The intexgral package

v3.0.1

Valentin Dao*

2 January 2026

Abstract

While integral composition is prevalent in LaTeX, it often proves to be impractical. When the expression becomes complex, it is then difficult to modify its various elements in an illegible source code. To address this problem, the *intexgral* package offers a central macro whose only argument is the integrand. Everything else (symbol, limits, variables...) can be modified using a ⟨key=value⟩ interface. Unlike the traditional method, where the user chooses the limit order with the active characters _ and ^, the package had to establish a convention. Thus, for the two keys that will deal with limits, the assumed input will be _⟨lower limit⟩^⟨upper limit⟩. Since this package is written in *expl3*, it depends on the LaTeX3 group of packages *l3kernel* and *l3package*. In addition, *intexgral* uses the *derivative*[1] package to facilitate the manipulation of differentials, as well as *pkginfograb*[2].

*E-mail: vdao.texdev@gmail.com

[1]Available on CTAN under: https://ctan.org/pkg/derivative

[2]Available on CTAN under: https://ctan.org/pkg/pkginfograb

Contents

License

Fonts

Chronicle Text G3 Roman
© 2002, 2007 Hoefler & Frere-Jones.

Whitney-Medium
© 1996, 2009 Hoefler & Frere-Jones.

Monaspace Argon Medium
© 2023, GitHub https://github.com/githubnext/monaspace

Repository

See GitHub repository.

Acknowledgements

Many thanks to Plante and Slurpy for accompanying me on this adventure of learning TeX, your pieces of advice were priceless. I'd also like to thank Anthony, who has always kindly reviewed the new versions and given me ideas for those to come.

1 Package options

`\intexgralsetup` `\intexgralsetup {⟨package options⟩}`

new: v2.0.0

These options can be declared in a classic way with `\usepackage` or with `\intexgralsetup` in the preamble.

`invert-limits` `invert-limits=⟨true|false⟩`

This key inverts the limit order convention. The entirety of the document can only follow one convention.

`invert-diff` `invert-diff=⟨true|false⟩`

updated: v3.0.0

It is common to see the differentials placed before the integrand in physics papers. This key hence swaps their position.

`limits-mode` `limits-mode=⟨limits|nolimits⟩`

new: v3.0.0

Applies `\limits` or `\nolimits` to every single integral.

`italic` `italic=⟨true|false⟩`
`upright` `upright=⟨true|false⟩`

These are the two keys of the *derivative* package.

2 Presentation of the main macro

`\integral` `\integral [⟨list of keys⟩] {⟨integrand⟩}`

updated: v3.0.0

This macro is used to typeset integrals. It must, of course, be used in math mode only. Here is a first example in its simplest form:

```
\begin{equation}
  \integral{x}
\end{equation}
```

$$\int x \,\mathrm{d}x \tag{1}$$

Since this macro focuses on the use of keys, the first part of this documentation will present them, grouping them by domain. The second part will introduce a few additional macros that complement the use of certain keys. The rest of this document will outline the other features of the package.

3 List of keys

3.1 Integration limits

3.1.1 Defining the limits

`limits` `limits= {⟨mixed-list⟩}, ⟨keyword⟩`
`limits*` `limits*= {⟨mixed-list⟩}, ⟨keyword⟩`

This key determines the integration limits to be used, following the convention explained in the abstract. In its simplest form, the value of the key takes as its argument a list of elements separated by commas (*comma-separated values* or ⟨csv-list⟩[1]).

```
\begin{equation}
  \integral[limits={1, 10}]{f(x)}
\end{equation}
```

$$\int_{1}^{10} f(x)\,\mathrm{d}x \tag{2}$$

The notable advantage of the `limits` key is that you can specify the integration limits for several integrals, and the package automatically typesets the corresponding symbols. To separate pairs of limits, we use a semicolon (*semicolon-separated values* or ⟨ssv-list⟩[1]).

```
\begin{equation}
  \integral[limits={1, 2; 3, 4}, variables={x, y}]{f(x, y)}
\end{equation}
```

$$\int_{1}^{2}\int_{3}^{4} f(x,y)\,\mathrm{d}x\,\mathrm{d}y \tag{3}$$

It is also possible to specify predefined limits using keywords (see section 4.2). Finally, the starred variant allows the limits of the integral to be expressed as an interval. The direction of the brackets automatically adjusts itself to the presence of ±`\infty`.

```
\begin{equation}
  \integral[limits*={1, 10}]{f(x)}
\end{equation}
```

$$\int_{[1,10]} f(x)\,\mathrm{d}x \tag{4}$$

[1] We therefore mean by ⟨mixed-list⟩ that `limits` can accept a set of ⟨csv-list⟩ in a more general ⟨ssv-list⟩.

3.2 Display modes

mode — `mode=⟨default|nested|product⟩`

new: v3.0.0

Solely when the `limits` key is used, it is possible to choose between three distinct display styles. Each of these is fully compatible with the `invert-diff` option.

`default` Typesets all the symbols, then the integrand, then the variables. As the macro operates by default in this mode, it is not necessary to use the key with this value.

`nested` Typesets a nested integral where the symbol and integrand alternate before all the variables are written.

`product` Typesets a product of integrals where the symbol, integrand, and variables alternate.

Of course, nothing prevents you from using the `\integral` macro several times in a row to produce the same result as in the example below. However, for those who would prefer to obtain the result with a single macro, this method is permitted.

> ***Important:*** for the last two modes, the integral will be *cut off* in a certain way. In order to perform this action correctly, the same method must be used as with `limits`, i.e. by using a semicolon.

MODE `default`

```
\begin{equation}
  \integral[limits={1, 2; 3, 4; 5, 6}, variables={x, y, z}, mode=default]{xyz}
\end{equation}
```

$$\int_1^2\int_3^4\int_5^6 xyz\,\mathrm{d}x\,\mathrm{d}y\,\mathrm{d}z \tag{5}$$

MODE `nested`

```
\begin{equation}
  \integral[limits={1, 2; 3, 4; 5, 6}, variables={z, y, x}, mode=nested]{x;y;z}
\end{equation}
```

$$\int_1^2 x\int_3^4 y\int_5^6 z\,\mathrm{d}z\,\mathrm{d}y\,\mathrm{d}x \tag{6}$$

MODE `product`

```
\begin{equation}
  \integral[limits={1, 2; 3, 4; 5, 6}, variables={x, y, z}, mode=product]{x;y;z}
\end{equation}
```

$$\int_1^2 x\,\mathrm{d}x\int_3^4 y\,\mathrm{d}y\int_5^6 z\,\mathrm{d}z \tag{7}$$

To function correctly, it is obvious that the keys `limits` and `variables` (see 3.4.1), as well as the integrand, must have the same number of elements. If this is not the case,

compilation will not fail, but the expression of the integral may no longer make sense. Numerous warning messages will be displayed to inform you of this.

3.3 Symbol (and more about limits)

The `limits(*)` keys are very useful when typesetting a definite integral. However, this only covers final expressions where the limits of each variable have been specified. For more general cases — indefinite integrals — they are relatively inconvenient. To typeset a double integral over a surface S, one would have to write `limits={,;S,}`. Needless to say, this is rather inconvenient for the user and not optimal for the package. Furthermore, the glyph would not be correct. The set of keys below therefore offers an easy way to modify both the symbol and the bounds.

3.3.1 Selecting a symbol

`symbol` `symbol=⟨control sequence⟩`

updated: v3.0.0

This key accepts a macro designating an integral symbol. Any symbol previously defined by a control sequence is acceptable. If you attempt to use one that is not defined, it will be substituted with `\int` and a warning message will be issued.

> ***Remark:*** apart from verifying the existence of the macro, no specific checks are performed, and the value of the key is used as is to compose the symbol. This implies two things in particular. First, the symbol will depend on how it is defined. For instance, *amsmath* and *unicode-math* do not define `\iint` in the same way. The result will naturally be different. Furthermore, if you assign a *defined* macro as the key argument but it does not correspond to an integral symbol, the package will ignore it and still use the given symbol. In addition to the mathematical inconsistency that this may represent, it is likely to lead, under certain circumstances, to low-level errors.

```
\begin{equation}
  \integral[symbol=\iint, llimit=S, variables={x, y}]{f(x, y)}
\end{equation}
```

$$\iint_S f(x,y)\,\mathrm{d}x\,\mathrm{d}y \tag{8}$$

3.3.2 Generating many integrals

`nint` `nint=⟨integer⟩`

This key accepts an integer n which allows n integrals to be composed. It is advisable to use this key only if the number of symbols exceeds 4 to favour the glyphs defined by the mathematical font used.

```
\begin{equation}
  \integral[nint=5, llimit=\Omega, variables={x_1, x_2, x_3, x_4, x_5}]{f(x_1, x_2,
   x_3, x_4, x_5)}
\end{equation}
```

$$\idotsint_\Omega f(x_1,x_2,x_3,x_4,x_5)\,\mathrm{d}x_1\,\mathrm{d}x_2\,\mathrm{d}x_3\,\mathrm{d}x_4\,\mathrm{d}x_5 \tag{9}$$

3.3.3 Defining boundaries on an ad hoc basis

`llimit` `llimit=⟨lower limit⟩`
`ulimit` `ulimit=⟨upper limit⟩`

updated: v3.0.0

These two keys allow you to specify the lower and upper limits, respectively. They are only suitable if a single symbol is displayed. If you need to specify both limits, the `limits` key should be used instead.

```
\begin{equation}
  \integral[llimit={x^2 + y^2 \leq 1}, variables={x, y}]{f(x, y)}
\end{equation}
```

$$\int_{x^2+y^2\leq 1} f(x,y)\,\mathrm{d}x\,\mathrm{d}y \tag{10}$$

3.3.4 Recurring limits motif

Lower limits sometimes follow common patterns that can be generalised with keys, thus avoiding overloading the argument of `llimit`.

