`diffcoeff`
a LaTeX package for writing differential coefficients easily

Andrew Parsloe
(aparsloe@clear.net.nz)

June 27, 2016

Abstract

`diffcoeff.sty` allows the easy writing of ordinary and partial differential coefficients of arbitrary (algebraic or numeric) order. For mixed partial derivatives, the overall order (the superscript on ∂ in the numerator) is calculated by the package. Optional arguments allow the easy specification of a point of evaluation for ordinary derivatives, or variables held constant for partial derivatives, and the placement of the differentiand (in the numerator or appended). Some tweaking of the display is possible through key = value settings. Secondary commands provide analogous coefficients constructed from D, Δ, and δ, and a command for writing Jacobians. The package uses `expl3` and `xparse` from the LaTeX3 bundles, `l3kernel` and `l3packages`.

1 Requirements

The LaTeX package `diffcoeff.sty` is written in the expl3 language of LaTeX3 and requires the bundles `l3kernel` and `l3packages` (the latter for the `xparse` module). However, granted the presence of these bundles in your TeX distribution, the LaTeX3 element should be invisible to the user.

The package is invoked in the usual way by entering

```
\usepackage{diffcoeff}
```

in the preamble of your document.

Note on terminology I refer throughout to the quantity or function being differentiated as the *differentiand* (in line with *integrand*, *operand*, etc.).

2 Ordinary differential coefficients

Writing `\diff{y}{x}` will produce $\frac{dy}{dx}$ in text style (i.e. placed between `$ $`) or

$$\frac{dy}{dx}$$

in display style (i.e. placed between `\[ \]`). In fact `\diff yx` (omitting the braces) will produce these results, with a saving on keystrokes. The braces are needed only when differentiand or variable of differentiation is more than a single token.

There is one other form: we can insert a slash, '/', between numerator and denominator: `\diff f/x` produces df/dx which may be preferred for textstyle differential coefficients on occasion. Nothing is gained in this particular instance. It is quicker to type the five keystrokes d, f, /, d, x than it is to type the nine of \, d, i, f, f, , f, /, x but there are occasions when this is not always the case.

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

$$\texttt{\textbackslash diff[2]\{y\}\{x\}} \implies \frac{d^2y}{dx^2},$$

$$\texttt{\textbackslash diff[n+1]\{y\}\{x\}} \implies \frac{d^{n+1}y}{dx^{n+1}}.$$

(And again the braces can be omitted for single letters like `x` and `y`.)

In slash style, `\diff[2]f/x` (11 keystrokes) produces d^2f/dx^2, not significantly more typing than `d^2f/dx^2` (9 keystrokes).

If you want to specify a point at which the derivative is evaluated, append a final optional argument, but note that it is given in *braces* rather than square brackets:

$$\texttt{\textbackslash diff[2]\{y\}\{x\}\{0\}} \implies \left.\frac{d^2y}{dx^2}\right|_0$$

(In this example it seems neater *not* to finish with a full stop or other punctuation.) The use of braces means that the differential coefficient can be followed immediately by a mathematical expression wrapped in `\{ \}`, or `[ ]`, without the expression being confused with the (final) optional argument. Note also that there must be *no space* before the argument: it follows *immediately* on the second mandatory argument (if it follows at all).

We could save a few keystrokes by writing this last example as `\diff[2]yx{0}`. The braces around the final optional argument can *not* be omitted – otherwise there is no way of knowing that it *is* the final optional argument and not part of a following expression.

In slash style, the trailing optional argument can be used, but perhaps should not be. It looks ugly:

$$\texttt{\textbackslash diff[2]y/x\{0\}} \implies d^2y/dx^2\big|_0$$

Slash style is a more casual rendering of the derivative, intended for inline use within text and it would be better to use a phrase like 'evaluated at zero'.

2.1 \diffset: formatting tweaks

There are a number of tweaks one can make to the display of a derivative. Many people now use upright (roman) forms for the 'd's of a differential coefficient, rather than math italic. To do this, put the command

```
\diffset[roman = true]
```

in the preamble of your document (following the `\usepackage{diffcoeff}` of course). The default is math italic.

It is possible that you may want more space between the 'd' in the numerator of a differential coefficient and the superscripted order of the derivative. Using an upright 'd' alleviates this problem, but if using the default math italic for the 'd's, the separation can be altered by using the

```
\diffset[d-sep = n]
```

command which adds an extra n mu to TEX's spacing. The default value for n is 1 (i.e. 1 mu). The new separation will affect all derivatives following the new setting. Put in the preamble, the new separation will be document-wide.

A third tweak changes the delimiters used to indicate the point of evaluation. By default there is nothing on the left side and a vertical rule with the point of evaluation subscripted to it on the right. You may prefer subscripted parentheses. In that case write

```
\diffset[d-delims = ()].
```

Whatever delimiters you choose need to work with LATEX's `\left` and `\right` commands and consist of exactly two tokens. `[` and `]` are acceptable as also are pairs like `\lceil \rceil`, `\lfloor \rfloor` but if you want to use `\{` and `\}` you need to place the `\diffset` command between maths delimiters. The default pair, as indicated, is `.    |`, t or full stop being LATEX's way of suppressing (in this case) the left delimiter.

If you change the delimiters, say to `( )`, then the position of the subscript may need adjusting. To do this, use the command

```
\diffset[d-nudge = n]
```

A suggested setting for parentheses `( )` is -6 (in fact -6 mu but the 'mu' is supplied by `diffcoeff`). Thus the total change would be

```
\diffset[d-delims = ( ), d-nudge = -6]
```

producing, for example,

$$\texttt{\\diff[n]\{y\}\{x\}\{0\}} \implies \left(\frac{d^n y}{dx^n}\right)_0.$$

The default setting for `.|` is 0. Simply writing

```
\diffset
```

will return all settings to their defaults.

