The intexgral package

v4.0.0

Valentin Dao*

June 6, 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 *l3kernel*.

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

Contents

Abstract 1

I Documentation 4

1 Package options 5

2 Main macro overview 5

3 List of keys 6
- 3.1 Integration limits 6
  - 3.1.1 Defining the limits 6
- 3.2 Display modes 7
- 3.3 Symbol (and more about limits) 8
  - 3.3.1 Selecting a symbol 8
  - 3.3.2 Generating many integrals 8
  - 3.3.3 Defining boundaries on an ad hoc basis 9
  - 3.3.4 Recurring limits motif 9
- 3.4 Differentials 10
  - 3.4.1 Specifying the integration variables 10
  - 3.4.2 Including the Jacobian 10
  - 3.4.3 Modifying the differential symbol 10
  - 3.4.4 Creating a path integral 11
  - 3.4.5 Adding an exponent 11

4 Additional macros 11
- 4.1 Custom differentials placement 12
- 4.2 Looking back at the *limits* key 12
- 4.3 Looking back at the *variables* key 14
- 4.4 Looking back at the *symbol* key 15

5 *Special* syntax 16
- 5.1 Presentation of the syntax 16
- 5.2 Functioning of the syntax 17

6 Additional parameters 17

Changelog 19

II Implementation 20

1 Initialisation 21
- 1.1 Pacakge declaration 21
- 1.2 Preliminary verifications 21

2 Package options 22
- 2.1 Variable definition 22
- 2.2 Package option declaration 22

3 Messages 23

4 Useful variants **25**

5 Internal variables **25**
5.1 Tokens 25
5.2 Booleans 26
5.3 Sequences and clists 27
5.4 Property lists 27
5.5 Musksip 27
5.6 String 28

6 Auxiliary macros and sequences **28**
6.1 Main sequences 29
6.2 Supplementary sequences 29

7 Integration limits **30**

8 Variables **33**

9 Special syntax **34**

10 Integral composition **36**
10.1 Mode *default* 37
10.2 Mode *nested* 38
10.3 Mode *product* 39
10.4 Final typesetting 40

11 Key declaration **40**

12 User-level macros **43**
12.1 Keywords for limits 43
12.2 Keywords for variables 44
12.3 Keywords for symbols 46
12.4 Configuration macros 47

13 Package configuration **48**
13.1 Symbol 48
13.2 Limits 48
13.3 Variables 48
13.4 Default parameters 48

Index **50**

License

© 2025 Valentin Dao, published under the LaTeX Project Public License (LPPL) 1.3c

Fonts

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

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

Monaspace Argon Medium
© 2023, GitHub

Repository

See GitHub repository.

Aknowledgements

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.

PART I

DOCUMENTATION

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⟩`

update: v3.0.0

In physical sciences, it is common to see the differentials placed before the integrand. This key hence swaps their position.

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

new: v3.0.0

Applies `\limits` or `\nolimits` to all integrals.

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

update: v4.0.0

Applies an upright or italic “d” for the differentials.

> ***Remark:*** Since the version v4.0.0, the *derivative* package is no longer used to manage differentials. The number of options is therefore more limited but should suit most use cases in the context of integrals.

2 Main macro overview

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

update: 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⟩).

```
\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⟩).

```
\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 **??**). 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}
```

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

[0]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 clear that the keys `limits` and `variables` (see 3.4.1), as well as the integrand, must have the same number of elements.

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⟩`

update: 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:*** aside from checking whether the macro exists, no specific checks are performed, and the value of the key is used as-is to typeset the symbol. This implies that the symbol will depend on how it is defined. For example, *amsmath* and *unicode-math* do not define `\iint` in the same way. The result will therefore naturally be different: *amsmath* concatenates two `\int` with some spacing to define `\iint`, whereas *unicode-math* defines the actual glyph `U+222C`.

```
\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` `ulimit`

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

update: 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

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

`domain` `domain*`

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

update: 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 limit.

```
\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`

update: 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 **??**).

> ***Remark:*** if no variable is specified, i.e. `variables` is not called, the package automatically sets "dx" (or "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 Including 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}
```

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

3.4.3 Modifying the differential symbol

`diff-symb` `diff-symb=` ⟨control sequence⟩

This key allows you to modify the symbol or style of the differentials.

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

$$\int e^{+\dfrac{iS[q(t)]}{\hbar}} \mathcal{D}q(t) \tag{15}$$

3.4.4 Creating a path integral

diff-vec `diff-vec`

This key applied a vector to each integration variable along with a centred dot. It is only compatible with the `default` mode. Using it with the `nested` or `product` options will have no effect.

```
\IntegralSetup{vectorstyle=\mathbf}
  \begin{equation}
    W = \integral[single=C, variables=s, diff-vec]{\mathbf F}
  \end{equation}
```

$$W = \int_C \mathbf{F} \cdot \mathrm{d}\mathbf{s} \tag{16}$$

3.4.5 Adding an exponent

diff-order `diff-order=`⟨csv-list⟩

This key places an exponent on each differential.

```
  \begin{equation}
    S[\phi] = \integral[diff-order=4]{\mathcal L (\phi, \partial_\mu \phi, x^
     \mu)}
  \end{equation}
```

$$S[\phi] = \int \mathcal{L}(\phi, \partial_\mu \phi, x^\mu)\, \mathrm{d}^4 x \tag{17}$$

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`, whose functioning varies slightly from one display mode to another:

- `default`: typesets all the differentials at once. It is compatible with the `invert-diff` option.
- `nested`: typesets all the differentials at once. The macro is *not* defined if the `invert-diff` option is active.
- `product`: typesets a single differential at a time. It will therefore need to be repeated between each semicolon. It is also compatible with the `invert-diff` option.

```
\begin{gather}
  \integral{\frac{\differentials}{x}}\\[1cm]
  \langle 0 \mid T\{ \phi(x) \phi(y) \} \mid 0 \rangle = \integral[diff-
   order=4, variables=p]{\frac{\differentials}{(2\pi)^4} \frac{e^{-i p \cdot
    (x-y)}}{p^2 - m^2 + i \epsilon}}
\end{gather}
```

$$\int \frac{\mathrm{d}x}{x} \tag{18}$$

$$\langle 0 \mid T\{\phi(x)\phi(y)\} \mid 0\rangle = \int \frac{\mathrm{d}^4p}{(2\pi)^4} \frac{e^{-ip\cdot(x-y)}}{p^2 - m^2 + i\epsilon} \tag{19}$$

4.2 Looking back at the *limits* key

`\NewLimitsKeyword` `\RenewLimitsKeyword` `\ProvideLimitsKeyword` `\DeclareLimitsKeyword`

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

update: v4.0.0

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.

When entering a new keyword, you must follow the convention for the order of delimiters. For example, you can write:

```
\usepackage{intexgral}
\NewLimitsKeyword{key1}{A, B}
\intexgralsetup{invert-limits=true}
\NewLimitsKeyword{key2}{B, A}
```

`key1` and `key2` will then produce the same integral. 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`	$0, \pi/2$
`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.

