The derivative package

Written by:
Simon Jensen
sjelatex@gmail.com

Released:
v0.97
2019/02/03

The derivative package provides a set of commands which makes writing ordinary and partial derivatives of arbitrary order in a straight forward manner. Additionally, this package provides a set of commands to define variants of the aforementioned derivatives. A set of optional arguments along with lots of package options allow for easy and great flexibility over the derivative's format, such as where the function is positioned, point of evaluation, and switching between fraction styles. Moreover, the mixed order of the partial derivative and variants hereof is automatically computed. This package is written in the expl3 language and requires therefore the LaTeX3 package bundles l3kernel and l3package. Additionally, the mleftright package is optional and provides the improved automatically scaling `\mleft` and `\mright`.

NB: This is a beta version and some elements are subject to change.

Contents

1 Introduction

This is a beta version meaning that some elements might be changed in the official release. Using this package in its current state is therefore on your own risk. The next update, version `v1.0`, will contain the changes specified in section 8.

This package started as a personal package a few years ago that I used in various projects. Firstly, it was written in TEX and LATEX, which gave rise to various errors as the complexity of the package grew larger and ended up as a spaghetti code. Therefore, the code was rewritten into the LATEX3 language and may now be easily maintained. Originally this package was written because there did not exist any great package for derivatives. It was first much later that I discovered the diffcoeff package, which does a good job. However, at this time, I had already written much of the code with no documentation. I decided to write one and make it into a package for the public.

As a note on terminology, I use the wording *infinitesimal* and abbreviated as `inf` for the operator symbol d, ∂, δ etc. used in various derivatives like $\frac{\mathrm{d}y}{\mathrm{d}x}, \frac{\partial y}{\partial x}, \frac{\delta y}{\delta x}$ etc. Moreover, I explicitly use *differential* d for d and *partial* for ∂. In the description of macros and options, I often write *cs-*⟨*placeholder*⟩ to denote a comma-separated list of ⟨*placeholder*⟩. For example, `[`⟨*cs-orders*⟩`]` is used in the partial derivative's description to denote the order differentiation and is read as a comma-separated list of orders. It should also be noted that whenever an argument reads ⟨*key=value*⟩, it means that it is a comma-separated list of key-value pairs.

2 Ordinary derivative

`\odv` `*[`⟨*order*⟩`]{`⟨*function*⟩`}/{`⟨*variable*⟩`}_{`⟨$point_1$⟩`}^{`⟨$point_2$⟩`}`

The ordinary derivative `\odv` is defined with a set of mandatory and optional arguments that either typeset specific parts or changing the style of the derivative. In this package, the ordinary derivative is defined with an upright lowercase d, because it is used by many nowadays books, as

```
\DeclareOdvVariant{\odv}{d}[style-inf=\mathrm]
\DeclareOdvVariant{\odv}{d}[style-inf=\symup]
```

pdfTEX
XETEX, LUATEX

`*` The first argument of `\odv` is an optional star that determines where the function is typeset; either in the numerator of the fraction or next to the fraction. Using the option `switch-*=false`, the function is typeset in the numerator when the star is absent, and next to the fraction when the star is present as shown below

$$\texttt{\textbackslash odv\{y\}\{x\}} \implies \frac{\mathrm{d}y}{\mathrm{d}x}$$
$$\texttt{\textbackslash odv*\{y\}\{x\}} \implies \frac{\mathrm{d}}{\mathrm{d}x}y$$

The effect of the star's presence can be switched around using `switch-*=true` i.e. the equations in the above example are exchanged, as described in section 4.2.

`[`⟨*order*⟩`]` The second argument is optional and is written inside square brackets. This argument is used to set the order of differentiation as seen below

$$\texttt{\textbackslash odv[2]\{y\}\{x\}} \implies \frac{\mathrm{d}^2y}{\mathrm{d}x^2}$$
$$\texttt{\textbackslash odv[n]\{y\}\{x\}} \implies \frac{\mathrm{d}^ny}{\mathrm{d}x^n}$$
$$\texttt{\textbackslash odv[n+2]\{y\}\{x\}} \implies \frac{\mathrm{d}^{n+2}y}{\mathrm{d}x^{n+2}}$$

The order may be a number, a symbol and a combination hereof, because it is simply typeset there. This is different from how the partial derivative's mixed order is typeset, which is computed. The order *is not* automatically sorted as the mixed order for the partial derivative. This argument is subject to change, see consideration 8.3(i) for more information.

`{`⟨*function*⟩`}` This is the first mandatory argument that typeset the function that is to be differentiated

$$\texttt{\textbackslash odv\{f(x)\}\{x\}} \implies \frac{\mathrm{d}f(x)}{\mathrm{d}x}$$
$$\texttt{\textbackslash odv\{e\^{}\{\textbackslash sin(x)\}\}\{x\}} \implies \frac{\mathrm{d}e^{\sin(x)}}{\mathrm{d}x}$$

The function is simply typeset in the numerator or next to the fraction.

`/` The fourth argument is an optional slash written *between the function and the variable* and determines which fraction style the derivative is typeset with, as described in section 4.2. Using the option `switch-/=false` and the package's default settings, the derivative is typeset with `\frac` when the slash is absent, and `\slashfrac`[1] when the slash is present, as shown below

$$\texttt{\textbackslash odv\{y\}\{x\}} \implies \frac{\mathrm{d}y}{\mathrm{d}x}$$
$$\texttt{\textbackslash odv\{y\}/\{x\}} \implies \mathrm{d}y/\mathrm{d}x$$

As for the star argument, the effect of the slash's presence can be switched around using `switch-/=true` i.e. the equations in the above example are exchanged, as described in section 4.2.

`{`⟨*variable*⟩`}` This is the second and final mandatory argument, which purpose is to typeset the variable in which the function is differentiated with respect to as shown below

$$\texttt{\textbackslash odv\{f\}\{x\}} \implies \frac{\mathrm{d}f}{\mathrm{d}x}$$
$$\texttt{\textbackslash odv\{f\}\{y\}} \implies \frac{\mathrm{d}f}{\mathrm{d}y}$$

The variable is always typeset in the denominator.

`_{`⟨*point*$_1$⟩`}^{`⟨*point*$_2$⟩`}` This is the last optional argument that specifies the point(s) of evaluation. It is an *e-type* argument in the `xparse` language and given as `e{_^}`. This means that the subscript `_` and superscript `^` accepts an argument given in braces. Moreover, the order of `_` and `^` is independent as shown below

$$\texttt{\textbackslash odv\{y\}\{x\}\_\{x\_1\}} \implies \left.\frac{\mathrm{d}y}{\mathrm{d}x}\right|_{x_1}$$
$$\texttt{\textbackslash odv\{y\}\{x\}\^{}\{x\_2\}} \implies \left.\frac{\mathrm{d}y}{\mathrm{d}x}\right|^{x_2}$$
$$\texttt{\textbackslash odv\{y\}\{x\}\_\{x\_1\}\^{}\{x\_2\}} \implies \left.\frac{\mathrm{d}y}{\mathrm{d}x}\right|_{x_1}^{x_2}$$
$$\texttt{\textbackslash odv\{y\}\{x\}\^{}\{x\_2\}\_\{x\_1\}} \implies \left.\frac{\mathrm{d}y}{\mathrm{d}x}\right|_{x_1}^{x_2}$$

The common way to specify the point of evaluation is using the subscript argument. If needed, the second point of evaluation can be specified with the superscript argument.

[1] which is a macro defined by the package, see section 7.1 for more information.

2.1 Variants

This package offers four variants of the ordinary derivative: Material derivative `\mdv`, functional derivative `\fdv`, the average rate of change `\adv` and the Jacobian `\jdv`. A unique feature of this package, is that you can define your own variants of the ordinary derivative as described in section 5.1.