2.2 Variations

Appending the differentiand: \diff*

If you want the differentiand to follow the differential coefficient rather than sit in the numerator, perhaps because it is a fraction itself or because it is long, like a polynomial (ax^2+bx+c), then one way to achieve that is to leave the first mandatory argument in the `\diff` command empty and immediately follow the differential operator with the differentiand:

`\diff{}{x}(ax^2+bx+c)` $\Longrightarrow \quad \dfrac{d}{dx}(ax^2+bx+c).$

Another is to use the star form of the `\diff` command,

`\diff*[2]{\frac{F(x)}{G(x)}}{x}` $\Longrightarrow \quad \dfrac{d^2}{dx^2}\dfrac{F(x)}{G(x)}.$

The LaTeX expression can be harder to read if, as here, one is using a command like `\frac` with its own pairs of braces, but it is much easier, if one isn't sure whether the differentiand should be appended or in the numerator, simply to insert or delete an asterisk than move the differentiand from one place to the other. The star form becomes especially useful if you want to both append the differentiand *and* indicate the point of evaluation, since it saves having to set up the `\left.` and `\right|` delimiters and the subscript:

`\diff*{\frac{F(x)}{G(x)}}{x}{0}` $\Longrightarrow \quad \left.\dfrac{d}{dx}\dfrac{F(x)}{G(x)}\right|_0$

In slash style with the star option, an example above becomes

`\diff*{(ax^2+bx+c)}/{x}` $\Longrightarrow \quad (d/dx)(ax^2+bx+c)$,

where the derivative is automatically enclosed in parentheses by `diffcoeff`.

Multi-character variables of differentiation

Derivatives of a function-of-a-function may require forming a differential coefficient in which the variable of differentiation is more complicated than a single symbol like `x` or `\alpha`. For instance, to differentiate $\ln x^2$ (the logarithm of x^2) one first differentiates in x^2 then in x. The initial differentiation can be rendered

`\diff{\ln x^2}{x^2}` $\Longrightarrow \quad \dfrac{d\ln x^2}{d(x^2)}$;

`diff{\ln x^2}/{x^2}` $\Longrightarrow \quad d\ln x^2/d(x^2).$

Because of the superscript in the variable of differentiation x^2, parentheses have been automatically inserted in the denominator. This does not happen in a first-order derivative unless there is a superscript present. For instance,

`\diff{\ln\sin x}{\sin x}` $\Longrightarrow \quad \dfrac{d\ln\sin x}{d\sin x}.$

displays without parentheses. However, for higher order derivatives parentheses are *always* inserted to avoid confusion:

`\diff[2]{\ln\sin x}{\sin x}` $\Longrightarrow \quad \dfrac{d^2\ln\sin x}{d(\sin x)^2}.$

Positioning the d in the numerator When appending a differentiand, you may want to change the position of the 'd' in the numerator, particularly if the variable of differentiation is a multi-character symbol or the order of differentiation is a multi-character value like $n+1$.

If you 'manually' append the differentiand, then there are various ways of altering the placement of the 'd' from the default midpoint: use `\hfill` to push it hard to the left; use `\hfil` to push it to the left an intermediate amount; use `\hphantom` or `\hspace`, both with a braced argument, to push it to the left some custom amount; use `\hspace` with a *negative* braced argument to push it to the right. These same means can be used to shift the 'd' when using the starred form of `\diff`. The effect is exactly the same, too:

`\diff[n+1]{\hphantom{\sin x}}{\sin x}\ln\sin x`

$$\Longrightarrow \quad \frac{d^{n+1}\hphantom{\sin x}}{d(\sin x)^{n+1}}\ln\sin x,$$

`\diff*[n+1]{\hphantom{\sin x}\ln\sin x}{\sin x}`

$$\Longrightarrow \quad \frac{d^{n+1}\hphantom{\sin x}}{d(\sin x)^{n+1}}\ln\sin x.$$

In the starred form `diffcoeff` understands that the formatting is not appended with the differentiand but stays in the numerator. (But a *second* `\hphantom` or `\hfil` etc. would be appended.) These are to be compared with

`\diff*[n+1]{\ln\sin x}{\sin x}` $\Longrightarrow \quad \dfrac{d^{n+1}}{d(\sin x)^{n+1}}\ln\sin x,$

where no phantom has been used. Which is better? Deleting the asterisk gives

`\diff[n+1]{\ln\sin x}{\sin x}` $\Longrightarrow \quad \dfrac{d^{n+1}\ln\sin x}{d(\sin x)^{n+1}},$

In slash style, the phantom (or `\hfil` etc.) is ignored:

`\diff*[n+1]{\hphantom{\sin x\sin x\sin x}\ln\sin x}/{\sin x}`

$$\Longrightarrow \quad (d^{n+1}/d(\sin x)^{n+1})\ln\sin x.$$

Iterated derivatives

A second derivative is an iterated derivative, i.e., one in which a differential coefficient forms the differentiand of another differential coefficient:

`\diff[2]yx = \diff*{\diff yx}x` $\Longrightarrow \dfrac{d^2y}{dx^2} = \dfrac{d}{dx}\dfrac{dy}{dx}$,

or even

`\diff[2]yx = \diff{\diff yx}x` $\Longrightarrow \dfrac{d^2y}{dx^2} = \dfrac{d\frac{dy}{dx}}{dx}$,

where omission of unnecessary braces has aided readability. Note how easy it is to switch between the different forms on the right, simply by inserting or removing an asterisk.