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

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

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

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

```
\begin{equation}
  \integral[limits={height;0,W;length}, variables={x, y, z}]{\kappa(T)\nabla
    T \cdot \mathbf{n}}
\end{equation}
```

$$\int_0^H \int_0^W \int_0^L \kappa(T)\nabla T \cdot \mathbf{n} \,\mathrm{d}x \,\mathrm{d}y \,\mathrm{d}z \tag{21}$$

4.3 Looking back at the *variables* key

\NewVariableKeyword
\RenewVariableKeyword
\ProvideVariableKeyword
\DeclareVariableKeyword

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

update: v4.0.0

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)
`cylindrical*`	ρ, θ, z (ρ)
`spherical`	r, θ, ϕ $(r^2, \sin\theta)$

```
\begin{equation}
  \integral[triple=V, variables=spherical]{f(r, \theta, \phi)}
\end{equation}
```

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

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` key

— 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}
```

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

One notable special case of this syntax is that if you want to modify the variable without specifying the bounds, you can do so more quickly as follows:

```
\begin{equation}
  \integral[:z]{f(z)}
\end{equation}
```

$$\int f(z) \,\mathrm{d}z \tag{29}$$

Even if it is not recommended.

5.2 Functioning of the syntax

From the user's perspective, it is relatively simple to choose which syntax to adopt. However, the package must be able to distinguish between the two. Without going too deeply into the details[2] of the implementation, it is worth noting that the package attempts to extract the first key from the optional argument[3] and checks if it exists. This is why, if you create a key with `\NewSymbolKeyword` whose name may probably serve as integration bounds, *intexrgal* may falsely evaluate the nature of the optional argument. In this kind of situation, the interpretation of the keys will be incorrect and the result will be wrong.

6 Additional parameters

`\IntegralSetup` `\IntegralSetup {⟨list of parameters⟩}`

new: v3.0.0
update: v4.0.0

This macro allows, under the syntax ⟨key=value⟩, to modify parameters related to certain keys. All assignments made are local and the macro can therefore be placed in a group if needed. Here is an example of the macro with the default assignments of the package:

[2] The more courageous among you are still invited to read the source code.
[3] Or at least, the group of tokens it thinks correspond to a key.

```
\IntegralSetup{
  diff-symb    = \l_@@_default_differential_symbol_tl, % see at the
   beginning of the implementation
  defaultvar   = {x},
  defaultvar*  = {r},
  varsep       = \thinmuskip,
  diffsep      = \thinmuskip,
  innersymbsep = {-5mu},
  postsymbsep  = {-1mu},
  vectorstyle  = \vec,
  domainstyle  = \mathbb,
  novar    = false
}
```

diff-symb `diff-symb=`⟨control sequence⟩

new: v4.0.0

This key controls the symbol used for differentials. If the option `italic` is used, `\mathnormal` is applied (see at the beginning of the implementation). Unlike the homonymous key of `\integral`, this one acts at a more global level.

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

new: v3.0.0

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`.

varsep `varsep=`⟨muexpr⟩

new: v4.0.0

This key modifies the spacing between the integrand (or the Jacobian if it is displayed) and the variables.

diffsep `diffsep=`⟨muexpr⟩

new: v4.0.0

This key modifies the spacing between the differentials themselves.

innersymbsep `innersymbsep=`⟨muexpr⟩

new: v4.0.0

When the `default` mode is in effect, that is to say when the `limits` key is used, the package inserts a gap between the symbols to slightly bring them together. This key therefore allows you to adjust this spacing.

postsymbsep `postsymbsep=`⟨muexpr⟩

new: v4.0.0

This key controls the spacing between the symbol and the element that follows (integrand or differential depending on `invert-limits`).

vectorstyle `vectorstyle=`⟨control sequence⟩

new: v3.0.0

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⟩

new: v3.0.0

Similarly, it is possible to change the macro to be applied for the `domain(*)` keys.

novar `novar=`⟨`true` | `false`⟩

new: v3.0.0
update: v4.0.0

When a large part of the document requires not to include the differentials, it is possible to extend the duration of the more punctual action of `variables=none`.

Changelog

4.0.0 (06-06-2026)

Added

- Detailed implementation.
- Keys `varsep`, `diffsep`, `postsymbsep` and `diff-order`.
- Variable keyword `cylindrical*`.

Changed

- The *`derivative`* is no longer a dependency. The differentials are now managed internally.
- Key `symbolskip`, renamed in `innersymbsep`.
- Key `hide-diff`, renamed in `novar`.

Removed

- Keys `diff-star` and `diff-options`.

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) — Version initiale.

PART II

IMPLEMENTATION

1 Initialisation

```
⟨*package⟩
⟨@@=intexgral⟩
```

1.1 Pacakge declaration

```
\NeedsTeXFormat{LaTeX2e}
\RequirePackage{expl3}[2025-05-14]

\def\intexgral@module{intexgral}
\def\intexgral@version{v4.0.0}
\def\intexgral@date{2026-05-26}
\def\intexgral@description{A LaTeX package for typesetting integrals}

\ProvidesExplPackage
  \intexgral@module
  \intexgral@date
  \intexgral@version
  \intexgral@description
```

1.2 Preliminary verifications

We check that the version of expl3 is recent enough before continuing. The package requires the macro `\regex_if_match:nnTF`, previously named `\regex_match:nnTF` in versions prior to May 2025.

```
\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 }
  }

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

Since *intexgral* uses `\mathbb` for the keys `domain(*)`, we must ensure that the macro exists. The verification is placed in a *hook* executed at the beginning of the document in case *amsfonts* or any package defining `\mathbb` is loaded after *intexgral*. This serves particularly to avoid conflicts between packages (*unicode-math* and *amssymb* for example).

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

2 Package options

2.1 Variable definition

`\l_@@_invert_limits_bool`

Booleen inverting the order of the limits.

```
\bool_new:N \l__intexgral_invert_limits_bool
```

`\l_@@_deactivate_variables_bool`

Determines whether the differentials should be displayed or not.

```
\bool_new:N \l__intexgral_deactivate_variables_bool
```

`\l_@@_invert_differential_bool`

Determines the order of display for the differentials.

```
\bool_new:N \l__intexgral_invert_differential_bool
```

`\l_@@_default_differential_symbol_tl`

Contains the default symbol for differentials.

```
\tl_new:N \l__intexgral_default_differential_symbol_tl
```

`\l_@@_limits_style_tl`

Contains the TEX primitive to use for the limits style.

```
\tl_new:N \l__intexgral_limits_style_tl
```

2.2 Package option declaration

We can now define the package options before loading them.

```
\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_differential_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  =
        {
          \tl_set:Nn \l__intexgral_default_differential_symbol_tl
```

```
            { \mathnormal{d} }
        },
      italic / false .code:n =
        {
          \tl_set:Nn \l__intexgral_default_differential_symbol_tl
            { \mathrm{d} }
        },
    italic .usage:n   = { preamble },
    italic .initial:n = { false },

    upright .choice:,
      upright / true .code:n  =
        {
          \tl_set:Nn \l__intexgral_default_differential_symbol_tl
            { \mathrm{d} }
        },
      upright / false .code:n =
        {
          \tl_set:Nn \l__intexgral_default_differential_symbol_tl
            { \mathnormal{d} }
        },
    upright .usage:n   = { preamble },
    upright .initial:n = { true }
  }

\ProcessKeyOptions[intexgral]
```

3 Messages

The next part of the implementation defines all the warning and error messages.

`\g_@@_integral_number_int`

We start by creating a global counter to number the integrals in the document.

```
\int_gzero_new:N \g__intexgral_integral_number_int
```

`\@@_warning_msg_header:` ⋆

We can now define a macro that acts as a header to warning messages. It uses the counter defined just before.

```
\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:
}
```

All the messages are then defined.