`\mdv` `*[`⟨*order*⟩`]{`⟨*function*⟩`}/{`⟨*variable*⟩`}_{`⟨*point*$_1$⟩`}^{`⟨*point*$_2$⟩`}`

The material derivative is used in some branches of physics (e.g. thermodynamic, fluid dynamic etc.). It is only natural for a physics student as me to define such a variant, that used an upright uppercase D. In this package, the material derivative is defined as

```
\DeclareOdvVariant{\mdv}{D}[style-inf=\mathrm]
\DeclareOdvVariant{\mdv}{D}[style-inf=\symup]
```

pdfTEX
XETEX, LUATEX

In physics, the material derivative is defined by

$$\frac{\mathrm{D}\varphi(\mathbf{r},t)}{\mathrm{D}t} := \frac{\partial\varphi(\mathbf{r},t)}{\partial t} + \dot{\mathbf{r}}\cdot\nabla\varphi(\mathbf{r},t)$$

`\fdv` `*[`⟨*order*⟩`]{`⟨*function*⟩`}/{`⟨*variable*⟩`}_{`⟨*point*$_1$⟩`}^{`⟨*point*$_2$⟩`}`

The functional derivative is used in the calculus of variation and uses a lowercase delta. With the package's default settings, it will use an italic delta. In this package, the functional derivative is defined as

```
\DeclareOdvVariant{\fdv}{\delta}
```

pdfTEX, XETEX, LUATEX

In physics, it is for example used in the Lagrange equation or in the derivation of the Hartree-Fock equation

$$\frac{\delta I}{\delta q_\alpha} = \frac{\partial L}{\partial q_\alpha} - \frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L}{\partial \dot{q}_\alpha} = 0, \qquad \frac{\delta\mathcal{L}}{\delta\psi_n^*} = \hat{F}|\psi_n\rangle - \epsilon_n|\psi_n\rangle = 0,$$

respectively.

`\adv` `*[`⟨*order*⟩`]{`⟨*function*⟩`}/{`⟨*variable*⟩`}_{`⟨*point*$_1$⟩`}^{`⟨*point*$_2$⟩`}`

The average rate of change is defined to use an upright uppercase delta with the packages default settings. In this package, the average rate of change is defined as

```
\DeclareOdvVariant{\adv}{\Delta}[style-inf=\mathrm]
\DeclareOdvVariant{\adv}{\Delta}
```

pdfTEX
XETEX, LUATEX

The average rate of change is used to determine the slope of a straight line

$$\frac{\Delta y}{\Delta x} = \frac{y_2 - y_1}{x_2 - x_1}$$

`\jdv` `*[`⟨*order*⟩`]{`⟨*function*⟩`}/{`⟨*variable*⟩`}_{`⟨*point*$_1$⟩`}^{`⟨*point*$_2$⟩`}`

The Jacobian is defined to use an italic partial differential with the package's default settings. Also a pair of parentheses is automatically inserted around the function and variable. In this package, the Jacobian is defined as

```
\DeclareOdvVariant{\jdv}{\partial}[misc-add-delims={fun,var}]
```
pdfTEX, XETEX, LuaTEX

which gives

$$\frac{\jdv(f,g,h)}{\jdv(x,y,z)}$$

3 Partial derivative

\pdv `*[`⟨*cs-orders*⟩`][`⟨*mixed order*⟩`]{`⟨*function*⟩`}/{`⟨*cs-variables*⟩`}_{`⟨$point_1$⟩`}^{`⟨$point_2$⟩`}`

The partial derivative `\pdv` is defined with a set of mandatory and optional arguments that either typeset specific parts or changing the style of the derivative. In this package, the partial derivative is defined with an italic partial differential as

```
\DeclarePdvVariant{\pdv}{\partial}[sep-inf-ord=1, sep-ord-fun=-2]
```

pdfTEX, XETEX, LUATEX

`*` The first argument of `\pdv` is an optional star that determines where the function is typeset; either in the numerator of the fraction or next to the fraction. Using the option `switch-*=false`, the function is typeset in the numerator when the star is absent, and next to the fraction when the star is present as shown below

$$\texttt{\textbackslash pdv\{f\}\{x,y\}} \implies \frac{\partial^2 f}{\partial x\,\partial y}$$

$$\texttt{\textbackslash pdv*\{f\}\{x,y\}} \implies \frac{\partial^2}{\partial x\,\partial y} f$$

The effect of the star's presence can be switched around using `switch-*=true` i.e. the equations in the above example are exchanged, as described in section 4.3.

`[`⟨*cs-orders*⟩`]` The second argument is optional and is written inside square brackets. This argument is used to set the order of differentiation for each of the variables and should be given in a comma-separated list e.g. `[1,2,3]` as shown below

$$\texttt{\textbackslash pdv[2,3]\{f\}\{x,y\}} \implies \frac{\partial^5 f}{\partial x^2\,\partial y^3}$$

$$\texttt{\textbackslash pdv[\textbackslash beta,a,n+2a]\{f\}\{x,y,z\}} \implies \frac{\partial^{3a+\beta+n} f}{\partial x^{\beta}\,\partial y^{a}\,\partial z^{n+2a}}$$

$$\texttt{\textbackslash pdv[2,n\^{}2,n\^{}2-1]\{f\}\{x,y,z\}} \implies \frac{\partial^{2n^2+1} f}{\partial x^2\,\partial y^{n^2}\,\partial z^{n^2-1}}$$

The orders may consist numbers, symbols and a combination hereof. Notice, that the mixed order is automatically calculated and sorted based on the variables orders by the package. See sections 4.3 and 6 for more information. This argument is subject to change, see consideration 8.3(i) for more information.

`[`⟨*mixed order*⟩`]` If you, unfortunately, come to the conclusion, that you are unsatisfied with how the package typeset the mixed order or if it gives a wrong result, then you can overwrite it with this argument. The use of this argument completely bypass the automatic calculation and sorting done by the package, and simply typeset the mixed order as you wrote it as shown below

$$\texttt{\textbackslash pdv[4n+2k, 3k+n]\{ f(x,y) \}\{ x,y \}} \implies \frac{\partial^{5n+5k} f(x,y)}{\partial x^{4n+2k}\,\partial y^{3k+n}}$$

$$\texttt{\textbackslash pdv[4n+2k, 3k+n][5(n+k)]\{ f(x,y) \}\{ x,y \}} \implies \frac{\partial^{5(n+k)} f(x,y)}{\partial x^{4n+2k}\,\partial y^{3k+n}}$$

This argument is also subject to change, see consideration 8.3(ii) for more information.

`{`⟨*function*⟩`}` This is the first mandatory argument that typeset the function that is to be partially differentiated as shown below

```
\pdv{ f(x,y,z) }{ x, y, z }
```
$$\Longrightarrow \quad \frac{\partial^3 f(x,y,z)}{\partial x\,\partial y\,\partial z}$$

```
\pdv{ e^x \sin(y) \ln(z) }{ x, y, z }
```
$$\Longrightarrow \quad \frac{\partial^3 e^x \sin(y)\ln(z)}{\partial x\,\partial y\,\partial z}$$

The function is simply typeset in the numerator or next to the fraction.

`/` The fifth argument is an optional slash written *between the function and the variables* and determines which fraction style the derivative is typeset with, as described in section 4.3. Using the option `switch-/=false` and the package's default settings, the derivative is typeset with `\frac` when the slash is absent and `\slashfrac`[1] when the slash is present, as shown below

```
\pdv{f}{x,y}
```
$$\Longrightarrow \quad \frac{\partial^2 f}{\partial x\,\partial y}$$

```
\pdv{f}/{x,y}
```
$$\Longrightarrow \quad \partial^2 f \big/ \partial x\,\partial y$$

As for the star argument, the effect of the slash's presence can be switched around using `switch-/=true` i.e. the equations in the above example are exchanged, as described in section 4.3.

`{`⟨*cs-variables*⟩`}` This is the second and final mandatory argument, which purpose is to typeset the variables in which the function is partially differentiated with respect to. The variables are given in a comma-separated list e.g. `{x, y, z}`, as shown below

```
\pdv{f}{x}
```
$$\Longrightarrow \quad \frac{\partial f}{\partial x}$$

```
\pdv{f}{x,y}
```
$$\Longrightarrow \quad \frac{\partial^2 f}{\partial x\,\partial y}$$

The variables are typeset in the denominator with a ∂ inserted to the left of them.

