# `diffcoeff`

a LaTeX package to ease the writing of differential coefficients

Version 5.6

Andrew Parsloe
(ajparsloe@gmail.com)

December 19, 2025

**Abstract**

`diffcoeff` is a LaTeX package to ease the writing of ordinary, partial and other derivatives of arbitrary algebraic or numeric order. For mixed partial derivatives, the total order of differentiation is calculated by the package. Optional arguments allow for points of evaluation (ordinary derivatives), or variables held constant (partial derivatives), and the placement of the differentiand in the numerator or appended. Besides $\frac{\mathrm{d}y}{\mathrm{d}x}$, forms like ${}^{\mathrm{d}y}/_{\mathrm{d}x}$, $\mathrm{d}y/\mathrm{d}x$ and $\partial_x y$ are also available, as well as derivatives built from $D$, $\Delta$, $\delta$, and configurable jacobians and differentials. Other notations like line elements ($\mathrm{d}x^2 + \mathrm{d}y^2 + \mathrm{d}z^2$) and bra-kets are easily produced.

### Version 5.6

Version 5.6 resolves a conflict that arises with version 5.5 and packages still using `xtemplate`.[a] Otherwise the versions are the same.

### Version 5.5

Since 2024/06/01, the functionality of `xtemplate` has been incorporated into the LaTeX kernel. Version 5.5 of `diffcoeff` does not require or load `xtemplate` unless your LaTeX system is earlier than 2024/06/01. In addition to some corrections and additions to documentation, new features of version 5.5 are

- the ability to write a generic mixed partial derivative $\partial^n F/\partial x_1 \dots \partial x_n$ 'with dots' in the straightforward way (§2.2.5.5);
- the ability to write a 'split level' inline derivative like $\mathrm{d}y/\mathrm{d}x$ (or the example in the previous item); see §3.3.4.6;
- the ability to write a 'doubly compact' derivative like $\partial_{x^2yz^3} F$; see §3.3.4.8.

#### For users of version 4

Version 5 of `diffcoeff` more fully embraces the configurability offered by templates than earlier versions, at the cost of some incompatibilities with version 4. That version is still available with the command `\usepackage[<options>]{diffcoeff}[=v4]`. In version 5, the `\diff` and `\diffp` commands of version 4 remain, but lack the 'spacing switch' `!` and 'slash switch' `/`. Slash-fraction derivatives are now created with the `\difs` and `\difsp` commands. New commands `\difc` and `\difcp` produce derivatives in 'compact notation' like $\mathrm{d}_x y$ and $\partial_x y$. The two-argument `\diffdef` command of earlier versions has been replaced by the three-argument command `\difdef`, the additional argument determining which of the `f`, `s`, `c`, `fp`, `sp` or `cp` forms the defined variant applies to. The differential command `\dl` has been rewritten and is now fully template-configurable (allowing easy writing of line elements like $\mathrm{d}x^2 + \mathrm{d}y^2 + \mathrm{d}z^2$), and the jacobian command `\jacob` is also configurable.

#### ISO defaults

The `ISO` package option is redundant. Version 5 of `diffcoeff` follows ISO recommendations (see the standard ISO 80000-2). In particular, this means upright 'd's, and subscripted parentheses enclosing a derivative to indicate a point of evaluation. This document is written with those defaults. But the defaults can be readily changed; see §§3.3, 3.3.2.

[a] My thanks to Jensen Tan for pointing out the conflict.

# Contents



[1] I thank Hans Schülein for first raising this issue with me and for subsequent thoughtful comments.

[2] I thank Christophe Bal for this suggestion.

# Chapter 1

# Introduction

### Requirements

A TEX distribution from or later than February 2020 is assumed. `diffcoeff` uses the `xtemplate` functionality of the LATEX3 project. Since June 2024 that has become part of the LATEX base – see `lttemplates-doc.pdf` which can be found in the `doc\latex\base` or `doc/latex/base` folder (depending on your operating system) of your TEX distribution. Consequently from version 5.5 of `diffcoeff`, `xtemplate` is a required package only if your TEX distribution is earlier than June 2024. The package `mleftright` is required (but is loaded by `diffcoeff`), and for some optional features (`style=tfrac` and `style=sfrac`) that use the `\tfrac` and `\text` commands of `amsmath`, you will need to load that package (which in any case is likely to have already been done).

### Version 4

Version 5 of `diffcoeff` marked a significant change from previous versions. If you want the behaviour of version 4, add to the `\usepackage` command a trailing optional argument like this,

```
\usepackage[<options>]{diffcoeff}[=v4]
```

(with no space after the '=' sign!). Working with version 4 is described in the document `diffcoeff4.pdf`.

## 1.1 Package options

The package is loaded in the usual way by entering

```
\usepackage[<options>]{diffcoeff}
```

in the preamble of your document.[1]

[1] Angle brackets indicate possible user input (*without* the angle brackets).

From version 5.5 of `diffcoeff`, there are five possible package options. If none are being used, omit the square-bracket term in the `\usepackage` call, otherwise the options are entered in a comma list in the square-bracket argument of the `\usepackage` command.

1. The `spaced` option takes one of three values:

   (a) `spaced=1` inserts a small space before the differentiand; this is the default so that entering `spaced` alone is equivalent to `spaced=1`;

   (b) `spaced=0` inserts no space before the differentiand; `diffcoeff` is initialized to this value so that if the `spaced` option is not used `spaced=0` is assumed;

   (c) `spaced=-1` inserts no space before the differentiand if it is a single token but otherwise inserts a small space; the present document uses `spaced=-1`;

   Space before the differentiand is discussed further in §2.2.1, where the commands `\spacedone`, `\spacednil` and `\spacedneg` are introduced, enabling local (within group) deviations from the global setting.

2. The second package option `def-file` takes as value the `<filename>` (which may include the path) of a file containing definitions of variant forms of derivative (see §3.3). The file has extension `.def`, `<filename>.def`; this is discussed in §3.3.3. If `<filename>` includes the path, any backslashes `\` (as used in windows systems) must be changed to forward slashes `/` (as in linux systems) to avoid 'Undefined control sequence' errors when compiling. (Because of the presence of earlier versions of `diffcoeff5.def` on the author's computer, the present document includes the path in the package option call: `def-file=<path to>/diffcoeff5`.)

3. The third package option `dif**` (from version 5.5 of `diffcoeff`) takes as value a comma list of identifiers drawn from `f`, `fp`, `s`, `sp`, `c`, `cp` (e.g. `dif**={c,cp}`). For the corresponding derivative commands (e.g. `\difc`, `\difcp`) specified by the identifiers the order of entry of the differentiand and differentiation variable(s) arguments is reversed; see §2.2.3.2. The present document does not use this package option.

4. By entering `mleftright` (no value required) in the options list, the command `\mleftright` is automatically inserted in the preamble. The effect is to change all occurrences of `\left`, `\right` in the document to `\mleft`, `\mright` so that the spacing around scalable delimiters modified by `\left`, `\right` is reduced; see §1.1.1 immediately below. The present document does not use this option.

5. The final package option `DIF` is a comma list of *key=value* statements amending the built-in defaults for the 'grandparent' template `DIF`; see §3.3.5. The present document does not use this option.

For the present document, the call is

```
\usepackage[def-file=<path to>/diffcoeff5,spaced=-1]{diffcoeff}
```

### 1.1.1 mleftright

To see the effect of the `mleftright` option, consider the expression

```
\[ \ln \left(\frac xy\right),\quad\sin\left(x^2\right). \] ⟹
```

$$\ln\left(\frac{x}{y}\right), \quad \sin\left(x^2\right).$$

in which there is significant whitespace before and after the parentheses. The package `mleftright` enables this whitespace to be reduced by using `\mleft`, `\mright` in place of `\left`, `\right`:

```
\[ \ln \mleft(\frac xy\mright),\quad\sin\mleft(x^2\mright). \]
                                ⟹
```

$$\ln\left(\frac{x}{y}\right), \quad \sin\left(x^2\right).$$

If you put `\mleftright` in the preamble, which is what the `mleftright` package option does, then all occurrences of `\left`, `\right` in the document will be affected. `\left`, `\right` can be restored to their normal behaviour by the command `\mleftrightrestore`. Rather than use `\mleft`, `\mright` explicitly, as in the example, the same effect can be obtained by using `\left`, `\right` and preceding the expression with the command `\mleftright`:

```
\mleftright
\[ \ln \left(\frac xy\right),\quad\sin \left(x^2\right). \]
\mleftrightrestore
```

⟹

$$\ln\left(\frac{x}{y}\right), \quad \sin\left(x^2\right).$$

## 1.2 A Rogues' Gallery of derivatives

Browsing through some (rather old) calculus textbooks, and texts on statistical mechanics, relativity and classical mechanics, I find the following choice examples of derivatives 'disporting every which way'.

- Upright fraction, slash fraction and compact form ordinary and partial derivatives,

$$\frac{\mathrm{d}^2y}{\mathrm{d}x^2}, \quad \mathrm{d}y/\mathrm{d}x, \quad \partial_x y.$$

- Multi-character variables of differentiation, un-parenthesized:

$$\frac{\partial\, \frac{\psi}{\Theta}}{\partial \frac{1}{\Theta}}, \quad \frac{\partial\, E/T}{\partial 1/T}, \quad \frac{\mathrm{d} \ln f}{\mathrm{d} \ln x_0}, \quad \frac{\partial^2 \psi}{\partial a_i\, \partial \frac{1}{\Theta}}, \quad \frac{\partial \mathcal{L}}{\partial \eta_{,i}^{(r)}}.$$

- Multi-character variables of differentiation parenthesized (or sometimes not) in *higher-order* derivatives, where the parentheses do not (or sometimes do) include the operator:

$$\frac{\partial^2 q}{\partial \left(\frac{1}{\Theta}\right)^2}, \quad \frac{\partial^2 q}{\partial (1/\Theta)^2}, \quad \frac{\partial^2 \varepsilon}{\partial {a_i}^2}, \quad \frac{\mathrm{d}^2\, \phi^i(x^i)}{(\mathrm{d}x^i)^2}.$$

  Should the d or $\partial$ be included within the parentheses, as in the last of these, or not, as in the first two? Logic says 'yes'; practice suggests (generally) 'no'; see §2.2.6 and §3.3.4.4.

- Indicating a point of evaluation is similarly varied (see §2.2.4, §3.3.4.1):

$$\left.\frac{\partial \phi}{\partial \varepsilon}\right|_{\varepsilon=\varepsilon_0}, \quad \left.\frac{\mathrm{d}^2 \phi}{\mathrm{d}\varepsilon^2}\right|_{\varepsilon=\varepsilon_0}, \quad \left[\frac{\partial\, b^\beta}{\partial a^\alpha}\right]_{b=0}, \quad \left(\frac{\mathrm{d}u}{\mathrm{d}v}\right)_{v=0}.$$

  ISO 80000-2 (item 2.11.13) favours the last of these – parentheses – for ordinary derivatives. Presumably, partial derivatives should follow suit, although parentheses are also used to indicate variables held constant:

$$\left(\frac{\partial}{\partial U}\, \frac{P}{T}\right)_V, \quad \left(\frac{\partial S}{\partial N_2}\right)_{U,V,N_1}, \quad (\partial S/\partial T)_V\,.$$

- We might want to write a generic mixed partial derivative (see §2.2.5.5),

$$\frac{\partial^n F}{\partial x_1\, \partial x_2 \cdots \partial x_n}, \quad \frac{\partial^{k_1+\cdots+k_n} F}{\partial {x_1}^{k_1} \cdots \partial {x_n}^{k_n}},$$

  or use a compact abbreviated notation like $\partial_{ij} F$ for $\partial^2 F/\partial x_i\, \partial x_j$, or $\partial_{x^3yz^2} F$ for $\partial^6 F/\partial x^3 \partial y\, \partial z^2$.

- Other symbols besides d and $\partial$ are used to denote derivative-like quantities. From introductory calculus and from classical mechanics and thermodynamics come $\delta$ and $\Delta$, from fluid mechanics comes D:

$$\frac{\delta y}{\delta x}, \quad \frac{\delta \mathcal{L}}{\delta \eta^{(r)}}, \quad \frac{D\rho}{Dt}, \quad \left(\frac{\Delta U}{\Delta T}\right)_V, \quad \Delta U/\Delta T.$$

  See §3.3.4.9 for these and for the $\mathbf{D}_x$ operator used sometimes in the discussion of differential equations:

$$\mathbf{D}_x^2 y + 2\mathbf{D}_x y - 4 = 0.$$

- There are those, like the International Organization for Standardization (ISO), who stipulate (or prefer) an upright d for their derivatives, and there are those (like the author, through sixty years of habit) who prefer a math-italic $d$ (see §3.3.4.2):

$$\frac{\mathrm{d}y}{\mathrm{d}x}, \qquad \frac{dy}{dx},$$

  and of course also in slash-fraction form $\mathrm{d}y/\mathrm{d}x$, $dy/dx$.

- When the differentiand is too big or awkward to sit in the numerator and is appended to the operator, the d or $\partial$ in the numerator is generally centred – but not always. In texts prior to the age of computerised typesetting one will sometimes find the symbol pushed to the *left*:

$$\frac{\partial}{\partial x^{l^*}}\frac{\partial x^{i^*}}{\partial x^{k^*}}, \quad \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{m\mathbf{q}_x}{\sqrt{1-q^2}}\right).$$

  The observant will note an italic adjustment with the first expression, so that the $\partial$ in the numerator and the $\partial$ in the denominator of the first term of the first example line up in a slanting column and the upright 'd's line up in the second example, rather than the operator symbols being centred as seems to be general modern practice.

- Then there is the case when the operator in the numerator differs from that in the denominator. For instance, in tensor calculus acceleration is sometimes written

$$\frac{\nabla v^i}{\mathrm{d}t} = \frac{\mathrm{d}^2x^i}{\mathrm{d}t^2} + \Gamma_{k\ h}^{\ i}\frac{\mathrm{d}x^h}{\mathrm{d}t}\frac{\mathrm{d}x^k}{\mathrm{d}t}$$

  where $\nabla v^i$ is the 'absolute differential' of the velocity $v^i$; see §3.3.4. (The $x^i$ are the coordinates.)

The `diffcoeff` package has the generative power to cope with all these variations – see §3.3 – although it is unlikely an author should need to call on this capacity to anything like the extent required for this Rogues' Gallery.

# Chapter 2

# Syntax and use

`diffcoeff` aims to ease the writing of derivatives (sometimes also called differential coefficients). There are long-established shorthands available in a few cases: $\dot{x}$ and $\ddot{x}$ for the time derivatives of a function $x$ of time $t$; $y'$ and $y''$ for the derivatives of a function $y$ (usually) of $x$. But mostly derivatives are expressed in fraction form and require more keystrokes to compose. It is here that `diffcoeff` is aimed. It uses three pairs of commands: `\diff` and `\diffp` to write (upright) fraction forms of ordinary and partial derivatives like

$$\frac{\mathrm{d}y}{\mathrm{d}x}, \quad \frac{\partial y}{\partial x},$$

generally intended for display-style environments; `\difs` and `\difsp` for slash-fraction forms of ordinary and partial derivatives like $\mathrm{d}y/\mathrm{d}x$, $\partial y/\partial x$, generally intended for text-style environments; and `\difc` and `\difcp` to write compact forms[1] of ordinary and partial derivatives like $\mathrm{d}_x y$ and $\partial_x y$.

**Note**

I refer throughout to the quantity or function being differentiated as the *differentiand* or *derivand* (in line with *integrand*, *operand*, etc.) and shall sometimes use `\difx` (resp. `\difxp`) to make general statements about any or all of `\diff`, `\difs` or `\difc` (resp.`\diffp`, `\difsp`, `\difcp`).

## 2.1 Syntax

All commands `\difx`, `\difxp`, share the same syntax. With all arguments present the syntax is

[1] Suggested by a question on TEX StackExchange: https://tex.stackexchange.com/questions/652223/write-a-derivative-operator-without-denominator-using-diffcoef/652298#652298

```
\difx.name.**[order-spec]<override>{variable(s)}
    {differentiand}[pt of eval]
```

The syntax is identical for `\difxp`. The eight arguments have the following meanings:

1. `name` (optional) A dot-delimited name to distinguish a variant form (non-default form) of derivative; not discussed further until §3 below, and specifically, §3.3.
2. `*` (optional) The presence of a star (asterisk) signals *append the differentiand*; its absence means the differentiand appears in the numerator of an upright- or slash-fraction form derivative; no effect for compact-form derivatives unless (see next) a second `*` is present; see §2.2.3.
3. `*` (optional) The presence of a *second* star signals that the argument specifying the variable(s) of differentiation comes *before* the argument specifying the differentiand; this is sometimes convenient when a complicated or lengthy differentiand is appended; see §2.2.3.2.
4. `order-spec` (optional) The order of differentiation when differentiating in a single variable, or a comma list of orders of differentiation for a mixed partial derivative; see §2.2.2 and §2.2.5. Not necessary if orders are specified with the colon notation in the variable list; see §§2.2.2.1 and 2.2.5.1.
5. `override` (optional) The total order of differentiation when it cannot be calculated by `diffcoeff` or is wanted in a different form from that calculated by `diffcoeff`; see §2.2.5.3.
6. `differentiand/derivand` (mandatory) The function being differentiated.
7. `variable(s)` (mandatory) The variable of differentiation or a comma list of variables of differentiation (for a mixed partial derivative); also, with the colon notation (see §§2.2.2.1, 2.2.5.1), provides an alternative method of specifying orders of differentiation.
8. `pt of eval` (optional) Point of evaluation or, for partial derivatives, variable or variables held constant; *no space* before the left square bracket; see §2.2.4.

Both mandatory arguments may be empty, but require empty brace pairs to indicate as much. (Omitting the differentiand makes sense for all forms of derivative, `\difx`, `\difxp`, but omitting the variable or variables of differentiation is sensible only for the compact forms, `\difc`, `\difcp` – see §3.3.4.8.)

## 2.2 General use

Writing `\diff{y}{x}` will produce $\frac{\mathrm{d}y}{\mathrm{d}x}$ in an inline math environment (i.e. placed between `\( \)` or `$ $`) or

$$\frac{\mathrm{d}y}{\mathrm{d}x}$$

in display style (placed, for instance, between `\[ \]` ). In fact `\diff yx` (omitting the braces) will produce these results, with a saving on keystrokes. The braces are needed only when a LaTeX argument – the variable of differentiation, or the differentiand – is multi-token:

`\[ \diff{f(x)}x \]` $\Longrightarrow$

$$\frac{\mathrm{d}\, f(x)}{\mathrm{d}x}$$

- The keen-eyed will note a space before the 'd' operator and the differentiand in this example. See §2.2.1 immediately below.
- Although conforming to the ISO standard (ISO 80000-2) the upright 'd's are not everyone's preference. If you want math-italic 'd's as default, see §3.3.2 on changing default settings.

General usage seems to favour `\diff` for display-style use and a slash fraction for inline text-style use. Slash fraction derivatives are produced with the `\difs` command: `$ \difs yx $` $\Longrightarrow$ $\mathrm{d}y/\mathrm{d}x$. From version 5.5 of `diffcoeff` (provided the `amsmath` package is loaded) it is also possible to produce a split level alternative, ${}^{\mathrm{d}y}\!/_{\mathrm{d}x}$, by employing a 'variant form' (`\difs.s.yx`; see §3.3.4.6). This scales to the smaller $e^{{}^{\mathrm{d}y}\!/_{\mathrm{d}x}}$ when used as a superscript and to the author's eye is neater than `$e^{\diff yx}$` $\Longrightarrow$ $e^{\frac{\mathrm{d}y}{\mathrm{d}x}}$. If you want still more compactness, you can use the `\difc` ('c' for *compact*) command: `$ \difc yx $` $\Longrightarrow$ $\mathrm{d}_x y$.

*Partial derivatives* follow the same pattern as ordinary derivatives. The commands this time are `\diffp`, `\difsp` and `\difcp` for (upright) fraction, slash fraction and compact forms of partial derivative. Thus `\diffp{F}{x}` (or `\diffp Fx`) produces $\frac{\partial F}{\partial x}$ in text style; `\diffp{F(x,y)}{x}` produces

$$\frac{\partial\, F(x,y)}{\partial x}$$

in display style. For inline use, `\difsp Fx`, displays as $\partial F/\partial x$ and `\difcp Fx` displays as $\partial_x F$. Given that `\partial` takes 8 keystrokes to type, all forms economise on keystrokes.