`domain` `domain=` {⟨*-list⟩}
`domain*` `domain*=` {⟨*-list⟩}

updated: v3.0.0

These keys accept a list delimited by an asterisk. Each element of the list is then parsed as follows:

- The first token of the item is passed `\mathbb` (preceded by `\uppercase` in case the SHIFT key is too far away for your fingers).
- The remaining tokens — which may be empty — are placed as superscript (or subscript for the starred variant of the key).

```
\begin{equation}
  \integral[symbol=\iint, domain={r*r}, variables={x, y}]{xy}
\end{equation}
```

$$\iint_{\mathbb{R}\times\mathbb{R}} xy\,\mathrm{d}x\,\mathrm{d}y \tag{11}$$

`boundary` `boundary=` {⟨lower limit⟩}

This key simply places the symbol ∂ before the lower bound.

```
\begin{equation}
  \integral[symbol=\oint, boundary=S, diff-vec]{G(\vec r)}
\end{equation}
```

$$\oint_{\partial S} G(\vec{r})\cdot\mathrm{d}\vec{r} \tag{12}$$

3.4 Differentials

3.4.1 Specifying the integration variables

`variables` `variables=` {⟨csv-list⟩}, ⟨keyword⟩, `none`

updated: v3.0.0

This key allows you to define the variables of the integral in the form of a ⟨csv-list⟩. Like `limits`, the key can accept a keyword as an argument. This behaviour is also explained later (see section 4.3).

> ***Remark:*** if no variable is specified, i.e. `variables` is not called, the package automatically sets "$\mathrm{d}x$" (or "$\mathrm{d}\vec{r}$" if `diff-vec` is active). Furthermore, if `none` is passed as a value, no variable will be displayed.

```
\begin{equation}
  \integral[variables=t]{t^2}
\end{equation}
```

$$\int t^2 \,\mathrm{d}t \tag{13}$$

3.4.2 Inlcuding the Jacobian

`jacobian` `jacobian`

This key enables the display of the Jacobian when defined by `\NewVariableKeyword`.

```
\begin{equation}
  \integral[limits={0, R; 0, 2\pi; 0, \pi}, variables=spherical, mode=product,
   jacobian]{;;}
\end{equation}
```

$$\int_0^R r^2 \,\mathrm{d}r \int_0^{2\pi} \sin\theta \,\mathrm{d}\theta \int_0^{\pi} \mathrm{d}\phi \tag{14}$$

As mentioned earlier, *derivative* is used by the package to compose differentials. The set of keys to follow, whose names will be prefixed with "`diff-`"[1], thus allows you to apply the features of the macro `\odif`.

3.4.3 Modifying the differential symbol

`diff-symb` `diff-symb=` ⟨differential macro⟩

This key allows you to change the style of differentials among those defined by `\NewDifferential`.

```
\NewDifferential{\feynmandiff}{\mathcal{D}}
\begin{equation}
  \integral[variables={q(t)}, diff-symb=\feynmandiff]{e^{+\dfrac{\imath S[q(t)]}{
   \hbar}}}
\end{equation}
```

[1] Although the key `diff-vec` is not linked to *derivative*.

$$\int e^{+\frac{\imath S[q(t)]}{\hbar}} \mathcal{D}q(t) \tag{15}$$

3.4.4 Creating a path integral

diff-vec `diff-vec`

This key applies a vector to each variable in addition to a centred dot. It is only compatible with the `default` mode.

```
\begin{equation}
  \integral[double=S, variables=S, diff-vec]{\vec F}
\end{equation}
```

$$\iint_S \vec{F} \cdot \mathrm{d}\vec{S} \tag{16}$$

3.4.5 Opting for the starred variant

diff-star `diff-star`

Activating this key is equivalent to using `\odif*` or its variants.

```
\begin{equation}
  \integral[double, variables={x, y}, diff-star]{x^2 \exp(y)}
\end{equation}
```

$$\iint x^2 \exp(y) \, \mathrm{d}_x \, \mathrm{d}_y \tag{17}$$

3.4.6 Designating options

diff-options `diff-options=` {⟨keyval list⟩}

This key accepts a list of keys as it would have been written as an optional argument of `\odif` or its variants. For example, `diff-options={order={2, 3}, var=all}` will be interpreted as `\odif[order={2, 3}, var=all]{...}`.

```
\begin{equation}
  \integral[diff-options={order=4}]{\bar\psi(x)(\imath\gamma^\mu\partial_\mu-m)\psi(
   x)}
\end{equation}
```

$$\int \bar{\psi}(x)(\imath\gamma^\mu\partial_\mu - m)\psi(x) \, \mathrm{d}^4x \tag{18}$$

4 Additional macros

In addition to the wide range of keys defined by the package, a few macros also enhance their uses.

4.1 Custom differentials placement

`\differentials` `\differentials`

Although the `invert-diff` option exists, it is often desirable to be able to place differentials wherever one wishes. For example, they are frequently seen in the numerator of a fraction when the numerator is 1. To meet this need, the package provides the macro `\differentials`. For the `default` and `nested` modes, the macro typesets all differentials at once. With `product`, the macro must be repeated between each semicolon.

```
\begin{equation}
  \integral{\frac{\differentials}{x}}
\end{equation}
```

$$\int \frac{\mathrm{d}x}{x} \tag{19}$$

4.2 Looking back at the *limits* key

\NewLimitsKeyword
\RenewLimitsKeyword
\ProvideLimitsKeyword
\DeclareLimitsKeyword

`\NewLimitsKeyword {⟨keyword⟩} {⟨limits⟩}`

It is common to have to specify the same limits in an integral. To make them easier to write, the `limits(*)` keys accept, in addition to explicit limits, a keyword designating a pair of them predefined by one of these four macros:

`\NewLimitsKeyword` Creates a new keyword and issues an error if it already exists.

`\RenewLimitsKeyword` Redefines a keyword and issues an error if it does not already exist.

`\ProvideLimitsKeyword` Only creates a new keyword if it does not already exist. No message will be issued if this is the case.

`\DeclareLimitsKeyword` Creates a new keyword regardless, overwriting any previous definition.

Here is the list of keywords already defined by the package and the limits they contain:

`ab`	a, b
`unit`	$0, 1$
`real`	$-\infty, +\infty$
`positive`	$0, +\infty$
`negative`	$-\infty, 0$
`circle`	$0, 2\pi$
`sircle`	$0, \pi$
`qcircle`	$0, \frac{\pi}{2}$
`height`	$0, H$
`radius`	$0, R$
`length`	$0, L$
`time`	$0, T$

> ***Remark:*** keywords contain both limits of an integral at the same time. They must therefore be preceded and/or followed by semicolons.
>
> ***Important:*** due to the implementation of `\NewLimitsKeyword` and its variants, the user *must* follow the package's limit order convention, even if `invert-limits` is set to `true`

We could therefore modify the expression of an integral as follows:

```
\begin{equation}
  \integral[limits={radius;circle;scircle}, variables={\rho, z, \phi}]{\rho z \phi}
\end{equation}
```

$$\int_0^R \int_0^{2\pi} \int_0^{\pi} \rho z \phi \,\mathrm{d}\rho \,\mathrm{d}z \,\mathrm{d}\phi \tag{20}$$

It is entirely possible to combine these keywords with the more explicit syntax of `limits`.

```
\begin{equation}
  \integral[limits={time; 34, 40; height}, variables={h, \omega, t}]{h \omega t}
\end{equation}
```

$$\int_0^T \int_{34}^{40} \int_0^H h\omega t \,\mathrm{d}h \,\mathrm{d}\omega \,\mathrm{d}t \tag{21}$$

4.3 Looking back at the *variables* key

\NewVariableKeyword
\RenewVariableKeyword
\ProvideVariableKeyword
\DeclareVariableKeyword

`\NewVariableKeyword {⟨keyword⟩} {⟨variables⟩} [⟨jacobian⟩]`

Similarly, the same groups of variables often reappear. We can therefore also define keywords containing a set of pre-registered variables. The variants of this macro behave in the same way as for `\NewLimitsKeyword`. The only difference is that here it is possible to define a Jacobian, in the form of a ⟨csv-list⟩ if necessary. Its display is then controlled by the `jacobian` key as explained above. Here is the list of variable keywords already defined by the package, with the Jacobian for some of them:

`cartesian`	x, y, z
`planar`	x, y
`polar`	r, θ (r)
`cylindrical`	r, θ, z (r)
`spherical`	r, θ, ϕ $(r^2, \sin\theta)$

```
\begin{equation}
  \integral[triple=V, variables=spherical]{f(x, y, z)}
\end{equation}
```

$$\iiint_V f(x, y, z) \,\mathrm{d}r \,\mathrm{d}\theta \,\mathrm{d}\phi \tag{22}$$

4.4 Looking back at the *symbol* key

\NewSymbolKeyword
\RenewSymbolKeyword
\ProvideSymbolKeyword
\DeclareSymbolKeyword