2.3 Forming 'derivatives' with D, \Delta, \delta

Often one wants to construct analogues of a differential coefficient but with symbols other than d or ∂. The `diffcoeff` package offers three alternatives, all with the same pattern of optional and mandatory arguments as for `\diff`, except for the slash form. There is *no* slash option.

An uppercase D is used in place of d for the *material* or *substantive* derivative of a quantity in (for example) fluid dynamics. Write `\Diff` to invoke this command:[1]

```
\Diff{\rho}{t}=\diffp\rho t + \mathbf{u\cdot}\nabla\rho
```

$$\Longrightarrow \frac{D\rho}{Dt} = \frac{\partial\rho}{\partial t} + \mathbf{u}\cdot\nabla\rho.$$

(The braces could also be removed from the arguments of `\Diff` as they have been from the arguments of `\diffp`.)

The 'D's are romanised (along with the 'd's of ordinary derivatives) with the

```
\diffset[roman = true]
```

command. The default is math italic.

The command `\diffd` will form a fraction often used in introductory calculus texts (and other places):[2]

`\diffd{y}{x}` $\Longrightarrow \dfrac{\delta y}{\delta x}$.

Similarly, `\Diffd` forms a fraction with Δ:

`\Diffd{y}{x}` $\Longrightarrow \dfrac{\Delta y}{\Delta x}$.

[1]The `\diffp` command, the partial derivative, in the example is discussed in the next section.

[2]I considered using `\diffg` for this command as in 'diff greek' but decided that the more likely mind-phrase is 'diff delta', leading to the use of 'd' rather than 'g'.

Higher order forms of these derivatives are produced in the same way as with `\diff`, using an optional argument to specify the order:

$$\texttt{\textbackslash diffd[2]\{y\}\{x\}} \implies \frac{\delta^2 y}{\delta x^2}.$$

A final optional argument, enclosed in braces, specifies a point of evaluation, care being taken, as ever, to ensure that there is no space between it and the second mandatory argument:

$$\texttt{\textbackslash Diffd\{y\}\{x\}\{x=0\}} \implies \left.\frac{\Delta y}{\Delta x}\right|_{x=0}.$$

3 Partial differential coefficients

Partial differential coefficients follow the same pattern as for ordinary derivatives, with some generalisations arising from the greater possibilities. The command this time is `\diffp`. Thus **\diffp{F}{x}** produces $\frac{\partial F}{\partial x}$ in text style and

$$\frac{\partial F}{\partial x}$$

in display style. Braces can be omitted for single token differentiands and variables: `\diffp Fx` does the job. As for `\diff`, there is a slash form for more casual use: `\diffp F/x` displaying as $\partial F/\partial x$. Given that `\partial` takes 8 keystrokes to type, the slash form *does* economise on keystrokes for a partial derivative.

Again an optional argument allows the specification of the order of differentiation and it may be numeric or algebraic or a mix of the two. For a second or, indeed, an $n+4$th-order partial derivative,

$$\texttt{\textbackslash diffp[n+4]\{F\}\{x\}} \implies \frac{\partial^{n+4} F}{\partial x^{n+4}},$$

$$\texttt{\textbackslash diffp[n+4]\{F\}/\{x\}} \implies \partial^{n+4} F/\partial x^{n+4},$$

In a subject like thermodynamics, there is a need to indicate which variable or variables are held constant when the differentiation occurs. To show this, append a final optional argument. Thus to differentiate the entropy S in the temperature T while holding the volume V constant, write

$$\texttt{\textbackslash diffp\{S\}\{T\}\{V\}} \implies \left(\frac{\partial S}{\partial T}\right)_V$$

As with `\diff` note how the final optional argument is given in braces rather than square brackets, and that there must be *no space* before the argument: if used, it follows *immediately* on the second mandatory argument. This means that the differential coefficient can be followed immediately by a mathematical expression wrapped in \{ \}, or [ ], without the expression being confused with the (final) optional argument.

We could save a few keystrokes by writing this last example as `\diffp ST{V}`. The braces around the optional argument can *not* be dispensed with (otherwise there is no way of knowing that it *is* the final optional argument and not part of a following expression).

Note that for the slash form of the derivative it is anticipated that there will be no trailing optional argument. If you *do* use one, you will need to change the nudge value either with the `\diffset` command or, better, by including a spacing command in the third argument:

$$\texttt{\textbackslash diffp\{S\}/\{T\}\{\textbackslash;V\}} \implies \quad (\partial S/\partial T)_V$$

Without the spacing command, the subscript encroaches on the right parenthesis.

Appending the differentiand

If you want to remove the differentiand from the numerator to instead follow the derivative, one way, as for ordinary derivatives, is to leave the first mandatory argument empty and manually append the differentiand:

$$\texttt{\textbackslash diffp[n]\{\}xf(x)} \implies \quad \frac{\partial^n}{\partial x^n} f(x).$$

However, you may wonder how that would look with the differentiand in the numerator, which is a good reason for preferring the starred form of the `\diffp` command to achieve an appended derivative:

$$\texttt{\textbackslash diffp*[n]\{f(x)\}x} \implies \quad \frac{\partial^n}{\partial x^n} f(x).$$

Now it is easy to switch between an appended differentiand and one in the numerator simply by inserting or deleting the asterisk. In the slash form, parentheses are automatically inserted around the differential operator:

$$\texttt{\textbackslash diffp*[n]\{f(x)\}/x} \implies \quad (\partial^n/\partial x^n)f(x).$$