### 2.2.1 Spacing before the differentiand[2]

There are (at least) two different ways in which to think of derivatives. We are all familiar with the argument presented in elementary calculus books where a curve is shown, and also a point on the curve through which a chord has been drawn. The chord is the hypotenuse of a small right-angled triangle, the other sides having lengths $\delta x$ and $\delta y$ parallel to the coordinate axes. The slope of the

[2] I thank Hans Schülein for first raising this issue with me and for subsequent thoughtful comments.

chord is $\frac{\delta y}{\delta x}$. By drawing shorter and shorter chords through the point, the ratio $\frac{\delta y}{\delta x}$ approaches the slope of the tangent to the curve at the point. We write

$$\frac{\mathrm{d}y}{\mathrm{d}x}$$

for the limit of $\frac{\delta y}{\delta x}$. It is natural following this line of argument to think of $\mathrm{d}y$ and $\mathrm{d}x$ as tiny lengths, like $\delta y$ and $\delta x$, in which case it would be quite wrong to insert space between the d and the $y$. $\mathrm{d}y$ is a single object, called a differential, and we write expressions like

$$\mathrm{d}y = \frac{\mathrm{d}y}{\mathrm{d}x}\ \mathrm{d}x$$

and justly call the fraction in this expression a *differential coefficient*.

But there is another way of viewing differentiation: as a process producing (or *deriving*) one function, $y'(x)$, from another, $y(x)$. Here the sense is of applying an operator $\frac{\mathrm{d}}{\mathrm{d}x}$ to a quite separate object, the function $y(x)$. Although we include $y(x)$ in the numerator it is distinct from the 'd' and should be separated from it by a small space:

$$y'(x) = \frac{\mathrm{d}\,y(x)}{\mathrm{d}x}.$$

Here the fraction on the right is another name for the derived function $y'$ and is justly called the *derivative* of $y$. As you can see a small space has been inserted between the 'd' and the $y$ in the numerator. By default the space is `3 mu` with the ability to stretch by `1 mu` or shrink by `2 mu` – `3 mu plus 1 mu minus 2 mu` – as TEX adjusts lines to fit on the page. (A 'mu' is a 'math unit' and is one eighteenth of a quad.) The size of the space inserted by default can be easily changed; see §3.3 and §3.3.2.

- You may want all your derivatives to have this space before the derivand. The `spaced=1` package option switches this behaviour on.
  - Note however that commands like `\sin`, `\cosh`, `\ln` and similar, including those defined by the user with `\DeclareMathOperator` from the `amsmath` package, come with a preceding space built in. You may want to insert a *negative* thin space, `\!`, before the command, to avoid *too much* space being inserted before the derivand, or use the `\spacednil` command – see §2.2.1.1 immediately below:

```
\[ \diff{\sin x}x, \quad \diff{\!\sin x}x \] ⟹
```

$$\frac{\mathrm{d}\,\sin x}{\mathrm{d}x},\quad \frac{\mathrm{d}\sin x}{\mathrm{d}x}$$

- I have used the `spaced=-1` package option (§1.1) for the present document which inserts space only if the derivand contains *more than one*

*token*. Thus $y(x)$ will have space inserted before it, but $y$ alone will not. This (generally) maintains the distinction between a differential coefficient thought of as a ratio of tiny lengths and a derivative thought of as an operator applied to a function.

  - If the derivand is *appended* when `spaced=-1`, $\frac{\mathrm{d}}{\mathrm{d}x}\,y$, then indisputably we have an operator operating on a function, and the space is inserted. (Note that version 5.4 of `diffcoeff` did not insert the space in this case; that was an error, corrected in version 5.5.)
  - A vector component $x_i$, say, of a vector $\mathbf{x} = (x_1, x_2, x_3)$ is multi-token but might well be thought of (e.g. by the author) as if it were a single-token variable and yet, when `spaced=-1`, a space is inserted. LaTeX provides some spacing shorthands like `\!` (a negative thin space) to adjust things; `diffcoeff` provides others; see §2.2.1.1 below.

- If `spaced=0` no space is inserted before the derivand, single token or multi-token, under any circumstances. If a space is wanted in a particular case, it is up to the user to insert it – perhaps by using `\,` (a thin space) within the braces defining the differentiand, e.g. `{\,f(x)}`. As noted, there are a number of (short) commands in LaTeX (like `\,`) to do the job and `diffcoeff` provides others; see §2.2.1.1 immediately below.

Slash-form derivatives also allow space before the derivand. By default this is `2 mu plus 1 mu minus 2 mu`, slightly reduced from the fraction-form value to avoid visually detaching the initial 'd' operator from the derivative as a whole. The value can be changed; see §3.3 and §3.3.2.

```
$ \difs{s(t)}t, \quad \difs st $
```
$\implies \mathrm{d}\,s(t)/\mathrm{d}t, \quad \mathrm{d}s/\mathrm{d}t.$

For *compact-form* derivatives the space before the derivand is *always* inserted when `spaced` is non-zero, since the subscript precludes the entire symbol ever being viewed as a differential – it is always an operator operating on a function. The inserted space, `1 mu plus 1 mu minus 1 mu` by default, can be changed should you wish; see §3.3 and §3.3.2. It is less for compact forms since the subscript already provides some visual separation. In the example the negative thin space `\!` cancels the positive thin space that comes with `\ln`:

```
$ \difc{\!\ln\sin x}x, \quad \difc st $
```
$\implies \mathrm{d}_x \ln\sin x, \quad \mathrm{d}_t s.$

The `spaced` package option has the same effect on partial derivatives as ordinary derivatives. Thus with `spaced=1` or `spaced=-1`, `3 mu` of space (with some stretch and shrink) is inserted before the differentiand `F(x,y)` in the first member of the following example, space of `2 mu` (with stretch and shrink) in the second, and space of `1 mu` (with stretch and shrink) in the third:

```
\[ \diffp{F(x,y)}x,\;\difsp{F(x,y)}x,\;\difcp{F(x,y)}x, \]
```
$\implies$
$$\frac{\partial\, F(x,y)}{\partial x},\ \partial\, F(x,y)/\partial x,\ \partial_x F(x,y).$$

But for single-token differentiands in this document (using `spaced=-1`) the space is not inserted for upright and slash-form derivatives:

```
\[ \diffp Fx,\quad \difsp Fx. \] ⟹
```

$$\frac{\partial F}{\partial x}, \quad \partial F/\partial x.$$

#### 2.2.1.1 Spacing commands

LaTeX has its own explicit spacing commands. In particular `\,` which is `3 mu` (a thin space) and `\!` which is `-3 mu` (a negative thin space) are convenient in math mode. The `diffcoeff` package adds four simple spacing commands to fill in the gap between these two. These are

**`\negmu`** insert spacing of `-1 mu`;

**`\nilmu`** insert spacing of `0 mu` (cf. use of an empty brace pair `{}` );

**`\onemu`** insert spacing of `1 mu`;

**`\twomu`** insert spacing of `2 mu`.

From version 5.5 of `diffcoeff`, each of these commands can be starred to produce a space of opposite sign but the same magnitude. Thus `\onemu*` is equivalent to `\negmu` and vice versa. The most useful is likely to be `\twomu*` which fills in the missing `-2 mu` negative space in the range between `\,` (`3 mu`) and `\!` (`-3 mu`). Thus if you do *not* want a vector component to be spaced, `$ \difs{\twomu*y_i}t $` ⟹ $\mathrm{d}y_i/\mathrm{d}t$, to be compared with the spaced version in which the `\twomu*` is omitted, $\mathrm{d}\,y_i/\mathrm{d}t$.

From version 5.5, `diffcoeff` also supports three commands which switch on one of the `spaced` package option values *within the current group*.

**`\spacedone`** is equivalent to `spaced=1` within the current group;

**`\spacednil`** is equivalent to `spaced=0` within the current group;

**`\spacedneg`** is equivalent to `spaced=-1` within the current group.

This document uses the package option `spaced=-1`. That means that a multi-token differentiand like $x^k$ will be separated from the $\partial$ symbol in the numerator of a partial differential coefficient by a noticeable whitespace. In the transformation coefficients of tensor calculus that is not desirable (in the author's opinion); `\spacednil` comes to the rescue – compare the first instance with the second in the following example:

```
\[
    \spacednil
    \bar{g}_{ij}=\diffp{x^k}{\bar{x}^i}
        \diffp{x^l}{\bar{x}^j}g_{kl},
\]
```

```
\[
    \bar{g}_{ij}=\diffp{x^k}{\bar{x}^i}
        \diffp{x^l}{\bar{x}^j}g_{kl}.
\]
```

$\Longrightarrow$

$$\bar{g}_{ij} = \frac{\partial x^k}{\partial \bar{x}^i}\frac{\partial x^l}{\partial \bar{x}^j}g_{kl}$$

$$\bar{g}_{ij} = \frac{\partial\, x^k}{\partial \bar{x}^i}\frac{\partial\, x^l}{\partial \bar{x}^j}g_{kl}$$

The effect of `\spacednil` has been confined to the first expression; the package option setting `spaced=-1` has taken effect in the second and the multi-token differentiands are noticeably spaced from the $\partial$ operator.

It is also worth recalling here the reduced spacing around scalable delimiters that results from using `\mleft`, `\mright` in place of `\left`, `\right`; see §1.1 on package options and specifically §1.1.1.

### 2.2.2 Higher order derivatives

An optional argument allows the order of differentiation to be specified. The order need not be a number; an algebraic order of differentiation is perfectly acceptable as is a mix of the two:

```
\[ \diff[2]yx, \quad \diff[n+1]yx. \] ⟹
```

$$\frac{\mathrm{d}^2 y}{\mathrm{d}x^2}, \quad \frac{\mathrm{d}^{n+1} y}{\mathrm{d}x^{n+1}}.$$

As mentioned, the braces can be and have been omitted around the $x$ and $y$ since they are single tokens. The square brackets around the optional order-of-differentiation argument are essential. In slash form,

`$ \difs[2]yx, \quad \difs[n+1]yx $` $\Longrightarrow$ $\mathrm{d}^2 y/\mathrm{d}x^2, \quad \mathrm{d}^{n+1} y/\mathrm{d}x^{n+1},$

the latter of which is a bit of an eyesore. In compact form,

`$ \difc[2]yx,\quad\difc[n+1]yx $` $\Longrightarrow$ $\mathrm{d}^2_x y, \quad \mathrm{d}^{n+1}_x y.$

Note that entering `1` as the optional argument has no effect:

`$ \diff[1]yx,\;\difs[1]yx,\;\difc[1]yx $` $\Longrightarrow$ $\frac{\mathrm{d}y}{\mathrm{d}x}$, $\mathrm{d}y/\mathrm{d}x$, $\mathrm{d}_x y$.

For partial derivatives when differentiating in only one variable the pattern is the same, e.g.

```
\[ \diffp[2]yx, \quad \difsp[n+1]yx, \quad \difcp[2]yx. \] ⟹
```

$$\frac{\partial^2 y}{\partial x^2}, \quad \partial^{n+1} y/\partial x^{n+1}, \quad \partial^2_x y.$$

For partial differentiation in more than one variable – so-called *mixed* partial derivatives – see §2.2.5.

#### 2.2.2.1 Alternative ( : ) notation[3]

From version 5.3 of `diffcoeff` it is also possible to specify the order of differentiation by appending a colon and the order to the differentiation variable:

```
\[ \diff y{x:2},\quad\difsp y{x:n+1},\quad\difcp y{x:2}. \] ⟹
```

$$\frac{\mathrm{d}^2 y}{\mathrm{d}x^2}, \quad \partial^{n+1} y/\partial x^{n+1}, \quad \partial_x^2 y.$$

This is more relevant for mixed partial derivatives when there is more than one variable, each potentially to a different order of differentiation; see §2.2.5.1.[4]

### 2.2.3 Appending the differentiand

Some differentiands are too big or awkward to be placed neatly in the numerator of a derivative and it is natural to *append* them to a preceding differential operator. One could leave the numerator argument empty in the `\difx` or `\difxp` command and follow the command with the differentiand, but `diffcoeff` offers a better way: star the `\difx` or `\difxp` command. This tells `diffcoeff` to append the differentiand. Thus suppose the differentiand is a polynomial, say $ax^2 + bx + c$. Add a star (an asterisk) to the `\diff` command:

```
\[ \diff*{(ax^2+bx+c)}x. \] ⟹
```

$$\frac{\mathrm{d}}{\mathrm{d}x}\,(ax^2 + bx + c).$$

Or, for a partial derivative, one might want to indicate in the differentiand all the variables on which it depends:

```
\[ \diffp*[2]{\Phi(x,y,z)}x \] ⟹
```

$$\frac{\partial^2}{\partial x^2}\,\Phi(x, y, z).$$

A virtue of using an asterisk to append the differentiand is that if one isn't sure whether to append or not, it is an easy matter to simply insert or delete the asterisk to compare the results. For instance, a second derivative is an iterated derivative – one in which a derivative forms the differentiand of another . Thus

```
\[ \diff[2]yx = \diff*{\diff yx}x \] ⟹
```

$$\frac{\mathrm{d}^2 y}{\mathrm{d}x^2} = \frac{\mathrm{d}}{\mathrm{d}x}\,\frac{\mathrm{d}y}{\mathrm{d}x}.$$

This result is more elegant to my eye than what results when removing the asterisk,

[3] I thank Christophe Bal for this suggestion.

[4] Note that there was a bug in version 5.4 of `diffcoeff` – fixed in 5.5 – with the implementation of this feature when the differentiation variable was subscripted, or when the order of differentiation was algebraic and subscripted.

```
\[ \diff[2]yx = \diff{\diff yx}x \] ⟹
```

$$\frac{\mathrm{d}^2 y}{\mathrm{d}x^2} = \frac{\mathrm{d}\,\frac{\mathrm{d}y}{\mathrm{d}x}}{\mathrm{d}x},$$

although whether the *meaning* is clearer is moot.

#### 2.2.3.1 Operator parenthesizing

For slash fraction derivatives, when the derivand is appended, the operator is parenthesized:

```
$ \difs*{(ax^2+bx+c)}x $ ⟹ (d/dx)(ax^2+bx+c).
```

Similarly, for slash-style partial derivatives,

```
$ \difsp*[2]{\Phi(x,y,z)}x $ ⟹ (∂^2/∂x^2)Φ(x,y,z)
```

parentheses are again inserted automatically around the differential operator. If you don't want the parentheses, like other elements of automatic formatting this behaviour can be changed; see §3.3, §3.3.2.

Since the differentiand is appended *by default* in compact-form derivatives, starring has no effect and `\difcp`, `\difcp*` produce identical results.

#### 2.2.3.2 Transposing the argument order

If a *second* asterisk follows the first, the order of the arguments specifying the differentiand and variable(s) of differentiation is reversed. Thus it is clearer to the eye to write

```
\[ \diffp**x{(ax^2+2bxy+cy^2)} \] ⟹
```

$$\frac{\partial}{\partial x}\,(ax^2 + 2bxy + cy^2)$$

than `\[ \diffp*{(ax^2+2bxy+cy^2)}x \]`, where the eye has to search for the variable of differentiation. This is especially the case if the differentiand contains multiple variables and includes commands like `\frac` or `\sqrt` requiring braced arguments. Compare `\diffp*{\frac1{\sqrt{x_i^2-x_j^2}}}{x_i}` with

```
\[ \diffp**{x_i}{\frac1{\sqrt{x_i^2-x_j^2}}}. \]
```

⟹

$$\frac{\partial}{\partial x_i}\,\frac{1}{\sqrt{x_i^2 - x_j^2}}.$$

For compact-form derivatives the initial, appending asterisk is always implicitly present. However, it must be *explicitly* present for the second asterisk to take effect:

```
$ \difcp yx,\quad\difcp*yx,\quad\difcp**yx $ ⟹ ∂_x y,  ∂_x y,  ∂_y x.
```

**The `dif**` package option** Since the natural inclination is to read $\mathrm{d}_x y$ as (something like) *d sub x y*, and yet the arguments of `\difc yx` are presented in the reverse order, you might wish to make the double asterisk state the default and *always* present the variable(s) of differentiation before the differentiand, at least for compact derivatives. This can be done with the package option `dif**`. By entering `dif**={c,cp}` in the optional argument of the `\usepackage` command when calling `diffcoeff`,

```
\usepackage[dif**={c,cp}]{diffcoeff}
```

the reversed order of arguments, first the differentiation variable(s) then the differentiand, becomes the default condition for `\difc` and `\difcp`. *Now* appending two asterisks to `\difc` and `\difcp` means reversing this *new* default, a double reversal overall, returning to the original state in which the differentiand precedes the variable(s) of differentiation. If you want to extend the reversed argument order also to, say, `\difs`, `\difsp`, set `dif**={c,cp,s,sp}`, or to extend it to all derivatives set `\dif**={c,cp,s,sp,f,fp}`.

### 2.2.4 Point of evaluation/variables held constant

If you want to specify a point at which a derivative is evaluated, append a final square-bracketed optional argument:

```
\[ \diff[2]yx[0] \]
```
$$\Longrightarrow$$
$$\left(\frac{\mathrm{d}^2 y}{\mathrm{d}x^2}\right)_0$$

Note that there must be *no space* before the left square bracket of the trailing argument, otherwise it will be treated as part of the wider mathematical expression of which the derivative is part and typeset as such. (But this should not cause a LaTeX error.)

- If you prefer to use subscripted *square* brackets
$$\left[\frac{\partial F(x,y)}{\partial x}\right]_{(0,0)}$$
or a subscripted vertical rule after the derivative
$$\left.\frac{\partial F(x,y)}{\partial x}\right|_{(0,0)}$$
to indicate a point of evaluation, then this can easily be done; see §3.3.4.1. Parentheses are the ISO recommendation; see ISO 80000-2.

Because the slash form spreads the derivative out horizontally, parentheses are the natural way in this case to indicate a point of evaluation:

```
$ \difs{\!\ln sin x}{sin x}[x=\pi/3] $
```
$\Longrightarrow$ $(\mathrm{d}\ln\sin x/\mathrm{d}\sin x)_{x=\pi/3}$.

A vertical rule can easily become too remote from the opening 'd' of the differential coefficient: $\mathrm{d}\ln\sin x/\mathrm{d}\sin x|_{x=\pi/3}$. Parentheses tie the whole cluster of symbols together.

Parentheses are also used with partial derivatives to indicate variables held constant, particularly in thermodynamics. In the following well-known relation in the subject, the differentiands are appended and the trailing argument is used to indicate the variables held constant:

```
\[ \diffp*{\frac PT}U[V] = \diffp*{\frac 1T}V[U] \] ⟹
```

$$\left(\frac{\partial}{\partial U}\,\frac{P}{T}\right)_V = \left(\frac{\partial}{\partial V}\,\frac{1}{T}\right)_U.$$

This is much easier to write than building the expressions 'by hand', starting with `\left(` and finishing with `_U`.

#### 2.2.4.1 Superscripts

It is easy to add a superscript to a derivative to indicate evaluation at two points and the difference between the values:

```
\[ \diff {\!\sin x}x[0]^{\mkern-11mu\pi/2} \] ⟹
```

$$\left(\frac{\mathrm{d}\sin x}{\mathrm{d}x}\right)_0^{\pi/2}$$

but to my eye either square brackets or a vertical rule are clearer for this purpose (and do not involve nudging the subscript or superscript closer to the right delimiter); see §3.3.

#### 2.2.4.2 Empty trailing argument

If the trailing argument is included but left empty it will, with the default set-up, wrap the derivative in parentheses but with no subscript. This fact can be exploited. Thus, for a particle of mass $m$ moving along a line, distant $x$ from the origin at time $t$, the kinetic energy is:

```
$ \tfrac 12 m \difs xt[]^2 $ ⟹
```

$\frac{1}{2}m(\mathrm{d}x/\mathrm{d}t)^2$.

Or, again exploiting the parentheses resulting from an empty trailing argument, Lagrange's equations of motion in analytic mechanics can be written,

```
\[ \diffp L{q_k}-\diff**t{ \diffp L{\dot{q}_k}[] } = 0 \] ⟹
```

$$\frac{\partial L}{\partial q_k} - \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L}{\partial \dot{q}_k}\right) = 0.$$

(See §2.2.3.2 for the double asterisk.)