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

4 Useful variants

\regex_if_match:nV*TF*
\str_if_eq:ne*TF*
\str_if_eq:neF
\keys_if_exist:nV*TF*
\seq_use:Ne
\str_if_eq_p:en
\prop_gput:Nne
\seq_gset_split:cnn
\seq_set_split:Nnf

```
\cs_generate_variant:Nn \regex_if_match:nnTF { nVTF }
\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   }
\cs_generate_variant:Nn \prop_gput:Nnn       { Nne  }
\cs_generate_variant:Nn \seq_gset_split:Nnn  { cnn  }
\cs_generate_variant:Nn \seq_set_split:Nnn   { Nnf }
```

5 Internal variables

5.1 Tokens

\l_@@_integrand_tl

Token containg the integrand. It is used as-is of the `default` mode, whereas for the other modes, it is subsequently separated into a sequence.

```
\tl_new:N \l__intexgral_integrand_tl
```

\l_@@_integral_symbol_tl

Token containing the integral symbol. When not modified by the `symbol` key, it should still be initialised to `\int`.

```
\tl_new:N \l__intexgral_integral_symbol_tl
\tl_set_eq:NN \l__intexgral_integral_symbol_tl \int
```

\l_@@_lower_limit_tl
\l_@@_upper_limit_tl

Tokens containg the limits

```
\tl_new:N \l__intexgral_lower_limit_tl
\tl_new:N \l__intexgral_upper_limit_tl
```

\l_@@_differential_symbol_tl

Token containing the symbol for differentials, which can be modified with the `diffsymb` key. It is initialised to the `\l_@@_default_differential_symbol_tl` token.

```
\tl_new:N \l__intexgral_differential_symbol_tl
```

\l_@@_default_differential_tl
\l_@@_default_vector_differential_tl

Tokens containing the default differentials, used when the key `variables` is not provided.

```
\tl_new:N \l__intexgral_default_differential_tl
\tl_new:N \l__intexgral_default_vector_differential_tl
```

\l_@@_vectorial_differential_style_tl
\l_@@_domain_style_tl

Tokens containing the macros to apply to the`diff-vec` and `domain(*)` keys.

```
\tl_new:N \l__intexgral_vectorial_differential_style_tl
\tl_new:N \l__intexgral_domain_style_tl
```

\l_@@_domain_char_tl
\l_@@_domain_dimension_tl

Token used for storing respectively the character and the dimension of a domain.

```
\tl_new:N \l__intexgral_domain_char_tl
\tl_new:N \l__intexgral_domain_dimension_tl
```

\c_@@_left_bracket_tl
\c_@@_right_bracket_tl

Constant tokens for the `limits*` key.

```
\tl_const:Nn \c__intexgral_left_bracket_tl { [ }
\tl_const:Nn \c__intexgral_right_bracket_tl { ] }
```

\l_@@_key_name_tl

Token containing the potential name of a key, used to differentiate the syntax ⟨key=value⟩ from the special syntax.

```
\tl_new:N \l__intexgral_key_name_tl
```

5.2 Booleans

\l_@@_manual_differential_bool

Boolean used to determine if the differentials are inserted through `\differentials`.

```
\bool_new:N \l__intexgral_manual_differential_bool
```

\l_@@_custom_variables_bool

Boolean used to determine if the variables are defined with `variables`.

```
\bool_new:N \l__intexgral_custom_variables_bool
```

\l_@@_separate_integral_bool

Boolean used to determine if the integral is *separated* into several parts, that is to say, if the modes `nested` or `product` are in effect.

```
\bool_new:N \l__intexgral_separate_integral_bool
```

\l_@@_vectorial_differential_bool

Boolean used to determine if vectors should be applied to the differentials.

```
\bool_new:N \l__intexgral_vectorial_differential_bool
```

\l_@@_jacobian_bool

Boolean used to determine if the Jacobian should be displayed.

```
\bool_new:N \l__intexgral_jacobian_bool
```

5.3 Sequences and clists

\l_@@_domain_seq

Sequence containing the integration domains and their dimensions (the main sequences will be explained later).

```
\seq_new:N \l__intexgral_domain_seq
```

\l_@@_differential_order_clist

List containing the powers of the differentials (clist and not sequence since `.seq_set:N` does not exist for keys).

```
\clist_new:N \l__intexgral_differential_order_clist
```

5.4 Property lists

\g_@@_limits_keyword_prop
\g_@@_differential_group_keyword_prop

A property list is created to store limits and variable keywords.

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

5.5 Musksip

\l_@@_symbol_inner_sep_muskip
\l_@@_symbol_post_sep_muskip
\l_@@_differential_sep_muskip
\l_@@_variable_sep_muskip

Internal muskips used to store the spacings for respectively:

1. The spacing between the symbols when several are generated by `limits(*)`.
2. The spacing that follows the symbol.
3. The spacing between the differentials when the key `variables` is used.
4. The spacing between the differentials and the integrand.

```
\muskip_new:N \l__intexgral_symbol_inner_sep_muskip
\muskip_new:N \l__intexgral_symbol_post_sep_muskip
\muskip_new:N \l__intexgral_variable_sep_muskip
\muskip_new:N \l__intexgral_differential_sep_muskip
```

5.6 String

`\l_@@_display_mode_str`

Internal variable used to store the integral's display mode. The reason for using a *string* stems from the lack of a `\tl_case:nn`*TF* macro.

```
\str_new:N \l__intexgral_display_mode_str
```

6 Auxiliary macros and sequences

Here we define the two TEX primitives `\mkern` and `\mathchoice` with an *expl3*-like syntax.

`\@@_mkern:N` `\@@_mkern:N` ⟨muskip⟩

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

`\@@_mathchoice:nnnn` `\@@_mathchoice:nnnn` {⟨display style⟩} {⟨inline style⟩} {⟨script style⟩} {⟨scriptscript style⟩}

```
\cs_new_eq:NN \__intexgral_mathchoice:nnnn \mathchoice
```

`\@@_invert_limits:w` `\@@_invert_limits:w` ⟨lower limit⟩,⟨upper limit⟩`\q_stop`

Macro inverting the limits for the `invert-limits` option. The action is fully expandable and is more efficient than a `\seq_inverse:N` in the context of this package, knowing that a maximum of two elements are present.

```
\cs_new:Npn \__intexgral_invert_limits:w #1,#2\q_stop{#2,#1}
```

`\@@_general_integral_symbol:`

We create a macro that contains the general expression for an integral: symbol, limits style, lower limit and upper limit. We just have to assign the tokens to compose a complete symbol, and possibly repeat the macro for integrals defined on several variables.

```
\cs_new:Npn \__intexgral_general_integral_symbol:
  {
    \l__intexgral_integral_symbol_tl
    \l__intexgral_limits_style_tl
    \c_math_subscript_token
      { \l__intexgral_lower_limit_tl }
    \c_math_superscript_token
      { \l__intexgral_upper_limit_tl }
  }
```

6.1 Main sequences

Given the large number of existing keys and possible combinations, the tokens must be carefully organized and stored in order to assemble the final integral. The approach taken was to create sequences dedicated to each element of an integral:

`\l_@@_integral_symbol_seq`

One for the symbol (or the *symbols*, if the key `limits*` is in use).

```
\seq_new:N \l__intexgral_integral_symbol_seq
```

`\l_@@_integrand_seq`

One for the integrand (or *the* integrands, if split into several parts by `nested` or `product`).

```
\seq_new:N \l__intexgral_integrand_seq
```