It also happens, for example in thermodynamics, that you may wish to both append the differentiand *and* indicate variables held constant. In that case, the starred `\diffp` command is much easier to use. Thus, to express a relation in thermodynamics,

```
\diffp*{\frac {P}{T}}{U}{V} = \diffp*{\frac{1}{T}}{V}{U}
```

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

where the starred form automatically takes care of the parentheses and subscripts. Again, not all the braces are necessary, with some help to readability:

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

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

3.1 Mixed partial derivatives

The new thing with partial derivatives, not present with ordinary derivatives, is *mixed* partial derivatives, where there is more than one variable of differentiation. If each variable is differentiated only to the first order, then it is easy to specify the derivative. Say $f(x, y, z)$ is a function of three variables, as indicated. Then

$$\texttt{\textbackslash diffp\{f\}\{x,y,z\}} \implies \frac{\partial^3 f}{\partial x\,\partial y\,\partial z}.$$

The variables of differentiation are listed in order in a comma list forming the second mandatory argument. The total order of differentiation (3 in this example) is inserted automatically – `diffcoeff` does the calculation itself. There is also a slash form:

$$\texttt{\textbackslash diffp\{f\}/\{x,y,z\}} \implies \partial^3 f/\partial x\,\partial y\,\partial z.$$

If we want to differentiate variables to higher order, then their orders need to be specified explicitly. To do so use a comma list also in the *optional* argument:

$$\texttt{\textbackslash diffp[2,3]\{f\}\{x,y,z\}} \implies \frac{\partial^6 f}{\partial x^2\,\partial y^3\,\partial z}.$$

Notice that the overall order of the derivative – 6 – is again automatically calculated and inserted as a superscript on the ∂ symbol in the numerator. In this example, the comma list of orders has only two members, even though there are three variables. It is assumed that the orders given in the comma list of orders 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, by default, 1. Thus we need to specify only 2 and 3 in the example; the order of z is 1 by default.

But you *cannot* use an order specification like `[,,2]`. This will be treated as if it were `[2]`. (This is a feature of comma lists in the expl3 language used by `diffcoeff.sty`.) Instead write `[1,1,2]`. It is only the *tail* of an order specification which can be omitted.

The automatic calculation of the overall order of differentiation remains true even when some or all of the orders for the individual variables are algebraic. For example, differentiating in three variables with orders `2k`, `m-k-2`, `m+k+3`, we have

$$\texttt{\textbackslash diffp[2k-1,m-k-2,m+k+3]\{F(x,y,z)\}\{x,y,z\}}$$
$$\implies \frac{\partial^{2k+2m} F(x, y, z)}{\partial x^{2k-1}\,\partial y^{m-k-2}\,\partial z^{m+k+3}},$$

3.2 The order-override option

In this example the overall order is presented as `2k+2m`. You might prefer this to be presented as `2(k+m)` instead. Although `diffcoeff` takes some steps to present the overall order appropriately, it does not factorise expressions. If you want to present the order in a manner distinct from that of `diffcoeff`, use the *order-override option*, which is a second optional argument immediately following the first:

```
\diffp[2k-1,m-k-2,m+k+3][2(k+m)]{F(x,y,z)}{x,y,z}
```

$$\implies \quad \frac{\partial^{2(m+k)}F(x,y,z)}{\partial x^{2k-1}\,\partial y^{m-k-2}\,\partial z^{m+k+3}}.$$

The order-override option does exactly that: overrides the presentation of the calculated order with the manually given one. (In fact the algorithm does not get called at all.)

Order specifications beyond the scope of `diffcoeff.sty`

The order specification can include signed integers, variables like k and α with signed integer coeffients, and products of any number of variables like mn or kmn with signed integer coefficients. The algorithm that calculates the overall order in `diffcoeff.sty` *cannot* handle exponents, subscripts or parentheses. For such constructs, or more exotic ones, the order-override option is always available. If it is present (even if empty), the algorithm is bypassed completely and one can include 'anything' there without causing error.

I doubt that these limitations matter in any practical sense. We are in 'overkill' territory here. Mixed partial derivatives are used far more rarely than the 'pure' ones, and mixed partial derivatives to 'exotic' orders of differentiation are used *vanishingly* rarely, and in any case the order-override option is always available. But should you, in some freak circumstance, find yourself needing to write such things, then I suggest you use `diffcoeffx.sty`, which is `diffcoeff.sty` 'on steroids'. It can handle the situations described above that are beyond the scope of `diffcoeff.sty`, and it uses exactly the same commands so there is nothing new to remember. It also provides additonal functionality for the trailing optional argument.

Presentation of the overall order

To take a grotesque example, that will never arise in practice, consider the following:

```
\diffp[kmn-mn+n-1,2kmn-mn+2n-1,n+1]{f}{x,y,z,w}
```

$$\implies \frac{\partial^{3kmn-2mn+4n}f}{\partial x^{kmn-mn+n-1}\,\partial y^{2kmn-mn+2n-1}\,\partial z^{n+1}\,\partial w}.$$

As noted earlier, since the final variable w is differentiated only to order 1, there is no need to specify it in the comma list of orders. The implicit 1 contributes to the vanishing of the numerical part in the overall order of differentiation. In this example, the overall order contains multivariable terms, kmn and mn. `diffcoeff` initially organises these in the sequence: ... 3-variable terms before 2-variable terms before single-variable terms, generally before the numerical term. However if a minus sign precedes the first many-variable term, and the numerical term is positive, it will be presented first:

$$\texttt{\textbackslash diffp[12-2km,k-1,km+1]\{f\}\{x,y,z,w\}} \implies \frac{\partial^{13-km+k} f}{\partial x^{12-2km}\,\partial y^{km-1}\,\partial z^{k+1}\,\partial w}.$$

Should the numerical term either vanish or be negative and the leading algebraic term is preceded by a minus sign, `diffcoeff` will look for an algebraic term with a preceding + sign and put that first:

```
\diffp[2km-3k-1,2k-1,-3km+4k+1]{f}{x,y,z,w}
```

$$\implies \frac{\partial^{3k-km} f}{\partial x^{2km-3k-1}\,\partial y^{2k-1}\,\partial z^{-3km+4k+1}\,\partial w}.$$

3.3 \diffset: formatting tweaks

As with ordinary derivatives, there are a number of tweaks one can make to the display of a partial derivative.