`_{`⟨*point*$_1$⟩`}^{`⟨*point*$_2$⟩`}` This is the last optional argument that specifies the point(s) of evaluation or variables held constant. It is an *e-type* argument in the xparse language and given as `e{_^}`. This means that the subscript `_` and superscript `^` accepts an argument given in braces. Moreover, the order of `_` and `^` is independent as shown below

```
\pdv{f}{x,y}_{(x_1,y_1)}
```
$\Longrightarrow$ $\left(\dfrac{\partial^2 f}{\partial x\,\partial y}\right)_{(x_1,y_1)}$

```
\pdv{f}{x,y}^{(x_2,y_2)}
```
$\Longrightarrow$ $\left(\dfrac{\partial^2 f}{\partial x\,\partial y}\right)^{(x_2,y_2)}$

```
\pdv{f}{x,y}_{(x_1,y_1)}^{(x_2,y_2)}
```
$\Longrightarrow$ $\left(\dfrac{\partial^2 f}{\partial x\,\partial y}\right)^{(x_2,y_2)}_{(x_1,y_1)}$

```
\pdv{f}{x,y}^{(x_2,y_2)}_{(x_1,y_1)}
```
$\Longrightarrow$ $\left(\dfrac{\partial^2 f}{\partial x\,\partial y}\right)^{(x_2,y_2)}_{(x_1,y_1)}$

The subscript argument is commonly used as the point of evaluation or variables held constant. If needed, the superscript argument may be used for the second point of evaluation.

3.1 Variants

This package does not define any variants of the partial derivative. However, a unique feature of this package, is that you can define your own variants of the partial derivative as described in section 5.2.

4 Package options

This package accepts its options using the well-known *key=value* syntax. The keys are divided into categories, for which each key have its associated category as a prefix. Each category and option is explained below.

`\derivset` `{`⟨*derivative*⟩`}[`⟨*key=value*⟩`]`

The package options can be set either in the preamble or in the document using the `\derivset` command. The options may also be set when defining a new derivative. Currently, there is no other way to set the options however this is subject to change, see consideration 8.3(i) for more information.

`{`⟨*derivative*⟩`}` A mandatory argument that determines which derivative the *key=value* pairs is assigned to, where the allowed ⟨*main-category*⟩ are the derivatives defined by the packages and you, see sections 5.1 and 5.2. The special value `all` is also allowed, which gives access to the options that applies to *all* derivatives.

`[`⟨*key=value*⟩`]` This optional argument accepts its input as a comma-separated list of *key=value* pairs. Leaving out `[`⟨*key=value*⟩`]` sets the options to the packages default settings for the chosen `{`⟨*derivative*⟩`}` e.g. `\derivset{\odv}` sets the options for the ordinary derivative to the packages default settings. This argument is subject to change, see consideration 8.3(iii) for more information.

4.1 Categories

This section seeks to give a detailed description of each category.

- The `style-`⟨…⟩ keys sets either the font style or the fraction style of ⟨…⟩. The font styles `\mathnormal`/`\symnormal` and `\mathrm`/`\symup` are commonly used in literature. The fraction style can be either `\frac`, `\dfrac`, `\tfrac`, `\sfrac`[R], `\slashfrac`[1] and many more.
- The `delims-`⟨…⟩ keys sets the delimiters used around the ⟨…⟩. The Rule of Two applies here: 'Always two there are, no more, no less. A left and a right delimiter'. The allowed delimiters are those that can be scaled with `\left`, `\big` etc.
- The `scale-`⟨…⟩ keys sets the size of the ⟨…⟩'s delimiters. The values `big`, `Big`, `bigg`, and `Bigg` are self-explanatory and internally a left and right version are used e.g. `\bigl` and `\bigr`. The value `none` inserts the delimiters (except periods) unscaled. While the value `auto` scales the delimiters automatically using `\left` and `\right`.
- The `sep-`⟨…⟩`-`⟨…⟩ keys inserts math space between ⟨…⟩ and ⟨…⟩ using `\mskip`. These keys accepts a comma-separated list of numbers `{x,y,z}` that are internally used to form the syntax `x mu plus y mu minus z mu`.
- The `switch-`⟨…⟩ keys serves to change an argument's behaviour by swapping the effect of an optional character's presence.

- The `sort-⟨…⟩` keys deals with the sorting algorithm behind the mixed order, where you can choose the sorting method that suits you best. These keys are unique to the partial derivative and variants of it.
- The `mics-⟨…⟩` keys are miscellaneous keys that does not belong to any of the above categories.

Note. A value with superscripted `U`, `P` and `R` refers to a unicode engine i.e. LuaTEX and XeTEX, pdfTEX and requires package, respectively. Additionally, some keys have two versions; with and without `-/` at the end. These keys are related to the slash argument. If `switch-/=false` then the keys with and without `-/` are used when the slash argument is present and absent, respectively. Setting `switch-/=true` then these keys are used in the opposite cases for the slash's presence.

4.2 Ordinary derivative

The options in this subsection are available for the ordinary derivative `\odv` and variants hereof that are defined by the package and you.

Style

`style-inf` ⟨*math-font-style*⟩ `\mathnormal`$^{\texttt{P}}$, `\symnormal`$^{\texttt{U}}$

The font style of the infinitesimal for the chosen derivative is set by this key. The default font style is `\mathnormal`$^{\texttt{P}}$ and `\symnormal`$^{\texttt{U}}$. If an upright font is preferred then `\mathrm`$^{\texttt{P}}$ and `\symup`$^{\texttt{U}}$ could be used, or if an italic font then `\mathnormal`$^{\texttt{P}}$ and `\symit`$^{\texttt{U}}$.

`style-frac` ⟨*fraction*⟩ `\frac`

The derivative uses the fraction set by this key whenever `switch-/=false` and the slash argument is absent. The key's default value is the usual fraction `\frac`.

`style-frac-/` ⟨*fraction*⟩ `\slashfrac`

The derivative uses the fraction set by this key whenever `switch-/=false` and the slash argument is present. The key's default value is a text-styled fraction `\slashfrac`[1] i.e. on the form $\mathrm{d}y/\mathrm{d}x$.

Scaling

`scale-eval` `auto, none, big, Big, bigg, Bigg` `auto`

This key sets the size of the delimiters used for the point of evaluation. This scaling is used whenever `switch-/=false` and the slash argument is absent. The key's default value is set to scale the delimiters automatically.

`scale-eval-/` `auto, none, big, Big, bigg, Bigg` `auto`

This key sets the size of the delimiters used for the point of evaluation. This scaling is used whenever `switch-/=false` and the slash argument is present. The key's default value is set to scale the delimiters automatically.

`scale-fun` `auto, none, big, Big, bigg, Bigg` `auto`

This key sets the size of the delimiters used around the function. The scaling is used whenever `misc-add-delims=fun` is used. The key's default value is set to scale the delimiters automatically.

`scale-var` `auto, none, big, Big, bigg, Bigg` `auto`

This key sets the size of the delimiters used around the variable. The scaling is used whenever `misc-add-delims=var` is used. The key's default value is set to scale the delimiters automatically.

`scale-frac` `auto, none, big, Big, bigg, Bigg` `auto`

This key sets the size of the delimiters used for around the fraction. This scaling is used whenever `switch-/=false`, the slash argument is absent and `misc-add-delims=frac` is used. The key's default value is set to scale the delimiters automatically.

`scale-frac-/` `auto, none, big, Big, bigg, Bigg` `auto`

This key sets the size of the delimiters used for around the fraction. This scaling is used whenever `switch-/=false`, the slash argument is present and `misc-add-delims=frac` is used. The key's default value is set to scale the delimiters automatically.

Delimiters

`delims-eval` ⟨*left delimiter*⟩⟨*right delimiter*⟩ `. \rvert`

This key sets the left and right delimiters used to indicate the point of evaluation. These delimiters are inserted whenever `switch-/=false` and the slash argument is absent. The default left and right delimiters is a period[2] and a vertical bar, respectively.

`delims-eval-/` ⟨*left delimiter*⟩⟨*right delimiter*⟩ `. \rvert`

This key sets the left and right delimiters used to indicate the point of evaluation. These delimiters are inserted whenever `switch-/=false` and the slash argument is present. The default left and right delimiters is a period[2] and a vertical bar, respectively.

`delims-fun` ⟨*left delimiter*⟩⟨*right delimiter*⟩ `( )`

This key sets the left and right delimiters used to around the function and these are inserted whenever `misc-add-delims=fun` is used. The key's default left and right delimiters are a left and a right parenthesis, respectively.