You may feel that there is too much whitespace before the large left parenthesis in this expression. The present document uses the `spaced=-1` package option which inserts a thin space before a multi-token differentiand. The `\spacednil` command (see §2.2.1.1) cancels that, just leaving the built-in spacing of `\left(`:

```
\[
   \spacednil
   \diffp L{q_k}-\diff**t{ \diffp L{\dot{q}_k}[] } = 0.
\]
```

$\Longrightarrow$

$$\frac{\partial L}{\partial q_k} - \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L}{\partial \dot{q}_k}\right) = 0.$$

### 2.2.5 Mixed partial derivatives

The new thing with partial derivatives, not present with ordinary derivatives, is the possibility of differentiation in more than one variable, so-called *mixed* partial derivatives. If each variable is differentiated only to the first order, then it is easy to specify the derivative. Suppose $F$ is a function of three variables, $x$, $y$, $z$. Then

```
\[ \diffp F{x,y,z},\quad \diffp{F(x,y,z)}{x,y,z}. \] ⟹
```

$$\frac{\partial^3 F}{\partial x\,\partial y\,\partial z}, \quad \frac{\partial^3 F(x,y,z)}{\partial x\,\partial y\,\partial z}.$$

The variables of differentiation are listed in order in a comma list – `{x,y,z}` – forming the second mandatory argument. The total order of differentiation (3 in the example) was calculated by `diffcoeff` and was inserted automatically. (The effect of the `spaced=-1` package option can be seen in the second member of this example.)

The slash form is

`$ \difsp F{x,y,z} $` ⟹ $\partial^3 F/\partial x\,\partial y\,\partial z$,

and the compact form is

`$ \difcp F{x,y,z} $` ⟹ $\partial_x \partial_y \partial_z F$.

One might wonder about an even more compact notation like $\partial^3_{xyz} F$ or $\partial_{xyz} F$? These can be readily achieved by means of a variant form; see §3.3.4.8.

To differentiate variables to higher order, their orders need to be specified explicitly. To do so use a comma list for the optional argument:

```
\[ \diffp[2,3]F{x,y,z},\quad \difsp[2,3]F{x,y,z}. \] ⟹
```

$$\frac{\partial^6 F}{\partial x^2 \partial y^3 \partial z}, \quad \partial^6 F/\partial x^2 \partial y^3 \partial z.$$

The overall order of the derivative – 6 – is again automatically calculated and inserted as a superscript on the $\partial$ symbol in the numerator.

In these examples, the comma list of orders has only *two* members, although there are *three* variables. It is assumed that the orders given in the comma list apply in sequence to the variables, the first order to the first variable, the

second to the second variable, and so on, and that any subsequent orders not listed in the optional argument are 1. Thus we need to specify only 2 and 3 in the example; the order of differentiation of $z$ is 1 by default. But you *cannot* use an order specification like `[,,2]`; instead write `[1,1,2]` (which is the natural thing to do in any case). It is only the *tail* of an order specification which can be omitted. In the other direction, if there are more orders of differentiation specified than there are variables, the list of orders is truncated to match the number of variables.

For compact-form derivatives, and using the colon notation to specify the orders of differentiation (available since version 5.3 of `diffcoeff`),

`$ \difcp F{x:2,y:3,z} $` $\Longrightarrow \partial_x^2 \partial_y^3 \partial_z F\,.$

A 'doubly compact' notation is also possible, producing $\partial_{x^2y^3z} F$ – see §3.3.4.8.

#### 2.2.5.1 Algebraic orders

Orders of differentiation can be algebraic as well as numerical:

`\[ \diffp[2m-1,m+1,2]F{x,y,z} \]` $\Longrightarrow$

$$\frac{\partial^{3m+2} F}{\partial x^{2m-1} \partial y^{m+1} \partial z^2}$$

The total order of differentiation is still calculated by `diffcoeff`. Or again,

`\[ \diffp[1,km+1,m+k-1]{F(x,y,z)}{x,y,z} \]` $\Longrightarrow$

$$\frac{\partial^{m+k+km+1} F(x,y,z)}{\partial x \, \partial y^{km+1} \partial z^{m+k-1}}\,.$$

Algebraic orders are sorted according to increasing number of characters – hence $km$ after $m$ and $k$ in the example – then followed by the total numeric order.

When there are two or more variables of differentiation, particularly when subject to different orders of differentiation, the colon notation (see §2.2.2.1) is clearer to both writer and reader; it avoids a lot of clutter:

`\[ \diffp{F(x,y,z)}{x,y:km+1,z:m+k-1} \]` $\Longrightarrow$

$$\frac{\partial^{m+k+km+1} F(x,y,z)}{\partial x \, \partial y^{km+1} \partial z^{m+k-1}}\,.$$

Note that it suffices to write `x` rather than `x:1`. If, in a fit of absent-mindedness, one specifies the orders of differentiation by both methods, it is the colon notation that prevails:

`\[ \diffp[1,2,3]{F(x,y,z)}{x:4,y:5,z:6} \]` $\Longrightarrow$

$$\frac{\partial^{15} F(x,y,z)}{\partial x^4 \partial y^5 \partial z^6}$$

#### 2.2.5.2 Parentheses

Auto-calculation of the total order can also handle the simple use of parentheses:

```
\[ \diffp[2m-(k+1),2(k+1)-m]{F(x,y,z)}{x,y,z} \] ⟹
```

$$\frac{\partial^{m+k+2} F(x,y,z)}{\partial x^{2m-(k+1)} \partial y^{2(k+1)-m} \partial z}$$

The left parenthesis in each case is preceded by a *number* or a *sign*. In evaluating the total order `diffcoeff` multiplies out the expression (or that is the effect). However, an order specification like `[f(n+1),f(n-1)]` is interpreted as the familiar 'function of' notation – in this case a function $f$ evaluated at $n \pm 1$. `diffcoeff` *always* interprets a left parenthesis preceded by something that is *not* a number or a sign in this way. It does not try to multiply out such expressions when calculating the total order. The following example combines both uses:

```
\[\diffp[2(f(n)-(m-1)),5-(f(n)+m)]F{x,y}\] ⟹
```

$$\frac{\partial^{7-3m+f(n)} F}{\partial x^{2(f(n)-(m-1))} \partial y^{5-(f(n)+m)}}$$

#### 2.2.5.3 Order-override option

Sometimes the total order of differentiation may not be calculable by `diffcoeff`. For instance, an order specification like `[m(k-1)+1,m(k+1)-1]` may not be meant in the 'function of' sense, but rather that `m` multiplies `k-1` and `k+1`. To cope with such situations, `diffcoeff` provides the *order-override option*.[5] This is an *angle-bracket* delimited (using the 'less than' and 'greater than' symbols, `< >`) optional argument between the order specification (if it is used) and the differentiand, in which is entered the desired form of the total order of differentiation. Angle brackets provide a necessary distinction from the square-bracket delimited order specification.

In an earlier example, the total order of differentiation $m + k + km + 1$ factorizes to $(k + 1)(m + 1)$. `diffcoeff` is not a computer algebra system and does not do such factorizations but if you would like to express the total order in that form, use the override option:

```
\[ \diffp<(k+1)(m+1)>{F(x,y,z)}{x,y:km+1,z:m+k-1}. \]
```

⟹

$$\frac{\partial^{(k+1)(m+1)} F(x,y,z)}{\partial x \partial y^{km+1} \partial z^{m+k-1}}.$$

When the override option is used, the algorithm that calculates the total order does not get called at all In this way not only can the total order be

[5] I thank Christophe Bal for urging the availability of this argument and the use of angle brackets.

presented in whatever manner you wish but essentially arbitrary material can be attached as a superscript to the $\partial$ (or other) symbol in the numerator.

(Note that for compact-form derivatives, prior to version 5.5 of `diffcoeff`, the override option was irrelevant. Version 5.5 introduced a 'doubly compact' notation and that is no longer so; see §3.3.4.8.)

**Order-override command (deprecated):** As an alternative to the override option, you can use the `\difoverride` command, although *this is now deprecated*. (It was introduced to avoid cluttered expressions, but that is reduced with the introduction of the colon notation and an angle-bracketed override optional argument.) The command takes one (mandatory) argument, the total order of differentiation, which it stores:

```
\[
  \difoverride{(k+1)(m+1)}
  \diffp[1,km+1,m+k-1]{F(x,y,z)}{x,y,z}
\]
```

$\Longrightarrow$

$$\frac{\partial^{(k+1)(m+1)} F(x,y,z)}{\partial x\, \partial y^{km+1} \partial z^{m+k-1}}$$

Note that in the example `\difoverride` has been used *within* the math environment. This is good practice. It prevents the contents of the command erroneously overriding the orders of later derivatives in other math environments, but it does mean that if you want to include a later derivative within *this* mathematical environment then you will need to precede the second derivative with the statement `\difoverride{}`.

#### 2.2.5.4 Error messages

The order-override option is also needed when calculating the total order is beyond the abilities of `diffcoeff`. The package is *not* a computer algebra system. It can cope with order specifications where variables are followed by diverse arithmetic operators: `n^2`, `m\times n`, `m/2` and the like cause no problems. But a *number* can be followed *only* by a sign or a variable or a left parenthesis. Anything beyond this will raise an error. For instance

```
\[ \diffp[2^k]F{x,y} \]
```

produces a message beginning '`! Package diffcoeff Error:`' and continuing,

```
number followed by ^ in the order spec. [2^k,1] on
line xx. Calculation of the total differentiation
order fails in this case. Use the order override
option to enter the total order. See the
diffcoeff documentation for further information.
```

(The `xx` will be replaced by a specific line number in each case. Line breaking may also differ from case to case.) To avoid such errors and enable compilation to proceed, do as the message suggests – use the override option.

There are limitations on what order specifications the `diffcoeff` package can 'digest', but in real life that is unlikely to be significant. Mixed partial derivatives are used far less often than the pure derivatives, and when they *are* used it is nearly always to low numerical orders like 1 or 2. For those rare other cases, the override option is always available.

#### 2.2.5.5 Generic mixed partial derivatives

The question was posed on TEX StackExchange[6] about how to write a generic mixed partial derivative. In version 5.4 of `diffcoeff`, the answer provided looked like this, `\diffp<n>{y}{x_1,x_2\dots,x_n}`, in which the `\dots` command (from `amsmath`) has been adjoined to the variable `x_2` in the specification. However, the enquirer's first instinct had been to insert `\dots` between commas in the variable list, as one would with a proper differentiation variable. The problem was that a $\partial$ operator was inserted immediately before the dots. From version 5.5 of `diffcoeff` one can achieve the desired effect in that 'first instinct' way but, rather than inserting `\dots` or `\ldots` or `\cdots`, one enters three dots (periods, full stops). These are converted by `diffcoeff` into a chosen dots command. By default `diffcoeff` uses `\cdots` for upright fraction derivatives (`\diffp`), and `\ldots` for slash-fraction (`\difsp`) and compact form (`\difcp`) derivatives.

**A dissertation on dots** To justify the `diffcoeff` defaults, the understanding is that one uses `\ldots` in a list-like context, $a, b, \dots, k$, and `\cdots` between binary relations, $a+b+\cdots+k$. In that sense, the clearest case with relevance for `diffcoeff` is a product of differentials (see §4.1) of the kind that might occur in a multiple integral:

$$\int\cdots\int F(x_1,\dots,x_n)\,\mathrm{d}x_1\cdots\mathrm{d}x_n$$

The relevant dots here are those between the differentials. (An explicit `\dotsi` from `amsmath` has been used between the integral signs and an explicit `\ldots` in the list of arguments of $F$.) Here $\mathrm{d}x_1\cdots\mathrm{d}x_n$ is an element of volume in an $n$-dimensional space, a product of $n$ infinitesimal lengths $\mathrm{d}x_i$. Between the $\mathrm{d}x_i$ is an implicit product sign, a `\cdot` ( $\cdot$ ), and so three dots entered in the argument of a `\dl` command are rightly converted to `\cdots`. Also clear to my mind is the compact form partial derivative, $\partial_{x_1}\partial_{x_2}\dots\partial_{x_n}F(\mathbf{x})$, which is not a product but a sequence of operators $\partial_{x_i}$ applied one after the other. Using `\ldots` in this case seems appropriate.

[6] https://tex.stackexchange.com/questions/745070/partial-derivatives-with-n-variables-in-diffcoeff

Generic slash- and upright fraction derivatives are also sequences of operations but the construction of the derivatives, distributed over numerator and denominator, disguises the fact. Since the slash in a slash fraction tends to isolate the denominator and so *visually* brings `\difsp` closer to `\difcp`, there seems no reason not to use `\ldots` in this case: $\partial^n F(\mathbf{x})/\partial x_1\,\partial x_2 \ldots \partial x_n$. But what to make of

$$\frac{\partial^n F(\mathbf{x})}{\partial x_1\,\partial x_2 \cdots \partial x_n}\,?$$

In an upright fraction like this, `\cdots` breaks up the whitespace better than `\ldots`, which leaves too much whitespace between the dots near the bottom of the denominator and the total order of differentiation $n$ near the top of the numerator, producing a 'hollowed out' symbol:

$$\frac{\partial^n F}{\partial x_1 \ldots \partial x_n},\quad \frac{\partial^n F}{\partial x_1 \cdots \partial x_n}.$$

Hence `\cdots` is the `diffcoeff` default. If you prefer different choices, see §3.3.4.3 on how to make them *your* defaults.

In the following examples, note the use of the override option.

```
\[ \diffp<n>F{x_1,x_2,...,x_n},\quad
   \diffp<n>F{u:a,v:b,...,z:k}. \]
```

$\Longrightarrow$

$$\frac{\partial^n F}{\partial x_1\,\partial x_2 \cdots \partial x_n},\quad \frac{\partial^n F}{\partial u^a \partial v^b \cdots \partial z^k}.$$

For slash-fraction and compact form derivatives,

```
\[ \difcp<n>F{x_1,x_2,...,x_n},\quad
   \difsp<n>F{u:a,v:b,...,z:k}. \]
```

$\Longrightarrow$

$$\partial_{x_1}\partial_{x_2}\ldots\partial_{x_n}F,\quad \partial^n F/\partial u^a \partial v^b \ldots \partial z^k.$$

It is sometimes possible to omit the override option by assigning an appropriate order to the dots:

```
\[ \diffp[1,1,n-3]F{x_1,x_2,...,x_n}. \]
```

$\Longrightarrow$

$$\frac{\partial^n F}{\partial x_1\,\partial x_2 \cdots \partial x_n},$$

where the order $n-3$ has been assigned to the dots, or even

```
\[ \diffp F{u:a,v:b,...:\cdots,z:k\relax }, \]
```

$\Longrightarrow$

$$\frac{\partial^{a+b+\cdots+k} F}{\partial u^a \partial v^b \cdots \partial z^k},$$

where the dots have been assigned the 'order' `\cdots`. `diffcoeff` stores the algebraic orders of differentiation as *strings* – sequences of characters. Since they are sorted by number of characters and `\cdots` (including the backslash) has 6, I have padded `k` with `\relax` which also has 6 characters and has no visual effect to ensure that $\cdots$ sorts *between* $b$ and $k$; (`\empty` would have served equally well).

#### 2.2.5.6 Comma list of variables of differentiation

In tensor calculus or analytic mechanics differentiations are almost always in terms of super- or subscripted coordinates. In many other contexts multi-token variables of differentiation are common too – the reciprocal of the temperature for instance in thermodynamics. This is why a comma list is used in `diffcoeff` for specifying variables of differentiation for mixed partial derivatives. Although it would be nice to write the minimal `{xy}` rather than `{x,y}` when two variables $x$ and $y$ are involved, the extra writing is trivial and the comma list allows a simpler handling of multi-character variables. Thus in tensor calculus we find expressions like

```
\[ \diffp{A_i}{x^j,x^k} \] ⟹
```

$$\frac{\partial^2 A_i}{\partial x^j \, \partial x^k}.$$

It is easier to write `{x^j,x^k}` here than, say, `{{x^j}{x^k}}` to distinguish the variables. It's also easier to read, particularly if the indices themselves get ornamented and need surrounding braces:

```
\[ \diffp{A_i}{x^{j'},x^{k'}} \] ⟹
```

$$\frac{\partial^2 A_i}{\partial x^{j'} \, \partial x^{k'}}.$$

Compare that variable specification, already requiring some care, with the thicket of braces of `{{x^{j'}}{x^{k'}}}`. Some extra whitespace would help here, but the point stands: the comma list requires fewer nested braces. Of course braces *are* required if a variable of differentiation includes a comma, for then the comma needs to be enclosed in them.

#### 2.2.5.7 Spacing in the denominator

In Chapter 18 of the *The TEXbook*, Knuth suggests inserting a thin space, `\,` (or `3 mu`), between differentials in appropriate contexts, giving as an example

$dx\,dy = r\,dr\,d\theta$. To my eye, that degree of extra spacing in the denominator of a derivative seems too great, interfering with seeing the derivative 'as a whole',

$$\frac{\partial^3 F}{\partial x\,\partial y\,\partial z},$$

especially for the slash-form: $\partial^3 F/\partial x\,\partial y\,\partial z$. Some spacing is desirable, but less. By default `diffcoeff` inserts `2 mu` (with stretch and shrink) between the terms, $\partial^3 F/\partial x\,\partial y\,\partial z$, bringing the extremities into the overall symbol.

Should a differentiation occur to higher order and so a variable acquire a superscript, an adjustment is made to the extra spacing because the superscript itself has a spacing effect. By default `1 mu` is subtracted from the default spacing. Thus in

$$\frac{\partial^4 F}{\partial x^2\partial y\,\partial z},$$

spacing of `2 mu` is inserted between the $\partial y$ and $\partial z$, but because the superscript already provides some separation between them, only `1 mu` is inserted between $\partial x^2$ and $\partial y$. The values used for the spacing and its adjustment in the presence of a superscript can be changed by the user; see Chapter 3.

When the variables themselves are superscripted, as happens in tensor calculus, no automatic adjustment is made. Any fine-tuning must be done by the user using explicit spacing commands:

```
\[ \diffp{A_i}{ x^j\negmu,x^k } \] ⟹
```

$$\frac{\partial^2 A_i}{\partial x^j\,\partial x^k}.$$

The `\negmu` reduces the space between the terms from the default `2 mu` to `1 mu`.

### 2.2.6 Multi-token variables: parenthesizing

Differentiating a function of a function may involve a multi-character differentiation variable – e.g. to differentiate $\ln\sin x$ in $x$ means forming the product

```
\[ \diff{\!\ln\sin x}{\twomu*\sin x}\diff{\!\sin x}x \] ⟹
```

$$\frac{\mathrm{d}\ln\sin x}{\mathrm{d}\sin x}\frac{\mathrm{d}\sin x}{\mathrm{d}x}.$$

(I have inserted preceding negative spaces to counteract the thin space built-in to `\sin`.) Forming the *second* derivative of $\ln\sin x$ will now involve forming, among other quantities,

```
\[\diff[2]{\!\ln\sin x}{\sin x}\] ⟹
```

$$\frac{\mathrm{d}^2\ln\sin x}{\mathrm{d}(\sin x)^2}$$

Parentheses have been inserted automatically around $\sin x$ in the denominator to avoid any visual hint that we are differentiating in the sine of $x^2$:

$$\frac{\mathrm{d}^2 \ln \sin x}{\mathrm{d} \sin x^2}$$

*That* is the problem with a long (multi-character) variable: the superscript in a higher order derivative may look as if it applies to only part – the last character – of a multi-character variable. Hence the parenthesizing; for first-order derivatives the problem does not arise and so there is no automatic parenthesizing – you need to do it yourself if you want it. In some cases it seems necessary:

```
\[\diff {f(x)}{x/k}, \quad \diff {f(x)}{(x/k)}.\] ⟹
```

$$\frac{\mathrm{d}\, f(x)}{\mathrm{d} x/k}, \quad \frac{\mathrm{d}\, f(x)}{\mathrm{d}(x/k)}.$$

You may prefer `diffcoeff` not to parenthesize by default in higher order derivatives. Changing the default setting is easily accomplished; see §3.3.2.

But if you do want parentheses, are they in the right place? Logically, they should include the 'd': $(\mathrm{d} \sin x)^2$ – it is the differential $\mathrm{d} \sin x$ that is of the second order. But as the examples in the Rogues' Gallery show the inclination seems to be to do otherwise. This may be because one wants in any case to parenthesize the variable to show that the 'd' symbol attaches to the whole variable and not just its *first* character. A second outer pair of parentheses,

$$\frac{\mathrm{d}^2 f(x)}{(\mathrm{d}(x/k))^2},$$

like this, then seems too fussy and detracts from comprehending the symbol 'at a glance'. Customary but illogical notations are familiar in mathematics – think of the position of the superscripts in an identity like $\sin^2 \theta + \cos^2 \theta = 1$.