`\NewSymbolKeyword {⟨key⟩} {⟨symbol⟩}`

new: v3.0.0

In order to typeset indefinite integrals, the package essentially offers two keys: `symbol` and `llimit`. Although they are simple to use, it is still laborious to write both of them with their values. This is why *intexgral* provides so-called *shortcut* keys, whose purpose is to combine the selection of the symbol and the limits. Unlike the two previous macros, this involves creating entirely new keys, rather than simply specific values assigned to `symbol`. On their own, these keys modify the integral symbol. The value of these keys corresponds to the lower bound. Here is the list of all *symbol-key* defined by the package:

`single`	used symbol = `\int`
`double`	used symbol = `\iint`
`triple`	used symbol = `\iiint`
`quadruple`	used symbol = `\iiiint`
`contour`	used symbol = `\oint`
`surface`	used symbol = `\oiint`
`volume`	used symbol = `\oiiint`

> ***Important:*** allowing users to create their own keys for the `\integral` macro poses a risk that will be explained in more detail in the next section. For now, just remember that all these macros[1] will issue a warning message if you attempt to create a new *symbol-key* with the same name as a group of defined limits.

```
\begin{equation}
  \integral[contour=\mathcal{C}, diff-vec]{f(\vec r)}
\end{equation}
```

$$\oint_{\mathcal{C}} f(\vec{r}) \cdot \mathrm{d}\vec{r} \tag{23}$$

5 *Special* syntax

5.1 Presentation of the syntax

`\integral` `\integral [⟨limits:variables(+j):mode⟩] {⟨integrand⟩}`

new: v3.0.0

With regard to definite integrals, the user will be required to use a maximum of four keys to compose them: `limits`, `variables`, `mode` and `jacobian`. However, they can be criticised for the same reason as before; writing these four keys, which can take quite long arguments, runs counter to the main objective of this package. Thus, the user can designate as an optional argument of `\integral` a syntax called *special* that is unique to *intexgral*. The logic is as follows:

- You specify a *valid* argument for the `limits` key.
- You specify a *valid* argument for the `variables`
- You specify a *valid* argument for the `mode` key

And you separate everything with a colon. Let's look at an example to better understand.

```
\begin{equation}
  \integral[1, 2; 4, 5:y, x:nested]{x; y}
\end{equation}
```

$$\int_{1}^{2} x \int_{4}^{5} y \,\mathrm{d}y \,\mathrm{d}x \tag{24}$$

We mean by *valid argument* that all forms of values accepted by the four keys can also be likewise adopted by the special syntax. This hence includes keywords.

```
\begin{equation}
  \integral[radius;circle;scircle:spherical:product]{r;\theta;\phi}
\end{equation}
```

$$\int_{0}^{R} r \,\mathrm{d}r \int_{0}^{2\pi} \theta \,\mathrm{d}\theta \int_{0}^{\pi} \phi \,\mathrm{d}\phi \tag{25}$$

To include the Jacobian, simply write `+j` after the argument of `variables`. The mode can also be indicated using only an initial.

```
\begin{equation}
  \integral[radius;circle;scircle:spherical+j:p]{;;}
\end{equation}
```

$$\int_{0}^{R} r^2 \,\mathrm{d}r \int_{0}^{2\pi} \sin\theta \,\mathrm{d}\theta \int_{0}^{\pi} \mathrm{d}\phi \tag{26}$$

It is important to reiterate that in the classic configuration of the optional argument, it is of course not mandatory to specify the four keys mentioned. The same applies to this syntax. Since the key `mode` is not necessary for `default`, this part can be omitted.

```
\begin{equation}
  \integral[1, 2; 3, 4:x, y]{xy}
\end{equation}
```

$$\int_{1}^{2}\int_{3}^{4} xy \,\mathrm{d}x \,\mathrm{d}y \tag{27}$$

You can also take advantage of the automatic placement of "$\mathrm{d}x$" with this syntax, ignoring `variables`.

```
\begin{equation}
  \integral[1, 10]{x}
\end{equation}
```

$$\int_{1}^{10} x \,\mathrm{d}x \tag{28}$$

5.2 Functioning of the syntax

From the user's point of view, it is relatively simple to choose which syntax to adopt. However, the package must be able to distinguish between the two. Without going into too much detail[1] about the implementation, it is brought to your attention that the package attempts to extract the first key from the optional argument[2] and checks whether it exists. This is why, if you create a key with `\NewSymbolKeyword` whose name can probably be used as integration limits, *intexrgal* may incorrectly evaluate the nature of the optional argument.

6 Additional parameters

`\IntegralSetup` `\IntegralSetup {⟨parameter list⟩}`

new: v3.0.0

This macro allows you to modify parameters related to certain keys using the syntax ⟨key=value⟩. All assignments made are local, so the macro can be placed in a group if necessary. Here is an example of the macro with the package's default assignments:

```
\IntegralSetup
  {
    defaultvar  = {x},
    defaultvar* = {r},
    vectorstyle = \vec,
    domainstyle = \mathbb,
    symbolskip  = {-6mu},
    hide-diff   = false
  }
```

`defaultvar` `defaultvar= {⟨variables⟩}`

`defaultvar*`

This key modifies the variable automatically inserted when `variables` is not used. The argument may correspond to a keyword defined by `\NewVariableKeyword`. The starred variant of the key controls the integration parameter to be typeset with `diff-vec`.

[1] The more courageous among you are still invited to read the source code.

[2] Or at least, the group of tokens that it thinks corresponds to a key.

`vectorstyle` `vectorstyle=`⟨`control sequence`⟩

This key modifies the macro to be applied to variables when `diff-vec` is in action. It is therefore possible to alter the style of the vector: bold, underlined, or even another style from a particular package, *esvect* for example.

`domainstyle` `domainstyle=`⟨`control sequence`⟩

Similarly, you can change the macro to be applied for keys `domain(*)`.

`symbolskip` `symbolskip=`⟨`muexpr`⟩

When the `default` mode is in effect, the package inserts kerning between symbols to bring them slightly closer together. The default dimension (-6mu) can thus be adjusted according to the mathematical font in use.

`hide-diff` `hide-diff=`⟨`true`|`false`⟩

When a large part of the document requires differentials not to be included, it is possible to extend the more specific action of `variables=none` over time.

Changelog

3.0.1 (02-01-2026)

Fixed

- Bug with jacobian in special syntax (issue #3).
- French and English documentations (issue #4, issue #6 and issue #7).

Changed

- Limits keywords `positive` and `real` now contain a + sign (issue #5).

3.0.0 (24-12-2025)

Added

- Special syntax.
- Keys `domain*` and `mode`.
- Macros `\IntegralSetup` and `\NewSymbolKeyword`.

Removed

- Macros `\defaultdiff`, `\defaultdiff`, `\defaultvdiff` et `\vdiffstyle` in favour of `\IntegralSetup`.
- Keys controlling both symbol and limit, now managed at a higher level by `\NewSymbolKeyword`.
- `int-split` key, in favour of `mode`.
- `\NewInegralSymbol` macro.

Changed

- The names of some keys (`variable` in `variables`, `lower-lim` and `upper-lim` in `llimit` and `ulimit`, `int-symb` in `symbol`, `invert-differintials` and `hide-differentials` in `invert-diff` and `hide-diff`).
- `jacobian` and `diff-star` no longer require a boolean value. Simply writing them will now enable their features.
- `hide-diff` assigned to a local option rather than a package one.
- `limits-mode` assigned to a package option rather than a macro key

v2.0.1 (13-09-2025)

Fixed

- Compatibiliy issue between unicode-math and amssyb depending on the loading order (problem #2).

2.0.0 (09-09-2025)

Added

- Macros `\intexgralsetup`, `\defaultdiff`,`\defaultvdiff` and `\vdiffstyle`.

Changed

- Warning messages related to non-existing symbols. They are now only triggered when the integral is typeset.

Removed

- `diff-vec-style` key in favour of `\vdiffstyle`.
- Warning message about misuse of semi-colon in conjunction with the `int-split` key.

Fixed

- Bug where the integrand was not reset when `\integral` was used successively in the same TEX group (details here).
- *expl3* log leftovers that shouldn't have been there.

1.1.0 (29-07-2025)

Added

- Starred variatn for the keys controlling both symbol and limit (keys `single`, `contour` etc).

1.0.0 (26-07-2025) — Initial version.