`delims-var` ⟨*left delimiter*⟩⟨*right delimiter*⟩ `( )`

This key sets the left and right delimiters used to around the variable and these are inserted whenever `misc-add-delims=var` is used. The key's default left and right delimiters are a left and a right parenthesis, respectively.

`delims-frac` ⟨*left delimiter*⟩⟨*right delimiter*⟩ `( )`

This key sets the left and right delimiters used to around the fraction in the derivative e.g. $\left(\frac{\mathrm{d}}{\mathrm{d}x}\right)y$ These delimiters are inserted whenever `switch-/=false`, the slash argument is absent and `misc-add-delims=frac` is used. The key's default left and right delimiters are a left and a right parenthesis, respectively.

`delims-frac-/` ⟨*left delimiter*⟩⟨*right delimiter*⟩ `( )`

[2] which doesn't output anything.

This key sets the left and right delimiters used to around the fraction in the derivative e.g. $(\mathrm{d}/\mathrm{d}x)y$ These delimiters are inserted whenever `switch-/=false`, the slash argument is present and `misc-add-delims=frac` is used. The key's default left and right delimiters are a left and a right parenthesis, respectively.

Math spacing

The options in this subsection inserts extra horizontal math space. The below equation illustrates where the space is inserted

$$\frac{\mathrm{d}\langle\texttt{inf-fun}\rangle y}{\mathrm{d}\langle\texttt{inf-var}\rangle x} \qquad \frac{\mathrm{d}^{\langle\texttt{inf-ord}\rangle 2}\langle\texttt{ord-fun}\rangle y}{\mathrm{d}x^{\langle\texttt{var-ord}\rangle 2}} \qquad \frac{\mathrm{d}y}{\mathrm{d}x}\Big|_{\langle\texttt{eval-sb}\rangle x_1}^{\langle\texttt{eval-sp}\rangle x_2},$$

where it has been split into three to give a better overview.

`sep-inf-ord` ⟨*cs-numbers*⟩ 0

This key sets the math space that is inserted in the infinitesimal's power left to the order. It is only inserted when the order is different from 1. The key's default value is 0 `mu`.

`sep-inf-fun` ⟨*cs-numbers*⟩ 0

This key sets the math space that is inserted between the infinitesimal and the function when the order *is equal* to 1. The space is only inserted when a non-blank function is printed *in the numerator*. The key's default value is 0 `mu`.

`sep-ord-fun` ⟨*cs-numbers*⟩ 0

This key sets the math space that is inserted between the infinitesimal and the function when the order *is different* from 1. The space is only inserted when a non-blank function is printed *in the numerator*. The key's default value is 0 `mu`.

`sep-inf-var` ⟨*cs-numbers*⟩ 0

This key sets the math space that is inserted between the infinitesimal and the variable. The space is only inserted when a non-blank variable is given. The key's default value is 0 `mu`.

`sep-var-ord` ⟨*cs-numbers*⟩ 0

This key sets the math space that is inserted in the variable's power left to the order. The space is only inserted when the order is different from 1. The key's default value is 0 `mu`.

`sep-eval-sb` ⟨*cs-numbers*⟩ 0

This key sets the math space that is inserted in the evaluation subscript left to the point of evaluation. The space is only inserted when a non-blank subscript is given. The key's default value is 0 `mu`.

`sep-eval-sp` ⟨*cs-numbers*⟩ 0

This key sets the math space that is inserted in the evaluation superscript left to the point of evaluation. The space is only inserted when a non-blank superscript is given. The key's default value is 0 `mu`.

Switches

`switch-*` `true, false` `false`

The effect of the star's presence can be switched with the value britishtrue. That is, the function is typeset next to the fraction when the star is absent and in the numerator when the star is present. As an example, compare below where the option is turned on (britishtrue) and off (britishfalse),

```
\derivset{\odv}[switch-*=false] \odv{y}{x}
```
$\Longrightarrow \quad \dfrac{\mathrm{d}y}{\mathrm{d}x}$

```
\derivset{\odv}[switch-*=true] \odv{y}{x}
```
$\Longrightarrow \quad \dfrac{\mathrm{d}}{\mathrm{d}x}y$

The key's default value is false.

`switch-/` `true, false` `false`

The effect of the slash's presence can be switched with the value britishtrue. That is, the derivative is typeset with the fraction set by `style-frac-/` when the slash is absent and with the fraction set by `style-frac` when the slash is present. As an example, compare below where the option is turned on (britishtrue) and off (britishfalse),

```
\derivset{\odv}[switch-/=false] \odv{y}{x}
```
$\Longrightarrow \quad \dfrac{\mathrm{d}y}{\mathrm{d}x}$

```
\derivset{\odv}[switch-/=true] \odv{y}{x}
```
$\Longrightarrow \quad \mathrm{d}y/\mathrm{d}x$

The key's default value is false.

Miscellaneous

`misc-add-delims` `fun, var, frac` `false`
`misc-remove-delims`

These two keys accepts its input as an comma-separated list of values such that

```
\derivset{\odv}[misc-add-delims=fun]
\derivset{\odv}[misc-add-delims={fun,var}]
\derivset{\odv}[misc-add-delims={fun,var,frac}]
```

are all valid, and the same goes for `misc-remove-delims`. The key `misc-add-delims` is used to insert the delimiters around the key's values, while `misc-remove-delims` is used to remove the inserted delimiters. The next example shows the key `misc-add-delims` in action

```
\derivset{\odv}[misc-add-delims=fun] \odv{y}{x}
```
$\Longrightarrow \quad \dfrac{\mathrm{d}(y)}{\mathrm{d}x}$

```
\derivset{\odv}[misc-add-delims={fun,var}] \odv{y}{x}
```
$\Longrightarrow \quad \dfrac{\mathrm{d}(y)}{\mathrm{d}(x)}$

```
\derivset{\odv}[misc-add-delims={fun,var,frac}] \odv{y}{x}
```
$\Longrightarrow \quad \left(\dfrac{\mathrm{d}(y)}{\mathrm{d}(x)}\right)$

All the above applies to `misc-remove-delims` as well, except it removes the delimiters. The effect of the key `misc-add-delims` is turned off (britishfalse) locally inside a derivative. This is to ensure that the option is not applied to more than one derivative when nested as shown below

`\derivset{\odv}[misc-add-delims=var] \odv*{\odv{y}{x}}{x}` $\Longrightarrow$ $\dfrac{\mathrm{d}}{\mathrm{d}(x)}\dfrac{\mathrm{d}y}{\mathrm{d}x}$

`\derivset{\odv}[misc-add-delims=fun] \odv*{\odv{y}{x}}{x}` $\Longrightarrow$ $\dfrac{\mathrm{d}}{\mathrm{d}x}\left(\dfrac{\mathrm{d}y}{\mathrm{d}x}\right)$

`\derivset{\odv}[misc-add-delims=frac] \odv*{\odv{y}{x}}{x}` $\Longrightarrow$ $\left(\dfrac{\mathrm{d}}{\mathrm{d}x}\right)\dfrac{\mathrm{d}y}{\mathrm{d}x}$

These two keys and this local behaviour are subject to change, see change 8.2(i) and consideration 8.3(v) for more information.

4.3 Partial derivative

The options in this subsection are available for the partial derivative `\pdv` and variants hereof that are defined by the package and you.

Style

`style-inf` ⟨*math-font-style*⟩ `\mathnormal`$^{\texttt{P}}$, `\symnormal`$^{\texttt{U}}$

The font style of the infinitesimal for the chosen derivative is set by this key. The default font style is `\mathnormal`$^{\texttt{P}}$ and `\symnormal`$^{\texttt{U}}$. If an upright font is preferred then `\mathrm`$^{\texttt{P}}$ and `\symup`$^{\texttt{U}}$ could be used, or if an italic font then `\mathnormal`$^{\texttt{P}}$ and `\symit`$^{\texttt{U}}$.

`style-frac` ⟨*fraction*⟩ `\frac`

The derivative uses the fraction set by this key whenever `switch-/=false` and the slash argument is absent. The key's default value is the usual fraction `\frac`.

`style-frac-/` ⟨*fraction*⟩ `\slashfrac`

The derivative uses the fraction set by this key whenever `switch-/=false` and the slash argument is present. The key's default value is a text-styled fraction `\slashfrac`[1] i.e. on the form $\partial^2 f/\partial x\,\partial y$.