The discussion applies equally to partial derivatives. In thermodynamics and statistical mechanics one may want to differentiate in the reciprocal of the temperature, $1/\Theta$ say:

```
\[ \diffp[2]q{\frac 1\Theta} \] ⟹
```

$$\frac{\partial^2 q}{\partial \left(\frac{1}{\Theta}\right)^2}.$$

As noted, when differentiating to first order, parenthesizing is up to the user:

```
\[ \diffp q{\bigl(\frac 1\Theta\bigr),V} \] ⟹
```

$$\frac{\partial^2 q}{\partial \left(\frac{1}{\Theta}\right) \partial V}.$$

In tensor calculus, indices are sometimes attached to variables as superscripts, leading in the default set-up to constructs like

$$\sum_{i=1}^{3} \frac{\partial^2 V}{\partial (x^i)^2} = \frac{\partial^2 V}{\partial (x^1)^2} + \frac{\partial^2 V}{\partial (x^2)^2} + \frac{\partial^2 V}{\partial (x^3)^2}.$$

If you prefer no parentheses (see §3.3.4.4), although the resulting expression is understandable to the experienced eye, it is not elegant and potentially ambiguous ($i^2 = -1$ anyone?):

$$\sum_{i=1}^{3} \frac{\partial^2 V}{\partial x^{i\,2}} = \frac{\partial^2 V}{\partial x^{1\,2}} + \frac{\partial^2 V}{\partial x^{2\,2}} + \frac{\partial^2 V}{\partial x^{3\,2}}.$$

# Chapter 3

# Templates, defaults & variants

`diffcoeff` was built on the facilities offered by the `xtemplate` package (included in the LaTeX3 bundle `l3packages`). These capabilities have now been absorbed into the LaTeX base (see `lttemplates-doc.pdf`). For `diffcoeff` a template is a list of parameter values determining what a derivative looks like in the pdf. The parameters may be broad-brush settings like whether the derivative is built from `\frac` or the slash `/` or in compact form, or whether the operator symbol is `d` or `\partial` (or `\nabla` or `\delta` or ...), or the parameters may be finer-grained, determining minutiae of spacing, easily missed at a casual glance but giving some cumulative overall effect. Access to the parameters is gained through the command `\difdef`[1], one argument of which is a *key=value* list of parameter values. Each such list is given a name (the second argument of `\difdef`) and is 'turned into a derivative' by placing the name between dots as the first argument of the appropriate `\difx`, `\difxp` command[2]. All this is discussed in §3.3 below.

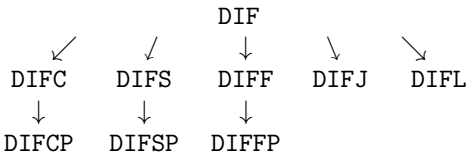
## 3.1 Template structure

To write a derivative one doesn't want to have to type a long list of *key=value* statements each time. The *default* values given to keys is crucial. Only some of the defaults appropriate for, say, an upright fraction ordinary derivative are going to be relevant for a slash-fraction partial derivative let alone a compact form partial derivative. This suggests creating a primary template as a 'super-repository' of default values and from this creating secondary or child templates in which (only) *some* of the defaults are changed – and, if necessary, creating

[1] In version 5; it has *three* arguments and replaces the two-argument command `\diffdef` in version 4 of `diffcoeff`.

[2] And – see Chapter 4 – of the differential and jacobian commands, `\dl` and `\jacob`.

Table 3.1: Template inheritance




from these child templates children of their own (grandchild templates) in which again some further defaults are adjusted.

In `diffcoeff`, the template that is the 'primogenitor' of the lines of default inheritance is named `DIF`. It is the repository of all possible keys used in all possible forms of derivative in `diffcoeff` and so has keys appropriate to upright-fraction, slash-fraction and compact forms of derivative; it has keys appropriate to multi-variable partial derivatives and single-variable ordinary derivatives, but it is not actually used to form derivatives. That is the role of its child templates `DIFF`, `DIFS` and `DIFC` corresponding to derivatives of upright-fraction, slash-fraction and compact forms. These child templates inherit the defaults of `DIF` save for some settings explicitly changed in the child template relevant to the specific forms of each.

Apart from the operator symbol, most of the settings in the child templates `DIFF`, `DIFS` and `DIFC` are also appropriate for partial derivatives. From a code design point of view, there is a certain neatness at not multiplying the number of templates in play, but actual use – for instance, forming the 'Rogues' gallery' of §1.2 – suggests the further step of creating additional templates specifically for *partial* derivatives in the three fraction forms. Apart from the operator symbol, the templates `DIFFP`, `DIFSP` and `DIFCP` inherit nearly all the defaults of their parents `DIFF`, `DIFS` and `DIFC` respectively.

Again, actual use suggests two further templates, both direct children of `DIF`, for the creation of jacobians, template `DIFJ`, and differentials, template `DIFL`, with default values appropriate to each. In all `diffcoeff` uses nine templates, the arrows in Table 3.2 indicating lines of inheritance of default values. Only the child and grandchild templates of `DIF` are used for actual construction of derivatives (and jacobians and differentials). `DIF` itself 'sits above the fray'.

## 3.2 Default values for template `DIF`

Table 3.2 lists the keys available for forming derivatives and the default values assigned in the 'grandparent' template, the primogenitor `DIF`. What values are appropriate will change with the math font used and the font size. The values shown have been chosen for

- 10pt, Computer Modern or Latin Modern fonts.

Other math fonts and sizes will require different defaults. Also, different forms of derivative demand different defaults for some keys. Where a key is relevant

for more than one style of derivative the default value is chosen according to the following precedence scheme:

1. *ordinary upright*-fraction derivatives in *display*-style environments
2. *ordinary slash*-fraction derivatives in *text*-style environments
3. *ordinary compact*-form derivatives in *text*-style environments

Users of version 4 of `diffcoeff` will notice similarities with and differences from that earlier version. Some key names remain (`op-symbol`), some names have changed (`multi-term-sep` for `denom-term-sep`), keys beginning with an asterisk, `*`, meaning they apply only when the derivand is appended, lack a following hyphen (`*derivand-sep` rather than `*-derivand-sep`), some keys have vanished (the `/` keys), and there are some new keys (`lvwrap-Ldelim`, `lvwrap-Rdelim`). The redesign of the user interface – `\difs`, `\difsp` for the `/` switch, the new compact form commands `\difc`, `\difcp`, and bringing the jacobian and differential within the `DIF` template structure – meant revisiting and rethinking the list of keys. In the end it seemed simpler (less confusing) to treat this as a completely new list rather than an amendment of the earlier one.

The first column in table 3.2 lists key names, the second column lists default values, and the third column lists the form or forms of derivative for which the key is *relevant* – meaning that assigning a different value to the key can change the appearance of the corresponding derivative in some way. The identifiers have these meanings:

**f, fp** upright fraction ordinary derivative, partial derivative;

**s, sp** slash fraction ordinary derivative, partial derivative;

**c, cp** compact ordinary derivative, partial derivative;

**j** jacobian;

**l** differential.

In Table 3.2 and following tables, all values specifying a space require the unit (`mu`) to be included; a number alone does not suffice. (A 'mu' is a 'math unit', 1/18 of a quad. A thin space `\,` is 3 mu.) 'Elastic' spaces with stretch and shrink can be written compactly like `3muplus1muminus2mu` or with spaces like `3 mu plus 1 mu minus 2 mu`.

Available keys and their defaults are listed in Table 3.2. If you are dissatisfied with various choices they can be changed; see the discussion at §3.3.2.

**style** the fraction form of the derivative;

- for upright-fraction derivatives, `\diff`, `\diffp`, a choice of `frac`, `tfrac` or `dfrac`:
  - `frac` results in a fraction formed from `\frac`, scalable
  - `tfrac` results in a fraction formed from `\tfrac`, not scalable
  - `dfrac` results in a fraction formed from `\dfrac`, not scalable
  - default in templates `DIFF`, `DIFFP` = `frac`
- for slash-fraction derivatives,`\difs`, `\difsp`, a choice of `/`, `sfrac`, `auto`, `big`, `Big`, `bigg` or `Bigg`
  - `/` forms the slash fraction with `/`, not scalable
  - `sfrac` forms a split-level inline fraction like $^{\mathrm{d}y}/_{\mathrm{d}x}$; scalable
  - `auto` forms the slash fraction with `\left. \middle/ \right.`, scalable
  - `big`, `Big`, `bigg` and `Bigg` form the slash fraction with `\big/`, `\Big/`, `\bigg/` and `\Bigg/` respectively, not scalable
  - default in templates `DIFS`, `DIFSP` = `/`
- for compact-form derivatives, `\difc`, `\difcp`, and differentials, a choice of `_`, `cc`, `dl` or `d^`
  - `_` or `d^` form derivatives of compact form like $\mathrm{d}_x y$, $\partial_x \partial_y^2 z$
  - `dl` with no differentiand, forms differentials like $\mathrm{d}x$ and $\partial x^2 \partial y \partial z$
  - `cc` forms derivatives in 'doubly compact' form like $\partial_{x^2yz^3} y$
  - default in templates `DIFC`, `DIFCP` = `_`
- overall default in template `DIF` = `frac`

**slash-tok** token or tokens used for the slash fraction; (see §3.3.6 for an assignment different from `/`); default = `/`

**slash-sep** space inserted on either side of the **slash-tok**; if two values are given (in a comma list), the first is inserted to the left of **slash-tok**, the second to the right; if only one value is provided, it is inserted on both sides of **slash-tok**; default = `0 mu`

**sfrac-scaling** used only for the `sfrac` variant of a slash-form derivative to enable scaling. For 10pt text-style characters, $0.7 \times 10 = 7$pt for script-style characters, and $0.7 \times 7 = 0.49 \approx 5$pt for scriptscript-style characters; default = 0.7

**derivand-sep** horizontal space added before the differentiand if the `spaced` package option is set to `1`, or before a multi-tokened differentiand if the `spaced` package option is set to `-1`; note that compact-form derivatives *always* have this space inserted; default (appropriate for an upright-fraction derivative) = `3mu plus 1mu minus 2mu`

Table 3.2: `DIF` defaults

| key | default | relevance |
|---|---|---|
| style | `frac` | f, fp, s, sp, c, cp, j |
| slash-tok | `/` | s, sp, j |
| slash-sep | `0 mu` | s, sp, j |
| sfrac-scaling | `0.7` | s, sp, j |
| derivand-sep | `3 mu plus 1 mu minus 2 mu` | f, fp, s, sp, c, cp |
| op-symbol | `\mathrm{d}` | f, fp, s, sp, c, cp, j, l |
| op-symbol-alt | `op-symbol` | f, fp, s, sp, j |
| op-order-nudge | `0 mu` | f, fp, s, sp, c, cp, l |
| op-sub-nudge | `0 mu` | c, cp |
| var-sup-nudge | `1 mu` | f, fp, s, sp, l |
| multi-term-sep | `2 mu plus 1 mu minus 1 mu` | f, fp, s, sp, c, cp, l |
| term-sep-adjust | `-1 mu` | f, fp, s, sp, c, cp, l |
| dots | `\cdots` | f, fp, s, sp, c, cp, l |
| long-var-wrap | `d(v)` | f, fp, s, sp, l |
| lvwrap-Ldelim | `\mleft (` | f, fp, s, sp, j, l |
| lvwrap-Rdelim | `\mright )` | f, fp, s, sp, j, l |
| lvwrap-sup-nudge | `-2 mu` | f, fp, s, sp, l |
| outer-Ldelim | `\left (` | f, fp, s, sp, c, cp, j, l |
| outer-Rdelim | `\right )` | f, fp, s, sp, c, cp, j, l |
| elbowroom | `0 mu` | f, fp, s, sp, c, cp, j, l |
| sub-nudge | `-5 mu` | f, fp, s, sp, c, cp |
| difcc-var-ord | `^{#1}` | c, cp |
| *derivand-sep | `derivand-sep` | f, fp, s, sp, c, cp |
| *op-set-left | `false` | f, fp, j |
| *italic-nudge | `0 mu` | f, fp, j |
| *slash-sep | `slash-sep` | s, sp |
| *inner-wrap | `false` | s, sp |
| *inner-Ldelim | `(` | s, sp |
| *inner-Rdelim | `)` | s, sp |
| *outer-Ldelim | `\big [` | s, sp |
| *outer-Rdelim | `\big ]` | s, sp |
| *sub-nudge | `0 mu` | s, sp |

**op-symbol** the operator symbol; for ordinary derivatives generally one of `d` or `\mathrm{d}`, for partial derivatives `\partial`; default = `\mathrm{d}`

**op-symbol-alt** if different from **op-symbol** then used in the denominator of a fraction-form derivative while `op-symbol` is used in the numerator; e.g. for acceleration $\frac{\nabla v^i}{\mathrm{d}t}$, `op-symbol = \nabla`, `op-symbol-alt = \mathrm{d}`; default = **op-symbol**

**op-order-nudge** extra horizontal space added between the operator symbol and the superscripted order of differentiation in higher order derivatives; for math-italic forms compare $d^2$ (no extra space) with $d^2$ (`1 mu` of extra space) ; since **op-symbol** defaults to an upright 'd', default = `0 mu`

**op-sub-nudge** horizontal adjustment of the position of the subscript in derivatives of compact form relative to the operator; e.g. if `\nabla` is the operator you might prefer $\nabla_x y$ (`-4 mu`) to $\nabla_x y$ (`0 mu`); since `\mathrm{d}` is the default operator, default = `0 mu`

**var-sup-nudge** extra horizontal space added between a variable in the denominator of a derivative and the superscripted order of differentiation in higher order derivatives; compare $\mathrm{d}T^2$ (`0 mu`) with $\mathrm{d}T^2$ (`1 mu`); default = `1 mu`

**multi-term-sep** horizontal spacing inserted between the differentials in, for example, the denominator of a mixed partial derivative to avoid a solid cluster like $\partial x\partial y\partial z$; with the default `2 mu` this is spread a little, $\partial x\partial y\partial z$; default = `2 mu plus 1 mu minus 1 mu`

**term-sep-adjust** adjustment (usually a reduction) to **multi-term-sep** when differentiation in a variable occurs to an order other than 1; e.g. the spacing between $\partial x^2$ and $\partial y$ in $\partial^4 F/\partial x^2\partial y\partial z$ is reduced by **term-sep-adjust**, because of the superscript, from the spacing between $\partial y$ and $\partial z$; default = `-1 mu`

**dots** dot command (e.g. `\ldots`, `\cdots`, ...) to use in a generic mixed partial derivative ($\partial^n F/\partial x_1 \ldots \partial x_n$). Because an upright fraction is the default form of derivative, default = `\cdots`

**long-var-wrap** it may aid clarity and avoid ambiguity in higher order derivatives to wrap multi-token variables of differentiation in parentheses; the choices are

- `dv` no wrapping, e.g. $\mathrm{d}x_i^2$ or $\partial\frac{1}{\Theta}^2$,
- `d(v)` wrap the variable only, e.g. $\mathrm{d}(x_i)^2$ or $\partial(\frac{1}{\Theta})^2$,
- `(dv)` wrap both op-symbol and variable, e.g. $(\mathrm{d}x_i)^2$ or $(\partial\frac{1}{\Theta})^2$;

default = `d(v)`

**lvwrap-Ldelim** left delimiter when wrapping a long variable, in a higher order derivative; also applies to the left delimiter used in a jacobian; default = `\mleft (`

**lvwrap-Rdelim** right delimiter when wrapping a long variable, in a higher order derivative; also applies to the right delimiter used in a jacobian; default = `\mright )`

**lvwrap-sup-nudge** horizontal adjustment to the superscript position in the denominator of a higher order derivative when a multi-token variable is wrapped in (e.g.) parentheses; default = `-2 mu`

**outer-Ldelim** the left member of a delimiter pair wrapping the derivative, the right member of which is subscripted to indicate a point of evaluation or variables held constant; ISO recommends parentheses for this purpose, hence default = `\left (`

**outer-Rdelim** the right member of a delimiter pair wrapping the derivative and subscripted to indicate a point of evaluation or variables held constant; ISO recommends parentheses for this purpose, hence default = `\right )`

**elbowroom** adjustment to the whitespace between the delimiter pair **outer-Ldelim**, **outer-Rdelim** and the enclosed derivative; negative values reduce the space; default = `0 mu`

**sub-nudge** horizontal adjustment of the subscript's placing relative to the **outer-Rdelim**iter for a point of evaluation or variable held constant; a negative value compensates for the curving inwards of a large right parenthesis; default = `-5 mu`

**difcc-var-ord** how the order of differentiation is indicated in the 'doubly compact' form of a compact form derivative (see §3.3.4.8); default = `^{#1}`

***derivand-sep** when the derivand is appended, horizontal space added before the differentiand (derivand) depending on the setting of the `spaced` package option; default = **derivand-sep**

***op-set-left** a choice of `true` or `false` indicating whether the op-symbol is left-aligned or not when the differentiand is appended; generally it is centred; applies only to upright-fraction forms of the derivative; default = `false`

***italic-nudge** if ***op-set-left** is `true`, makes an italic adjustment in the numerator, so that the op-symbols in numerator and denominator align in the same slanting column; for `d` or `\partial` an appropriate value might be `3 mu`; because of the default `\mathrm{d}`, default = `0 mu`

***slash-sep** when the derivand is appended, space inserted on either side of the **slash-tok**; if two values are given (in a comma list), the first is inserted to the left of **slash-tok**, the second to the right; if only one value is provided, it is inserted on both sides of **slash-tok**; default = **slash-sep**

***inner-wrap** when the differentiand is appended, a choice of `true` or `false` dictating whether the differential operator is wrapped in parentheses, as here $(\partial/\partial x)\,F(x,y)$, or not; for a slash-fraction derivative `true` is an appropriate default, but the overall default, appropriate for an upright-fraction derivative = `false`

***inner-Ldelim** if ***inner-wrap** is `true`, the left member of a delimiter pair around the differential operator; default = `(`

***inner-Rdelim** if ***inner-wrap** is `true`, the right member of a delimiter pair around the differential operator ; default = `)`

***outer-Ldelim** if ***inner-wrap** is `true`, the left member of a delimiter pair around both the differential operator and appended differentiand, the right member of which may be subscripted to indicate a point of evaluation or variables held constant; to avoid too many parentheses, given the default values of ***inner-Ldelim**, ***inner-Rdelim**, default = `\bigl [`

***outer-Rdelim** if ***inner-wrap** is `true`, the right member of a delimiter pair around the differential operator and appended differentiand; may be subscripted to indicate a point of evaluation or variables held constant; to avoid too many parentheses, given the default values of ***inner-Ldelim**, ***inner-Rdelim**, default = `\bigr ]`

***sub-nudge** if ***inner-wrap** is `true`, horizontal adjustment of the subscript's placing relative to the ***outer-Rdelim**iter for a point of evaluation or variable held constant; a negative value compensates for the curving inwards of a large right parenthesis; since the default ***outer-Rdelim** is a square bracket, default = `0 mu`

### 3.2.1 Ordinary upright-fraction derivatives; template `DIFF`

The defaults assigned in template `DIF` are inherited by template `DIFF` without change. Template `DIFF` is therefore strictly unnecessary but, with templates `DIFS` and `DIFC` in mind, was created for the sake of a consistent naming scheme.

The `\diff` command uses the values in the `DIFF` template to form an upright-fraction derivative. Only keys with an 'f' in the third column of Table 3.2 are used in this process. Keys without an 'f' play no part in the process and their default values are ignored. See §5.2.2 for the complete list of *relevant* `DIFF` defaults.

### 3.2.2 Ordinary slash-fraction derivatives; template `DIFS`

When you use the command `\difs` to form a slash-fraction derivative it is the keys in template `DIF` with an 's' in the third column of Table 3.2 which are used. Table 3.3a records those keys assigned default values *different* from those in `DIF` which are used for this purpose. See §5.2.3 for the complete list of *relevant* `DIFS` defaults.