You may want more space between the ∂ symbol in the numerator of a partial derivative and the superscripted order of the derivative. The separation can be altered by using the

`\diffset[p-sep = `n`]`

command which adds an extra n mu to TEX's spacing. The default value is 1 (i.e. 1 mu). The new separation will affect all derivatives following the new setting. Put in the preamble, the new separation will be document-wide.

You may also want to adjust the spacing between the terms in the denominator. This can be done with the command

`\diffset[sep=`n`]`

which adds an extra n mu to TEX's spacing. The default value is 2 mu.

If you wish to indicate the point at which a partial derivative is evaluated, you may not want to use parentheses, since these when subscripted are widely held to indicate variables held constant. To change the delimiter on the right to a vertical line, use

```
\diffset[p-delims = .  | ],
```

the dot suppressing the delimiter on the left. (Note that to use `\{` and `\}` as delimiters, `\diffset` must be placed between maths delimiters.)

Changing the delimiters will usually require a repositioning of the subscript. The command is

`\diffset[p-nudge = `n`].`

For parentheses the default value of n is -6, but for the vertical rule a zero value is appropriate. Thus the overall command for . | would be

```
\diffset[p-delims = .  |, p-nudge = 0] .
```

Writing

```
\diffset
```

will return all settings to their default values.

3.4 Variations

Multi-character variables of differentiation

In thermodynamics one may want to differentiate in the reciprocal of the temperature, $1/T$. In tensor calculus the differentiations are almost always in terms of super- or subscripted coordinates, and in many other contexts this is the case too. This is why a comma list is used in `diffcoeff` for specifying the variables of differentiation for partial derivatives. Although it would be nice to write the minimal `{xy}` for this rather than `{x,y}`, the extra writing is trivial and the comma list allows the simplest handling of multi-character variables:

$$\texttt{\textbackslash diffp\{A\_i\}\{ x\^{}j,x\^{}k \}} \implies \frac{\partial^2 A_i}{\partial x^j\,\partial x^k},$$

taken from tensor calculus, or this strange object taken from statistical mechanics:

$$\texttt{\textbackslash diffp[2]q\{\textbackslash frac 1\textbackslash Theta\}} \implies \frac{\partial^2 q}{\partial(\frac{1}{\Theta})^2}.$$

The parentheses have been inserted automatically by `diffcoeff` to clarify exactly what the variable of differentiation is.

Use of phantoms when appending differentiands

As for ordinary derivatives, when appending a differentiand you may want to include a phantom (`\hphantom` etc.) in the numerator of the differential coefficient to alter the placement of the ∂ symbol. This may be particularly relevant if the order of differentiation is a multi-character symbol or if there are a number of variables of differentiation.

Either means of achieving the appended differentiand achieve the same result:

$$\texttt{\textbackslash diffp[m,2]\{\textbackslash hphantom\{\textbackslash partial y \textbackslash partial \}\}\{x,y,z\} (\textbackslash ln \textbackslash cos x + \textbackslash ln \textbackslash sin y)z} \implies \frac{\partial^{m+3}}{\partial x^m\,\partial y^2\,\partial z}(\ln\cos x + \ln\sin y)z,$$

$$\texttt{\textbackslash diffp*[m,2]\{\textbackslash hphantom\{\textbackslash partial y \textbackslash partial \}(\textbackslash ln \textbackslash cos x + \textbackslash ln \textbackslash sin y)z\}\{x,y,z\}} \implies \frac{\partial^{m+3}}{\partial x^m\,\partial y^2\,\partial z}(\ln\cos x + \ln\sin y)z,$$

which is to be compared with the derivative without the phantom,

```
\diffp*[m,2]{(\ln \cos x + \ln \sin y)z}{x,y,z}
```

$$\Longrightarrow \quad \frac{\partial^{m+3}}{\partial x^m\,\partial y^2\,\partial z}(\ln\cos x+\ln\sin y)z.$$

In the starred form, `diffcoeff` understands that the phantom is not appended with the differentiand but stays in the numerator. (But a *second* phantom would be appended.)

Iterated derivatives

Partial derivatives can be iterated. For example,

`\diffp f{x,y} = \diffp*{\diffp fy}x` $\Longrightarrow \dfrac{\partial^2 f}{\partial x\,\partial y}=\dfrac{\partial}{\partial x}\dfrac{\partial f}{\partial y},$

`\diffp f{x,y} = \diffp{\diffp fy}x` $\Longrightarrow \dfrac{\partial^2 f}{\partial x\,\partial y}=\dfrac{\partial\frac{\partial f}{\partial y}}{\partial x}.$

It is easy to switch between these forms by inserting or deleting the asterisk.

3.5 Jacobians

`diffcoeff` provides a command `\jacob` for constructing Jacobians. For example

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

The comma lists can contain any number of variables. `\jacob` does *not* check that the two arguments contain the same number of variables, so it is perfectly possible to form an object like

```
\jacob{u,v,w}{x,y} ,
```

which as far as I know has no meaning.