Scaling

`scale-eval` `auto, none, big, Big, bigg, Bigg` `auto`

This key sets the size of the delimiters used for the point of evaluation. This scaling is used whenever `switch-/=false` and the slash argument is absent. The key's default value is set to scale the delimiters automatically.

`scale-eval-/` `auto, none, big, Big, bigg, Bigg` `auto`

This key sets the size of the delimiters used for the point of evaluation. This scaling is used whenever `switch-/=false` and the slash argument is present. The key's default value is set to scale the delimiters automatically.

`scale-fun` `auto, none, big, Big, bigg, Bigg` `auto`

This key sets the size of the delimiters used around the function. The scaling is used whenever `misc-add-delims=fun` is used. The key's default value is set to scale the delimiters automatically.

`scale-var` `auto, none, big, Big, bigg, Bigg` `auto`

This key sets the size of the delimiters used around the variable. The scaling is used whenever `misc-add-delims=var` is used. The key's default value is set to scale the delimiters automatically.

`scale-frac` `auto, none, big, Big, bigg, Bigg` `auto`

This key sets the size of the delimiters used for around the fraction. This scaling is used whenever `switch-/=false`, the slash argument is absent and `misc-add-delims=frac` is used. The key's default value is set to scale the delimiters automatically.

`scale-frac-/` `auto, none, big, Big, bigg, Bigg` `auto`

This key sets the size of the delimiters used for around the fraction. This scaling is used whenever `switch-/=false`, the slash argument is present and `misc-add-delims=frac` is used. The key's default value is set to scale the delimiters automatically.

Delimiters

`delims-eval` ⟨*left delimiter*⟩⟨*right delimiter*⟩ `( )`

This key sets the left and right delimiters used to indicate the point of evaluation. These delimiters are inserted whenever `switch-/=false` and the slash argument is absent. The key's default left and right delimiters are a left and a right parenthesis, respectively.

`delims-eval-/` ⟨*left delimiter*⟩⟨*right delimiter*⟩ `( )`

This key sets the left and right delimiters used to indicate the point of evaluation. These delimiters are inserted whenever `switch-/=false` and the slash argument is present. The key's default left and right delimiters are a left and a right parenthesis, respectively.

`delims-fun` ⟨*left delimiter*⟩⟨*right delimiter*⟩ `( )`

This key sets the left and right delimiters used to around the function and these are inserted whenever `misc-add-delims=fun` is used. The key's default left and right delimiters are a left and a right parenthesis, respectively.

`delims-var` ⟨*left delimiter*⟩⟨*right delimiter*⟩ `( )`

This key sets the left and right delimiters used to around the variable and these are inserted whenever `misc-add-delims=var` is used. The key's default left and right delimiters are a left and a right parenthesis, respectively.

`delims-frac` ⟨*left delimiter*⟩⟨*right delimiter*⟩ `( )`

This key sets the left and right delimiters used to around the fraction in the derivative e.g. $\left(\frac{\partial}{\partial x}\right)y$. These delimiters are inserted whenever `switch-/=false`, the slash argument is absent and `misc-add-delims=frac` is used. The key's default left and right delimiters are a left and a right parenthesis, respectively.

`delims-frac-/` ⟨*left delimiter*⟩⟨*right delimiter*⟩ `( )`

This key sets the left and right delimiters used to around the fraction in the derivative e.g. $(\partial/\partial x)y$. These delimiters are inserted whenever `switch-/=false`, the slash argument is present and `misc-add-delims=frac` is used. The key's default left and right delimiters are a left and a right parenthesis, respectively.

Math spacing

The options in this subsection inserts extra horizontal math space. The below equation illustrates where the space is inserted

$$\frac{\partial\langle\texttt{inf-fun}\rangle f}{\partial\langle\texttt{inf-var}\rangle x} \qquad \frac{\partial^{\langle\texttt{inf-ord}\rangle 2}\langle\texttt{ord-fun}\rangle f}{\partial x\langle\texttt{var-inf}\rangle\partial y} \qquad \frac{\partial^3 f}{\partial x^{\langle\texttt{var-ord}\rangle 2}\langle\texttt{ord-inf}\rangle\partial y} \qquad \left(\frac{\partial f}{\partial x}\right)_{\langle\texttt{eval-sb}\rangle(x_1,y_1)}^{\langle\texttt{eval-sp}\rangle(x_2,y_2)}$$

where it have been split into four to give a better overview.

`sep-inf-ord` ⟨*cs-number*⟩ 0

This key sets the math space that is inserted in the infinitesimal's power left to the mixed order. It is only inserted when the mixed order is different from 1. The key's default value is 0 `mu`.

`sep-inf-fun` ⟨*cs-number*⟩ 0

This key sets the math space that is inserted between the infinitesimal and the function when the mixed order *is equal* to 1. The space is only inserted when a non-blank function is printed *in the numerator*. The key's default value is 0 `mu`.

`sep-ord-fun` ⟨*cs-number*⟩ 0

This key sets the math space that is inserted between the infinitesimal and the function when the mixed order *is different* from 1. The space is only inserted when a non-blank function is printed *in the numerator*. The key's default value is 0 `mu`.

`sep-inf-var` ⟨*cs-number*⟩ 0

This key sets the math space that is inserted between the infinitesimal and the following variable. The space is only inserted when a non-blank variable is given. The key's default value is 0 `mu`.

`sep-var-ord` ⟨*cs-number*⟩ 0

This key sets the math space that is inserted in a variable's power left to the order. The space is only inserted when the order is different from 1. The key's default value is 0 `mu`.

`sep-var-inf` ⟨*cs-number*⟩ 3

They key sets the math space that is inserted between a variable and the following infinitesimal. The space is only inserted when the variable's order *is equal* to 1. It is only inserted when more than one non-blank variables are given. The key's default value is 3 `mu`.

`sep-ord-inf` ⟨*cs-number*⟩ 3

They key sets the math space that is inserted between a variable and the following infinitesimal. The space is only inserted when the variable's order *is different* from

1. It is only inserted when more than one non-blank variables are given. The key's default value is 3 `mu`.

`sep-eval-sb` ⟨*cs-number*⟩ `-4`

This key sets the math space that is inserted in the evaluation subscript left to the point of evaluation. The space is only inserted when a non-blank subscript is given. The key's default value is -4 `mu`.

`sep-eval-sp` ⟨*cs-number*⟩ `-4`

This key sets the math space that is inserted in the evaluation superscript left to the point of evaluation. The space is only inserted when a non-blank superscript is given. The key's default value is -4 `mu`.

Switches

`switch-*` `true, false` `false`

The effect of the star's presence can be switched with the value britishtrue. That is, the function is typeset next to the fraction when the star is absent and in the numerator when the star is present. As an example, compare below where the option is turned on (britishtrue) and off (britishfalse),

```
\derivset{\pdv}[switch-/=false] \pdv{y}{x}
```
$\Longrightarrow \quad \dfrac{\partial y}{\partial x}$

```
\derivset{\pdv}[switch-/=true] \pdv{y}{x}
```
$\Longrightarrow \quad \partial y/\partial x$

The key's default value is false.

`switch-/` `true, false` `false`

The effect of the slash's presence can be switched with the value britishtrue. That is, the derivative is typeset with the fraction set by `style-frac-/` when the slash is absent and with the fraction set by `style-frac` when the slash is present. As an example, compare below where the option is turned on (britishtrue) and off (britishfalse),

```
\derivset{\pdv}[switch-/=false] \pdv{y}{x}
```
$\Longrightarrow \quad \dfrac{\partial y}{\partial x}$

```
\derivset{\pdv}[switch-/=true] \pdv{y}{x}
```
$\Longrightarrow \quad \partial y/\partial x$

The key's default value is false.