Table 3.3: Defaults differing from the parent template

(a) `DIFS`

| key | default |
|---|---|
| style | `/` |
| derivand-sep | `2 mu plus 1 mu minus 2 mu` |
| dots | `\ldots` |
| outer-Ldelim | `(` |
| outer-Rdelim | `)` |
| sub-nudge | `0 mu` |
| *inner-wrap | `true` |

(b) `DIFC`

| key | default |
|---|---|
| style | `-` |
| derivand-sep | `1 mu plus 1 mu minus 1 mu` |
| multi-term-sep | `1 mu` |
| term-sep-adjust | `0 mu` |
| outer-Ldelim | `\bigl (` |
| outer-Rdelim | `\bigr )` |
| sub-nudge | `-2 mu` |

(c) `DIFFP`

| key | default |
|---|---|
| op-symbol | `\partial` |
| op-order-nudge | `1 mu` |
| *italic-nudge | `3 mu` |

(d) `DIFSP`

| key | default |
|---|---|
| op-symbol | `\partial` |
| op-order-nudge | `1 mu` |

(e) `DIFCP`

| key | default |
|---|---|
| op-symbol | `\partial` |
| op-order-nudge | `1 mu` |

### 3.2.3 Ordinary compact-form derivatives; template DIFC

When you use the command \difc to form a compact derivative it is the keys in template DIF with a 'c' in the third column of Table 3.2 which are used. Table 3.3b records those keys assigned default values *different* from those in DIF which are used for this purpose. See §5.2.4 for the complete list of *relevant* DIFC defaults.

### 3.2.4 Partial derivatives; templates DIFFP, DIFSP, DIFCP

The default values given in the tables so far apply to ordinary derivatives. For *partial* derivatives, only a few defaults change. These are listed in Tables 3.3c, 3.3d, 3.3e. All other keys take the default values of the respective parent templates, DIFF, DIFS and DIFC.

## 3.3 Defining variants, changing defaults

You may well have the need to write an occasional derivative that deviates from the default form – a *variant* form. That was what I needed to do to write the range of different examples displayed in the Rogues' Gallery (§1.2). Alternatively, you may be dissatisfied with the scheme of default values listed in the preceding tables and wish to 'Re-mould it nearer to the Heart's Desire'. Both processes – changing a default, defining and using a variant – are very similar and make use of a command \difdef which has three *mandatory* arguments:

```
\difdef{id-list}{variant-name}{key-value list}
```

1. id-list A comma-list of identifiers, one or more of f, s, c, fp, sp, cp, j, l distinguishing the respective templates DIFF, DIFS, DIFC, DIFFP, DIFSP, DIFCP, DIFJ and DIFL (for the last two see Chapter 4).

2. variant-name A (preferably brief) name for the variant form; it may include characters other than letters, like numbers, punctuation marks (excluding full stops), mathematical symbols like + and =, but not control sequences or active characters, nor \, % , # or braces.

   (a) If a name is not provided – if the variant-name braces are left empty – then the changes signalled in key-value list, the third argument, become the new *default values* of the derivatives identified in the first argument, id-list.

3. key-value list A *key=value* list of settings differing from the default settings for the relevant template or templates (as determined by the id-list).

The present document comes with a number of variant definitions. These are divided into two groups. One, in the preamble (and reproduced at §5.4), contains definitions designed to illustrate various effects for this document – as in the

Rogues' Gallery. The other, in the associated file `diffcoeff5.def`, contains definitions that may be of more general use. (For use of a `.def` file see §3.3.3; `diffcoeff5.def` is reproduced at §5.3.)

The command `\difdef` in version 5 of `diffcoeff` takes *three* mandatory arguments. Do not confuse with the command `\diffdef` of version 4 and earlier which took *two* mandatory arguments for this purpose. The additional argument is required to identify which one or more of the fraction forms of the commands `\difx`, `\difxp`, the variant applies to. In earlier versions this was not necessary since there was only the one primary derivative command `\diff`.

### 3.3.1 Creating and using a variant

To illustrate the process of creating and using a variant consider the definition in the file `diffcoeff5.def` enabling the writing of the acceleration in tensor calculus. This has `\nabla` in the numerator and `\mathrm{d}` in the denominator:

```
\difdef { f, s } { n }
  {
    op-symbol     = \nabla,
    op-symbol-alt = \mathrm{d},
    *slash-sep    = { -2mu, 0mu }
  }
```

The second argument shows that this variant has been given the name `n`, obviously from 'nabla', and can be used with both upright and slash fraction derivatives – the `f` and the `s` in the first argument. Using the new variant is as simple as writing `\diff.n.` or `\difs.n.` followed by the usual arguments:

```
\[ \diff.n.{v^{i}}t,\quad \difs.n.{v^{i}}t,\quad
   \diff.n.*{v^{i}}t,\quad \difs.n.*{v^{i}}t. \]
```

$\Longrightarrow$

$$\frac{\nabla v^i}{\mathrm{d}t}, \quad \nabla v^i/\mathrm{d}t, \quad \frac{\nabla}{\mathrm{d}t} v^i, \quad (\nabla/\mathrm{d}t)\, v^i.$$

(In the Rogues' Gallery I also included the command `\spacednil` inside the `\[` to cancel the extra space inserted before the differentiand (§2.2.1) with the `spaced=-1` package option used for this document.)

The fourth of these derivatives makes use of the `*slash-sep` setting which applies only when the differentiand is appended and only to slash-fraction derivatives. Like `slash-sep`, it takes either a single value which is applied to both sides of the slash token or, as here, a comma list (hence between braces) of two values applied respectively to the left and the right side of the slash token. By default it is set to the value of the `slash-sep` setting but for this variant the `-2 mu` avoids the appearance of too much whitespace between the similarly sloping $\nabla$ and / symbols.

A second example shows the effect of the key `sub-nudge` which adjusts the positioning of the subscript denoting a point of evaluation. The preamble contains the definition,

```
\difdef { f, fp } { wsp } { sub-nudge = 0 mu }
```

the name `wsp` derived from 'whitespace'. The definition applies only to upright fraction forms of derivative (the `f, fp` in the first argument). The default value of `sub-nudge` is `-5 mu` which has the effect of moving the point-of-evaluation subscript in towards the in-curling right parenthesis. By giving `sub-nudge` a zero value, the subscript is cast adrift in a sea of whitespace. Compare the two settings, using `\diffp` and `\diffp.wsp.` side-by-side:

```
\[ \diffp Fx[0],\quad \diffp.wsp.Fx[0]. \]
```

$\Longrightarrow$

$$\left(\frac{\partial F}{\partial x}\right)_0, \quad \left(\frac{\partial F}{\partial x}\right)_0.$$

### 3.3.2 Changing a default

The process of changing a default is even simpler than creating a variant: the second argument of the `\difdef` command is left empty and no dot-delimited name is required for the `\difx`, `\difxp` commands.

By default, `diffcoeff` parenthesizes multi-token variables in the denominators of higher order derivatives – this sort of thing

$$\frac{\mathrm{d}^2 \ln \sin x}{\mathrm{d}(\sin x)^2}$$

in order to avoid this sort of thing

$$\frac{\mathrm{d}^2 \ln \sin x}{\mathrm{d} \sin x^2},$$

where the superscript in the denominator looks as if it is attached to $x$ rather than $\sin x$ (or $\mathrm{d} \sin x$). Nonetheless you may still prefer to have *no* parentheses inserted, the superscript in the numerator resolving any ambiguity in the denominator. To make this the default behaviour, all you need is the following definition

```
\difdef { f, fp, s, sp } {} { long-var-wrap = dv }
```

placed either in the preamble of your document or your `.def` file (see below). Then when you next form a derivative like `\diff[2]{\!\ln\sin x}{\sin x}` involving a multi-token variable in a higher-order derivative, no parenthesizing will occur; see §3.3.4.4 for more on this.

The procedure is general if you wish to change a default: use `\difdef` with an *empty* second argument.

### 3.3.3 The `.def` file

The problem with the procedures just discussed is that if you want the new definitions of variants or defaults to also apply in other documents, you are faced with either entering them anew or copying and pasting from one preamble to another.

It is much easier to use a `.def` file. This is a text file containing a list of definitions of variant derivatives or new defaults after the fashion of the examples above. The reason for placing such variant definitions in a file is that they can be easily transferred from document to document by means of the `def-file` package option. If the name of your `.def` file is `myfile`, then invoking `diffcoeff` with the call

```
\usepackage[def-file=myfile]{diffcoeff}
```

makes the definitions in `myfile.def` available to your current document – provided `diffcoeff` can find `myfile.def`.

One obvious way is to include the full path in the name. If you are using a windows operating system then the path separator is the backslash \, e.g. `E:\MyDocuments\...`. To avoid 'Undefined control sequence' errors when compiling, these backslashes must be changed to forward slashes, as on linux systems: `E:/MyDocuments/...`. Including the full path in the filename means the file can be placed anywhere you like.

If you don't use the full path in the filename, then the question is where to put the file so that `diffcoeff` can find it? The directory of the current document is an obvious candidate and for the current document serves well, but it does mean copying the `.def` file from directory to directory to work on *different* documents. To make a definition file available for *all* documents, place it in the `texmf` tree, preferably not the one created by your TEX distribution, but your own *personal* `texmf` tree. Provided your TEX distribution knows about your personal `texmf` tree and the files it contains – which may mean explicitly refreshing the filename database of your TEX distribution – then a `.def` file placed within it will be accessible to all documents.

**Personal `texmf` tree?**

This is a directory created by you for 'waifs and strays' of the TEX system that are not included in standard distributions like MiKTEX or TEXLive. For instance, it is the place for personal packages designed for your own particular circumstances or preferences, and is structured like the standard `texmf` hierarchy but placed in another location so that there is no chance of its being overwritten when your TEX distribution is updated. But that distribution needs to be alerted to the existence of your personal `texmf` tree and any new files added to it. TEX distributions should have information accessible on how to add your personal `texmf` tree to its search path and how to refresh its filename database when you add a file to your `texmf` tree.

Be aware that it is best to place any particular `.def` file in no more than *one* place in the search path – no copies with the same name in different places. It is too easy to change only one such copy and forget to update the others and then wonder why the change you made is not reflected in your document – `diffcoeff` has found a file with the same name, just not the one with the changed definitions.

If the `.def` file named in the package option statement cannot be located by TEX, a message to that effect is sent to the terminal and log file, but `diffcoeff` continues loading.

**`diffcoeff.def`**

In version 4 of `diffcoeff`, if there was no explicit `def-file=<filename>` package option statement, then a file `diffcoeff.def` was sought and if found loaded. This is no longer the case. Version 5 of `diffcoeff` searches for a `.def` *only if it is explicitly named* in a package option statement. (This decision was made at least in part to avoid conflict with a `diffcoeff.def` file from an earlier version of `diffcoeff` tucked away in some non-obvious place and producing obscure errors in the current version 5.)

The present document uses a `.def` file `diffcoeff5.def`, reproduced in §5.3. Because the author has a number of files with this name on his computer, the full path to the file was included in the `\usepackage` statement in the preamble.

### 3.3.4 Examples of variants

The dot-delimited name argument must always be the *first* argument of the `\difx` or `\difxp` command, even preceding an asterisk (star) signalling 'append the differentiand'. Now for some examples.

#### 3.3.4.1 Point of evaluation

Although ISO recommends subscripting parentheses to indicate a point of evaluation, some (like the author) prefer to use a vertical rule with a subscript and save parentheses for the case of variables held constant in partial derivatives. The file `diffcoeff5.def` contains the definition

```
\difdef { f, fp, s, sp } { | }
  {
    outer-Ldelim  = \left . ,
    outer-Rdelim = \right |,
    sub-nudge    = 0 mu
  }
```

where the 'pipe' character is used for the name of the variant:

```
\[ \diffp.|.{F(x,y)}x[x=1] \] ⟹
```

$$\left.\frac{\partial F(x,y)}{\partial x}\right|_{x=1}$$

By omitting the 'pipe' symbol name from the second argument in the `\difdef` command, this variant would become the default (for `\diff`, `\diffp`, `\difs`, `\difsp`) for indicating a point of evaluation. For slash fractions,

`$ \difs yx[0],\quad \difs.|.yx[0] $` ⟹ $(\mathrm{d}y/\mathrm{d}x)_0, \quad \mathrm{d}y/\mathrm{d}x|_0$.

As a *default*, you might prefer to limit use of the 'pipe' to `\diff`, `\diffp` (by omitting the `s, sp` identifiers from the first argument of the `\difdef` command). Parentheses neatly tie the whole expression together. Conversely, it is easy to create expressions that suffer from 'parenthesis overload':

```
$ \difs{F(x)}{(1/x)}[x=1],\quad\difs.|.{F(x)}{(1/x)}[x=1] $ ⟹
```

$$(\mathrm{d}F(x)/\mathrm{d}(1/x))_{x=1}, \quad \mathrm{d}F(x)/\mathrm{d}(1/x)|_{x=1}$$

The vertical rule is clearer in this case. The solution may be to use *square* brackets. The file `diffcoeff5.def` contains the definition

```
\difdef { f, fp, s, sp } { ] }
  {
    outer-Ldelim = \left [ ,
    outer-Rdelim = \right ],
    elbowroom    = 1 mu,
    sub-nudge    = 0 mu
  }
```

giving the result

`$ \difs.].{F(x)}{(2x)}[x=0] $` ⟹ $[\mathrm{d}F(x)/\mathrm{d}(2x)]_{x=0}$,

which both avoids 'parenthesis overload' and is 'tied together' by the square brackets and at least gives a nod in the direction of the ISO standard. If you want more separation between the square brackets and the differential coefficient they enclose, increase the value of the `elbowroom` key – perhaps to `2 mu`.

#### 3.3.4.2 Math-italic 'd's

If you prefer math-italic 'd's rather than the upright ISO recommendation, then the file `diffcoeff5.def` contains the definition

```
\difdef { f, s, c, l } { d' }
  {
    op-symbol      = d ,
    op-order-nudge = 1 mu,
    *slash-sep     = { 0 mu, -1 mu }
  }
```

The list of identifiers includes `l` in the first argument; this refers to the template `DIFL` of the differential – see §4.1.

```
\[ \diff.d'.yx,\quad \difs.d'.yx,\quad \difc.d'.yx. \]
```

$\Longrightarrow$

$$\frac{dy}{dx}, \quad dy/dx, \quad d_x y.$$

If you want this form of 'd' to be the *default* leave the second argument of the `\difdef` command empty.

#### 3.3.4.3 Generic slash fraction with \cdots

In §2.2.5.5 it was remarked that it was not clear whether to use `\ldots` or `\cdots` as default in a generic *slash-fraction* form of mixed partial derivative. In fact `\ldots` has been chosen as the `diffcoeff` default, $\partial^n F/\partial x_1 \ldots \partial x_n$, but it is easy to provide a variant that uses `\cdots`:

```
\difdef { sp } { cd } { dots = \cdots }
```

Indeed, just this definition can be found in the file `diffcoeff5.def` so let's try it:

```
$ \difsp.cd.<n>F{x_{1},...,x_{n}} $
```
$\Longrightarrow \partial^n F/\partial x_1 \cdots \partial x_n.$

#### 3.3.4.4 Parenthesizing multi-token variables

To illustrate the different modes of parenthesizing 'long' variables in higher order derivatives, I have put these two definitions in `diffcoeff5.def`:

```
\difdef { f, fp } { (dv) }
  { long-var-wrap = (dv) }

\difdef { f, fp }{ dv }
  { long-var-wrap = dv }
```

The three possibilities for wrapping multitoken variables can now be illustrated:

```
\[ \diffp[2]F{1/T},\quad
       \diffp.dv.[2]F{1/T},\quad
           \diffp.(dv).[2]F{1/T} \]
```

$\Longrightarrow$

$$\frac{\partial^2 F}{\partial(1/T)^2}, \quad \frac{\partial^2 F}{\partial 1/T^2}, \quad \frac{\partial^2 F}{(\partial 1/T)^2}$$

The first is the default setting; it takes a moment to decipher $T^2$ in the second and the third is unfamiliar. In other cases it is less clear-cut.

**Subscripted variables** Consider the following example, where the 'out of the box' default means multi-token variables, and hence the subscripted variables in the example, are parenthesized:

```
\[ \sum_{i=0}^3 \diffp[2]V{x_i}=\diffp[2]V{x_1}
   +\diffp[2]V{x_2}+\diffp[2]V{x_3}. \]
```

$\Longrightarrow$

$$\sum_{i=0}^{3} \frac{\partial^2 V}{\partial (x_i)^2} = \frac{\partial^2 V}{\partial (x_1)^2} + \frac{\partial^2 V}{\partial (x_2)^2} + \frac{\partial^2 V}{\partial (x_3)^2}.$$

You may well prefer no parenthesizing here – the `dv` variant:

```
\[ \sum_{i=0}^3 \diffp.dv.[2]V{x_i}=\diffp.dv.[2]V{x_1}
   +\diffp.dv.[2]V{x_2}+\diffp.dv.[2]V{x_3}. \]
```

$\Longrightarrow$

$$\sum_{i=0}^{3} \frac{\partial^2 V}{\partial {x_i}^2} = \frac{\partial^2 V}{\partial {x_1}^2} + \frac{\partial^2 V}{\partial {x_2}^2} + \frac{\partial^2 V}{\partial {x_3}^2}.$$

But are the superscripts in the denominators too detached now?

When there is no parenthesizing, the `var-sup-nudge` setting is used to adjust this spacing. The default value, `1 mu`, is intended for differentiation variables which are *not* subscripted. So let's adjust the setting for variables which *are* subscripted and define a variant with name, `dv_`, say, the name suggesting 'not parenthesized and subscripted':

```
\difdef { f, fp, s, sp } {dv_}
  {
    long-var-wrap = dv,
    var-sup-nudge = -1 mu
  }
```

This definition can be found in the file `diffcoeff5.def`, giving the result

```
\[ \sum_{i=0}^3 \diffp.dv_.[2]V{x_i}=
       \diffp.dv_.[2]V{x_1}+\diffp.dv_.[2]V{x_2}+
           \diffp.dv_.[2]V{x_3}. \]
```

$\Longrightarrow$

$$\sum_{i=0}^{3} \frac{\partial^2 V}{\partial {x_i}^2} = \frac{\partial^2 V}{\partial {x_1}^2} + \frac{\partial^2 V}{\partial {x_2}^2} + \frac{\partial^2 V}{\partial {x_3}^2}.$$

The superscripts are no longer detached from the variables.

**Superscripted variables** *Super*scripted variables, as you get in tensor calculus, unavoidably give less elegant results. The three possibilities are

```
\[ \diffp[2]f{x^1},\quad
       \diffp.dv.[2]f{x^1},\quad
           \diffp.(dv).[2]f{x^1}. \]
```

$\Longrightarrow$

$$\frac{\partial^2 f}{\partial(x^1)^2}, \quad \frac{\partial^2 f}{\partial x^{1\,2}}, \quad \frac{\partial^2 f}{(\partial x^1)^2}.$$

The third of these is unlikely to be favoured. The first (the default) to my eye gives a noticeably better result for the Laplacian than the second:

```
\[ \sum_{i=1}^3\diffp[2]V{x^i}=
       \diffp V{x^1:2}+\diffp V{x^2:2}+
           \diffp V{x^3:2}, \]
\[ \sum_{i=1}^3\diffp.dv.[2]V{x^i}=
       \diffp.dv.V{x^1:2}+\diffp.dv.V{x^2:2}+
           \diffp.dv.V{x^3:2}. \]
```

$\Longrightarrow$

$$\sum_{i=1}^{3} \frac{\partial^2 V}{\partial(x^i)^2} = \frac{\partial^2 V}{\partial(x^1)^2} + \frac{\partial^2 V}{\partial(x^2)^2} + \frac{\partial^2 V}{\partial(x^3)^2},$$

$$\sum_{i=1}^{3} \frac{\partial^2 V}{\partial x^{i\,2}} = \frac{\partial^2 V}{\partial x^{1\,2}} + \frac{\partial^2 V}{\partial x^{2\,2}} + \frac{\partial^2 V}{\partial x^{3\,2}}.$$

#### 3.3.4.5 Upright text-style derivatives

`diffcoeff` assumes that derivatives of upright-fraction form will be used mainly in display-style expressions and that the slash form will be used mainly for inline (text-style) use. But if one does want to use the upright form in an inline expression, then `$ \diffp ST $` displaying as $\frac{\partial S}{\partial T}$ is fine, but adding a trailing optional argument, `$ \diffp ST[V] $`, to indicate (in the present example) a variable held constant is not: $\left(\frac{\partial S}{\partial T}\right)_V$. Clearly the subscript is too close to the right parenthesis and (to my eye) there is too much 'elbowroom' between the derivative and the enclosing parentheses. Hence the file `diffcoeff5.def` contains the following definition for text-style upright fraction derivatives (note that since the `\tfrac` command is used here, this requires the `amsmath` package to have been loaded):

```
\difdef { f, fp } { t }
  {
    style           = tfrac  ,
    derivand-sep    = 1 mu plus 1 mu minus 1 mu,
    multi-term-sep  = 0 mu   ,
    term-sep-adjust = 0 mu   ,
```

```
    wrap-sup-nudge  = 0 mu   ,
    outer-Ldelim    = \bigl (,
    outer-Rdelim    = \bigr ),
    elbowroom       = -2 mu  ,
    sub-nudge       = -3 mu
  }
```

Now the variant form `$ \diffp.t.ST[V] $` displays as $\left(\frac{\partial S}{\partial T}\right)_V$; the subscript is better positioned and there is a better fit between parentheses and derivative. Note that `style=tfrac` in the definition means `\diffp.t.` will not scale.