Index

S

T

token commands:

V

Implementation

The implementation will be explained in more detail in a later version...

```
⟨*package⟩
⟨@@=intexgral⟩
\NeedsTeXFormat{LaTeX2e}
\RequirePackage{expl3}[2025-05-14]
\RequirePackage{pkginfograb}

% Package declaration

\pkginfograbProvidesExplPackage{intexgral}
  {
    name         = {intexgral}   ,
    author       = {Valentin Dao},
    date         = {2026-01-02}  ,
    creation     = {2025-07-26}  ,
    version      = {3.0.1}       ,
    description  = {A LaTeX package for typesetting integrals.}
  }

\msg_new:nnn { intexgral } { expl3-too-old }
  {
    Your~expl3~installation~is~too~old,~
    "intexgral"~requires~expl3~dated~2025-05-14~or~later.\iow_newline:
    Package~loading~has~been~aborted.\iow_newline:
    \msg_see_documentation_text:n { intexgral }
  }

% expl3 version verification

\@ifpackagelater{expl3}{2025-05-14}{}
  { \msg_critical:nn { intexgral } { expl3-too-old } }

% Verifies is \mathbb is defined (default domain style)

\hook_gput_code:nnn { begindocument/before } { . }
  { \cs_if_exist:NF \mathbb { \RequirePackage{amsfonts} } }

% Definition of data types for package options

\bool_new:N \l__intexgral_invert_limits_bool
\bool_new:N \l__intexgral_deactivate_differentials_bool
\bool_new:N \l__intexgral_invert_differentials_bool
\tl_new:N \l__intexgral_limits_style_tl

% Package options

\keys_define:nn { intexgral }
  {
    invert-limits .bool_set:N = \l__intexgral_invert_limits_bool,
    invert-limits .usage:n    = { preamble },
    invert-limits .initial:n  = { false },

    invert-diff .bool_set:N = \l__intexgral_invert_differentials_bool,
    invert-diff .usage:n    = { preamble },
    invert-diff .initial:n  = { false },
```

```
    limits-mode .choice:,
    limits-mode / limits .code:n =
      { \tl_set_eq:NN \l__intexgral_limits_style_tl \tex_limits:D },
    limits-mode / nolimits .code:n =
      { \tl_set_eq:NN \l__intexgral_limits_style_tl \tex_nolimits:D },

    limits-mode .usage:n   = { preamble },
    limits-mode .initial:n = { nolimits },

    italic .choice:,
      italic / true .code:n  =
        { \PassOptionsToPackage{italic=true}{derivative} },
      italic / false .code:n =
        { \PassOptionsToPackage{italic=false}{derivative} },

    upright .choice:,
      upright / true .code:n  =
        { \PassOptionsToPackage{upright=true}{derivative} },
      upright / false .code:n =
        { \PassOptionsToPackage{upright=false}{derivative} },
  }

% Loads the package options and "derivative" package

\ProcessKeyOptions[intexgral]

\RequirePackage{derivative}

% Warning/Error/Info messages

\cs_new:Nn \__intexgral_warning_msg_header: {
  (
    integral~no.~
    \int_use:N\g__intexgral_integral_number_int\c_space_tl
    \msg_line_context:
  )
  \iow_newline:
}

\msg_new:nnnn { intexgral } { symb-key-alr-def }
  { Symbol~key~"\tl_trim_spaces:n{#1}"~is~already~defined. }
  { Use~\token_to_str:N \RenewSymbolKeyword\ instead. }

\msg_new:nnnn { intexgral } { symb-key-not-def }
  { Symbol~key~"\tl_trim_spaces:n{#1}"~is~not~defined. }
  { Use~\token_to_str:N \NewSymbolKeyword\ instead. }

\msg_new:nnnn { intexgral } { diff-group-alr-def }
  { Differential~group~"\tl_trim_spaces:n{#1}"~is~already~defined. }
  { Use~\token_to_str:N \RenewVarKeyword\ instead. }

\msg_new:nnnn { intexgral } { diff-group-not-def }
  { Differential~group~"\tl_trim_spaces:n{#1}"~is~not~defined. }
  { Use~\token_to_str:N \NewVarKeyword\ instead. }

\msg_new:nnnn { intexgral } { lim-group-alr-def }
  { Limits~group~"\tl_trim_spaces:n{#1}"~is~already~defined. }
```

```
  { Use~\token_to_str:N \RenewLimitsKeyword\ instead. }

\msg_new:nnnn { intexgral } { lim-group-not-def }
  { Limits~group~"\tl_trim_spaces:n{#1}"~is~not~defined. }
  { Use~\NewLimitsKeyword\ instead. }

\msg_new:nnn { intexgral } { key-exists-for-lim }
  {
    Symbol~key~already~defined~with~
    \token_to_str:N \NewLimitsKeyword\ \__intexgral_warning_msg_header:
    Key~"#1"~already~exists~for~predefined~limits.~
    This~can~disrupt~the~functioning~of~the~special~syntax.
  }

\msg_new:nnn { intexgral } { unknown-symb }
  {
    Unknown~integral~symbol~\__intexgral_warning_msg_header:
    The~symbol~\tl_trim_spaces:n{#1}~has~been~replaced~with~
    \token_to_str:N\int.
  }

\msg_new:nnn { intexgral } { unequal-dim }
  {
    Inconsistent~structure~\__intexgral_warning_msg_header:
    The~number~of~integrals~(#1),~integrands~(#2),~and~
    differentials~(#3)~does~not~match.
  }

\msg_new:nnn { intexgral } { no-jacobian }
  {
    Jacobian~unavailable~\__intexgral_warning_msg_header:
    A~Jacobian~was~requested~for~the~"#1"~keyword,
    ~but~none~was~declared~for~the~latter.
  }

\msg_new:nnn { intexgral } { diff-not-sup }
  {
    Macro~\token_to_str:N\differentials\
    not~supported~\__intexgral_warning_msg_header:
    You~can't~use~\token_to_str:N\differentials\
    with~"nested"~mode~alongside~inverted~differentials.~
    I~will~simply~ignore~them.
  }

\msg_new:nnn { intexgral } { diff-vec-not-supp }
  {
    Key~"diff-vec"~not~supported~\__intexgral_warning_msg_header:
    You~can't~use~"diff-vec"~with~"#1"~mode.~I~will~simply~ignore~this~key.
  }

\msg_new:nnn { intexgral } { use-glyph-instead }
  {
    Consider~using~the~dedicated~glyph~\__intexgral_warning_msg_header:
    The~key~"nint"~was~used~with~fewer~than~
    5~regular~integral~signs.
  }

% Argument specifiers variants
```

```
\cs_generate_variant:Nn \regex_if_match:nnTF    { nVTF }
\cs_generate_variant:Nn \__intexgral_new_limits_group:nn { nV   }
\cs_generate_variant:Nn \str_if_eq:nnTF         { neTF }
\cs_generate_variant:Nn \str_if_eq:nnF          { neF  }
\cs_generate_variant:Nn \keys_if_exist:nnTF     { nVTF }
\cs_generate_variant:Nn \seq_use:Nn             { Ne   }
\cs_generate_variant:Nn \str_if_eq_p:nn         { en   }

% DATA TYPES

\tl_new:N \l__intexgral_key_name_tl
\tl_new:N \l__intexgral_integrand_tl
\tl_new:N \l__intexgral_domain_style_tl
\tl_new:N \l__intexgral_lower_limit_tl
\tl_new:N \l__intexgral_upper_limit_tl
\tl_new:N \l__intexgral_integral_symbol_tl
\tl_new:N \l__intexgral_differential_options_i_tl
\tl_new:N \l__intexgral_differential_options_ii_tl
\tl_new:N \l__intexgral_vectorial_differential_style_tl
\tl_new:N \l__intexgral_independent_differential_options_tl
\tl_new:N \l__intexgral_domain_char_tl
\tl_new:N \l__intexgral_domain_dimension_tl
\tl_new:N \l__intexgral_default_differential_tl
\tl_new:N \l__intexgral_default_vector_differential_tl
\tl_const:Nn \c__intexgral_left_bracket_tl { [ }
\tl_const:Nn \c__intexgral_right_bracket_tl { ] }

\tl_set_eq:NN \l__intexgral_integral_symbol_tl \int

\bool_new:N \l__intexgral_manual_differentials_bool
\bool_new:N \l__intexgral_custom_variables_bool
\bool_new:N \l__intexgral_separate_integral_bool
\bool_new:N \l__intexgral_vectorial_differential_bool
\bool_new:N \l__intexgral_jacobian_bool
\bool_new:N \l__intexgral_starred_differential_bool
\bool_new:N \l__intexgral_has_order_bool
\bool_new:N \l__intexgral_has_var_bool

\bool_set_false:N \l__intexgral_vectorial_differential_bool
\bool_set_false:N \l__intexgral_manual_differentials_bool
\bool_set_false:N \l__intexgral_custom_variables_bool
\bool_set_false:N \l__intexgral_separate_integral_bool
\bool_set_false:N \l__intexgral_has_order_bool
\bool_set_false:N \l__intexgral_has_var_bool

\clist_new:N \l__intexgral_variable_i_clist
\clist_new:N \l__intexgral_variable_ii_clist

\seq_new:N \l__intexgral_integrand_seq
\seq_new:N \l__intexgral_integral_symbol_seq
\seq_new:N \l__intexgral_jacobian_seq
\seq_new:N \l__intexgral_differential_seq
\seq_new:N \l__intexgral_limits_seq
\seq_new:N \l__intexgral_upper_limits_seq
\seq_new:N \l__intexgral_lower_limits_seq
\seq_new:N \l__intexgral_differential_var_i_seq
\seq_new:N \l__intexgral_differential_var_ii_seq
```

```
\seq_new:N \l__intexgral_differential_order_i_seq
\seq_new:N \l__intexgral_differential_order_ii_seq
\seq_new:N \l__intexgral_domain_seq

\str_new:N \l__intexgral_display_mode_str

\muskip_new:N \l__intexgral_inline_symbol_muskip

\prop_new:N \g__intexgral_limits_keyword_prop
\prop_new:N \g__intexgral_differential_group_keyword_prop