Miscellaneous

`misc-add-delims` `fun, var, frac` `false`

`misc-remove-delims`

These two keys accepts its input as an comma-separated list of values such that

```
\derivset{\pdv}[misc-add-delims=fun]
\derivset{\pdv}[misc-add-delims={fun,var}]
\derivset{\pdv}[misc-add-delims={fun,var,frac}]
```

are all valid, and the same goes for `misc-remove-delims`. The key `misc-add-delims` is used to insert the delimiters around the key's values, while `misc-remove-delims` is used to remove the inserted delimiters. The next example shows the key `misc-add-delims` in action

```
\derivset{\pdv}[misc-add-delims=fun] \pdv{y}{x}
```
$$\Longrightarrow \quad \frac{\partial(y)}{\partial x}$$

```
\derivset{\pdv}[misc-add-delims={fun,var}] \pdv{y}{x}
```
$$\Longrightarrow \quad \frac{\partial(y)}{\partial(x)}$$

```
\derivset{\pdv}[misc-add-delims={fun,var,frac}] \pdv{y}{x}
```
$$\Longrightarrow \quad \left(\frac{\partial(y)}{\partial(x)}\right)$$

All the above applies to `misc-remove-delims` as well, except it removes the delimiters. The effect of the key `misc-add-delims` is turned off (britishfalse) locally inside a derivative. This is to ensure that the option is not applied to more than one derivative when nested as shown below

```
\derivset{\pdv}[misc-add-delims=var] \pdv*{\pdv{y}{x}}{x}
```
$$\Longrightarrow \quad \frac{\partial}{\partial(x)}\frac{\partial y}{\partial x}$$

These two keys and this local behaviour are subject to change, see change 8.2(i) and consideration 8.3(v) for more information.

Sort

The keys given in this subsection will be briefly described here and a more in-depth description will be given in section 6.

`sort-method` `abs, sign, symbol` `sign, symbol, abs`

This key sets the sorting method behind the mixed order using build in algorithms. The symbolic terms in the mixed order is sorted by sign, symbolic length and absolute value by the britishsign, britishsymbol and britishabs algorithms, respectively. The key takes its input as a comma-separated list of values, accepting up to three values e.g.

```
\derivset{\pdv}[sort-method=sign] \pdv[c+kn,-b+2a]{f}{x,y}
```
$$\Longrightarrow \quad \frac{\partial^{c+kn+2a-b} f}{\partial x^{c+kn}\,\partial y^{-b+2a}}$$

```
\derivset{\pdv}[sort-method={sign,symbol}] \pdv[c+kn,-b+2a]{f}{x,y}
```
$$\Longrightarrow \quad \frac{\partial^{kn+c+2a-b} f}{\partial x^{c+kn}\,\partial y^{-b+2a}}$$

```
\derivset{\pdv}[sort-method={sign,symbol,abs}] \pdv[c+kn,-b+2a]{f}{x,y}
```
$$\Longrightarrow \quad \frac{\partial^{kn+2a+c-b} f}{\partial x^{c+kn}\,\partial y^{-b+2a}}$$

are valid inputs. Note how the terms in the mixed order are ordered using different sorting methods. This key is further described in section 6.1. The key's default value uses all three algorithms as sign, symbol, abs.

`sort-numerical` `auto, first, last` `auto`

This key determines where the numerical term[3] is placed in the mixed order. Using the values britishfirst and britishlast, then the numerical term will *always* be placed as the first and last term in the mixed order, respectively. While the value britishauto will automatically determine where to place the numerical term depending on the sign of the first symbolic term; it is placed as the first and last term if the sign is negative and positive, respectively. This is shown below

```
\derivset{\pdv}[sort-numerical=first] \pdv[n,2]{f}{x,y}
```

$$\Longrightarrow \qquad \frac{\partial^{2+n} f}{\partial x^n \, \partial y^2}$$

```
\derivset{\pdv}[sort-numerical=last] \pdv[-n,2]{f}{x,y}
```

$$\Longrightarrow \qquad \frac{\partial^{-n+2} f}{\partial x^{-n} \, \partial y^2}$$

where the value britishauto would have placed the numerical term opposite in both equations. The key is further described in section 6.2, and its default value is auto.

`sort-sign-reverse` `true, false` `false`

The sorting algorithm britishsign separates the positive and negative terms where the former is placed first and the latter last. The value britishtrue will reverse this ordering such that the negative terms is placed first and the positive terms last. See section 6.3 for more information.

`sort-symbol-reverse` `true, false` `false`

The sorting algorithm britishsymbol separates the terms by their symbolic length, where the terms are ordered in a descending manner according to their length. If the value britishtrue is used, then the terms will ordered in a ascending manner instead. See section 6.3 for more information.

`sort-abs-reverse` `true, false` `false`

The sorting algorithm britishabs separates the terms by their absolute value, where the terms are ordered in a descending manner. If the value britishtrue is used, then the terms will ordered in a ascending manner instead. See section 6.3 for more information.

4.4 All derivatives

The options in this subsection are applied to all of the derivatives that are defined by the package and you, because some options should be consisting regardless of the derivatives. The options are accessed using `\derivset{all}[`⟨*key*=*value*⟩`]`.

`scale-auto` `leftright, mleftmright`[R] `leftright`

[3] The numerical term is the sum of all the orders that does not contain any symbols, but solely consist of numbers.

This package uses `\left` and `\right` to automatically scale delimiters. The value britishmleftmright requires the mleftright package, which makes this package use `\mleft` and `\mright` instead.

5 Defining variants

This section goes into detail with how to define variants of the ordinary and partial derivative based on the package's internal commands, as mentioned earlier. The derivative package provides a LaTeX 2_ε way of defining the derivatives. Thought it is preferable to define variants of the ordinary derivative over the partial derivative, unless one needs more than one variable of differentiation and access to the mixed order.

5.1 Variants of the ordinary derivative

`\NewOdvVariant`
`\RenewOdvVariant`
`\ProvideOdvVariant`
`\DeclareOdvVariant`

⟨*control-sequence*⟩`{`⟨*infinitesimal*⟩`}[`⟨*key=value*⟩`]`

This family of commands are used to define a variant of the ordinary derivative with the macro name ⟨*control-sequence*⟩. Moreover, the new derivative will use `{`⟨*infinitesimal*⟩`}` as its infinitesimal and inherit the package's default settings given in section 4.2, but they may be overwritten with `[`⟨*key=value*⟩`]`. The difference between them, is as follows:

- `\NewOdvVariant` is similar to `\NewDocumentCommand` of the xparse package and will issue an error if ⟨*control-sequence*⟩ has already been defined.
- `\RenewOdvVariant` is similar to `\RenewDocumentCommand` of the xparse package and will issue an error if ⟨*control-sequence*⟩ has not previously been defined.
- `\ProvideOdvVariant` is similar to `\ProvideDocumentCommand` of the xparse package and will define ⟨*control-sequence*⟩ if it does not have an existing definition. It will not issue any errors.
- `\DeclareOdvVariant` is similar to `\DeclareDocumentCommand` of the xparse package and will always define the ⟨*control-sequence*⟩ with the new definition regardless of whether it already exists.

⟨*control-sequence*⟩ The first argument is the macro name of the derivative that is to be defined.

`{`⟨*infinitesimal*⟩`}` While the second argument makes the derivative ⟨*control-sequence*⟩ use `{`⟨*infinitesimal*⟩`}` as the infinitesimal, like 'd' is used as the infinitesimal in the ordinary derivative `\odv`.

`[`⟨*key=value*⟩`]` The optional argument accepts its input as a comma-separated list of *key=value* pairs, which overrides the package's default options for the keys given. Leaving out this argument, then the derivative will use the package's default settings.

5.2 Variants of the partial derivative

`\NewPdvVariant`
`\RenewPdvVariant`
`\ProvidePdvVariant`
`\DeclarePdvVariant`

⟨*control-sequence*⟩`{`⟨*infinitesimal*⟩`}[`⟨*key=value*⟩`]`

This family of commands are used to define a variant of the partial derivative with the macro name ⟨*control-sequence*⟩. Moreover, the new derivative will use `{`⟨*infinitesimal*⟩`}` as its infinitesimal and inherit the package's default options given in section 4.3, but they may be overwritten with `[`⟨*key=value*⟩`]`. The difference between them is as follows:

- `\NewPdvVariant` is similar to `\NewDocumentCommand` of the xparse package and will issue an error if ⟨*control-sequence*⟩ has already been defined.

- `\RenewPdvVariant` is similar to `\RenewDocumentCommand` of the `xparse` package and will issue an error if ⟨*control-sequence*⟩ has not previously been defined.
- `\ProvidePdvVariant` is similar to `\ProvideDocumentCommand` of the `xparse` package and will define ⟨*control-sequence*⟩ if it does not have an existing definition. It will not issue any errors.
- `\DeclarePdvVariant` is similar to `\DeclareDocumentCommand` of the `xparse` package and will always define the ⟨*control-sequence*⟩ with the new definition regardless of whether it already exists.