#### 3.3.4.6 Split-level text-style derivative

A 'halfway house' between upright and slash-fraction forms for inline derivatives is a split-level slash fraction, available from version 5.5 of `diffcoef`.[3] (This requires the `amsmath` package to have already been loaded.) Then

```
\difdef { s, sp } { s }
  {
    style             = sfrac,
    slash-sep         = { -2 mu, -1 mu },
    derivand-sep      = 0 mu ,
    multi-term-sep    = 0 mu ,
    term-sep-adjust   = 0 mu ,
    var-sup-nudge     = 0 mu ,
    lvwrap-sup-nudge  = 0 mu
  }
```

defines inline derivatives of the form

```
\[ \difs.s.yx,\quad\difs.s.yx[0],\quad
     \difsp.s.*yx,\quad\difsp.s.*yx[0]. \]
```

$\Longrightarrow$

$$ {\rm d}y/{\rm d}x, \quad ({\rm d}y/{\rm d}x)_0, \quad (\partial/\partial x)y, \quad \big[(\partial/\partial x)y\big]_0. $$

The $y$ in the first two is `\scriptstyle` size but when appended and hence outside the fraction it is normal size. Since it remains within the fraction in all four examples, the $x$ is `\scriptstyle` size. Should you need to write a derivative as an exponent, this split-level slash fraction form gives a neater result than a pure slash fraction or an upright fraction:

```
$ e^{\difs.s.yx},\quad e^{\difs yx},\quad e^{\diff yx} $
```

$\Longrightarrow e^{{\rm d}y/{\rm d}x}, \quad e^{{\rm d}y/{\rm d}x}, \quad e^{\frac{{\rm d}y}{{\rm d}x}}$.

In LaTeX if your default font size is 10pt, then script-style size is 7pt ($= 10 \times 0.7$) and script-script-style size is 5pt ($\approx 10 \times 0.7 \times 0.7$), which explains the default value 0.7 of **sfrac-scaling**. If your default font size is 11pt or 12pt then in either case the script-style size is 8pt and script-script-style size is 6pt.

[3] I have copied and modified code defining the `\sfrac` command from the `xfrac` package; hence the `style=sfrac` line in the definition.

**Editing a variant form** If you wish to edit an already defined variant form, you don't need to repeat the full definition. It suffices to change the setting only of the relevant key or keys – but please ensure your TEX distribution is from 2022-12-17 or later. Earlier versions of `xtemplate` contained a bug that didn't otherwise affect the workings of `diffcoeff` but did prevent changes being made to already defined variants.

Thus, to the definition just given, in `diffcoeff5.def` I have made the change

```
\difdef { s } { s } { *slash-sep = { -1 mu, -1 mu } }
```

to accommodate the different way the upright d relates to the slash symbol in the numerator of an appended derivative compared with the math-italic $\partial$:

`$ \difs.s.*yx,\quad\difs.s.*yx[0] $` $\Longrightarrow$ $(\mathrm{d}/\mathrm{d}x)y, \quad \big[(\mathrm{d}/\mathrm{d}x)y\big]_0$.

**\sfrac analogue** The `xfrac` package defines a command `\sfrac` that produces split-level fractions. The thought arises that perhaps by omitting the operator symbol in `\difs.s.` we could create an analogue of `\sfrac`. Indeed, from version 5.5 of `diffcoeff`, the file `diffcoeff5.def` contains the definition

```
\NewDocumentCommand \difsfrac { O{-2mu,-1mu} m m }
  {{
     \difdef { s } { s } { op-symbol= , slash-sep={#1} }
     \difs.s.{#2}{#3}
  }}
```

The reason for the double braces in the definition is so that the changes to `\difs.s.` do not 'leak' beyond `\difsfrac` and affect a later use of `\difs.s.`. Indeed, `$ \difsfrac yx,\ \difs.s.yx $` $\Longrightarrow$ $y/x,\ \mathrm{d}y/\mathrm{d}x$, so that there is no 'leakage'. `\difsfrac` can be used within or without math delimiters,

`\difsfrac 34, $\difsfrac 1{16}$` $\Longrightarrow$ $3/4,\ 1/16$.

If you want to adjust the spacing of numerator or denominator from the slash, an initial optional argument is available taking two math lengths in mu, separated by a comma. Generally these lengths will be negative to bring numerator and denominator closer to the slash; positive lengths push them away, like this: `\difsfrac[10mu,10mu] 34` $\Longrightarrow$ $3\ /\ 4$.

#### 3.3.4.7 Slash-fraction styles

The default slash-fraction form `$ \difs yx $` displaying as $\mathrm{d}y/\mathrm{d}x$ does not scale. It is intended for inline use, but sometimes you may want a slash fraction of a different size – perhaps a fraction is present in the differentiand or in the variable of differentiation. The file `diffcoeff5.def` contains a definition of a scaling slash fraction (name `0`) and a slightly larger-than-default slash fraction (name `1`):

```
\difdef { s, sp } { 0 }
  {
    style         = auto     ,
    outer-Ldelim  = \left [  ,
    outer-Rdelim  = \right ] ,
    sub-nudge     = 0 mu     ,
    *inner-Ldelim = \mleft ( ,
    *inner-Rdelim = \mright ),
    *outer-Ldelim = \left [  ,
    *outer-Rdelim = \right ]
  }
```

```
\difdef { s, sp } { 1 }
  {
    style          =  big   ,
    outer-Ldelim   = \bigl (,
    outer-Rdelim   = \bigr ),
    sub-nudge      = -2   mu,
    *inner-Ldelim  = \bigl (,
    *inner-Rdelim  = \bigr ),
    *outer-Ldelim  = \bigl [,
    *outer-Rdelim = \right ]
  }
```

The names arise from the sequence `\big/`, `\Big/`, `\bigg/`, `\Bigg/`, hence `1`, `2`, `3`, `4`, which leaves `0` for the scaling form `auto` (which is built around `\left.`, `\middle/`, `\right.`). `diffcoeff5.def` does not contain definitions for the `2`, `3`, `4` variants, only the two shown, because the larger sizes give ridiculous results. For the scaling variant, it is also easy to produce eyesores:

`\[ \difsp.0.{\frac1Y}{\frac1X} \]` $\Longrightarrow$

$$\partial \frac{1}{Y} \Big/ \partial \frac{1}{X}$$

But for small size increases, the results can be pleasing. To the author's eye, both `0` and `1` variants (the first two of the following) give better results than the default (the last):

`$ \difsp.1.{F(x,y)}{\tfrac1x}[x=0] $` $\Longrightarrow \big(\partial F(x,y)/\partial \frac{1}{x}\big)_{x=0}$

`$ \difsp.0.{F(x,y)}{\tfrac1x}[x=0] $` $\Longrightarrow \big[\partial F(x,y)/\partial \frac{1}{x}\big]_{x=0}$

`$ \difsp{F(x,y)}{\tfrac1x}[x=0] $` $\Longrightarrow (\partial F(x,y)/\partial \frac{1}{x})_{x=0}$

Subscripted *square* brackets are chosen for the scaling variant so that the setting `sub-nudge=0 mu` is appropriate at all scales. They provide good visual contrast with the parentheses of $F(x,y)$.

#### 3.3.4.8 Compact–form derivatives

Three styles are available for compact-form derivatives, `style=_`, the default as displayed in $\mathrm{d}_x y$; `style=dl`, a 'differential style' as displayed in $\mathrm{d}x^2$ (with no differentiand), and from version 5.5 of `diffcoeff`, a 'doubly compact' style, `style=cc` as displayed in $\partial_{x^2yz^3} f$. (Nominally there is a fourth style, `style=d^`, but that is equivalent to `style=_`.)

In the default style (for a partial derivative) the orders of differentiation are applied to the operator symbol:

`$ \difcp[3,1,2]f{x,y,z} $` $\Longrightarrow \partial_x^3 \partial_y \partial_z^2 f\,.$

If no differentiation variable is specified, only an empty brace pair,[4]

$$\texttt{\$ \difc[2]f\{\} \$} \Longrightarrow \mathrm{d}^2 f\,.$$

(One could also write `$ \difc f{:2} $` to get this result.) If you want to write 'differentials of differentials', second order differentials, then this is a possible way of doing so – but `diffcoeff` has a dedicated differential command, `\dl`, see §4.1, and in relation to the example, see §4.1.3.

Suppose now we define a variant form (as is done in `diffcoeff5.def`),

```
\difdef { cp } { dl }
  {
    style           = dl   ,
    multi-term-sep  =  1 mu,
    term-sep-adjust = -1 mu
  }
```

and use it to form a similar expression but without the differentiand this time, an empty argument in *its* place:

$$\texttt{\$ \difcp.dl.[3,2]\{\}\{x,y,z\} \$} \Longrightarrow \partial x^3 \partial y^2 \partial z$$

In this `dl` (or *differential*) style, the orders of differentiation in the displayed output are applied to the *variables*. This allows discussion of, for example, the denominator of a mixed partial derivative – perhaps a remark about minutiae of spacing. (But see §4.1, on differentials, which more conveniently allows the writing of, for example, $\mathrm{d}x^3$.)

**Doubly compact forms** In the search for ever more concision, one might wonder if a form like $\partial_x^3 \partial_y \partial_z^2 f$ can be compressed further to use only one operator symbol. This is the purpose of `style=cc`. The file `diffcoeff5.def` contains the definition

```
\difdef { cp } { cc }
  {
    style           =  cc  ,
    multi-term-sep  =  0 mu,
    term-sep-adjust = -1 mu,
    var-sup-nudge   =  0 mu
  }
```

Now one can write `$\difcp.cc.f{x:3,y,z:2}$` $\Longrightarrow \partial^6_{x^3yz^2} f$. Is the total order – the 6 – really necessary? We can use the order-override option to suppress it: `$\difcp.cc.<1>f{x:3,y,z:2}$` $\Longrightarrow \partial_{x^3yz^2} f$, which to my eye is cleaner. You can make this your *default* compact derivative by omitting the `cc` from the second argument of `\difdef`, leaving it empty; then it would suffice to write `$\difcp<1>f{x:3,y,z:2}$` to obtain the last result.

[4] This is the behaviour from version 5.2 of `diffcoeff`; in version 5.1, a nested pair of empty braces, {{}}, was required.

But do you want always to be writing `<1>` to suppress the superfluous total order of differentiation? Obviously not. For that reason, `diffcoeff5.def` contains the lines

```
\NewDocumentCommand \difccp { s s O{1} }
  {
    \IfBooleanTF #1
      { \difcp.cc.**[#3]<1> } { \difcp.cc.[#3]<1> }
  }
```

which now means you can write `$\difccp f{x:3,y,z:2}$` $\Longrightarrow \partial_{x^3yz^2} f$ or `$\difccp[3,1,2]f{x,y,z}[(x,y,z)=\mathbf0]$` $\Longrightarrow \left(\partial_{x^3yz^2} f\right)_{(x,y,z)=\mathbf{0}}$.

`\difccp` can be double starred to reverse the order of its arguments, variables of differentiation before differentiand. (In fact a single star has the same effect here: present the arguments to `\difccp` in reverse order.) Thus, doubly starred, `$\difccp**{x,y}f[y=1]$` $\Longrightarrow \left(\partial_{xy} f\right)_{y=1}$ (although a single star would have sufficed). However, `\difccp` does not accept a dot delimited first argument with which to define a variant.

There is one key in template `DIF` solely relevant to doubly compact derivatives: `difcc-var-ord` specifies how the orders of differentiation are displayed between the variables of differentiation. It's default value is `^{#1}` meaning that the orders are presented as superscripts – as shown in the examples. If you do not want superscripting but, say, only the order then edit the above variant in this way: either add the line `difcc-var-ord = ##1` (a double `#` must be used within the `\difdef` command) to the *key-value* list defining the `\difcp.cc.` variant above (ensuring all lines except possibly the last end in commas), or *after* the definition above add a definition to your `.def` file that edits it:

```
\difdef { cp } { cc } { difcc-var-ord = ##1 }
```

If you prefer the form $\partial_{x:3yz:2}$ replace `##1` here with `:##1`; if you prefer $\partial_{x\mathbf{3}yz\mathbf{2}}$ replace `##1` with `\mathbf{##1}`.

*Subscripts* (actually sub-subscripts) to my eye give a poor result: $\partial_{x_{3}yz_{2}}$, but if this is what you want, then a complication emerges. Placed in the *preamble* of your document,

```
\difdef { cp } { cc } { difcc-var-ord = _{##1} }
```

will achieve the desired effect; placed in your `.def` file, it will not. There you will need to write

```
\difdef { cp } { cc }
    { difcc-var-ord = \c_math_subscript_token {##1} }
```

#### 3.3.4.9 D, \delta, \Delta derivatives

In introductory calculus texts a derivative-like symbol is created with the lowercase Greek delta, $\delta$ . An uppercase Greek delta, $\Delta$, is often used in a derivative-like symbol for an average. In fluid dynamics the *material* (also *substantive* or

*total* ) derivative uses an uppercase D in place of d. Texts on differential equations often use a D operator. The file `diffcoeff5.def` contains the definitions

```
\difdef { f, s } { gd }
  { op-symbol = \delta }
\difdef { f, s } { gD }
  { op-symbol = \Delta }
\difdef { f, s } { D }
  { op-symbol = \mathrm{D} }
\difdef { c } { bD }
  {
    op-symbol    = \mathbf{D},
    op-sub-nudge = -2mu
  }
```

(where the 'g' in the first two suggests 'greek'), meaning one can write expressions like `$ \difs.gd.yx $` $\Longrightarrow \delta y/\delta x$, or `$ \difs.gD.yx $` $\Longrightarrow \Delta s/\Delta t$ (for the average speed), or

`\[ \diff.D.{\rho}t=\diffp\rho t + \mathbf{u\cdot}\nabla\rho \]`$\Longrightarrow$

$$\frac{\mathrm{D}\rho}{\mathrm{D}t} = \frac{\partial\rho}{\partial t} + \mathbf{u}\cdot\nabla\rho$$

for the total derivative of $\rho$ (perhaps in fluid dynamics), or, in a course on solving differential equations,

`$ \difc.bD.[2]y{x\,}+2\difc.bD.y{x\,}-4=0 $` $\Longrightarrow \mathbf{D}_x^2\, y + 2\mathbf{D}_x\, y - 4 = 0$

### 3.3.5 Changing defaults in `DIF`

Leaving the second argument of the `\difdef` command empty changes default values in the templates specified in the first argument. But the grandparent template `DIF` is unaffected. Intuitively, one might expect that leaving the first argument of the `\difdef` command empty would change a `DIF` default, but that is not the case. To change a default in `DIF` so that it 'infects' – is inherited by – all other templates can be done only at load time.

There are two ways of doing this. The first is to create a text file with the specific name `diffcoeff.DIF` with the desired settings. For example, if we want math-italic 'd's and a subscripted vertical rule for points of evaluation, then the file might look like

```
op-symbol      = d,
op-order-nudge = 1 mu,
outer-Ldelim   = \left . ,
outer-Rdelim   = \right |,
sub-nudge      = 0 mu
```

By locating the file in a place where your TEX distribution can find it – either in the directory of the current document or in your personal `texmf` tree (see the earlier discussion at §3.3.3, and in particular the need to alert your TEX distro to the presence of the file) – `diffcoeff.DIF` will be read at load time and the new defaults not only incorporated into template `DIF` but inherited by all child and grandchild templates unless explicitly countermanded (for example by `op-symbol = \partial` and similar statements in the definitions of those templates).

Note that placing `diffcoeff.DIF` in your current document directory means that only for documents sharing that directory will it be found. If you want `diffcoeff.DIF` to be found whatever the location of the document you are working on, then place `diffcoeff.DIF` in your personal `texmf` tree and alert your TEX distro to its presence ('refresh the filename database').

The second method of changing defaults in the grandparent template `DIF` is to use the *package option* `DIF`. For instance loading `diffcoeff` with the call

```
\usepackage
  [ DIF =
      {
        op-symbol = d,
        op-order-nudge  = 1 mu,
        outer-Ldelim    = \left . ,
        outer-Rdelim    = \right |,
        sub-nudge       = 0 mu
      }
  ]{diffcoeff}
```

will overwrite the built-in defaults with these new values, which will be inherited by child (and grandchild) templates unless explicitly countermanded. Notice that since `DIF` is a comma list it requires braces around the list of *key=value* statements. If using the package option `DIF` and employing commands within the *key=value* list from the `mleftright` package, you will need to explicitly load that package before the call to `diffcoeff`. (This issue does not affect use of the file `diffcoeff.DIF`.)

If both methods of changing the template `DIF` are employed, the order of use is, first, read and act on the file `diffcoeff.DIF`, then read and act on the package option `DIF`. (In other words, to avoid complicating the preamble, preferably use the file `diffcoeff.DIF`; use the package option `DIF` only for fine-tuning – perhaps a setting specific to that particular document.)

### 3.3.6 Other notations

The `diffcoeff` package is about defining *derivatives* but it is worth observing that other notations can be built from the `diffcoeff` constituents, in particular from the slash fraction forms. For example, some other token than `/`, or indeed series of tokens, can be used to link numerator and denominator. It could be `\vert` or `\Vert`, displaying as | and ‖ respectively, or `\otimes` (requiring for

example `\usepackage{stmaryrd}` in the preamble), displaying as $\otimes$, or the sequence of tokens `\otimes\cdots\otimes` displaying as $\otimes\cdots\otimes$. The critical key is `slash-tok`, with possible extra spacing on either side through the key `slash-sep`. Or, one may want to void the `op-symbol` key by giving it an empty value or do something like `op-symbol=\mathbf`, or give `outer-Ldelim`, `outer-Rdelim` special values, e.g.,`\langle`, `\rvert`.

In the file `diffcoeff5.def` I have included the following definition, in order to mimic the `\Braket` command of the `braket` package,

```
\difdef{ s }{ bk }
  {
    slash-tok = ,
    op-symbol = ,
    multi-term-sep = 3mu\middle|\mskip3mu ,
    outer-Ldelim = \left\langle ,
    outer-Rdelim = \right\rangle
  }
```

and supplemented it with the definition:

```
\NewDocumentCommand \Braket { m }
  { \difs.bk.<\negmu>{}{#1}[] }
```

Testing the new command, `\Braket`, gives this display:

```
\[ \Braket{\phi,\diffp[2]{}t,\psi} \] ⟹
```

$$\left\langle \phi \middle| \frac{\partial}{\partial t^2} \middle| \psi \right\rangle$$

Comparison with the `\Braket` command of the `braket` package, which uses `|` as the separator in the argument rather than commas, shows the displayed results to be the same (as far as I can judge).

# Chapter 4

# Differentials and jacobians

In addition to the six derivative commands, `\difx` and `\difxp`, the `diffcoeff` package has two further commands, `\dl` (also `\difl`) and `\jacob` (also `\difj`), for writing differentials and jabobian determinants respectively. These commands use the settings of the templates `DIFL` and `DIFJ`, and both are correspondingly configurable by means of the `\difdef` command.