4 Discussion of the code

I set about creating this package when faced with trying to parse LaTeX expressions involving derivatives for another program I was working on. Trying to parse `\frac{d<something>}{d<something else>}`, perhaps with `\mathrm{d}`'s, and a superscript on the first `d`, perhaps with a `\tfrac` or `\dfrac` for the `\frac`, wasn't quite hopeless, but it was certainly *messy.* (I used regular expressions to transform the fraction into something more systematic.)

4.1 Other packages

Looking through the MiKTEX distribution and, less assiduously, through CTAN, produced the following packages which provide macros for derivatives. (Strangely, AMS packages do not touch this subject, as far as I can see.)

- `bropd`
  - `\od[n]{y}{x}` and `\pd[n]{y}{x}` for ordinary and partial derivatives of order `n` in one variable
  - `\pd{u}{x,x,t}` for a mixed partial derivative, order 2 in `x`, 1 in `t`
  - `\pd{}{z}{x+y}` for appending `(x+y)`
  - `\pd{!}{z}{x+y}` for appending `x+y`
- `commath`
  - `\od[n]{y}{x}` and `\pd[n]{y}{x}` for ordinary and partial derivatives of order `n` in one variable
  - `\md{f}{5}{x}{2}{y}{3}` for a 5th order mixed partial derivative
  - `\tmd`, `\dmd` and similar commands for forcing text and display styles
- `esdiff`
  - `\diff[n]{y}{x}` and `\diffp[n]{y}{x}` for ordinary and partial derivatives of order `n` in one variable
  - `\diffp{f}{{x^2}{y}{z^3}}` for a mixed partial derivative of order 6 in three variables
  - `\diff*[n]{y}{x}{0}` for indicating the point of evaluation of the derivative (using a subscript on parentheses)
  - `\diffp*{P}{T}{V}` to indicate a variable held constant
- `physymb`
  - `\ud{y}{x}` and `\pd{y}{x}` for ordinary and partial derivatives of first order
  - `\udd{y}{x}`, `\uddd{y}{x}` and `\pdd{y}{x}`, `\pddd{y}{x}` for second and third order ordinary and partial derivatives
  - higher order derivatives not catered for

None of the packages quite gave what I wanted (but for all that, I suspect cope with well over 90% of use cases). `esdiff` comes closest but failed when it came to combining algebraic and numeric orders of differentation in a mixed partial derivative. Also the need to em-brace variables in a mixed partial derivative in `esdiff` was another (small) count against it.

4.2 diffcoeff.sty

- The distinctive feature of `diffcoeff.sty` is that it will automatically form the overall order of a mixed partial derivative, including those containing both algebraic and numeric contributions to the order:

$$\texttt{\textbackslash diffp[m-k-1,m+k]\{F(x,y,z)\}\{x,y,z\}} \implies \frac{\partial^{2m} F(x,y,z)}{\partial x^{m-k-1}\,\partial y^{m+k}\,\partial z}.$$

- Ease of use was another major consideration, trying to avoid the unnecessary writing of superscripts and subscripts and brace pairs. In this example, no superscripts are written and only the two inescapable brace pairs are required.

  – The use of a comma list for the second mandatory argument in a partial derivative is another example. That makes differentiations in super- or subscripted symbols easier to both write and read by avoiding 'entanglements' of braces.

- I've also tried to make the options 'natural' and consistent across both ordinary and partial derivatives. Looking at the other packages listed above, writing something like `\diff[n]{f}{x}` (which can be trimmed to `\diff[n]fx` in this instance) seems 'natural' – only `physymb` deviates from the pattern. It seems consistent with this pattern to use a comma list as an optional argument for mixed partial derivatives.

- I debated whether to include provision for points of evaluation and variables held constant into the `\diff` and `\diffp` commands. `esdiff` certainly allows this. I think a case can be made, in subjects like thermodynamics, to consider the parentheses and subscript as part of the overall symbol. The partial derivative itself doesn't give the full story; it is ambiguous. Hence provision for these extra elements was included in `\diff` and `\diffp`. It's positioning as a final optional argument also felt natural given the position of the resulting symbol in the displayed derivative:

$$\texttt{\textbackslash diffp ST\{V\}} \implies \left(\frac{\partial S}{\partial T}\right)_V$$

- Although initially I used standard square brackets for this trailing optional argument, the possibility of an immediately following mathematical expression being enclosed in square brackets convinced me to use braces for the argument. An immediately following expression can now be enclosed in `[ ]`, or `\{ \}`, without ambiguity.

- The star option also prompted the reflection: is it needed? One can always leave the first mandatory argument empty and append the differentiand 'by hand'. But once the provision for points of evaluation or variables held constant was incorporated into the `\diff` and `\diffp` commands, the star option became the simplest way of handling appended differentiands using the extra provision. (Note that it conflicts with the star option in `esdiff`, but I can't see the packages ever being used together.) And once the option is available, it provides a simple way to switch between differentiand in the numerator/differentiand appended.

- The final option added to the package was the slash option. This was prompted after seeing the expression $(d/dz)[\log f(z)]$ in a text on statistical mechanics. Alerted to the form, I then skimmed through various texts and found this form of the derivative was used sufficiently often to justify inclusion. The placement of the slash, between the two mandatory arguments, seemed more-or-less self-evident.

4.3 The mixed partial derivatives algorithm

It occurred to me, after I had created an algorithm for splitting a linear expression composed of signs, integers and variables into its numerical and algebraic parts, that the same algorithm could be used in a recursive way to simplify the algebraic part of the expression.