⟨*control-sequence*⟩ The first argument is the macro name of the derivative that is to be defined.

`{`⟨*infinitesimal*⟩`}` While the second argument makes the derivative ⟨*control-sequence*⟩ use `{`⟨*infinitesimal*⟩`}` as the infinitesimal, like '∂' is used as the infinitesimal in the partial derivative `\pdv`.

`[`⟨*key=value*⟩`]` The optional argument accepts its input as a comma-separated list of *key=value* pairs, which overrides the package's default options for the keys given. Leaving out this argument, then the derivative will use the package's default settings.

6 The mixed order

6.1 Sorting algorithms

A unique feature of this package is that the sorting method behind the mixed order may be changed using built-in algorithms and there are currently three algorithms available: britishsign, britishsymbol and britishabs which are explained below. The algorithms have been chosen such that it the terms in the mixed order may be arranged in many ways with as few algorithms as possible to suit your liking. A sorting method may consist of up to three algorithms which are applied in layers.

- The sorting algorithm britishsign separates the positive and negative terms, putting the former terms first and the latter terms last in the mixed order. Using the option `sort-sign-reverse=true`, then this ordering is reversed such that the negative terms appear before the positive terms.
- The sorting algorithm britishsymbol separates the terms by their symbolic length, where the terms are ordered in a descending manner according to their length. Using the option `sort-symbol-reverse=true`, then the terms are ordered in a ascending manner instead.
- The sorting by britishabs separates the terms by their absolute value, where the terms are ordered in a descending manner according to their absolute value. Using the option `sort-abs-reverse=true`, then the terms are ordered in an ascending manner instead.

When using the option `sort-method={...}`, the number of algorithms determines the number of layers e.g. `sort-method=sign,symbol` is a sorting method with two layers, where the algorithms britishsign and britishsymbol are applied in layer 1 and layer 2, respectively. It should be understood as that the britishsign algorithm splits the terms into two groups, one with positive terms and one with negative terms. Then the britishsymbol algorithm will order the terms within each groups according to the terms symbolic length. The mixed order is then formed by combining these two groups such that the positive terms comes first.

The next subsubsection is dedicated to give some examples of how the algorithms in this package sorts the mixed order.

6.1.1 Examples

The examples are constructed of a partial derivative with the comma separated list of orders given by `[3a-3hh-2b, 4c+4gg+2ff, -5d-5ee]` and a close up view of the mixed order. Square brackets are used to indicate grouping of terms and the text below a group refers to the algorithm applied. Here the words positive and negative, long and short, and big and low refers to the britishsign, britishsymbol, and britishabs algorithms, respectively. A row of square brackets represents a layer.

The packages default sorting method is `sort-method=sign,symbol,abs`, which is used below. The britishsign algorithm separate the positive and negative terms in the first layer. While in the second layer, the terms are separated by symbol length etc.

$$\frac{\partial^{4gg+2ff+4c+3a-5ee-3hh-5d-2b} f}{\partial x^{3a-3hh-2b}\,\partial y^{4c+4gg+2ff}\,\partial z^{-5d-5ee}}$$

$$\underbrace{\underbrace{\underbrace{4gg}_{\text{big}}+\underbrace{2ff}_{\text{low}}}_{\text{long}}+\underbrace{\underbrace{4c}_{\text{big}}+\underbrace{3a}_{\text{low}}}_{\text{short}}}_{\text{positve}}-\underbrace{\underbrace{\underbrace{5ee}_{\text{big}}-\underbrace{3hh}_{\text{low}}}_{\text{long}}-\underbrace{\underbrace{5d}_{\text{big}}-\underbrace{2b}_{\text{low}}}_{\text{short}}}_{\text{negative}}$$

as expected. Interchanging britishsign and britishsymbol algorithms from the previous example i.e `sort-method=symbol, sign, abs`, then the britishsymbol sign algorithms are now applied in the first and second layer, respectively. The result is shown below

$$\frac{\partial^{4gg+2ff-5ee-3hh+4c+3a-5d-2b} f}{\partial x^{3a-3hh-2b}\,\partial y^{4c+4gg+2ff}\,\partial z^{-5d-5ee}}$$

$$\underbrace{\underbrace{\underbrace{4gg}_{\text{big}}+\underbrace{2ff}_{\text{low}}}_{\text{positive}}-\underbrace{\underbrace{\underbrace{5ee}_{\text{big}}-\underbrace{3hh}_{\text{low}}}_{\text{negative}}}_{\text{long}}+\underbrace{\underbrace{\underbrace{4c}_{\text{big}}+\underbrace{3a}_{\text{low}}}_{\text{positive}}-\underbrace{\underbrace{5d}_{\text{big}}-\underbrace{2b}_{\text{low}}}_{\text{negative}}}_{\text{short}}$$

where a clear difference is seen between the above two examples.

The above two examples used 3-layer sorting methods but as mentioned earlier, the sorting method can also be constructed with one and two algorithms i.e. 1-layer and 2-layer sorting methods. A sorting method with two layers could be constructed as `sort-method=sign,symbol` and with it, the terms are ordered by sign and then symbolic length. In this case, there is not a third layer. Instead, the terms 'appear randomly' according to their absolute value as shown below

$$\frac{\partial^{4gg+2ff+3a+4c-3hh-5ee-2b-5d} f}{\partial x^{3a-3hh-2b}\,\partial y^{4c+4gg+2ff}\,\partial z^{-5d-5ee}}$$

$$\underbrace{\underbrace{\underbrace{4gg}_{\text{long}}+\underbrace{2ff}_{\text{short}}}_{\text{positive}}+\underbrace{\underbrace{\underbrace{3a}_{\text{long}}+\underbrace{4c}_{\text{short}}}_{\text{positive}}-\underbrace{\underbrace{\underbrace{3hh}_{\text{long}}-\underbrace{5ee}_{\text{short}}}_{\text{negative}}}_{\text{negative}}-\underbrace{\underbrace{2b}_{\text{long}}-\underbrace{5d}_{\text{short}}}_{\text{negative}}}$$

It is seen that the first two terms came out in a descending manner according to their absolute value, while the next two terms came out in a ascending manner. Hence the choice of word: 'appear randomly'. The last example shows a 1-layer sorting method given as `sort-method=symbol` which gives the result

$$\frac{\partial^{-3hh+4gg+2ff-5ee+3a-2b+4c-5d} f}{\partial x^{3a-3hh-2b}\,\partial y^{4c+4gg+2ff}\,\partial z^{-5d-5ee}}$$

$$\underbrace{-3hh+4gg+2ff-5ee}_{\text{long}}+\underbrace{3a-2b+4c-5d}_{\text{short}}$$

where it can be see that the terms are not ordered by sign nor absolute value. A fourth sorting algorithm might be implemented in the future, see consideration 8.3(viii) for more information.

6.2 The numerical term

With the symbolic part of the mixed order treated, it is time to look at the numerical term[3]. For this reason it is treated differently than the symbolic terms. Currently it is only possible to control its position in the mixed order.

The numerical term may be placed either at the beginning or at the end of the mixed order with the values britishfirst and britishlast, respectively. Thought it is also possible to have the package automatically determine where to place the term with the value britishauto. Compare below

`sort-numerical=auto`:	$\frac{\partial^{\beta+2} f}{\partial x^{\beta}\,\partial y^{2}}$	$\frac{\partial^{\beta-2} f}{\partial x^{\beta}\,\partial y^{-2}}$	$\frac{\partial^{2-\beta} f}{\partial x^{-\beta}\,\partial y^{2}}$	$\frac{\partial^{-2-\beta} f}{\partial x^{-\beta}\,\partial y^{-2}}$
`sort-numerical=first`:	$\frac{\partial^{2+\beta} f}{\partial x^{\beta}\,\partial y^{2}}$	$\frac{\partial^{-2+\beta} f}{\partial x^{\beta}\,\partial y^{-2}}$	$\frac{\partial^{2-\beta} f}{\partial x^{-\beta}\,\partial y^{2}}$	$\frac{\partial^{-2-\beta} f}{\partial x^{-\beta}\,\partial y^{-2}}$
`sort-numerical=last`:	$\frac{\partial^{\beta+2} f}{\partial x^{\beta}\,\partial y^{2}}$	$\frac{\partial^{\beta-2} f}{\partial x^{\beta}\,\partial y^{-2}}$	$\frac{\partial^{-\beta+2} f}{\partial x^{-\beta}\,\partial y^{2}}$	$\frac{\partial^{-\beta-2} f}{\partial x^{-\beta}\,\partial y^{-2}}$

These three options should satisfy every need, but there might come more options, see consideration 8.3(vii).