`\l_@@_jacobian_seq`

One for the Jacobian.

```
\seq_new:N \l__intexgral_jacobian_seq
```

`\l_@@_differential_seq`

And finally, one for the differentials.

```
\seq_new:N \l__intexgral_differential_seq
```

This way, we can construct the final integral by varying how we extract the different elements from these sequences. Here is a more concrete example:
For an indefinite integral.

$$\overset{\mathrm{S}}{\boxed{\iint_S}}\ \overset{\mathrm{I}}{\boxed{f(x,y)}}\ \overset{\mathrm{D}}{\boxed{\mathrm{d}x\,\mathrm{d}y}}$$

For a definite integral.

$$\overset{\mathrm{S}_1}{\boxed{\int_a^b}}\ \overset{\mathrm{S}_2}{\boxed{\int_c^d}}\ \overset{\mathrm{I}_1}{\boxed{f(x)}}\ \overset{\mathrm{I}_2}{\boxed{g(y)}}\ \overset{\mathrm{D}_1}{\boxed{\mathrm{d}x}}\ \overset{\mathrm{D}_2}{\boxed{\mathrm{d}y}}$$

6.2 Supplementary sequences

A few additional sequences will be required during the intermediate stages, which will involve preparing the four main sequences listed above.

`\l_@@_variables_seq`

Firstly, a sequences containing the variables.

```
\seq_new:N \l__intexgral_variables_seq
```

`\l_@@_limits_seq`

Then, a sequence containing the pairs of integration limits, that is to say the elements of the `limits` key separated in two by the semicolon.

```
\seq_new:N \l__intexgral_limits_seq
```

\l_@@_upper_limits_seq
\l_@@_lower_limits_seq

Two sequences that contain the upper and lower limits respectively. Thus, if the key `limits` contains the elements `a;b`, `c;d` and `e;f`, then the sequence of lower limits will contain the elements `a`, `c` and `e`, while the sequence of upper limits will contain the elements `b`, `d` and `f`.

```
\seq_new:N \l__intexgral_upper_limits_seq
\seq_new:N \l__intexgral_lower_limits_seq
```

7 Integration limits

\@@_parse_limits:n `\@@_parse_limits:n {⟨lower limit⟩}{⟨ upper limit⟩}, ⟨keyword⟩`

This macro checks if its argument corresponds to a keyword defined for the limits, in which case it expands to its content, inverting it if necessary (according to the status of `\l_@@_invert_limits_bool`). Otherwise, the macro simply expands to its argument.

```
\cs_new:Npn \__intexgral_parse_limits:n #1
  {
    \prop_if_in:NnTF \g__intexgral_limits_keyword_prop { #1 }
      {
        \bool_if:NTF \l__intexgral_invert_limits_bool
          {
            \exp_last_unbraced:Ne \__intexgral_invert_limits:w
              { \prop_item:Nn \g__intexgral_limits_keyword_prop { #1 } }
              \q_stop
          }
          { \prop_item:Nn \g__intexgral_limits_keyword_prop { #1 } }
      }
      { #1 }
  }
```

\@@_set_limits_regular:

Once the argument of the key `limits` has been converted into a sequence (action performed within `.code:n` in the key declaration), each pair of limits is assigned to a temporary sequence. This sequence is created so that if a keyword is present, it is replaced by its corresponding limits according to the previous macro. The action is executed within a `\romannumeral`-type expansion to avoid expanding limits containing fragile macros (such as `\mathbb`). The lower and upper limits can then be extracted from this sequence and assigned to their respective sequences.

```
\cs_new_protected:Nn \__intexgral_set_limits_regular:
  {
    \seq_map_inline:Nn \l__intexgral_limits_seq
      {
        \seq_set_split:Nnf \l_tmpa_seq { , }
          { \__intexgral_parse_limits:n { ##1 } }
        \seq_put_right:Ne \l__intexgral_lower_limits_seq
          { \seq_item:Nn \l_tmpa_seq { 1 } }
        \seq_put_right:Ne \l__intexgral_upper_limits_seq
          { \seq_item:Nn \l_tmpa_seq { 2 } }
      }
```

```
}
```

\@@_set_limits_starred:

The following macro is similar to the previous one, but it handles limits in the form of intervals when the key `limits*` is used.

```
\cs_new_protected:Npn \__intexgral_set_limits_starred:
  {
    \seq_map_indexed_inline:Nn \l__intexgral_limits_seq
      {
        \seq_set_split:Nnf \l_tmpa_seq { , }
          { \__intexgral_parse_limits:n { ##2 } }
        \bool_if:NTF \l__intexgral_invert_limits_bool
          {
            \tl_set:Ne \l__intexgral_lower_limit_tl
              {
                \seq_item:Nn \l_tmpa_seq { 2 }
                \tex_mathpunct:D,
                \seq_item:Nn \l_tmpa_seq { 1 }
              }
          }
          {
            \tl_set:Ne \l__intexgral_lower_limit_tl
              {
                \seq_item:Nn \l_tmpa_seq { 1 }
                \tex_mathpunct:D,
                \seq_item:Nn \l_tmpa_seq { 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 { \seq_item:Nn \l_tmpa_seq { 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 { \seq_item:Nn \l_tmpa_seq { 2 } }
              {
                { +\infty } { \tex_right:D \c__intexgral_left_bracket_tl }
              }
              { \tex_right:D \c__intexgral_right_bracket_tl }
            }
      }
  }
```

\@@_set_limits:nn \@@_set_limits:nn {⟨lower limit⟩} {⟨upper limit⟩}

According to the `invert-limits` option, this macro assigns the lower and upper limits to their respective tokens. If the option is active, the assignment becomes somewhat counter-intuitive (the lower limit is assigned to `\l_@@_upper_limit_tl` and vice versa), but will still produce the expected result. The expansion of the limits is prevented to avoid issues with fragile macros as seen previously.

```
\cs_new_protected:Npn \__intexgral_set_limits:nn #1#2
  {
    \bool_if:NTF \l__intexgral_invert_limits_bool
      {
        \tl_set:Ne \l__intexgral_lower_limit_tl { \exp_not:n { #2 } }
        \tl_set:Ne \l__intexgral_upper_limit_tl { \exp_not:n { #1 } }
      }
      {
        \tl_set:Ne \l__intexgral_lower_limit_tl { \exp_not:n { #1 } }
        \tl_set:Ne \l__intexgral_upper_limit_tl { \exp_not:n { #2 } }
      }
  }
```

\@@_generate_integral_sequence:

Once the sequences of lower and upper limits are filled, we can use the structure of the sequences to our advantage to automatically generate as many symbols as necessary. We proceed with an incremented loop in which we perform two actions:

1. With the help of \@@_set_limits:nn, we assign the integration limits to the \l_@@_lower_limit_tl and \l_@@_upper_limit_tl tokens from the i^{th} element of the \l_@@_lower_limits_seq and \l_@@_upper_limits_seq sequences respectively.