## 4.1 Differentials

Forms like $\mathrm{d}x$ occur not only as components of derivatives but also in other contexts like the expression for a total differential,

$$\mathrm{d}P = \frac{\partial P}{\partial x}\,\mathrm{d}x + \frac{\partial P}{\partial y}\,\mathrm{d}y + \frac{\partial P}{\partial z}\,\mathrm{d}z,$$

or in integrals, like $\int \sin x\,\mathrm{d}x$, or multi-variable integrals like

$$\iiint_{-\infty}^{\infty} V(x,y,z)\,\mathrm{d}x\,\mathrm{d}y\,\mathrm{d}z,$$

or, with subscripted variables, rendered more cryptically as

$$\iiint_{-\infty}^{\infty} V(x_1,x_2,x_3)\,\mathrm{d}x^3, \text{ or } \iiint_{-\infty}^{\infty} V(x_1,x_2,x_3)\,\mathrm{d}^3x.$$

They also occur in differential geometry and elsewhere in the form of line elements like

$$\mathrm{d}x^2 + \mathrm{d}y^2 + \mathrm{d}z^2 \quad \texttt{and} \quad c^2\,\mathrm{d}t^2 - \mathrm{d}x^2 - \mathrm{d}y^2 - \mathrm{d}z^2\,.$$

Surely we want the ‘d’s in these expressions to correspond to their form (upright or math italic) in derivatives?

To this end, `diffcoeff` provides a command `\dl` to write the 'd' in a differential in a manner consistent with the default form used in derivatives. In the present document, the default form is upright and so

$$\texttt{\$ \textbackslash dl x \$} \implies \mathrm{d}x.$$

(From version 5.4 of `diffcoeff`, following the pattern of `\diff`, `\difs` and `\difc`, the command `\difl` is also available: `$ \difl x $` $\implies \mathrm{d}x$.) To use the command before a multi-token variable of differentiation, put the variable in braces:

$$\texttt{\$ \textbackslash dl\{\textbackslash vec\{x\}\},\textbackslash quad \textbackslash dl\{\textbackslash mathbf\{x\}\} \$} \implies \mathrm{d}\vec{x}, \quad \mathrm{d}\mathbf{x}.$$

For the first triple integral above, writing the differentials required not three but just the *one* command:

$$\texttt{\$ \textbackslash dl\{x,y,z\} \$} \implies \mathrm{d}x\,\mathrm{d}y\,\mathrm{d}z.$$

For the second triple integral, $\mathrm{d}x^3$ was just `\dl[3]x`, and for the third I used a dot-delimited argument producing a variant form of the differential `\dl.dn.[3]x` (which could be compacted further into a macro if it were to be used often; see §4.1.3 below).

To write the line elements I again made use of a variant form of the differential (again see §4.1.3.1):

$$\texttt{\$ \textbackslash dl.+.\{x,y,z\}\^{}2 \$} \implies \mathrm{d}x^2 + \mathrm{d}y^2 + \mathrm{d}z^2,$$
$$\texttt{\$ c\^{}2\textbackslash dl.-.\{t,x,y,z\}\^{}2 \$} \implies c^2\,\mathrm{d}t^2 - \mathrm{d}x^2 - \mathrm{d}y^2 - \mathrm{d}z^2.$$

### 4.1.1 Template `DIFL`

The differential command `\dl` gives access to a template `DIFL` which inherits the default values of the fundamental template `DIF` with the (few) changes shown in Table 4.1. In prior versions of `diffcoeff` the `style` key was fixed at the value `dl`; from version 5.4 it can also take the value `d^`. With the default style `dl`, `$\dl[2]x$` $\implies \mathrm{d}x^2$; with style `d^` the superscript attaches to the 'd', $\mathrm{d}^3x$. This form is sometimes used when abbreviating the product of differentials in a multiple integral. The `outer-Ldelim` key inserts a thin space before the differential; the `outer-Rdelim` key does nothing. For the differential, both `outer-Ldelim` and `outer-Rdelim` are *always inserted*. This differs from the derivative for which `outer-Ldelim` and `outer-Rdelim` are inserted only if there is a trailing optional argument. It is as if the differential command `\dl` had a built-in empty trailing optional argument.

Table 4.1: `DIFL` defaults

| key | default |
|---|---|
| style | `dl` |
| outer-Ldelim | `\,` |
| outer-Rdelim | |

That so few of the DIF defaults are changed in DIFL indicates that much of the machinery of derivative formation is irrelevant for forming a differential. A list of *relevant* keys for the creation of differentials – those that have some effect on the appearance of the thing – can be found at §5.2.5.

### 4.1.2 Syntax and options

If all arguments are present the differential command has the syntax

```
\dl.name.[order-spec]{variable(s)}^{exponent}
```

where the arguments have the following significance:

1. `name` (optional) A dot-delimited name to distinguish a variant form (non-default form) of differential; see §4.1.3 below.
2. `order-spec` (optional) The power or comma-list of powers to which the differential or differentials will be raised. If all powers are `1` then no specification is needed; indeed, if fewer powers are specified than there are variables, all 'missing' powers are assumed to be `1`; see the discussion for mixed partial derivatives, §2.2.5.
3. `variable(s)` (mandatory) The variable or comma-list of variables the differential operator applies to. `\dl x`, `\dl{\vec{x}}`, `\dl{x,y,z}`, are all valid variable specifications, displaying as $\mathrm{d}x$, $\mathrm{d}\vec{x}$ and $\mathrm{d}x\,\mathrm{d}y\,\mathrm{d}z$ respectively.
4. `exponent` (optional) An exponent to which all differentials will be raised; overrides the `order-spec`; see §4.1.3.1 for examples of use.

Only the third argument is mandatory, although it may be empty. As with derivatives, the square-bracket delimited order spec. can be replaced by colon-modified arguments in the variable specification:

```
$ \dl[3]x,\quad \dl{x:3} $
```
$\Longrightarrow\quad \mathrm{d}x^3, \quad \mathrm{d}x^3.$

### 4.1.3 Variant forms of differential

The first argument of the differential command `\dl` is the optional `name` which is used like the corresponding argument in the derivative commands to define *variant forms*.

To create such variant forms, the `\defdif` command is again used, but with `l` (lowercase `L`) used as the identifier in the first argument. For example, you may want a 'partial' differential, using `\partial` in place of `d`. It seems natural to give this the name `p`:

```
\difdef { l } { p }
  { op-symbol = \partial }
```

In fact just this definition can be found in the file `diffcoeff5.def`, so that

$ \dl.p.x $ $\Longrightarrow$ $\partial x$

which is seven keystrokes in all versus ten (space included) for `\partial x`. Defining `\dlp` by writing

```
\NewDocumentCommand \dlp {} { \dl.p. }.
```

saves another keystroke. This command can be found in the file `diffcoeff5.def`. The command eases the writing of expressions like $\partial x^3 \partial y^2 \partial z$ – perhaps in a document like the present one to discuss the minutiae of spacing in the denominators of mixed partial derivatives.

```
$\dlp[3,2]{x,y,z},\quad \dlp{x:3,y:2,z}$ ⟹
```

$\partial x^3 \partial y^2 \partial z, \quad \partial x^3 \partial y^2 \partial z$

As you can see from the example, just as for mixed partial derivatives, if more than one variable is specified but the `order-spec` contains fewer than that number of entries, `diffcoeff` assumes the missing entries on the right of the `order-spec` are `1`.

A second example of a variant form of differential is provided by the definition

```
\difdef { l } { b }
   { op-symbol = \mathrm{d}\mathbf }
```

which can be found in the file `diffcoeff5.def`. If you distinguish vectors by boldface type then you can avoid the repetitive writing of `\mathbf` for differentials of vectors by using the variant form `\dl.b.`:

```
$ \dl.b.x,\quad \dl.b.{x,y,z} $ ⟹
```

$\mathrm{d}\mathbf{x}, \quad \mathrm{d}\mathbf{x}\,\mathrm{d}\mathbf{y}\,\mathrm{d}\mathbf{z}.$

A third example is of a differential raised to a power in which the superscript is attached to the `d`, as provided by the definition

```
\difdef { l } { dn } { style = d^ }
```

I have added this definition to the file `diffcoeff5.def`. With this definition

```
\[ \iiintop_{-\infty}^{\infty}
     V(x_{1},x_{2},x_{3})\dl.dn.[3]x. \]
```

$\Longrightarrow$

$$\iiint_{-\infty}^{\infty} V(x_1, x_2, x_3)\ \mathrm{d}^3x.$$

If you are going to need this form of differential often, you could save some keystrokes with a macro definition like

```
\NewDocumentCommand \dn { m m } { \dl.dn.[#1] #2 }
```

which I have added to the preamble of the current document. Then to obtain $\mathrm{d}^2x$ one writes `\dn2x`. (But note that to obtain $\mathrm{d}x$ with no superscript means reverting to `\dl x` or remembering the `1` when writing `\dn1x`.)

#### 4.1.3.1 Line elements

Variant forms can be used to write line elements of Pythagorean or Minkowskian form. The definition

```
\difdef { l } { + }
  {
    multi-term-sep  = 0 mu +,
    term-sep-adjust = 0 mu  ,
    outer-Ldelim      =
  }
```

which can be found in the file `diffcoeff5.def`, inserts a `+` sign between terms in the variable specification. Note that the value of the `multi-term-sep` key begins with `0 mu`. A dimension here *initially* is essential. The intriguing feature of the definition is what follows the `0 mu`: a `+` sign. Also note that the thin space inserted by default before a differential by means of the `outer-Ldelim` setting is now removed. Applying this variant to `{x,y,z}` the result is $\mathrm{d}x + \mathrm{d}y + \mathrm{d}z$, which may be mildly interesting but definitely becomes so when we add an exponent to the variable spec.:

$$\texttt{\$ \dl.+.\{x,y,z\}\^{}2 \$} \implies \mathrm{d}x^2 + \mathrm{d}y^2 + \mathrm{d}z^2\,.$$

The exponent acts as if an order specification `[2,2,2]` had been included. If an order specification *is* included, whatever the values listed, the trailing exponent overrides it.

Similarly, the file `diffcoeff5.def` contains a definition where the plus sign is replaced by a minus. This enables the writing of a Minkowski metric:

$$\texttt{\$ c\^{}2\dl.-.\{t,x,y,z\}\^{}2 \$} \implies c^2\,\mathrm{d}t^2 - \mathrm{d}x^2 - \mathrm{d}y^2 - \mathrm{d}z^2\,.$$

### 4.1.4 Changing defaults

To change the *default* values of the `DIFL` template use the `\difdef` command but, as with derivatives, leave its second argument empty. For instance if you want slightly less space by default before a differential than the thin space (`\,` or `3 mu`) specified in the `DIFL` template – say you want `2 mu` – then write

```
\difdef { l } {} { outer-Ldelim = \twomu }
```

and ensure that this is in your `.def` file or in the preamble of your document. If you want a rubber length, say `3 mu plus 1 mu minus 2 mu` (which can also be written more compactly as `3muplus1muminus2mu`), then write (notice the `\mskip`)

```
\difdef { l } {}
  { outer-Ldelim = \mskip 3muplus1muminus2mu }
```

The crucial point is to leave the second argument of `\difdef`, the variant `name`, empty. That changes the *default* values in `DIFL` of the keys listed in the third argument of `\difdef`.

### 4.1.5 Rationale

But why bother with the differential command at all? It only seems to complicate the simple typing of `d` followed by `x`. Admittely typing `\dl x` requires fewer keystrokes than typing `\mathrm{d}x` (or even `\mathrm dx`), but there are other, more substantive, reasons why one might prefer an explicit command.

1. *Consistency* with the derivative.
2. *Spacing* is inserted automatically before the differential, and between differentials in (e.g.) multiple integrals.
3. *Parsing integrals* for some other package or program is much easier to do when looking for a concluding differential command `\dl` than when looking for `d` or `\mathrm{d}` or `\mathnormal{d}` (or whatever) followed by an arbitrary variable name.
4. *Configurability.* There are values other than the defaults that can be given to keys to give novel effects for variant forms of differential – see the examples `\dl.b.`, `\dl.+.` and `\dl.-.` above.

## 4.2 Jacobians

`diffcoeff` provides a command `\jacob` for writing jacobians – not the determinant as such but the symbol conventionally used to denote the determinant. (From version 5.4 of `diffcoeff`, the equivalent command `\difj` is also available for this purpose.) For example

```
\[ \jacob{u,v}{x,y},\quad \jacob{u,v,w}{x,y,z}. \] ⟹
```

$$\frac{\partial(u,v)}{\partial(x,y)}, \quad \frac{\partial(u,v,w)}{\partial(x,y,z)}.$$

The comma lists can contain any number of variables, even one or none, nor need the number of variables in numerator and denominator be equal. `\jacob` does *not* check such things. (It may be possible to exploit this fact when defining variant forms of jacobian.) The content of the variable lists in both numerator and denominator of a jacobian are used exactly as entered by you. That means that if you wish to include dots in the variable lists then you need to include (e.g.) `\ldots` explicitly yourself – entering three dots as in other `diffcoeff` commands will simply give you three dots in the output. It is fair to say that – as of version 5.6 – the jacobian is not *fully* integrated into the `diffcoeff` scheme, although it is still (partly) configurable.

Table 4.2: `DIFJ` defaults

| key | default |
|---|---|
| op-symbol | `\partial` |
| outer-Ldelim | |
| outer-Rdelim | |

### 4.2.1 Template DIFJ

The command \jacob (also \difj) gives access to a template, in this case DIFJ, which is a child of the fundamental template DIF and inherits most of its default values with only a few changes as shown in Table 4.2. Note that the keys outer-Ldelim and outer-Rdelim are both empty and, as with the differential, are *always inserted* – which is why they are empty by default.

The lack of entries in Table 4.2 is because many keys are irrelevant for forming jacobians – it doesn't matter what their values are. For a list of *relevant* keys – ones that have some effect on the appearance of a jacobian – see §5.2.6.

### 4.2.2 Syntax and variant forms

The \jacob command has only three arguments. The syntax is simple:

```
\jacob.name. {numer} {denom}
```

The arguments have the following significance:

1. name (optional) The dot-delimited name of a variant form of jacobian.
2. numer (mandatory) A comma list of variables forming the numerator of the jacobian.
3. denom (mandatory) A comma list of variables forming the denominator of the jacobian.

The default form of jacobian is an upright fraction with \partial operators and parentheses around the variable lists in both numerator and denominator.

If you want a jacobian in, say, slash-fraction form then once again the \difdef command is used. The file diffcoeff5.def contains the definition

```
\difdef { j } { / } { style = / }
```

To access this style, use the name – which is at your discretion but here I have chosen / – between dots after the \jacob command:

$ \jacob./.{u,v,w}{x,y,z} $ $\Longrightarrow$ $\partial(u,v,w)/\partial(x,y,z)$.

Or you might want an inline split-level jacobian, available from version 5.5 of diffcoeff, with the definition

```
\difdef { j } { s } { style = sfrac }
```

(to be found in diffcoeff5.def). With this definition

$ \jacob.s.{u,v,w}{x,y,z} $ $\Longrightarrow$ $\partial(u,v,w)/\partial(x,y,z)$.

If you want to change the operator symbol from \partial to math-italic D, as I have seen used, then the definition is:

```
\difdef { j } { D } { op-symbol = D }
```

(Again the name is at your discretion but `D` seems obvious.) I have added this to the preamble of the present document, so that

```
\[ \jacob.D.{u,v,w}{x,y,z} \] ⟹
```

$$\frac{D(u,v,w)}{D(x,y,z)}.$$

If you want square brackets rather than parentheses around the variable lists, then `lvwrap-Ldelim` and `lvwrap-Rdelim` (perhaps not intuitively) are the keys to change:

```
\difdef { j } { [ }
  {
    lvwrap-Ldelim = \onemu\mleft [,
    lvwrap-Rdelim = \mright ]
  }
```

the `\onemu` giving, to my eye, better spacing between the `\partial` symbols and the left brackets. This definition, too, has been added to the preamble so that

```
\[ \jacob.[.{u,v,w}{x,y,z} \] ⟹
```

$$\frac{\partial[u,v,w]}{\partial[x,y,z]}.$$

### 4.2.3 Changing defaults

To change *default* values of the `DIFJ` template leave the second argument of the `\difdef` command – the variant `name` – empty. For instance, if you want split-level to be your default setting, the `\difdef` command would be

```
\difdef { j } {} { style = sfrac }
```

The only difference from the definition shown earlier is the absence of the name from the now empty second argument. If this definition were added to the preamble or to the `.def` file of your current document then writing `\jacob{u,v,w}{x,y,z}` would give the same result as obtained above with the variant `\jacob.s.{u,v,w}{x,y,z}`.

# Chapter 5

# Reference

For convenience I list here the commands of `diffcoeff`, the template defaults, and the files and preamble definitions associated with this document.

## 5.1 Commands

### 5.1.1 Derivatives

`\diff`, `\diffp`, `\difs`, `\difsp`, `\difc`, `\difcp` (sometimes summarised as `\difx` and `\difxp`), ordinary and partial derivatives of upright-fraction, slash-fraction and compact forms respectively. With all arguments present the syntax is

```
\difx.name.**[order-spec]<override>{variable(s)}
    {differentiand}[pt of eval]
```

The syntax is identical for `\difxp`. The eight arguments have the following meanings:

1. `name` (optional) A dot-delimited name to distinguish a variant form (non-default form) of derivative; see §3 and specifically §3.3.
2. `*` (optional) The presence of a star (asterisk) signals: *append* the differentiand; its absence means the differentiand appears in the numerator of an upright- or slash-fraction form derivative; no effect for compact-form derivatives unless (see next) a second `*` is present; see §2.2.3.
3. `*` (optional) The presence of a *second* star signals that the argument specifying the variable(s) of differentiation comes *before* the argument specifying the differentiand; this is sometimes convenient when a complicated or lengthy differentiand is appended; see §2.2.3.2.
4. `order-spec` (optional) The order of differentiation when differentiating in a single variable, or a comma list of orders of differentiation for a mixed partial derivative; see §2.2.2 and §2.2.5. Not necessary if orders are specified with the colon notation in the variable list, §§2.2.2.1 and 2.2.5.1.

5. `override` (optional) Specifies the total order of differentiation when it cannot be calculated by `diffcoeff`, or is wanted in a different form from the calculated value; see §2.2.5.3.

6. `differentiand/derivand` (mandatory) The function or expression being differentiated.

7. `variable(s)` (mandatory) The variable of differentiation or a comma list of variables of differentiation (for a mixed partial derivative); also, with the colon notation (see §§2.2.2.1, 2.2.5.1), provides an alternative method of specifying orders of differentiation.

8. `pt of eval` (optional) Point of evaluation or, for partial derivatives, variable or variables held constant; *no space* before the left square bracket; see §2.2.4.

Both mandatory arguments may be empty, but require empty brace pairs to indicate as much. (Omitting the differentiand makes sense for all forms of derivative, `\difx`, `\difxp`, but omitting the variable or variables of differentiation is sensible only for the compact forms, `\difc`, `\difcp` – see §3.3.4.8.)

### 5.1.2 Differential

`\dl` (also `\difl`) differential. If all arguments are present the differential command has the syntax

```
\dl.name.[order-spec]{variable(s)}^{exponent}
```

where the arguments have the following significance:

1. `name` (optional) A dot-delimited name to distinguish a variant form of differential; see §4.1.3.

2. `order-spec` (optional) The power or comma-list of powers to which the differential or differentials will be raised. If all powers are `1` then no specification is needed; indeed, if fewer powers are specified than there are variables, all 'missing' powers are assumed to be `1`; see the discussion for mixed partial derivatives, §2.2.5. Or, the alternative colon notation can be used instead.

3. `variable(s)` (mandatory) The variable or comma-list of variables the differential operator applies to. `\dl x`, `\dl{\vec{x}}`, `\dl{x,y,z}`, are all valid variable specifications, displaying as $\mathrm{d}x$, $\mathrm{d}\vec{x}$ and $\mathrm{d}x\,\mathrm{d}y\,\mathrm{d}z$ respectively.

4. `exponent` (optional) An exponent to which all differentials will be raised; overrides the `order-spec`; see §4.1.3.1 for examples of use.

### 5.1.3 Jacobian

`\jacob` (also `\difj`) jacobian with syntax

```
\jacob.name. {numer} {denom}
```

where the arguments have the following significance:

1. `name` (optional) The dot-delimited name of a variant form of jacobian.
2. `numer` (mandatory) A comma list of variables forming the numerator of the jacobian.
3. `denom` (mandatory) A comma list of variables forming the denominator of the jacobian.

### 5.1.4 Supporting commands

`\difdef` Means of defining variant forms of derivatives, differentials and jacobians, with syntax

```
\difdef { id(s) } { name } { settings }
```

The arguments are all mandatory and have the following significance

1. `id(s)` Non-empty comma list of one, some or all of the identifiers `f`, `s`, `c`, `fp`, `sp`, `cp`, `j`, `l` identifying upright fraction, slash fraction and compact ordinary derivatives; upright fraction, slash fraction and compact partial derivatives, and jacobians and differentials;
2. `name` Name for a variant form of derivative; as well as letters may include numbers and other keyboard characters, but not braces, `%` or `#`; may be empty, in which case the `{settings}` determine (new) default values for the template(s) identified in `{id(s)}`.
3. `settings` Comma list of changed *key=value* settings.