\int_new:N \l__intexgral_integral_sequence_int
\int_gzero_new:N \g__intexgral_integral_number_int

% TeX macros adapted to expl3 syntax

\cs_new_protected:Npn \__intexgral_mkern:N #1
  { \tex_mkern:D #1 \scan_stop: }

\cs_new_protected:Npn \__intexgral_mathchoice:nnnn #1#2#3#4
  { \mathchoice{#1}{#2}{#3}{#4} }

% SYMBOL

% General form of the symbol (glyph, limits mode, lower limit, upper limit)

\cs_new_protected:Npn \__intexgral_parse_integral_symbol:
  {
    \tl_use:N \l__intexgral_integral_symbol_tl
    \tl_use:N \l__intexgral_limits_style_tl
    \c_math_subscript_token
      { \l__intexgral_lower_limit_tl }
    \c_math_superscript_token
      { \l__intexgral_upper_limit_tl }
  }

% Generates the symbols after <limits> is executed

\cs_new_protected:Nn \__intexgral_generate_integral_sequence:
  {
    \int_set:Nn \l__intexgral_integral_sequence_int
      {
        \seq_if_empty:NTF \l__intexgral_lower_limits_seq
          { \seq_count:N \l__intexgral_upper_limits_seq }
          { \seq_count:N \l__intexgral_lower_limits_seq }
      }
    \exp_args:NV
    \int_step_inline:nn \l__intexgral_integral_sequence_int
      {
        \seq_put_right:Nn \l__intexgral_integral_symbol_seq
          {
            \__intexgral_set_limits:nn
              { \seq_item:Nn \l__intexgral_lower_limits_seq { ##1 } }
              { \seq_item:Nn \l__intexgral_upper_limits_seq { ##1 } }
            \__intexgral_parse_integral_symbol:
          }
      }
  }
```

```
% LIMITS

% Inverts the limit order convention

\cs_new_protected:Npn \__intexgral_set_limits:nn #1#2
  {
    \bool_if:NTF \l__intexgral_invert_limits_bool
      {
        \tl_set:Nn \l__intexgral_lower_limit_tl { #2 }
        \tl_set:Nn \l__intexgral_upper_limit_tl { #1 }
      }
      {
        \tl_set:Nn \l__intexgral_lower_limit_tl { #1 }
        \tl_set:Nn \l__intexgral_upper_limit_tl { #2 }
      }
  }

% Creates a keyword containing predefined limits

\cs_new_protected:Npn \__intexgral_new_limits_group:nn #1#2
  { \prop_put:Nnn \g__intexgral_limits_keyword_prop { #1 } { #2 } }

% Verifies if the argument corresponds to a valid keyword

\cs_new:Npn \__intexgral_parse_integral_limit:n #1
  {
    \prop_if_in:NnTF \g__intexgral_limits_keyword_prop { #1 }
      { \prop_item:Nn \g__intexgral_limits_keyword_prop { #1 } }
      { #1 }
  }

% Splits the argument of the key to prepare for parsing

\cs_new_protected:Npn \__intexgral_parse_limits:n #1
  {
    \tl_set:Nn \l_tmpa_tl { #1 }
    \seq_clear:N \l__intexgral_lower_limits_seq
    \seq_clear:N \l__intexgral_upper_limits_seq
    \seq_set_split:NnV \l__intexgral_limits_seq { ; } \l_tmpa_tl
  }

% Limits under regular form

\cs_new_protected:Nn \__intexgral_set_limits_regular:
  {
    \seq_map_inline:Nn \l__intexgral_limits_seq
      {
        \clist_set:Ne \l_tmpa_clist
          { \__intexgral_parse_integral_limit:n { ##1 } }
        \seq_put_right:Ne \l__intexgral_lower_limits_seq
          { \clist_item:Nn \l_tmpa_clist { 1 } }
        \seq_put_right:Ne \l__intexgral_upper_limits_seq
          { \clist_item:Nn \l_tmpa_clist { 2 } }
      }
  }

% Limits under interval form
```

```
\cs_new_protected:Npn \__intexgral_set_limits_starred:
  {
    \seq_map_indexed_inline:Nn \l__intexgral_limits_seq
      {
        \clist_set:Ne \l_tmpa_clist
          { \__intexgral_parse_integral_limit:n { ##2 } }
        \bool_if:NTF \l__intexgral_invert_limits_bool
          {
            \tl_set:Ne \l__intexgral_lower_limit_tl
              {
                \clist_item:Nn \l_tmpa_clist { 2 }
                \tex_mathpunct:D,
                \clist_item:Nn \l_tmpa_clist { 1 }
              }
          }
          {
            \tl_set:Ne \l__intexgral_lower_limit_tl
              {
                \clist_item:Nn \l_tmpa_clist { 1 }
                \tex_mathpunct:D,
                \clist_item:Nn \l_tmpa_clist { 2 }
              }
          }
            \bool_if:NTF \l__intexgral_invert_limits_bool
              { \seq_put_right:Ne \l__intexgral_upper_limits_seq }
              { \seq_put_right:Ne \l__intexgral_lower_limits_seq }
          {
            \str_case_e:nnF { \clist_item:Nn \l_tmpa_clist { 1 } }
              {
                { -\infty } { \tex_left:D \c__intexgral_right_bracket_tl }
              }
              { \tex_left:D \c__intexgral_left_bracket_tl }
            \tl_use:N \l__intexgral_lower_limit_tl
            \str_case_e:nnF { \clist_item:Nn \l_tmpa_clist { 2 } }
              {
                { +\infty } { \tex_right:D \c__intexgral_left_bracket_tl }
              }
              { \tex_right:D \c__intexgral_right_bracket_tl }
            }
      }
  }

% VARIABLES

% Creates a keyword containing predefined variables

\cs_new_protected:Npn \__intexgral_new_differential_group:nnn #1#2#3
  {
    \prop_put:Nnn \g__intexgral_differential_group_keyword_prop
      { #1 } { #2 }
    \tl_if_blank:nF { #3 }
      {
        \clist_set:Nn \l_tmpa_clist { #3 }
        \seq_set_from_clist:cN
          { g__intexgral_#1_jacobian_seq }
          \l_tmpa_clist
      }
```

```
  }

% expl3 equivalent of \odif(*)

\cs_new:Npn \__intexgral_differentials:nn #1#2
  {
    \bool_if:NTF \l__intexgral_starred_differential_bool
      { \odif*[#1]{#2} }
      { \odif[#1]{#2} }
  }

% Extracts the whole "order" <keyval>

\cs_new_protected:Npn \__intexgral_extract_differential_order:n #1
  {
    \regex_extract_once:nnNT
      { order\s*=\s*\{?\s*(\d+(?:\s*,\s*\d+)*)\s*\}? }
      { #1 } \l__intexgral_differential_order_i_seq
      { \bool_set_true:N \l__intexgral_has_order_bool }
    \clist_set:Ne \l_tmpa_clist
      { \seq_item:Nn \l__intexgral_differential_order_i_seq { 2 } }
    \seq_set_from_clist:NN
      \l__intexgral_differential_order_ii_seq
      \l_tmpa_clist
  }

% Extracts the whole "var" <keyval>

\cs_new_protected:Npn \__intexgral_extract_differential_var:n #1
  {
    \regex_extract_once:nnNTF
      { var\s*=\s*(none|all|\{\s*(\d+(?:\s*,\s*\d+)*\s*)\}) }
      { #1 }
      \l__intexgral_differential_var_i_seq
      { \bool_set_true:N \l__intexgral_has_var_bool }
      {
        \str_if_in:nnT { #1 } { var }
          {
            \seq_put_left:Nn \l__intexgral_differential_var_i_seq { var }
            \bool_set_true:N \l__intexgral_has_var_bool
          }
      }
    \clist_set:Ne \l_tmpa_clist
      { \seq_item:Nn \l__intexgral_differential_var_i_seq { 3 } }
    \seq_set_from_clist:NN
      \l__intexgral_differential_var_ii_seq
      \l_tmpa_clist
  }

% Extracts all keys other than "order" and "var"

\cs_new_protected:Npn \__intexgral_remove_differential_order_and_var:n #1
  {
    \tl_set:Nn \l__intexgral_independent_differential_options_tl { #1 }
    \regex_replace_all:nnN
      {
        order\s*=\s*\{?\s*\d+(\s*,\s*\d+)*\s*\}?\s*,?
        |
```

```
        [^-]var(?:\s*=\s*(?:none|all|\{\s*\d+(?:\s*,\s*\d+)*\s*\}))?,?
      }
      {  }
      \l__intexgral_independent_differential_options_tl
  }

% Reassembles the variables

\cs_new_protected:Npn \__intexgral_parse_variable:nn #1#2
  {
    \clist_set:Nn \l_tmpa_clist { #2 }
    \seq_set_from_clist:NN \l_tmpa_seq \l_tmpa_clist
    \tl_if_empty:nF { #1 }
      {
        \__intexgral_extract_differential_order:n { #1 }
        \__intexgral_extract_differential_var:n { #1 }
        \__intexgral_remove_differential_order_and_var:n { #1 }
      }
    \seq_map_indexed_inline:Nn \l_tmpa_seq {
      \tl_if_empty:nF { #1 }
        {
          \tl_put_right:NV
            \l__intexgral_differential_options_ii_tl
            \l__intexgral_independent_differential_options_tl
          \bool_if:NT \l__intexgral_has_order_bool
            {
              \tl_put_right:Ne \l__intexgral_differential_options_ii_tl
                {
                  ,order=
                  \seq_item:Nn \l__intexgral_differential_order_ii_seq { ##1 }
                }
            }
          \bool_if:NT \l__intexgral_has_var_bool
            {
              \int_compare:nNnTF
              { \seq_count:N \l__intexgral_differential_var_i_seq }
              =
              { 1 }
              {
                \tl_put_right:Nn