2. We place to the right (i.e. in order) of the \l_@@_integral_symbol_seq sequence the \@@_general_integral_symbol: macro, which contains the general form of an integral. Note that we fully expand its content in order to retrieve the immediate assignment of tokens.

```
\cs_new_protected:Nn \__intexgral_generate_integral_sequence:
  {
    \int_set:Nn \l_tmpa_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 }
      }
    \int_step_inline:nn { \l_tmpa_int }
      {
        \__intexgral_set_limits:nn
          { \seq_item:Nn \l__intexgral_lower_limits_seq { ##1 } }
          { \seq_item:Nn \l__intexgral_upper_limits_seq { ##1 } }
        \seq_put_right:Ne \l__intexgral_integral_symbol_seq
          { \__intexgral_general_integral_symbol: }
      }
  }
```

8 Variables

\@@_parse_variables:n \@@_parse_variables:n {⟨integration variable⟩}, ⟨key⟩

This macro is applied to each element of the argument of the variables key to check whether it is a key or an explicit list, in the same way as for the limits. It mainly serves to transfer the elements of \l_@@_variables_seq (containing for example "x, y, z") to \l_@@_differential_seq (which will contain "$\mathrm{d}x, \mathrm{d}y, \mathrm{d}z$"); while taking into account the different options that have been applied. In the case where no variable is entered, the macro takes care of placing x or $\vec{r}$ depending on the value of the boolean \l_@@_vectorial_differential_bool.

```
\cs_new_protected:Nn \__intexgral_parse_variables:
  {
    \bool_if:NTF \l__intexgral_custom_variables_bool
      {
        \seq_map_indexed_inline:Nn \l__intexgral_variables_seq
          {
            \seq_put_right:Ne \l__intexgral_differential_seq
              {
                \exp_not:V \l__intexgral_differential_symbol_tl
                \clist_if_empty:NF \l__intexgral_differential_order_clist
                  {
                    \c_math_superscript_token
                      {
                        \clist_item:Nn
                        \l__intexgral_differential_order_clist
                        { ##1 }
                      }
                  }
                \bool_if:NT \l__intexgral_vectorial_differential_bool
                  { \exp_not:V \l__intexgral_vectorial_differential_style_tl }
                  { ##2 }
              }
          }
      }
      {
        \seq_put_right:Ne \l__intexgral_differential_seq
          {
            \exp_not:V \l__intexgral_differential_symbol_tl
            \clist_if_empty:NF \l__intexgral_differential_order_clist
              {
                % TODO: allow passing a sequence for default variables; currently o
                \c_math_superscript_token
                  {
                    \clist_item:Nn
                    \l__intexgral_differential_order_clist
                    { 1 }
                  }
              }
            \bool_if:NTF \l__intexgral_vectorial_differential_bool
              {
                \exp_not:V \l__intexgral_vectorial_differential_style_tl
                  {\l__intexgral_default_vector_differential_tl}
              }
              { \l__intexgral_default_differential_tl }
          }
```

```
    }
}
```

9 Special syntax

`\@@_retrieve_key_name:w` ⋆ `\@@_retrieve_key_name:w` ⟨key⟩=⟨value⟩ `\q_stop`

In order to handle the special syntax, we start by creating a delimited macro that extracts the key name and its value, and that only returns the former.

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

`\@@_extract_first_key_name:n` `\@@_extract_first_key_name:n` {⟨`\integral`'s optional argument⟩}

We then associate the optional argument received by `\integral` to a temporary *comma list*, from which we extract the first element; we then check if it contains an "`=`" sign. If so, we use the previous macro to retrieve only the key name; otherwise, we simply copy the group of tokens retrieved into `\l_@@_key_name_tl`.

```
\cs_new_protected:Npn \__intexgral_extract_first_key_name:n #1
  {
    \seq_set_split:Nnn \l_tmpa_seq { : } { #1 }
    \tl_set:Ne \l_tmpa_tl { \seq_item:Nn \l_tmpa_seq { 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 }
  }
```

`\@@_parse_integral_keys:n` `\@@_parse_integral_keys:n` {⟨first key of the optional argument⟩}

Using the *l3keys* module, we can check if the captured group of tokens corresponds to a defined key, in which case we directly apply `\keys_set:nn`. If the test is negative, we manually assign the keys by considering the optional argument as a sequence to be delimited with a colon. Temporary sequences are used to manipulate the different parts of the argument. We notably check for the presence of the token `+j` to activate the `jacobian` key, and we add the default variable if it is not specified. Finally, we assign values to the `limits`, `variables` and `mode` keys based on the extracted elements.

```
\cs_new_protected:Npn \__intexgral_parse_integral_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 }
          }
      }
  }
```

10 Integral composition

`\@@_integral_preconfiguration:`

Before being able to typeset the integral in the document, a number of checks must be performed. Indeed, depending on the combination of keys specified in the optional argument, the sequences may not be filled yet. We therefore perform the following checks:

- We check if the integrand should be split into several parts by the `nested` and `product` modes, in which case we populate the integrand sequence by splitting the elements at the semicolon. Otherwise, we simply add the integrand (`\l_@@_integrand_tl`) to the sequence.
- We verify if the sequence of symbols is empty, indicating that the `limits*` key has not been called. Indeed, only the use of this key triggers the execution of all macros presented in section section 7. In this case, we simply add to the sequence the general form of the integral, whose symbol and bounds will be optionally modified by the keys `symbol`, `ulimit` and `llimit`.
- We then control the deactivation of differentials in addition to detecting the presence of `\differentials` using a regular expression.

```
\cs_new_protected:Nn \__intexgral_integral_preconfiguration:
  {
    \bool_if:NTF \l__intexgral_separate_integral_bool
      {
        \seq_set_split:NnV
          \l__intexgral_integrand_seq
          { ; }
          \l__intexgral_integrand_tl
      }
      {
        \seq_put_right:NV
          \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_general_integral_symbol: }
      }

    \bool_if:NF \l__intexgral_deactivate_variables_bool
      { \__intexgral_parse_variables: }

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

We can now define the six macros that will be used to typeset a precise integral; three modes allowed by `mode`, each accompanied by a variant for the inverted differentials. It should also be noted that `\differentials` is only created if the boolean `\l_@@_manual_differential_bool` is true, in order to avoid unnecessary definitions.

10.1 Mode *default*

`\@@_print_default_integral:`

For the default mode, we typeset in the following order: integral symbol, integrand, Jacobian (if applicable), and differentials. Since the elements are successive, we apply `\seq_use:Nn` for each of the sequences, inserting a kern when necessary. Note that if `\l_@@_vectorial_differential_bool` is true, we add a middle dot and remove the previous kern (for spacing reasons related to `\cdot`).

```
\cs_new_protected:Npn \__intexgral_print_default_integral:
  {
    \bool_if:NT \l__intexgral_manual_differential_bool
      {
        \cs_set_protected:Npn \differentials
          {
            \seq_use:Nn \l__intexgral_differential_seq
              { \__intexgral_mkern:N \l__intexgral_differential_sep_muskip }
          }
      }
    \seq_use:Nn \l__intexgral_integral_symbol_seq
      { \__intexgral_mkern:N \l__intexgral_symbol_inner_sep_muskip }
    \__intexgral_mkern:N \l__intexgral_symbol_post_sep_muskip
    \seq_use:Nn \l__intexgral_integrand_seq {  }
    \bool_if:NT \l__intexgral_jacobian_bool
      { \seq_use:Nn \l__intexgral_jacobian_seq {  } }
    \__intexgral_mkern:N \l__intexgral_variable_sep_muskip
    \bool_if:NT \l__intexgral_vectorial_differential_bool
      { \tex_unkern:D \cdot}
    \bool_if:NF \l__intexgral_manual_differential_bool
      {
        \seq_use:Nn \l__intexgral_differential_seq
          { \__intexgral_mkern:N \l__intexgral_differential_sep_muskip }
      }
  }
```

`\@@_print_default_integral_inv:`

For the inverted differentials, nothing is very different, except that `\l_@@_integrand_seq` and `\l_@@_differential_seq` are swapped.

