The fixdif Package

Zhang Tingxuan

2022/7/19 Version 1.3a[*]

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

Contents

[*]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[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 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 $\mathrm{d}y/\,\mathrm{d}x$;
2. This `\d` command cannot be used out of math mode. In another word, `\d{o}` would not produce "ọ" 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 $a^{f(x)\,\mathrm{d}x}$ but not $a^{f(x)\mathrm{d}x}$.

To solve these problems, you can try this package.

[1] See 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,

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

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 XƎTEX/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 LATEX or plain TEX as an accent command. For example,

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

will produce "$\mathrm{d}x$ and ẉ" .

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 $f(x)\,\mathrm{d}x$. 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 in preamble:

```
\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!

3.1 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 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 ⟨*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.

3.2 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{\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`, 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 "$-\,\mathrm{d}x$" in your document. However, if you think "$-\mathrm{d}x$" 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 "$-\,\mathrm{d}x$" is just okay.

6 The source code

```
⟨*package⟩
```

Check the TEX format and provides the package name.

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{fixdif}[2022/7/19 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.

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

If the unicode-math package has been loaded, use the XƎTEX/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}
}
```

6.2 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{}}}
```

6.3 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}}}
```

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.

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

\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

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

6.5 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}
```

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

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

End of the package.

```
⟨/package⟩
```