                  \l__intexgral_differential_options_ii_tl
                  { var }
              }
              {
                \str_if_eq:neF { none }
                  { \seq_item:Nn \l__intexgral_differential_var_i_seq { 2 } }
                    {
                      \str_if_eq:neTF { all }
                        { \seq_item:Nn
                            \l__intexgral_differential_var_i_seq
                            { 2 }
                        }
                        {
                          \tl_put_right:Nn
                            \l__intexgral_differential_options_ii_tl
                            { var }
                        }
                        {
```

```
                          \seq_if_in:NnT
                            \l__intexgral_differential_var_ii_seq
                            { ##1 }
                            {
                              \seq_pop_left:NN
                                \l__intexgral_differential_var_ii_seq
                                \l_tmpa_tl
                              \tl_put_right:Nn
                                \l__intexgral_differential_options_ii_tl
                                { ,var }
                            }
                        }
                    }
                }
            }
        }
      \seq_put_right:Ne \l__intexgral_differential_seq
        {
          \__intexgral_differentials:nn
            { \exp_not:V \l__intexgral_differential_options_ii_tl }
            { \seq_item:Nn \l_tmpa_seq { ##1 } }
        }
      \tl_clear:N \l__intexgral_differential_options_ii_tl
    }
  }

% Executes all previous macros to push tokens into the variable sequence

\cs_new_protected:Npn \__intexgral_parse_differentials:
  {
    \bool_if:NTF \l__intexgral_vectorial_differential_bool
      {
        \clist_map_inline:Nn \l__intexgral_variable_i_clist
          {
            \clist_put_right:Nn \l__intexgral_variable_ii_clist
              { \l__intexgral_vectorial_differential_style_tl{##1} }
          }
        \bool_if:NTF \l__intexgral_custom_variables_bool
          {
            \exp_args:NVV \__intexgral_parse_variable:nn
              \l__intexgral_differential_options_i_tl
              \l__intexgral_variable_ii_clist
          }
          {
            \exp_args:NV \__intexgral_parse_variable:nn
              \l__intexgral_differential_options_i_tl
              {
                \l__intexgral_vectorial_differential_style_tl
                  { \l__intexgral_default_vector_differential_tl }
              }
          }
      }
      {
        \bool_if:NTF \l__intexgral_custom_variables_bool
          {
            \exp_args:NVV \__intexgral_parse_variable:nn
              \l__intexgral_differential_options_i_tl
              \l__intexgral_variable_i_clist
```

```
          }
          {
            \exp_args:NV \__intexgral_parse_variable:nn
              \l__intexgral_differential_options_i_tl
              { \l__intexgral_default_differential_tl }
          }
      }
  }

% SPECIAL SYNTAX

% Cuts off the value of a key

\cs_new:Npn \__intexgral_retrieve_key_name:w #1=#2\q_stop { #1 }

% Extracts first key name of optional argument

\cs_new_protected:Npn \__intexgral_extract_first_key_name:n #1
  {
    \clist_set:Nn \l_tmpa_clist { #1 }
    \tl_set:Ne \l_tmpa_tl { \clist_item:Nn \l_tmpa_clist { 1 } }
    \tl_if_in:NnTF \l_tmpa_tl { = }
      {
        \tl_set:Ne \l__intexgral_key_name_tl
          {
            \exp_last_unbraced:NV
            \__intexgral_retrieve_key_name:w \l_tmpa_tl \q_stop
          }
      }
      { \tl_set_eq:NN \l__intexgral_key_name_tl \l_tmpa_tl }
  }

% Sets the keys as usual, or sets them up manually if special syntax is in use

\cs_new_protected:Npn \__intexgral_parse_macro_keys:n #1
  {
    \keys_if_exist:nVTF { integral } \l__intexgral_key_name_tl
      { \keys_set:nn { integral } { #1 } }
      {
        \regex_split:nnN { : } { #1 } \l_tmpa_seq
        \str_if_eq:eeTF
          { \seq_item:Nn \l_tmpa_seq { 2 } }
          { +j }
          {
            \exp_args:NNne \seq_set_item:Nnn \l_tmpa_seq { 2 }
              { \l__intexgral_default_differential_tl+j }
          }
          {
            \int_compare:nNnT { \seq_count:N \l_tmpa_seq } < { 2 }
              {
                \seq_put_right:NV
                  \l_tmpa_seq
                  \l__intexgral_default_differential_tl
              }
          }
        \int_compare:nNnT { \seq_count:N \l_tmpa_seq } < { 3 }
          { \seq_put_right:Nn \l_tmpa_seq { d } }
        \seq_set_split:Nne \l_tmpb_seq
```

```
          { + }
          { \seq_item:Nn \l_tmpa_seq { 2 } }
        \keys_set:ne { integral }
          {
            limits    = { \seq_item:Nn \l_tmpa_seq { 1 } },
            variables = { \seq_item:Nn \l_tmpb_seq { 1 } },
            mode      =
              {
                \str_case:enF { \seq_item:Nn \l_tmpa_seq { 3 } }
                  {
                    { d       } { default }
                    { n       } { nested  }
                    { p       } { product }
                    { default } { default }
                    { nested  } { nested  }
                    { product } { product }
                  }
                { default }
              },
            \bool_if:nT
              {
                \int_compare_p:nNn { \seq_count:N \l_tmpb_seq } > { 1 }
                &&
                \str_if_eq_p:en { \seq_item:Nn \l_tmpb_seq { 2 } } { j }
              }
              { jacobian }
          }
      }
  }

% PRELIMINARY CONFIGURATIONS

% Verifies that <limits> and <variables> (and <integrand>) have the same size

\cs_new_protected:Nn \__intexgral_check_sequence_size:
  {
    \int_compare:nNnTF
      { \seq_count:N \l__intexgral_integral_symbol_seq }
      =
      { \seq_count:N \l__intexgral_integrand_seq }
        {
          \int_compare:nNnF
            { \seq_count:N \l__intexgral_integrand_seq }
            =
            { \seq_count:N \l__intexgral_differential_seq }
              {
                \msg_warning:nneee { intexgral } { unequal-dim }
                  { \seq_count:N \l__intexgral_integral_symbol_seq }
                  { \seq_count:N \l__intexgral_integrand_seq       }
                  { \seq_count:N \l__intexgral_differential_seq    }
              }
        }
        {
          \msg_warning:nneee { intexgral } { unequal-dim }
            { \seq_count:N \l__intexgral_integral_symbol_seq }
            { \seq_count:N \l__intexgral_integrand_seq       }
            { \seq_count:N \l__intexgral_differential_seq    }
        }
```

```
  }

% Performs preparatory actions before executing the main macro

\cs_new_protected:Nn \__intexgral_integral_preconfiguration:
  {
    \int_gincr:N \g__intexgral_integral_number_int

    \bool_if:NT \l__intexgral_separate_integral_bool
      {
        \seq_set_split:NnV
          \l__intexgral_integrand_seq
          { ; }
          \l__intexgral_integrand_tl
      }

    \seq_if_empty:NT \l__intexgral_integral_symbol_seq
      {
        \seq_put_right:Nn \l__intexgral_integral_symbol_seq
          { \__intexgral_parse_integral_symbol: }
      }

    \seq_if_empty:NT \l__intexgral_integrand_seq
      {
        \seq_put_right:NV
          \l__intexgral_integrand_seq
          \l__intexgral_integrand_tl
      }

    \bool_if:NF \l__intexgral_deactivate_differentials_bool
      { \__intexgral_parse_differentials: }

    \regex_if_match:nVTF { \c{differentials} } \l__intexgral_integrand_tl
      { \bool_set_true:N \l__intexgral_manual_differentials_bool }
      { \bool_set_false:N \l__intexgral_manual_differentials_bool }

    \bool_if:NT \l__intexgral_separate_integral_bool
      { \__intexgral_check_sequence_size: }
  }

% Default mode

\cs_new_protected:Nn \__intexgral_print_default_integral:
  {
    \bool_if:NT \l__intexgral_manual_differentials_bool
      {
        \cs_set_protected:Npn \differentials
          { \seq_use:Nn \l__intexgral_differential_seq {  } }
      }
    \seq_use:Nn \l__intexgral_integral_symbol_seq
      { \__intexgral_mkern:N \l__intexgral_inline_symbol_muskip }