```
\cs_new_protected:Npn \__intexgral_print_default_integral_inv:
  {
    \bool_if:NT \l__intexgral_manual_differential_bool
      {
        \cs_set_protected:Npn \differentials
          {
            \seq_use:Nn \l__intexgral_differential_seq
              { \__intexgral_mkern:N \l__intexgral_differential_sep_muskip }
          }
      }
    \seq_use:Nn \l__intexgral_integral_symbol_seq
      { \__intexgral_mkern:N \l__intexgral_symbol_inner_sep_muskip }
    \__intexgral_mkern:N \l__intexgral_symbol_post_sep_muskip
    \bool_if:NF \l__intexgral_manual_differential_bool
      {
        \seq_use:Nn \l__intexgral_differential_seq
          { \__intexgral_mkern:N \l__intexgral_differential_sep_muskip }
```

```
      }
    \__intexgral_mkern:N \l__intexgral_variable_sep_muskip
    \bool_if:NT \l__intexgral_vectorial_differential_bool
      { \tex_unkern:D \cdot}
    \seq_use:Nn \l__intexgral_integrand_seq {  }
    \bool_if:NT \l__intexgral_jacobian_bool
      { \seq_use:Nn \l__intexgral_jacobian_seq {  } }
  }
```

10.2 Mode *nested*

`\@@_print_nested_integral:`

In the nested mode, we initiate an incremented loop where we extract the i^{th} element of each of the sequences of symbol, integrand and Jacobian. After the iteration, we compose all the differentials again using `\seq_use:Nn`.

```
\cs_new_protected:Npn \__intexgral_print_nested_integral:
  {
    \bool_if:NT \l__intexgral_manual_differential_bool
      {
        \cs_set_protected:Npn \differentials
          {
            \seq_use:Nn \l__intexgral_differential_seq
              { \__intexgral_mkern:N \l__intexgral_differential_sep_muskip }
          }
      }
    \int_step_inline:nn { \seq_count:N \l__intexgral_integral_symbol_seq }
      {
        \seq_item:Nn \l__intexgral_integral_symbol_seq { ##1 }
        \__intexgral_mkern:N \l__intexgral_symbol_post_sep_muskip
        \seq_item:Nn \l__intexgral_integrand_seq { ##1 }
        \bool_if:NT \l__intexgral_jacobian_bool
          { \seq_item:Nn \l__intexgral_jacobian_seq { ##1 } }
      }
    \__intexgral_mkern:N \l__intexgral_variable_sep_muskip
    \bool_if:NF \l__intexgral_manual_differential_bool
      {
        \seq_use:Nn \l__intexgral_differential_seq
          { \__intexgral_mkern:N \l__intexgral_differential_sep_muskip }
      }
  }
```

`\@@_print_nested_integral_inv:`

Here, it is the sequence of differentials that appears in the loop and the sequence of the integrand that is used at the end, after the iteration.

```
\cs_new_protected:Npn \__intexgral_print_nested_integral_inv:
  {
    \int_step_inline:nn { \seq_count:N \l__intexgral_integral_symbol_seq }
      {
        \seq_item:Nn \l__intexgral_integral_symbol_seq { ##1 }
        \__intexgral_mkern:N \l__intexgral_symbol_post_sep_muskip
        \seq_item:Nn \l__intexgral_differential_seq { ##1 }
      }
    \__intexgral_mkern:N \l__intexgral_variable_sep_muskip
    \seq_use:Nn \l__intexgral_integrand_seq {  }
```

```
    \bool_if:NT \l__intexgral_jacobian_bool
      { \seq_use:Nn \l__intexgral_jacobian_seq {  } }
  }
```

10.3 Mode *product*

\@@_print_product_integral:

The product mode is very similar to the nested mode, in the sense that all sequences appear in the loop.

```
\cs_new_protected:Npn \__intexgral_print_product_integral:
  {
    \bool_if:NT \l__intexgral_manual_differential_bool
      {
        \cs_set_protected:Npn \differentials
          {
            \seq_pop_left:NN \l__intexgral_differential_seq \l_tmpa_tl
            \tl_use:N \l_tmpa_tl
          }
      }
    \int_step_inline:nn { \seq_count:N \l__intexgral_integral_symbol_seq }
      {
        \seq_item:Nn \l__intexgral_integral_symbol_seq { ##1 }
        \__intexgral_mkern:N \l__intexgral_symbol_post_sep_muskip
        \seq_item:Nn \l__intexgral_integrand_seq { ##1 }
        \bool_if:NT \l__intexgral_jacobian_bool
          { \seq_item:Nn \l__intexgral_jacobian_seq { ##1 } }
        \__intexgral_mkern:N \l__intexgral_variable_sep_muskip
        \bool_if:NF \l__intexgral_manual_differential_bool
          { \seq_item:Nn \l__intexgral_differential_seq { ##1 } }
      }
  }
```

\@@_print_product_integral_inv:

Similarly, with a few adjustments.

```
\cs_new_protected:Npn \__intexgral_print_product_integral_inv:
  {
    \bool_if:NT \l__intexgral_manual_differential_bool
      {
        \cs_set_protected:Npn \differentials
          {
            \seq_pop_left:NN \l__intexgral_differential_seq \l_tmpa_tl
            \tl_use:N \l_tmpa_tl
          }
      }
    \int_step_inline:nn { \seq_count:N \l__intexgral_integral_symbol_seq }
      {
        \seq_item:Nn \l__intexgral_integral_symbol_seq { ##1 }

        \bool_if:NF \l__intexgral_manual_differential_bool
          {
            \__intexgral_mkern:N \l__intexgral_symbol_post_sep_muskip
            \seq_item:Nn \l__intexgral_differential_seq { ##1 }
          }
        \__intexgral_mkern:N \l__intexgral_variable_sep_muskip
```

```
        \seq_item:Nn \l__intexgral_integrand_seq { ##1 }
        \bool_if:NT \l__intexgral_jacobian_bool
          { \seq_item:Nn \l__intexgral_jacobian_seq { ##1 } }
      }
  }
```

10.4 Final typesetting

\@@_print_integral:

The main macro typesetting the integral first increments the global counter allowing each integral to be numbered. Then, it carries out the preparatory actions explained previously. Finally, depending on the placement of the differentials and the requested display mode, one of the six macros is called.

```
\cs_new_protected:Nn \__intexgral_print_integral:
  {
    \int_gincr:N \g__intexgral_integral_number_int
    \__intexgral_integral_preconfiguration:
    \bool_if:NTF \l__intexgral_invert_differential_bool
      {
        \str_case:VnF \l__intexgral_display_mode_str
          {
            { default } { \__intexgral_print_default_integral_inv: }
            { nested  } { \__intexgral_print_nested_integral_inv:  }
            { product } { \__intexgral_print_product_integral_inv: }
          }
          { \__intexgral_print_default_integral_inv: }
      }
      {
        \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: }
      }
  }
```

11 Key declaration

The creation of keys for `\integral` and `\IntegralSetup` is relatively trivial. The few elementary actions for the keys `limits` and `variables` that were not covered by the macros are performed here.