Given an order specification like, say, [**2m+k–1,2m–k+1,2k,1**], the idea is to concatenate the terms with intervening + signs, thus **2m+k–1+2m–k+1+2k+1**, then split this expression into numeric and algebraic parts, giving **–1+1+1** for the numeric part and **2m+k+2m–k+2k** for the algebraic part. The numeric part, assumed to be a combination of integers, is evaluated and the result stored. For the algebraic part, remove throughout all instances of one of the variables, say **m**. The result is **2+k+2–k+2k**. Split this into numeric and algebraic parts: **2+2** for the numeric part and **k–k+2k** for the algebraic part. Evaluate the numeric part, **+4**, and you have the overall coefficient of the variable **m**. Repeat the process for the next variable, and so on until all variables have been accounted for.

In fact repeating the process for the next variable, **k** in this example, immediately reveals a problem. Removing **k** from **k–k+2k** leaves **–+2** which evaluates to **–2** whereas the correct coefficient for **k** should be **+2**. The solution is to insert **1** before any 'bare' variable – a variable preceded only by a sign rather than a number. In that case the expression we remove **k** from is **1k–1k+2k** giving the correct overall coefficient **+2**.

A second problem may arise if there are terms involving products of variables as in the order specification [**mk–2,2m+1,2k+1**]. This splits into a numeric part **–2+1+1** evaluating to **0**, and an algebraic part **mk+2m+2k**. If we choose **m** as the first variable to remove from this expression, we get **+2** for the numeric part (and hence the overall coefficient of **m**) and **k+2k** for the algebraic part, which is wrong, since that will lead to the wrong overall coefficient **+3** for **k**, and the 2-variable term **mk** will not get treated at all. The cure is to treat **mk** as a variable itself, count the number of tokens in each such product and start the removal process with the largest.

Table 1: State transitions

	Curr. state	Curr. token	Action	Next state
1	**s**	s	$Ts \to s'$; $T = s'$	**s**
2	**s**	d	Td	**n**
3	**s**	v	Vv; $T1v$	**a**
4	**n**	s	$\mathbf{N}T$; $T = s$	**s**
5	**n**	d	Td	**n**
6	**n**	v	Vv; Tv	**a**
7	**a**	s	$\mathbf{V}V$,; $V = \emptyset$; $\mathbf{A}T$; $T = s$	**s**
8	**a**	d	error	**!!**
9	**a**	v	Vv; Tv	**a**

The splitting algorithm

Write s for a sign, one of $+$, $-$, and **s** for the state of assembling a signed term; a signed term is a string of one or more signs. Write d for a digit, one of 0123456789, and **n** for the state of assembling a numeric term; a numeric term is a signed term followed by a string of one or more digits. Write v for a variable, usually a letter from the roman alphabet but in principle any single token that is not a sign or a digit, and **a** for the state of assembling an algebraic term; an algebraic term is a numeric term followed by a string of one or more variables. Rather than referring to a signed-term-assembling state, we shall (obviously) simply refer to a *signed state*, and similarly to a *numeric state* and an *algebraic state*.

We also want to record the variables in the extended sense of products of same. Call a one-token variable a prime variable. Then in this desired sense, a variable is a string of one or more prime variables.

Let **E** be the initial expression. Let **A** be a container for the algebraic part of **E**; let **N** be a container for the numeric part of **E**; and let **V** be a container for the extended variables in **E**. Let T be a container in which to accumulate the current term, and V a container in which to accumulate the current extended variable (if any). Initially all these containers are empty ($\emptyset$).

We work through **E** token by token from the left. The table shows the alternatives.

- Row 1. The current token is a sign s and the system is in a signed state **s**. We append s to the current term, Ts, then resolve the juxtaposition of signs according to the familiar rules: $++ \to +$, $-- \to +$, $+- \to -$, $-+ \to -$, so that T contains only the resolved sign s'. The system remains in a signed state.

- Row 2. The current token is a digit d and the system is in a signed- state **s**. We append d to the current term, Td (which will now consist of a sign and a digit), and the system shifts to a numeric state **n**.

- Row 3. The current token is a prime variable v and the system is in a signed state **s**. We start assembling an extended variable, Vv, and append $1v$ to the current term, $T1v$, where the 1 is necessary as discussed earlier (and in any case 'sign variable' is not a recognised *term* – neither signed, numeric or algebraic). The system shifts to an algebraic state **a**.

- Row 4. The current token is a sign s and the system is in a numeric state **n**. The current term is a numeric term, a sign followed by at least one digit, and is complete. We append it to the numeric part **N** of **E**, $\mathbf{N}T$, then initialise T to s. The system shifts to a signed state.

- Row 5. The current token is a digit d and the system is in a numeric state **n**. We append d to the current term, Td, and remain in a numeric state.

- Row 6. The current token is a prime variable v and the system is in a numeric state **n**. We start assembling a variable, Vv, and also append v to the current term, Tv. The system shifts to an algebraic state **a**.

- Row 7. The current token is a sign s and the system is in an algebraic state **a**. The current term is an algebraic term, a sign followed by at least one digit followed by at least one prime variable, and is complete. We append it to the algebraic part **A** of **E**, $\mathbf{A}T$, then initialise T to s. We also append V, in which we have been accumulating the (extended) variable, to **V**, $\mathbf{V}V$,, then empty V in preparation for the next (extended) variable. Attention is drawn to the comma following V also appended to **V**, so that we can distinguish where one variable ends and the next begins. The system shifts to a signed state.

- Row 8. The current token is a digit d and the system is in an algebraic state **a**. This situation should not arise. We don't write $k2$; we write $2k$ – number precedes variable. An error is generated.

- Row 9. The current token is a variable v and the system is in an algebraic state **a**. We append v to the current extended variable, Vv, and also append v to the current term, Tv. The system remains in an algebraic state **a**.