    \bool_if:nT
      {
        \l__intexgral_invert_differentials_bool
        &&
        !\l__intexgral_manual_differentials_bool
      }
      {
```

```
        \seq_use:Nn \l__intexgral_differential_seq {  } \tex_mathop:D{} \!
        \bool_if:NT \l__intexgral_vectorial_differential_bool { \cdot \: }
      }
    \seq_use:Nn \l__intexgral_integrand_seq {  }
    \bool_if:NT \l__intexgral_jacobian_bool
      { \seq_use:Nn \l__intexgral_jacobian_seq {  } }
    \bool_if:nT
      {
        !\l__intexgral_invert_differentials_bool
        &&
        !\l__intexgral_manual_differentials_bool
      }
      {
        \bool_if:NT \l__intexgral_vectorial_differential_bool { \cdot }
        \seq_use:Nn \l__intexgral_differential_seq {  }
      }
  }

% Nested mode

\cs_new_protected:Nn \__intexgral_print_nested_integral:
  {
    \bool_if:NT \l__intexgral_manual_differentials_bool
      {
        \cs_set_protected:Npn \differentials
          { \seq_use:Nn \l__intexgral_differential_seq {  } }
      }
    \bool_if:NT \l__intexgral_vectorial_differential_bool
      { \msg_warning:nnn { intexgral } { diff-vec-not-supp } { nested } }
    \int_step_inline:nn { \seq_count:N \l__intexgral_integral_symbol_seq }
      {
        \seq_item:Nn \l__intexgral_integral_symbol_seq { ##1 }
        \bool_if:NTF \l__intexgral_invert_differentials_bool
          {
            \bool_if:NT \l__intexgral_manual_differentials_bool
              {
                \msg_warning:nn { intexgral } { diff-not-sup }
                \cs_set_eq:NN \differentials \scan_stop:
              }
            \seq_item:Nn \l__intexgral_differential_seq { ##1 }
            \tex_mathop:D{} \!
          }
          {
            \seq_item:Nn \l__intexgral_integrand_seq { ##1 }
            \bool_if:NT \l__intexgral_jacobian_bool
              { \seq_item:Nn \l__intexgral_jacobian_seq { ##1 } }
          }
      }
    \bool_if:NTF \l__intexgral_invert_differentials_bool
      {
        \seq_use:Nn \l__intexgral_integrand_seq {}
        \bool_if:NT \l__intexgral_jacobian_bool
          { \seq_use:Nn \l__intexgral_jacobian_seq {  } }
      }
      {
        \bool_if:NF \l__intexgral_manual_differentials_bool
          {
            \seq_use:Nn \l__intexgral_differential_seq {}
```

```
          }
      }
  }

% Product mode

\cs_new_protected:Nn \__intexgral_print_product_integral:
  {
    \bool_if:NT \l__intexgral_manual_differentials_bool
      {
        \cs_set_protected:Npn \differentials
          {
            \seq_gpop_left:NN \l_tmpa_tl
            \tl_use:N \l_tmpa_tl
          }
      }
    \bool_if:NT \l__intexgral_vectorial_differential_bool
      { \msg_warning:nnn { intexgral } { diff-vec-not-supp } { product } }
    \int_step_inline:nn { \seq_count:N \l__intexgral_integral_symbol_seq }
      {
        \seq_item:Nn \l__intexgral_integral_symbol_seq { ##1 }
        \bool_if:nT
          {
            \l__intexgral_invert_differentials_bool
            &&
            !\l__intexgral_manual_differentials_bool
          }
          {
            \seq_item:Nn
              \l__intexgral_differential_seq { ##1 }
            \tex_mathop:D{} \!
          }
      \seq_item:Nn \l__intexgral_integrand_seq { ##1 }
      \bool_if:NT \l__intexgral_jacobian_bool
        { \seq_item:Nn \l__intexgral_jacobian_seq { ##1 } }
      \bool_if:nT
        {
          !\l__intexgral_invert_differentials_bool
          &&
          !\l__intexgral_manual_differentials_bool
        }
        { \seq_item:Nn \l__intexgral_differential_seq { ##1 } }
      }
  }

% General macro for typesetting

\cs_new_protected:Nn \__intexgral_print_integral:
  {
    \__intexgral_integral_preconfiguration:
    \str_case:VnF \l__intexgral_display_mode_str
      {
        { default } { \__intexgral_print_default_integral: }
        { nested  } { \__intexgral_print_nested_integral:  }
        { product } { \__intexgral_print_product_integral: }
      }
      { \__intexgral_print_default_integral: }
    \seq_gclear:N \l__intexgral_differential_seq
```

```
  }

% KEYS

\keys_define:nn { integral } {

  % LIMITS

  mode .choice:,
    mode / default .code:n =
      {
        \bool_set_false:N \l__intexgral_separate_integral_bool
        \str_set:Nn \l__intexgral_display_mode_str { default }
      },
    mode / nested .code:n =
      {
        \bool_set_true:N \l__intexgral_separate_integral_bool
        \str_set:Nn \l__intexgral_display_mode_str { nested }
      },
    mode / product .code:n =
      {
        \bool_set_true:N \l__intexgral_separate_integral_bool
        \str_set:Nn \l__intexgral_display_mode_str { product }
      },
    mode .default:n = { default },

  limits .code:n =
    {
      \__intexgral_parse_limits:n { #1 }
      \__intexgral_set_limits_regular:
      \__intexgral_generate_integral_sequence:
    },

  limits* .code:n =
    {
      \__intexgral_parse_limits:n { #1 }
      \__intexgral_set_limits_starred:
      \__intexgral_generate_integral_sequence:
    },

  llimit .tl_set:N = \l__intexgral_lower_limit_tl,

  ulimit .tl_set:N = \l__intexgral_upper_limit_tl,

  symbol .code:n =
    {
      \cs_if_exist:NTF #1
        { \tl_set_eq:NN \l__intexgral_integral_symbol_tl #1 }
        {
          \msg_warning:nnn { intexgral } { unknown-symb } { #1 }
          \tl_set_eq:NN \l__intexgral_integral_symbol_tl \int
        }
    },

  % SYMBOL SHORTCUT

  nint .code:n =
    {
```

```
      \int_compare:nNnT { #1 } < { 5 }
        { \msg_warning:nn { intexgral } { use-glyph-instead } }

      \tl_clear:N \l__intexgral_integral_symbol_tl

      \int_step_inline:nn { #1 }
        {
          \tl_put_right:Nn \l__intexgral_integral_symbol_tl
            { \int }
          \int_compare:nNnT { ##1 } < { #1 }
            {
              \tl_put_right:Nn \l__intexgral_integral_symbol_tl
                {
                  \__intexgral_mathchoice:nnnn
                    { \tex_mkern:D -12mu \scan_stop: }
                    { \tex_mkern:D -8mu \scan_stop: }
                    { \tex_mkern:D -4mu \scan_stop: }
                    { \tex_mkern:D -2mu \scan_stop: }
                }
            }
        }
    },

  % LIMITS SHORTCUTS

  domain .code:n =
    {
      \tl_set:Nn \l_tmpa_tl { #1 }
      \seq_set_split:NnV \l__intexgral_domain_seq { * } \l_tmpa_tl
      \seq_map_inline:Nn \l__intexgral_domain_seq
        {
          \tl_if_empty:NF \l__intexgral_lower_limit_tl
            { \tl_put_right:Nn \l__intexgral_lower_limit_tl { \times } }
          \tl_set:Ne \l__intexgral_domain_char_tl
            { \exp_args:Ne \str_uppercase:n { \tl_head:n { ##1 } } }
          \tl_set:Ne \l__intexgral_domain_dimension_tl
            { \tl_tail:n { ##1 } }
          \tl_put_right:Ne \l__intexgral_lower_limit_tl
            {
              \exp_not:N \l__intexgral_domain_style_tl
                { \l__intexgral_domain_char_tl }
              \c_math_superscript_token { \l__intexgral_domain_dimension_tl }
            }
        }
    },

  domain* .code:n =
    {
      \tl_set:Nn \l_tmpa_tl { #1 }
      \seq_set_split:NnV \l__intexgral_domain_seq { * } \l_tmpa_tl
      \seq_map_inline:Nn \l__intexgral_domain_seq
        {
          \tl_if_empty:NF \l__intexgral_lower_limit_tl
            { \tl_put_right:Nn \l__intexgral_lower_limit_tl { \times } }
          \tl_set:Ne \l__intexgral_domain_char_tl
            { \exp_args:Ne \str_uppercase:n { \tl_head:n { ##1 } } }
          \tl_set:Ne \l__intexgral_domain_dimension_tl
            { \tl_tail:n { ##1 } }
```

```
          \tl_put_right:Ne \l__intexgral_lower_limit_tl
            {
              \exp_not:N \l__intexgral_domain_style_tl
                { \l__intexgral_domain_char_tl }
              \c_math_subscript_token { \l__intexgral_domain_dimension_tl }
            }
        }
    },

  boundary .code:n =
    { \tl_set:Nn \l__intexgral_lower_limit_tl { \partial #1 } },

  % VARIABLES

  variables .code:n =
    {
      \bool_set_true:N \l__intexgral_custom_variables_bool
      \str_if_eq:nnTF { #1 } { none }
        { \bool_set_true:N \l__intexgral_deactivate_differentials_bool }
        {
          \prop_get:NnNTF