```
\keys_define:nn { integral }
  {
    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 =
  {
    \seq_set_split:Nnn \l__intexgral_limits_seq { ; } { #1 }
    \__intexgral_set_limits_regular:
    \__intexgral_generate_integral_sequence:
  },

limits* .code:n =
  {
    \seq_set_split:Nnn \l__intexgral_limits_seq { ; } { #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
      }
  },

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: }
```

```
                }
            }
        }
    },

  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:V \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:V \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 .code:n =
    {
      \bool_set_true:N \l__intexgral_custom_variables_bool
      \str_if_eq:nnTF { #1 } { none }
        { \bool_set_true:N \l__intexgral_deactivate_variables_bool }
        {
          \prop_get:NnNTF
            \g__intexgral_differential_group_keyword_prop { #1 } \l_tmpa_tl
```

```
                {
                  \seq_set_split:NnV
                  \l__intexgral_variables_seq
                  { , }
                  \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 } }
                }
                { \seq_set_split:Nnn \l__intexgral_variables_seq { , } { #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-symb .tl_set:N = \l__intexgral_differential_symbol_tl,
    diff-order .clist_set:N = \l__intexgral_differential_order_clist,
  }

\keys_define:nn { IntegralSetup }
  {
    diff-symb    .tl_set:N     = \l__intexgral_differential_symbol_tl,
    defaultvar   .tl_set:N     = \l__intexgral_default_differential_tl,
    defaultvar*  .tl_set:N     = \l__intexgral_default_vector_differential_tl,
    postsymbsep  .muskip_set:N = \l__intexgral_symbol_post_sep_muskip,
    varsep       .muskip_set:N = \l__intexgral_variable_sep_muskip,
    diffsep      .muskip_set:N = \l__intexgral_differential_sep_muskip,
    vectorstyle  .tl_set:N     = \l__intexgral_vectorial_differential_style_tl,
    domainstyle  .tl_set:N     = \l__intexgral_domain_style_tl,
    innersymbsep .tl_set:N     = \l__intexgral_symbol_inner_sep_muskip,
    novar        .bool_set:N   = \l__intexgral_deactivate_variables_bool,
  }
```

12 User-level macros

12.1 Keywords for limits

\@@_new_limits_group:nn \@@_new_limits_group:nn {⟨keyword⟩} {⟨limits⟩}

For the functioning of \NewLimitsKeyword, we register the keyword and its value in a property list. A particularity that nevertheless needs to be explained is that there is a double inversion of the limits: at the time of the creation of the keyword, and at the time of extracting its content (cf. \@@_parse_limits:nn). The reason is that it is difficult to attach to a keyword the status of *inverted limits*. Thus, whatever convention is chosen, the limits are stored *in the same order*. This also allows the keywords defined by the package to be in the correct order, even if the convention is changed after loading it.

```
\cs_new_protected:Npn \__intexgral_new_limits_group:nn #1#2
```

```
  {
    \prop_gput:Nne \g__intexgral_limits_keyword_prop { #1 }
      {
        \bool_if:NTF \l__intexgral_invert_limits_bool
          { \__intexgral_invert_limits:w #2\q_stop }
          { #2 }
      }
  }
```

\NewLimitsKeyword

This macro is documented in sub-section 4.2.

```
\NewDocumentCommand\NewLimitsKeyword{ m m }
  {
    \prop_if_in:NnTF \g__intexgral_limits_keyword_prop { #1 }
      { \msg_error:nnn { intexgral } { lim-group-alr-def } { #1 } }
      { \__intexgral_new_limits_group:nn { #1 } { #2 } }
  }
```

\RenewLimitsKeyword

This macro is documented in sub-section 4.2.

```
\NewDocumentCommand\RenewLimitsKeyword{ m m }
  {
    \prop_pop:NnNTF \g__intexgral_limits_keyword_prop { #1 } \l_tmpa_tl
      { \__intexgral_new_limits_group:nn { #1 } { #2 } }
      { \msg_error:nnn { intexgral } { lim-group-not-def } { #1 } }
  }
```

\ProvideLimitsKeyword

This macro is documented in sub-section 4.2.

```
\NewDocumentCommand\ProvideLimitsKeyword{ m m }
  {
    \prop_if_in:NnF \g__intexgral_limits_keyword_prop { #1 }
      { \__intexgral_new_limits_group:nn { #1 } { #2 } }
  }
```

\DeclareLimitsKeyword

This macro is documented in sub-section 4.2.

```
\NewDocumentCommand\DeclareLimitsKeyword{ m m }
  {
    \prop_remove:Nn \g__intexgral_limits_keyword_prop { #1 }
    \__intexgral_new_limits_group:nn { #1 } { #2 }
  }
```

12.2 Keywords for variables

\@@_new_variables_group:nnn `\@@_new_variables_group:nnn {⟨keyword⟩} {⟨variable⟩} [⟨jacobian⟩]`

Just like for the limits, we register the groups of differentials in a property list, with their name, value and eventually their associated jacobian.

```
\cs_new_protected:Npn \__intexgral_new_variables_group:nnn #1#2#3
  {
    \prop_gput:Nnn \g__intexgral_differential_group_keyword_prop
      { #1 } { #2 }
    \tl_if_blank:nF { #3 }
      {
        \seq_new:c { g__intexgral_#1_jacobian_seq }
        \seq_gset_split:cnn { g__intexgral_#1_jacobian_seq } { , } { #3 }
      }
  }
```

\NewVariableKeyword

This macro is documented in sub-section 4.3.

```
\NewDocumentCommand\NewVariableKeyword{ m m o }
  {
    \prop_if_in:NnTF \g__intexgral_differential_group_keyword_prop { #1 }
      { \msg_error:nnn { intexgral } { diff-group-alr-def } { #1 } }
      { \__intexgral_new_variables_group:nnn { #1 } { #2 } { #3 } }
  }
```

\RenewVariableKeyword

This macro is documented in sub-section 4.3.

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

\ProvideVariableKeyword

This macro is documented in sub-section 4.3.

```
\NewDocumentCommand\ProvideVariableKeyword{ m m o }
  {
    \prop_if_in:NnF \g__intexgral_differential_group_keyword_prop { #1 }
      { \__intexgral_new_variables_group:nnn { #1 } { #2 } { #3 } }
  }
```

\DeclareVariableKeyword

This macro is documented in sub-section 4.3.

```
\NewDocumentCommand\DeclareVariableKeyword{ m m o }
  {
    \prop_remove:Nn \g__intexgral_differential_group_keyword_prop { #1 }
    \seq_gclear:c { g__intexgral_#1_jacobian_seq }
    \__intexgral_new_variables_group:nnn { #1 } { #2 } { #3 }
  }
```

12.3 Keywords for symbols

This section does not require an intermediate macro, since we modify the keys of the *integral* module directly.

`\NewSymbolKeyword`

This macro is documented in sub-section 4.4.

```
\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
              },
          }
      }
  }
```

`\RenewSymbolKeyword`

This macro is documented in sub-section 4.4.

```
\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 } }
  }
```

`\ProvideSymbolKeyword`

This macro is documented in sub-section 4.4.

```
\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
            },
        }
    }
}
```

\DeclareSymbolKeyword

This macro is documented in sub-section 4.4.

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

12.4 Configuration macros

\IntegralSetup

This macro is documented in section 6.

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

\intexgralsetup

Just like the previous macro, but without forgetting to restrict its usage to the preamble.

This macro is documented in section 1.

```
\NewDocumentCommand\intexgralsetup{ m }
  { \keys_set:nn { intexgral } { #1 } }
\@onlypreamble\intexgralsetup
```

\integral

The final macro is only a few lines long. If an optional argument is provided, it performs the necessary checks to determine the nature of that argument (whether it follows special syntax or not) and assigns the keys accordingly. It then assigns the main argument to the corresponding token before calling the central macro that constructs the entire expression. The entire process is executed within a group so that all modifications remain local.

This macro is documented in section 2.