6.3 Reversing the sort algorithm

The three reverse keys serves to reverse the sorting order of the sorting algorithms. This allows for even greater flexibility over the sorting method by 'doubling' the number of algorithms. So if the default ordering of an algorithm is not wished it may be reversed with the corresponding reverse key. For example, the britishsign algorithm places the positive terms first and the negative terms last, but using the option `sort-sign-reverse=true` will place the negative terms first and positive terms last. The reverse keys in action are shown below

`sort-sign-reverse=false`:	$\frac{\partial^{a-b} f}{\partial x^{a}\,\partial y^{-b}}$	`sort-sign-reverse=true`:	$\frac{\partial^{-b+a} f}{\partial x^{a}\,\partial y^{-b}}$
`sort-symbol-reverse=false`:	$\frac{\partial^{ab+c} f}{\partial x^{ab}\,\partial y^{c}}$	`sort-symbol-reverse=true`:	$\frac{\partial^{c+ab} f}{\partial x^{ab}\,\partial y^{c}}$
`sort-abs-reverse=false`:	$\frac{\partial^{2a+b} f}{\partial x^{2a}\,\partial y^{b}}$	`sort-abs-reverse=true`:	$\frac{\partial^{b+2a} f}{\partial x^{2a}\,\partial y^{b}}$

7 Miscellaneous

7.1 Slashfrac

`\slashfrac` `[`⟨*scale*⟩`]{`⟨*numerator*⟩`}{`⟨*denominator*⟩`}`

A text-styled fraction i.e. a/b, is commonly used in text-mode. While `\slashfrac{a}{b}` is certainly longer to write than `a/b`, which only requires three keystrokes, but for the implementation of text-styled derivatives, a macro is needed anyway.

`[`⟨*scale*⟩`]` The optional argument is the scaling parameter for the slash and takes the inputs as shown below. Leaving out `[`⟨*scale*⟩`]` sets the scaling parameter to britishauto

`\slashfrac[auto]{y_f}{x}`	$\Longrightarrow$	y_f/x
`\slashfrac[none]{y_f}{x}`	$\Longrightarrow$	y_f/x
`\slashfrac[big]{y_f}{x}`	$\Longrightarrow$	$y_f\big/x$
`\slashfrac[Big]{y_f}{x}`	$\Longrightarrow$	$y_f\Big/x$
`\slashfrac[bigg]{y_f}{x}`	$\Longrightarrow$	$y_f\bigg/x$
`\slashfrac[Bigg]{y_f}{x}`	$\Longrightarrow$	$y_f\Bigg/x$

This argument is subject to change, see considerations 8.3(ix) and 8.3(x) for more information.

`{`⟨*numerator*⟩`}` This argument typeset the fraction's numerator.

`{`⟨*denominator*⟩`}` This argument typeset the fraction's denominator.

8 To do

Entries marked with ✓ will be implemented in a future release specified in the entry. Entries marked with ✗ will not be implemented.

8.1 Future implementation

The list describes what *will be added* to the package in a later release.

(i) Define commands that can create variants of infinitesimal like d, ∂, δ etc. for use in integrals $\int f(x)\,\mathrm{d}x$, in differential equations $\frac{\partial f}{\partial x}\,\mathrm{d}x + \frac{\partial f}{\partial y}\,\mathrm{d}y$ and a shorthand notation of the partial derivative $\partial_{xx} f(x, y)$.

(ii) Implement the option to turn off the sorting of the terms in the mixed order. This could be made as a britishdraft / britishfinal option.

8.2 Future changes

The list describes what *will be changed* in the package in a later release.

(i) A complete rewrite of the options `misc-add-delims` and `misc-remove-delims` is necessary so that they have a clearer syntax. Moreover, the value britishvar needs to work differently (and correctly) for the partial derivatives and variants hereof (pretty much useless here).

8.3 Future considerations

The list describes what *might be changed or added* to the package in a later release.

(i) As mentioned earlier in the documentation, I heavily consider to change the optional argument `[`⟨*order*⟩`]` to a `[`⟨*key=value*⟩`]` argument. Of course this will lead to some changes:

- I would need a way to set the order of differentiation. I am thinking of something like a key value option; `order={`⟨*cs-orders*⟩`}`.
- Such an option should not be possible to be set with `\derivset` or in the definition of a new derivative (or maybe if one really needs a lot of higher order derivative).

✓**Conclusion:** *I truly believe this is the right way forward and is coming in the next release. More features can easily be added in a key-value style. The order argument will be replaced with a key-value argument and a order key introduced;* `ord=`⟨*order*⟩.

(ii) With the above consideration, than it would only be natural to remove the mixed order and then have an option `mixed-order={`⟨*mixed-order*⟩`}`. Likewise, it should not be possible to be set with `\derivset` or in the definition of a new derivative.

✓**Conclusion:** *The mixed order argument will be removed due to the introduction of a key-value argument as mentioned above in the next release. The mixed order can be set with the knew key* `m-ord=`⟨*mixed-order*⟩.

(iii) Currently `\derivset`'s optional argument `[`⟨*key=val*⟩`]`, when not given, sets the derivatives options to the package default settings. This should probably be changed to the options set in the definition of the derivative.
✓**Conclusion:** *I believe this would a nice change for the future, but I do not have the time to make the implementation for the next release.*

(iv) It can become rather cumbersome to read a derivative with a very long function. To ease this up, I am considering to add a switch that exchange the variable and the function arguments. But this might be against the LaTeX way of using arguments, which is the reason it have not been done and I would like feedback on it.

(v) Should I add a key, possible a switch, that turn on/off the local behaviour `misc-add-delims` and `misc-remove-delims` inside derivative?

(vi) It was mentioned in section 5, that the second argument of `\…Variant` should be a single token, e.g. meaning that `\NewOdvVariant{\myvariant}{\partial_\mu}` is not allowed but `\NewOdvVariant{\myvariant}{{\partial_\mu}}` is. It can easily be change to accept a multi token input.
✓**Conclusion:** *This have already been changed in version 0.95, since it is an unnecessary constraint.*

(vii) Should I add the option to `sort-numerical`, so that it is treated as a symbolic term instead of a numerical term?
✓**Conclusion:** *Yes. Yes I should. This feature will be added in the next release.*

(viii) Should I add a sorting algorithm that order the terms according to their number (including sign)?
✓**Conclusion:** *Yes. Yes I should. This feature will be added in the next release.*

(ix) I am considering making the optional argument of `\slashfrac` into a `[`⟨*key=value*⟩`]` argument with something like `scale={`⟨*scaling*⟩`}`.
✓**Conclusion:** *I want an key-value implementation like this in the future, but it will not be added it the next release.*

(x) Should the default scaling for `\slashfrac` be britishauto or britishnone? Or should it be possible to set it with something like `\slashfracset[`⟨*key=value*⟩`]`?
✗**Conclusion:** *The package default should be britishauto, but it will be possible to change it, when the key-value is implemented or* `\slashfrac`.

Index

Numbers in bold refer to the page where the entry is defined.

Index of Options

Index of Commands

Change history

v0.9
2019-07-21
Beta

- First release of the package. The package is currently in a beta version.

v0.95
2019-09-18
Beta

- *Please ignore this version, since it contained the wrong* .sty *and* .pdf *files* :(.
- Removed the single token restriction of the infinitesimal since it made no sense.
- Fixed the documentation errors and typos.
- Minor fixes to the code.

v0.95b
2019-09-21
Beta

- Contains the correct .sty and .pdf files :).
- One minor code fix.

v0.96
2019-12-22
Beta

- Fixed the double superscript issue for higher order derivative when the variable contained a superscript.

v0.97
2020-02-03
Beta

- Fixed the argument specifier of `\__deriv_scale_big:nnnn` when it was used (it was used with `:nnnm`).