`\difoverride` (see §2.2.5.3) order-override command (now deprecated in favour of the order-override option) with syntax

```
\difoverride { total order }
```

where the one mandatory argument has the significance

1. `total order` desired value of total order of differentiation; may be (and generally is) empty.

#### 5.1.4.1 Spacing commands

**\negmu** insert a −1 mu space

**\nilmu** insert a 0 mu space

**\onemu** insert a 1 mu space

**\twomu** insert a 2 mu space

From version 5.5 of `diffcoeff`, each of these commands can be starred to produce a space of opposite sign but the same magnitude, e.g. `\twomu*` gives a space of `-2 mu`.

From version 5.5, `diffcoeff` also supports three commands which switch on one of the `spaced` package option values *within the current group*.

**\spacedone** is equivalent to `spaced=1` within the current group;

**\spacednil** is equivalent to `spaced=0` within the current group;

**\spacedneg** is equivalent to `spaced=-1` within the current group.

### 5.1.5 Variant-form-derived commands

Definitions in terms of variant forms of derivative:

**\difsfrac** split-level inline fraction, §3.3.4.6,

**\difccp** doubly compact derivative, §3.3.4.8,

**\Braket** bra-ket command (as used in quantum mechanics), §3.3.6,

**\dlp** partial derivative differential, §4.1.3,

**\dn** superscripted-'d' differential, §4.1.3.

All save the last can be found in the file `diffcoeff5.def`, §5.3.

## 5.2 Templates

The following lists record the default values of the templates used by `diffcoeff`. For templates other than `DIF`, only *relevant* keys have been listed – those which affect the appearance of the derivative (or jacobian or differential).

### 5.2.1 DIF (primogenitor)

```
style            = frac,
slash-tok        = /,
slash-sep        = 0 mu,
sfrac-scaling    = 0.7,
derivand-sep     = 3 mu plus 1 mu minus 2 mu,
op-symbol        = \mathrm{d},
op-symbol-alt    = \KeyValue{ op-symbol },
op-order-nudge   = 0 mu,
op-sub-nudge     = 0 mu,
var-sup-nudge    = 1 mu,
multi-term-sep   = 2 mu plus 1 mu minus 1 mu,
term-sep-adjust  = -1 mu,
dots             = \cdots,
long-var-wrap    = d(v),
lvwrap-Ldelim    = \mleft (,
lvwrap-Rdelim    = \mright ),
lvwrap-sup-nudge = -2 mu,
outer-Ldelim     = \left (,
outer-Rdelim     = \right ),
elbowroom        = 0 mu,
sub-nudge        = -5 mu,
difcc-var-ord    = ^{#1},
*derivand-sep    = \KeyValue{ derivand-sep },
*op-set-left     = false,
*italic-nudge    = 0 mu,
*inner-wrap      = false,
*inner-Ldelim    = (,
*inner-Rdelim    = ),
*outer-Ldelim    = \big [,
*outer-Rdelim    = \big ],
*sub-nudge       = 0 mu
```

### 5.2.2 DIFF (upright-fraction derivative)

Relevant keys and default values for template DIFF (all values correspond to the defaults in DIF):

```
style            = frac,
derivand-sep     = 3 mu plus 1 mu minus 2 mu,
op-symbol        = \mathrm{d},
op-symbol-alt    = \KeyValue { op-symbol },
op-order-nudge   = 0 mu,
var-sup-nudge    = 1 mu,
multi-term-sep   = 2 mu plus 1 mu minus 1 mu,
term-sep-adjust  = -1 mu,
```

```
    dots             = \cdots,
    long-var-wrap    = d(v),
    lvwrap-Ldelim    = \mleft (,
    lvwrap-Rdelim    = \mright ),
    lvwrap-sup-nudge = -2 mu,
    outer-Ldelim     = \left (,
    outer-Rdelim     = \right ),
    elbowroom        = 0 mu,
    sub-nudge        = -5 mu,
    *derivand-sep    = \KeyValue { derivand-sep },
    *op-set-left     = false,
    *italic-nudge    = 0 mu
```

#### 5.2.2.1 DIFFP

DIFF defaults as above with the following changes (a marginal >> indicates where a setting differs from that in DIFF):

```
>>  op-symbol        = \partial,
>>  op-order-nudge   = 1 mu,
>>  *italic-nudge    = 3 mu
```

### 5.2.3 DIFS (slash-fraction derivative)

Relevant keys and default values for template DIFS (a marginal > indicates where a setting differs from that in DIF):

```
>   style            = /,
    slash-tok        = /,
    slash-sep        = 0 mu,
    sfrac-scaling    = 0.7,
>   derivand-sep     = 2 mu plus 1 mu minus 1 mu,
    op-symbol        = \mathrm{d},
    op-symbol-alt    = \KeyValue { op-symbol },
    op-order-nudge   = 0 mu,
    var-sup-nudge    = 1 mu,
    multi-term-sep   = 2 mu plus 1 mu minus 1 mu,
    term-sep-adjust  = -1 mu,
>   dots             = \ldots,
    long-var-wrap    = d(v),
    lvwrap-Ldelim    = \mleft (,
    lvwrap-Rdelim    = \mright ),
    lvwrap-sup-nudge = -2 mu,
>   outer-Ldelim     = (,
>   outer-Rdelim     = ),
    elbowroom        = 0 mu,
>   sub-nudge        = 0 mu,
```

```
    *derivand-sep    = \KeyValue { derivand-sep },
>   *inner-wrap      = true,
    *inner-Ldelim    = (,
    *inner-Rdelim    = ),
    *outer-Ldelim    = \big [,
    *outer-Rdelim    = \big ],
    *sub-nudge       = 0 mu
```

#### 5.2.3.1 DIFSP

DIFS defaults as above with the following changes (a marginal >> indicates where a setting differs from that in DIFS):

```
>>  op-symbol        = \partial,
>>  op-order-nudge   = 1 mu
```

### 5.2.4 DIFC (compact derivative)

Relevant keys and default values for template DIFC (a marginal > indicates where a setting differs from that in DIF):

```
>   style            = _ ,
>   derivand-sep     = 1 mu plus 1 mu minus 1 mu,
    op-symbol        = \mathrm{d},
    op-order-nudge   = 0 mu,
    op-sub-nudge     = 0 mu,
>   multi-term-sep   = 1 mu,
>   term-sep-adjust  = 0 mu,
>   dots             = \ldots,
>   outer-Ldelim     = \bigl (,
>   outer-Rdelim     = \bigr ),
    elbowroom        = 0 mu,
>   sub-nudge        = -2 mu,
    *derivand-sep    = \KeyValue { derivand-sep }
```

#### 5.2.4.1 DIFCP

DIFC defaults as above with these changes (a marginal >> indicates where a setting differs from that in DIFC):

```
>>  op-symbol        = \partial,
>>  op-order-nudge   = 1 mu
```

### 5.2.5 DIFL (differential)

Relevant keys and default values for template DIFL (a marginal > indicates where a setting differs from that in DIF):

```
    op-symbol        = \mathrm{d},
    op-order-nudge   = 0 mu,
    var-sup-nudge    = 1 mu,
    multi-term-sep   = 2 mu plus 1 mu minus 1 mu,
    term-sep-adjust  = -1 mu,
>   long-var-wrap    = dv,
    lvwrap-Ldelim    = \mleft (,
    lvwrap-Rdelim    = \mright ),
    lvwrap-sup-nudge = -2 mu,
>   outer-Ldelim     = \, ,
>   outer-Rdelim     = ,
    elbowroom        = 0 mu
```

### 5.2.6 `DIFJ` (jacobian)

Relevant keys and default values for template `DIFJ` (a marginal `>` indicates where a setting differs from that in `DIF`):

```
    style            = frac,
    slash-tok        = /,
    slash-sep        = 0 mu,
    sfrac-scaling    = 0.7,
>   op-symbol        = \partial,
    op-symbol-alt    = \KeyValue{ op-symbol },
    lvwrap-Ldelim    = \mleft (,
    lvwrap-Rdelim    = \mright ),
>   outer-Ldelim     = ,
>   outer-Rdelim     = ,
    elbowroom        = 0 mu ,
    *op-set-left     = false,
    *italic-nudge    = 0 mu
```

## 5.3 The file `diffcoeff5.def`

```
% file 'diffcoeff5.def'
% definitions for variant forms
% 2025/10/01
% Andrew Parsloe ajparsloe@gmail.com
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% material derivative
\difdef { f, s } { D }
  { op-symbol = \mathrm{D} }
% math italic
\difdef { f, s, c } { d' }
  {
```

```
    op-symbol      = d,
    op-order-nudge = 1 mu,
    *slash-sep     = { 0 mu, -1 mu }
  }
\difdef { f, s, c } { D' }
  {
    op-symbol  = D,
    op-order-nudge = 1 mu,
    *slash-sep     = -1 mu
  }
% Greek
\difdef { f, s } { gd }
  { op-symbol = \delta }
\difdef { f, s } { gD }
  { op-symbol = \Delta }
% nabla, tensor calc. acceleration
\difdef { f, s } { n }
  {
    op-symbol     = \nabla,
    op-symbol-alt = \mathrm{d},
    *slash-sep    = { -2mu, 0mu }
  }
% tfrac, nonscalable
\difdef { f, fp } { t }
  {
    style             = tfrac  ,
    derivand-sep      = 1 mu plus 1 mu minus 1 mu,
    multi-term-sep    = 0 mu   ,
    term-sep-adjust   = 0 mu   ,
    lvwrap-sup-nudge  = 0 mu   ,
    outer-Ldelim      = \bigl (,
    outer-Rdelim      = \bigr ),
    elbowroom         = -2 mu  ,
    sub-nudge         = -3 mu
  }
% pt of eval., sq. bracket
\difdef { f, fp, s, sp } { ] }
  {
    outer-Ldelim = \left [ ,
    outer-Rdelim = \right ],
    elbowroom    = 1 mu,
    sub-nudge    = 0 mu
  }
% long var wrap
\difdef { f, fp } { (dv) }
  { long-var-wrap = (dv) }
```

```
\difdef { f, fp } { dv }
  { long-var-wrap = dv }
\difdef { f, fp } { dv_ }
  {
    long-var-wrap = dv,
    var-sup-nudge = -2 mu
  }
%%%%%%%%% slash fraction %%%%%%%%%
% sfrac split-level
\difdef { s, sp } { s }
  {
    style             = sfrac,
    slash-sep         = { -2 mu, -1 mu },
    derivand-sep      = 0 mu ,
    multi-term-sep    = 0 mu ,
    term-sep-adjust   = 0 mu ,
    var-sup-nudge     = 0 mu ,
    lvwrap-sup-nudge  = 0 mu
  }
\difdef { s } { s } { *slash-sep = { -1 mu, -1 mu } }
% sfrac, no op symbols
\NewDocumentCommand \difsfrac { O{-2mu,-1mu} m m }
  {{
    \difdef { s } { s } { op-symbol=, slash-sep={#1} }
    \difs.s.{#2}{#3}
  }}
% \NewDocumentCommand \difsfrac { D[,{-2} D-]{1} m m }
  % {{
    % \difdef{s}{s}{op-symbol=,slash-sep={#1mu,-#2mu} }
    % \difs.s.{#3}{#4}
  % }}
% slash frac: 0=scalable, 1=big,
% % 2=Big, 3=bigg, 4=Bigg but > 1
% generally gives eyesores
\difdef { s, sp } { 0 }
  {
    style         = auto     ,
    outer-Ldelim  = \left [  ,
    outer-Rdelim  = \right ] ,
    sub-nudge     = 0 mu     ,
    *inner-Ldelim = \mleft ( ,
    *inner-Rdelim = \mright ),
    *outer-Ldelim = \left [  ,
    *outer-Rdelim = \right ]
  }
\difdef { s, sp } { 1 }
```

```
  {
    style          =  big   ,
    outer-Ldelim   = \bigl (,
    outer-Rdelim   = \bigr ),
    sub-nudge      = -2 mu  ,
    *inner-wrap    = true   ,
    *inner-Ldelim  = \bigl (,
    *inner-Rdelim  = \bigr ),
    *outer-Ldelim  = \bigl [,
    *outer-Rdelim  = \bigr ]
  }
% \cdots generic slash fraction
\difdef { s, sp } { cd } { dots = \cdots }
% pt of eval -- pipe
\difdef { f, fp, s, sp } { | }
  {
    outer-Ldelim = \left . ,
    outer-Rdelim = \right |,
    sub-nudge    = 0 mu
  }
%%%%%%%%%% compact %%%%%%%%%%
% D operator
\difdef { c } { D }
  {
    op-symbol    = \mathrm{D},
    op-sub-nudge = -2 mu
  }
\difdef { c } { D' }
  {
    op-symbol    = D,
    op-sub-nudge = -2 mu
  }
% bold
\difdef { c } { bD }
  {
    op-symbol    = \mathbf{D},
    op-sub-nudge = -2 mu
  }
% differential style
\difdef { c, cp } { dl }
  {
    style           = dl,
    multi-term-sep  = 1 mu,
    term-sep-adjust = -1 mu
  }
% attach superscript to the d
```

```
\difdef { c, cp } { d^ }
  {
    style           = d^,
    multi-term-sep  = 1 mu,
    term-sep-adjust = -1 mu
  }
% doubly compact
\difdef { cp } { cc }
  {
    style           =  cc  ,
    op-order-nudge  =  1 mu,
    multi-term-sep  =  0 mu,
    term-sep-adjust = -1 mu,
    var-sup-nudge   =  0 mu
  }
% \difdef { cp } { cc } { difcc-var-ord = ##1 }
% \difdef { cp } { cc } { difcc-var-ord = :##1 }
% \difdef { cp } { cc } { difcc-var-ord = \mathbf{##1} }
\NewDocumentCommand \difccp { s s O{1}}
  {
    \IfBooleanTF #1
      { \difcp.cc.**[#3]<1> }
      { \difcp.cc.[#3]<1> }
  }
%%%%%%%%%% differentials %%%%%%%%%%
% partial
\difdef { l } { p } { op-symbol = \partial }
\NewDocumentCommand \dlp {} { \dl.p. }
% bold variable
\difdef { l } { b }
  { op-symbol = \mathrm{d}\mathbf }
% d^n x
\difdef { l } { dn }{ style = d^ }
% line element: Pythagoras
\difdef { l } { + }
  {
    multi-term-sep  = 0 mu +,
    term-sep-adjust = 0 mu  ,
    outer-Ldelim    =
  }
% line element: Minkowski
\difdef { l } { - }
  {
    multi-term-sep  = 0 mu -,
    term-sep-adjust = 0 mu  ,
    outer-Ldelim    =
```

```
  }
%%%%%%%%%% jacobian %%%%%%%%%%
% slash fraction
\difdef { j } { / } { style = / }
% split level
\difdef { j } { s } { style = sfrac }
%%%%%%%%%% \Braket %%%%%%%%%%
\difdef{ s }{ bk }
  {
      slash-tok = ,
      op-symbol =  ,
      multi-term-sep = 3mu\middle|\mskip3mu ,
      outer-Ldelim=\left\langle ,
      outer-Rdelim=\right\rangle
  }
\NewDocumentCommand \Braket { m }
  { \difs.bk.<\negmu>{}{#1}[] }
```

## 5.4 Preamble definitions

The preamble to the present document contains the command

```
\usepackage[def-file=<path to>/diffcoeff5,%
            spaced=-1]{diffcoeff}
```

where `<path to>` is the path on my computer (beginning `E:/`), and definitions:

```
% no sub nudge (a sea of white space)
\difdef { f, fp } { wsp }
  { sub-nudge = 0 mu }
% align op left; no italic nudge
\difdef { f,fp } { left0 }
  {
    *op-set-left  = true,
    *italic-nudge = 0 mu
  }
% align op left; italic nudge
\difdef { f,fp } { left }
  { *op-set-left  = true }
% partial variant of \diff
\difdef { f } { p }
  {
    op-symbol      = \partial,
    op-order-nudge = 1 mu
  }
% partial, 3mu sep of terms
```

```
\difdef { fp, sp } { 3mu }
  {  multi-term-sep  = 3 mu }
% partial, compact, -2 mu subscr. var. adjustment
\difdef { cp } { 2 } { term-sep-adjust = -2 mu }
% partial, use \ldots
\difdef { fp } { ld } { dots = \!\ldots }
% differential d^n x
\NewDocumentCommand \dn { m m } { \dl.dn.[#1] #2 }
% D jacobian
\difdef { j } { D }
  { op-symbol = D }
% square bracket jabobian
\difdef { j } { [ }
  {
    lvwrap-Ldelim = \onemu\mleft [,
    lvwrap-Rdelim = \mright ]
  }
```

## 5.5 Version history

Version 5 was conceived as a new package (under the name `diffcoefx`) and only at the end, after discussion with CTAN maintainers, changed to version 5.0 of `diffcoeff`.

1. Version 5.0 (2023-01-02) of `diffcoeff`
   (a) splits the `\diff` command of version 4 into three pairs of commands: `\diff` and `\diffp` for upright-fraction derivatives; `\difs` and `difsp` for slash-fraction derivatives, and `\difc` and `\difcp` for 'compact form' derivatives;
   (b) replaces the order-override option by a new command `\difoverride` (to avoid cluttering formulas with a second square-bracket delimited optional argument before the differentiand);
   (c) adds a second star option to reverse the order of differentiand and variable(s) of differentiation when the differentiand is appended;
   (d) replaces the two-argument `\diffdef` command of earlier versions with the three-argument command `\difdef` command, the additional argument determining which one or more of the `f`, `s`, `c`, `fp`, `sp` or `cp` forms the defined variant applies to;
   (e) rewrites the differential command `\dl` which is now template-configurable (e.g. allowing easy writing of line elements like $\mathrm{d}x^2 + \mathrm{d}y^2 + \mathrm{d}z^2$);
   (f) rewrites the jacobian command `\jacob` which is now template-configurable;
   (g) uses ISO defaults;
   (h) includes version conflict messages.

2. Version 5.1 (2023-01-16)

   (a) adds a now-redundant `ISO` package option and related version conflict message;

   (b) makes some corresponding tweaks to documentation (including this version 5 history).

3. Version 5.2 (2023-01-24)

   (a) Simplifies the treatment of the empty argument of an absent differentiation variable;

   (b) initializes (clears) two sequence variables that otherwise caused error when `scrbook` class was used;

   (c) amends documentation.

4. Version 5.3 (2023-04-10)

   (a) Fixes a bug when `\dl` was used in a particular way in `beamer` (e.g. `\[\alert{\dl x}\]`).

   (b) Provides an alternative method of specifying orders of differentiation by means of colon separators in the variable argument.

   (c) Reinstates (from v.4) the order-override option as an alternative to `\difoverride` but now angle-bracket delimited.

5. Version 5.4 (2023-11-08)

   (a) Adjusts both code and documentation about the differential to enable forms like $\mathrm{d}^3x$ (sometimes used in multiple integrals).

6. Version 5.5 (2025-10-01)

   (a) Adds the ability to write a generic mixed partial derivative (e.g. $\partial^n y/\partial x_1 \ldots \partial x_n$) in $n$ variables in a natural way by entering 3 dots between commas in the variable list; adds an associated key `dots`.

   (b) Adds the option of a split-level slash-fraction form derivative (like `\sfrac` of the `xfrac` package), an associated key `sfrac-scaling`, and an adjustment to the key `slash-sep`.

   (c) Adds the option of a 'doubly compact' form of compact derivative and an associated key `difcc-var-ord`.

   (d) Adds the key `op-sub-nudge` to adjust the position of the subscript in a compact form derivative (e.g. for an operator like `\nabla`).

   (e) Adds a package option `dif**` making permanent the double star option (reversing the order of the derivand and differentiation variable(s) arguments) for specified derivatives (like `\difc`, `\difcp`).

   (f) Fixes a bug when using colon notation to indicate an order of differentiation when either the variable or the order is multi-character.

(g) Adds to the documentation and corrects a few typos.

7. Version 5.6 (2025-12-10)

   (a) Resolves an error arising if version 5.5 is used at the same time as a package still using `xtemplate`.