 Interface for defining the differential operator.]
```

4.1 Control the skip between slash 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 `XYTeX/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 }

```

4.2 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{}}}

```

4.3 Declare the package options

Declare the options of the package and execute them.

```

20 \DeclareOption{rm}{\@ifpackageloaded{unicode-math}{}{\def\@@dif{\symrm{d}}}{\def\@@dif{\mathrm{d}}}}
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}}}

```

4.4 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{\ifmmode%
34 \mathinner{\@dif}\mup@tch%
35 \fi}

```

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

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

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

```
37 \gdef\d{\ifmode\@dif\else\expandafter\d@accent\fi}
```

4.5 User's interface for defining new differential operators

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

```
38 \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`.

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

Finally let `#1` be the new command.

```
43 \gdef#1{\mathinner{\csname #2@old\endcsname}\mup@tch}
44 }}
```

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

```
45 \def\s@letdif#1#2{\AtBeginDocument{
46 \ifcsname #2@old\endcsname\else
47 \expandafter\let\csname #2@old\expandafter\endcsname%
48 \csname #2\endcsname
49 \fi
50 \gdef#1{\mathinner{\s@beforep@tch\csname #2@old\endcsname\mbox{}}}\mup@tch}
51 }}
52 \def\letdif{\@ifstar\s@letdif\@letdif}
```

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

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

```

66     \long\gdef#1{\s@beforep@tch\mathinner{#2\mbox{}}}\mup@tch}
67   \fi
68 }}
69 \def\newdif{\@ifstar\s@newdif\@newdif}

\renewdif Define the \renewdif and \renewdif* commands.
\renewdif*
70 \long\def\@renewdif#1#2{\AtBeginDocument{
71   \ifdefined#1
72     \long\gdef#1{\mathinner{#2}\mup@tch}
73   \else
74     \PackageError{fixdif}{\string#1 has not been defined yet.}
75     {You should use \string\newdif instead of \string\renewdif.}
76   \fi
77 }}
78 \long\def\s@renewdif#1#2{\AtBeginDocument{
79   \ifdefined#1
80     \long\gdef#1{\s@beforep@tch\mathinner{#2\mbox{}}}\mup@tch}
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 \def\renewdif{\@ifstar\s@renewdif\@renewdif}
87 \end{package}

```