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 as differential operators.

The package is compatible with pdfTeX, XeLaTeX and LuaTeX. Furthermore, the package is compatible with unicode-math package in XeLaTeX and LuaTeX.

Contents

1 The background

It's usually recommended that one should reserve a small skip between the differential operator and the expression before it[1]. Take the following cases as an example:

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

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 $\mathrm{d}x$ can be regarded as a symbol of the product of $f(x)$ and $\mathrm{d}x$.

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 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 $\mathrm{d}y/\mathrm{d}x$;
2. This `\d` command cannot be used out of math mode. In another word, `\d{o}` would not produce "o̩" in text;

*https://github.com/AlphaZTX

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

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 $a^{f(x)\,\mathrm{d}x}$ but not $a^{f(x)\mathrm{d}x}$.

To solve these problems, you can try this package.

2 Introduction

To load this package, write

```
\usepackage{fixdif}
```

in the preamble. In your document,

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

will produce

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

2.0.1 When using unicode-math

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

- If you want to 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 $\mathrm{d}y/\mathrm{d}x$.
- Load the fixdif package *after* unicode-math.

Therefore the correct order is

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

2.0.2 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.

2.0.3 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 LaTeX or plain TeX as an accent command. For example,

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

will produce "dx and x̣".

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 \$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:

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

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

\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!

3.0.1 Define commands with a single command name

\letdif \letdif{⟨*cmd*⟩}{⟨*csname*⟩} (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 ⟨*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.

3.0.2 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 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.

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

```
⟨*package⟩
```

Check the TEX format and provides the package name.

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

4.0.1 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.

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

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

```
  \Umathcode`\/="4 "0 "002F
  \Umathcode"2044="4 "0 "2044
  \Umathcode"2215="4 "0 "2215
  \Umathcode"2F98="4 "0 "2F98
  \Umathcode`\\="4 "0 "005C
  \Umathcode"2216="4 "0 "2216
  \Umathcode"29F5="4 "0 "29F5
  \Umathcode"29F9="4 "0 "29F9
}{
```

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.

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

4.0.2 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{}}}
```

4.0.3 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}}}
```

4.0.4 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{\ifmmode%
  \mathinner{\@@dif}\mup@tch%
\fi}
```

`\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`

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

4.0.5 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}
⟨/package⟩
```