            \g__intexgral_differential_group_keyword_prop { #1 } \l_tmpa_tl
              {
                \clist_set:NV \l__intexgral_variable_i_clist \l_tmpa_tl
                \seq_if_exist:cTF { g__intexgral_#1_jacobian_seq }
                  {
                    \seq_set_eq:Nc
                      \l__intexgral_jacobian_seq
                      { g__intexgral_#1_jacobian_seq }
                  }
                  { \msg_warning:nnn { intexgral } { no-jacobian } { #1 } }
              }
              { \clist_set:Nn \l__intexgral_variable_i_clist { #1 } }
        }
    },

  jacobian .code:n =
    { \bool_set_true:N \l__intexgral_jacobian_bool },
  diff-vec .code:n =
    { \bool_set_true:N \l__intexgral_vectorial_differential_bool },
  diff-star .code:n =
    { \bool_set_true:N \l__intexgral_starred_differential_bool },

  diff-symb .code:n =
    {
      \cs_set:Npn \__intexgral_differentials:nn ##1##2 {
        \bool_if:NTF \l__intexgral_starred_differential_bool
          { #1*[##1]{##2} }
          { #1[##1]{##2} }
      }
    },

  diff-options .tl_set:N = \l__intexgral_differential_options_i_tl,
}

\keys_define:nn { IntegralSetup }
  {
    defaultvar  .tl_set:N   = \l__intexgral_default_differential_tl,
```

```
    defaultvar* .tl_set:N   = \l__intexgral_default_vector_differential_tl,
    vectorstyle .tl_set:N   = \l__intexgral_vectorial_differential_style_tl,
    domainstyle .tl_set:N   = \l__intexgral_domain_style_tl,
    symbolskip  .tl_set:N   = \l__intexgral_inline_symbol_muskip,
    hide-diff   .bool_set:N = \l__intexgral_deactivate_differentials_bool,
  }

% END USER MACROS

% Limits keywords

\NewDocumentCommand\NewLimitsKeyword{ m m }
  {
    \prop_get:NnNTF \g__intexgral_limits_keyword_prop { #1 } \l_tmpa_tl
      { \msg_error:nnn { intexgral } { lim-group-alr-def } { #1 } }
      {
        \bool_if:NTF \l__intexgral_invert_limits_bool
          {
            \clist_set:Nn \l_tmpa_clist { #2 }
            \clist_reverse:N \l_tmpa_clist
            \__intexgral_new_limits_group:nV { #1 } \l_tmpa_clist
          }
          { \__intexgral_new_limits_group:nn { #1 } { #2 } }
      }
  }

\NewDocumentCommand\RenewLimitsKeyword{ m m }
  {
    \prop_pop:NnNTF \g__intexgral_limits_keyword_prop { #1 } \l_tmpa_tl
      {
        \bool_if:NTF \l__intexgral_invert_limits_bool
          {
            \clist_set:Nn \l_tmpa_clist { #2 }
            \clist_reverse:N \l_tmpa_clist
            \__intexgral_new_limits_group:nV { #1 } \l_tmpa_clist
          }
          { \__intexgral_new_limits_group:nn { #1 } { #2 } }
      }
      { \msg_error:nnn { intexgral } { lim-group-not-def } }
  }

\NewDocumentCommand\ProvideLimitsKeyword{ m m }
  {
    \prop_get:NnNF \g__intexgral_limits_keyword_prop { #1 } \l_tmpa_tl
      {
        \bool_if:NTF \l__intexgral_invert_limits_bool
          {
            \clist_set:Nn \l_tmpa_clist { #2 }
            \clist_reverse:N \l_tmpa_clist
            \__intexgral_new_limits_group:nV { #1 } \l_tmpa_clist
          }
          { \__intexgral_new_limits_group:nn { #1 } { #2 } }
      }
  }

\NewDocumentCommand\DeclareLimitsKeyword{ m m }
  {
    \prop_pop:NnNT \g__intexgral_limits_keyword_prop { #1 } \l_tmpa_tl
```

```
      {
        \bool_if:NTF \l__intexgral_invert_limits_bool
          {
            \clist_set:Nn \l_tmpa_clist { #2 }
            \clist_reverse:N \l_tmpa_clist
            \__intexgral_new_limits_group:nV { #1 } \l_tmpa_clist
          }
          { \__intexgral_new_limits_group:nn { #1 } { #2 } }
      }
  }

% Variable keywords

\NewDocumentCommand\NewVariableKeyword{ m m o }
  {
    \prop_get:NnNTF \g__intexgral_differential_group_keyword_prop { #1 }
      \l_tmpa_tl
      { \msg_error:nnn { intexgral } { diff-group-alr-def } { #1 } }
      { \__intexgral_new_differential_group:nnn { #1 } { #2 } { #3 } }
  }

\NewDocumentCommand\RenewVariableKeyword{ m m o }
  {
    \prop_pop:NnNTF \g__intexgral_differential_group_keyword_prop { #1 }
      \l_tmpa_tl
      {
        \seq_clear:c { g__intexgral_#1_jacobian_seq }
        \__intexgral_new_differential_group:nnn { #1 } { #2 } { #3 }
      }
      { \msg_error:nnn { intexgral } { diff-group-not-def } { #1 } }
  }

\NewDocumentCommand\ProvideVariableKeyword{ m m o }
  {
    \prop_get:NnNF \g__intexgral_differential_group_keyword_prop { #1 }
      \l_tmpa_tl
      { \__intexgral_new_differential_group:nnn { #1 } { #2 } { #3 } }
  }

\NewDocumentCommand\DeclareVariableKeyword{ m m o }
  {
    \prop_pop:NnNT \g__intexgral_differential_group_keyword_prop { #1 }
      \l_tmpa_tl
      {
        \seq_clear:c { g__intexgral_#1_jacobian_seq }
        \__intexgral_new_differential_group:nnn { #1 } { #2 } { #3 }
      }
  }

% Symbol keys

\NewDocumentCommand\NewSymbolKeyword{ m m }
  {
    \prop_if_in:NnT \g__intexgral_limits_keyword_prop { #1 }
      { \msg_warning:nnn { intexgral } { key-exists-for-lim } { #1 } }
    \keys_if_exist:nnTF { integral } { #1 }
      { \msg_error:nnn { intexgral } { symb-key-alr-def } { #1 } }
      {
```

```
        \keys_define:nn { integral }
          {
            #1 .meta:n =
              {
                symbol=#2,
                llimit=##1
              },
          }
      }
  }

\NewDocumentCommand\RenewSymbolKeyword{ m m }
  {
    \prop_if_in:NnT \g__intexgral_limits_keyword_prop { #1 }
      { \msg_warning:nnn { intexgral } { key-exists-for-lim } { #1 } }
    \keys_if_exist:nnTF { integral } { #1 }
      {
        \keys_define:nn { integral }
          {
            #1 .undefine:
            #1 .meta:n =
              {
                symbol=#2,
                llimit=##1
              }
          }
      }
      { \msg_error:nnn { intexgral } { symb-key-not-def } }
  }

\NewDocumentCommand\ProvideSymbolKeyword{ m m }
  {
    \prop_if_in:NnT \g__intexgral_limits_keyword_prop { #1 }
      { \msg_warning:nnn { intexgral } { key-exists-for-lim } { #1 } }
    \keys_if_exist:nnF { integral } { #1 }
      {
        \keys_define:nn { integral }
          {
            #1 .meta:n =
              {
                symbol=#2,
                llimit=##1
              },
          }
      }
  }

\NewDocumentCommand\DeclareSymbolKeyword{ m m }
  {
    \keys_define:nn { integral }
      {
        #1 .undefine:
        #1 .meta:n =
          {
            symbol=#2,
            llimit=##1
          },
      }
```

```
  }

\NewDocumentCommand\IntegralSetup{ m }
  { \keys_set:nn { IntegralSetup } { #1 } }

\NewDocumentCommand\integral{ O{} m }
  {
    \group_begin:
    \tl_if_empty:nF { #1 }
      {
        \__intexgral_extract_first_key_name:n { #1 }
        \__intexgral_parse_macro_keys:n { #1 }
      }
    \tl_set:Nn \l__intexgral_integrand_tl { #2 }
    \__intexgral_print_integral:
    \group_end:
  }

% Preamble only

\@onlypreamble\intexgralsetup

% Included keywords

% Symbol keywords
\NewSymbolKeyword{single}     {\int}
\NewSymbolKeyword{double}     {\iint}
\NewSymbolKeyword{triple}     {\iiint}
\NewSymbolKeyword{quadruple}  {\iiiint}
\NewSymbolKeyword{contour}    {\oint}
\NewSymbolKeyword{surface}    {\oiint}
\NewSymbolKeyword{volume}     {\oiiint}

% Limits keywords
\NewLimitsKeyword{ab}        {a, b}
\NewLimitsKeyword{real}      {-\infty, +\infty}
\NewLimitsKeyword{positive}  {0, +\infty}
\NewLimitsKeyword{negative}  {-\infty, 0}
\NewLimitsKeyword{unit}      {0, 1}
\NewLimitsKeyword{circle}    {0, 2\pi}
\NewLimitsKeyword{scircle}   {0, \pi}
\NewLimitsKeyword{qcircle}   {0, \frac{\pi}{2}}
\NewLimitsKeyword{height}    {0, H}
\NewLimitsKeyword{radius}    {0, R}
\NewLimitsKeyword{length}    {0, L}
\NewLimitsKeyword{time}      {0, T}

% Variable keywords
\NewVariableKeyword{cartesian}    {x, y, z}
\NewVariableKeyword{planar}       {x, y}
\NewVariableKeyword{polar}        {r, \theta}        [r]
\NewVariableKeyword{cylindrical}  {r, \theta, z}     [r]
\NewVariableKeyword{spherical}    {r, \theta, \phi}  [r^2, \sin\theta]

% Default parameters

\IntegralSetup
  {
```

```
    defaultvar  = {x},
    defaultvar* = {r},
    vectorstyle = \vec,
    domainstyle = \mathbb,
    symbolskip  = {-6mu},
    hide-diff   = false
  }

\ExplSyntaxOff
```