To get things under way, an initial plus sign is put in T, $T = +$, and the system is set to the signed state **s**. In order that *all* terms of **E** are recorded in either **N** or **A**, and all extended variables in **V**, we append a plus sign to **E**: **E**+. Since an expression doesn't end with a trailing sign (we don't write, e.g., **2m+k–**), the process necessarily terminates either in row 4 or row 7 with the final term appended either to **N** or **A** and with $T = +$; if it terminates in row 7, the final extended variable is appended to **V**, $\mathbf{V}V$ (and V is emptied, although that hardly matters at this point).

An enlarged scheme?

Row 8 of our table generates an error: a digit following a variable. But having allowed products of variables like `mn` (mn), it is very tempting to allow `mm`, i.e. `m^2` (m^2) and, indeed, `m^n` (m^n). And if we allow `m^2` and `m^n`, how can we say no to subscripted forms like `k_2` (k_2) and `k_n` (k_n)? Or, for that matter, `k_+` (k_+) and `k_-` (k_-), and therefore `m^+` (m^+) and `m^-` (m^-)? And having extended the scheme in this way to exponents of *variables*, surely it should also encompass exponents of *numbers*, not only an obvious case like `2^2` (2^2) but less obviously, yet still compellingly, `2^n` (2^n)?

Each of these extensions produces its own problems, but they can all be accommodated within an enlarged scheme, as can the use of parentheses (with numerical coefficients). Table 1 translates neatly into code. Rather than add these complications to `diffcoeff.sty`, I have transferred the enlarged scheme to `diffcoeff.sty`'s 'big brother', `diffcoeffx.sty`. The comparable table and routine resulting from it in `diffcoeffx.sty` is much bigger and less obvious than in `diffcoeff.sty`.

Some code details

In the code, the states are distinguished by integers as indicated in Table 2a. Tokens are assigned similar integer indexes, as indicated in the table. The relevant routine is `\__diffco_get_curr_index:NN`. The actions embodied in Table 1 are encoded in `\__diffco_compare_states:NNNNN` which is a direct translation of the table into expl3 code.

Table 2: Some code details

(a) State integers

State	Index	Tokens
signed	0	$+ -$
numeric	1	0123456789
algebraic	2	variables

(b) Translations

Symbol	Code variable
s,d,v	`\l__diffco_curr_tok_tl`
T	`\l__diffco_curr_term_tl`
V	`\l__diffco_curr_var_tl`
N	`\l__diffco_nos_tl`
A	`\l__diffco_alg_tl`
V	`\l__diffco_vars_prop`

A property list is used to store the variables, organised by size – the number of tokens composing an extended variable. This enables the sorting by size needed for the determination of the overall coefficients of variables by removing them in turn from the algebraic part of the expression. That process is conducted in the routine `\__diffco_eval_vars:NN`. The variables are recorded only on the first scan through the order specification expression. This is the function of the boolean `\l__diffco_vars_noted_bool` which is set in `\__diffco_eval_vars:NN`. Evaluation of the numeric parts of expressions is provided by `\__diffco_eval_nos:N`.

5 Summary of main commands

Ordinary derivatives

The syntax is

```
\diff[order]{differentiand}{variable}{point of evaluation}
```

for the differentiand in the numerator, and where the final argument, although using braces, is an *optional* argument. A starred form appends the differentiand:

```
\diff*[order]{differentiand}{variable}{point of evaluation}
```

No space must occur between the final optional argument, if it is used, and the second mandatory argument.

There are also slash forms of both these commands:

```
\diff[order]{differentiand}/{variable}{point of evaluation}
\diff*[order]{differentiand}/{variable}{point of evaluation}
```

For the starred form, the differential coefficient is enclosed in parentheses.

Precisely similar definitions, but without the slash forms, apply to `\Diff`, forming a differential coefficient with D, `\diffd`, forming a differential coefficient with δ, and `\Diffd`, forming a differential coefficient with Δ.

Partial derivatives

The syntax is

```
\diffp[order spec.][order override]{differentiand}{variables}{constant
variables}
```

for the differentiand in the numerator and where the final argument, although in braces, is an *optional* argument. No space must occur between the final optional argument, if it is used, and the second mandatory argument. The **order spec.** is a comma-separated list; the **variables** is also a comma-separated list. A starred form appends the differentiand:

```
\diffp*[order][order override]{differentiand}{variables}{constant
variables}
```

Slash forms exist also for these commands:

```
\diffp[order spec.][order override]{differentiand}/{variables}{constant
variables}
\diffp*[order][order override]{differentiand}/{variables}{constant
variables}
```

For the starred version of the slash form, the differential coefficient is enclosed in parentheses.

Settings

```
\diffset[option1=<value1>,option2=<value2>,...]
```

All numerical values should be integers (`diffcoeff` interprets this in units of mu, 1/18 of an em). To return all options to default values, write

```
\diffset
```

The options and defaults are

roman = false **true** gives upright (roman) **d** and **D**

d-delims = . | delimiters which, when subscripted, indicate the point of evaluation of an ordinary derivative

p-delims = ( ) delimiters which, when subscripted, indicate variables held constant for partial derivatives

d-nudge = 0 adjustment for positioning the subscript to the preceding delimiters

p-nudge = −6 adjustment for positioning the subscript to the preceding delimiters

d-sep = 1 additional separation between the d and its superscript in the numerator of a second or higher order ordinary derivative

p-sep = 1 additional separation between the ∂ and its superscript in the numerator of a second or higher order partial derivative

sep = 2 additional separation between the terms in the denominator of a mixed partial derivative