```
\NewDocumentCommand\integral{ O{} m }
```

```
{
  \group_begin:
  \tl_if_empty:nF { #1 }
    {
      \__intexgral_extract_first_key_name:n { #1 }
      \__intexgral_parse_integral_keys:n { #1 }
    }
  \tl_set:Nn \l__intexgral_integrand_tl { #2 }
  \__intexgral_print_integral:
  \group_end:
}
```

13 Package configuration

The last part of the package is dedicated to defining the default keywords for symbols, limits and variables, as well as the default parameters for the typesetting of integrals.

13.1 Symbol

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

13.2 Limits

```
\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, \pi/2}
\NewLimitsKeyword{height}    {0, H}
\NewLimitsKeyword{radius}    {0, R}
\NewLimitsKeyword{length}    {0, L}
\NewLimitsKeyword{time}      {0, T}
```

13.3 Variables

```
\NewVariableKeyword{cartesian}    {x, y, z}
\NewVariableKeyword{planar}       {x, y}
\NewVariableKeyword{polar}        {r, \theta}        [r]
\NewVariableKeyword{cylindrical}  {r, \theta, z}     [r]
\NewVariableKeyword{cylindrical*} {\rho, \theta, z}   [\rho]
\NewVariableKeyword{spherical}    {r, \theta, \phi}  [r^2, \sin\theta]
```

13.4 Default parameters

```
\IntegralSetup{
  diff-symb    = \l__intexgral_default_differential_symbol_tl,
  defaultvar   = {x},
  defaultvar*  = {r},
```

```
  varsep       = \thinmuskip,
  diffsep      = \thinmuskip,
  innersymbsep = {-5mu},
  postsymbsep  = {-1mu},
  vectorstyle  = \vec,
  domainstyle  = \mathbb,
  novar        = false
}
```
⟨/package⟩

Index

Symbols

\␣ 93, 97, 101, 105, 109, 113, 118

B

bool commands:
- \bool_if:NTF 206, 234, 251, 271, 300, 316, 336, 415, 434, 443, 455, 458, 460, 468, 479, 485, 488, 493, 506, 510, 526, 531, 544, 547, 553, 565, 572, 580, 768
- \bool_if:nTF 403
- \bool_new:N 28, 29, 30, 167, 168, 169, 170, 171
- \bool_set_false:N . 439, 605
- \bool_set_true:N 438, 610, 615, 720, 722, 744, 746

C

\c 437
\cdot 459, 486
clist commands:
- \clist_if_empty:NTF 307, 326
- \clist_item:Nn 311, 331
- \clist_new:N 173

cs commands:
- \cs_generate_variant:Nn 143, 144, 145, 146, 147, 148, 149, 150, 151
- \cs_if_exist:NTF ... 27, 640
- \cs_new:Nn 83
- \cs_new:Npn 184, 185, 202, 345
- \cs_new_eq:NN 183
- \cs_new_protected:Nn .. 216, 281, 298, 413, 576
- \cs_new_protected:Npn . 181, 228, 269, 346, 360, 441, 466, 491, 516, 529, 551, 764, 795
- \cs_set_protected:Npn . 445, 470, 495, 533, 555

D

\DeclareLimitsKeyword 790
\DeclareSymbolKeyword 883
\DeclareVariableKeyword .. 826
\def 6, 7, 8, 9
\differentials 445, 470, 495, 533, 555

E

exp commands:
- \exp_args:Ne 682, 703
- \exp_args:NNne 370
- \exp_last_unbraced:Ne . 208
- \exp_last_unbraced:NV . 354
- \exp_not:n . 273, 274, 277, 278, 306, 317, 325, 338, 687, 708

G

group commands:
- \group_begin: 902
- \group_end: 910

H

hook commands:
- \hook_gput_code:nnn 26

I

\iiiint 915
\iiint 914
\iint 913
\infty 257, 263, 920, 921, 922
\int 127, 154, 644, 658, 912
int commands:
- \int_compare:nNnTF . 374, 381, 650, 659
- \int_compare_p:nNn 405
- \int_gincr:N 578
- \int_gzero_new:N 82
- \int_set:Nn 283
- \int_step_inline:nn 289, 501, 518, 539, 561, 655
- \int_use:N 86
- \l_tmpa_int 283, 289

\integral 900
\IntegralSetup 895, 937
intexgral internal commands:
- \l__intexgral_custom_-variables_bool 168, 300, 720
- \l__intexgral_deactivate_-variables_bool . 29, 434, 722, 762
- \l__intexgral_default_-differential_symbol_tl ... 31, 55, 60, 69, 74, 938
- \l__intexgral_default_-differential_tl 158, 341, 371, 378, 754

\l__intexgral_default_vector_differential_tl 159, 339, 755
\g__intexgral_differential_group_keyword_prop . 175, 725, 797, 807, 813, 823, 828
\l__intexgral_differential_order_clist . 173, 307, 312, 326, 332, 748
\l__intexgral_differential_sep_muskip . 179, 448, 463, 473, 482, 498, 513, 758
\l__intexgral_differential_seq 197, 304, 323, 447, 462, 472, 481, 497, 512, 522, 535, 548, 557, 568
\l__intexgral_differential_symbol_tl 157, 306, 325, 747, 753
\l__intexgral_display_mode_str 180, 582, 591, 606, 611, 616
\l__intexgral_domain_char_tl 162, 681, 688, 702, 709
\l__intexgral_domain_dimension_tl 163, 683, 689, 704, 710
\l__intexgral_domain_seq . . . 172, 676, 677, 697, 698
\l__intexgral_domain_style_tl . . . 161, 687, 708, 760
__intexgral_extract_first_key_name:n . . 346, 905
__intexgral_general_integral_symbol: 185, 295, 431
__intexgral_generate_integral_sequence: . 281, 624, 631
\g__intexgral_integral_number_int 82, 86, 578
__intexgral_integral_preconfiguration: . . 413, 579
\l__intexgral_integral_symbol_seq 194, 294, 428, 430, 451, 476, 501, 503, 518, 520, 539, 541, 561, 563
\l__intexgral_integral_symbol_tl 153, 154, 187, 641, 644, 653, 657, 661
\l__intexgral_integrand_seq . 195, 418, 424, 454, 487, 505, 525, 543, 571
\l__intexgral_integrand_tl . . 152, 420, 425, 437, 908
\l__intexgral_invert_differential_bool 30, 39, 580
__intexgral_invert_limits:w 184, 208, 769
\l__intexgral_invert_limits_bool . . 28, 35, 206, 234, 251, 271, 768
\l__intexgral_jacobian_bool . 171, 455, 488, 506, 526, 544, 572, 744
\l__intexgral_jacobian_seq . 196, 456, 489, 507, 527, 545, 573, 734
\l__intexgral_key_name_tl 166, 352, 358, 362
\c__intexgral_left_bracket_tl 164, 259, 263
\g__intexgral_limits_keyword_prop 174, 204, 209, 212, 766, 775, 781, 787, 792, 834, 851, 869
\l__intexgral_limits_seq . . . 199, 218, 230, 622, 629
\l__intexgral_limits_style_tl 32, 45, 47, 188
\l__intexgral_lower_limit_tl 155, 190, 236, 244, 260, 273, 277, 634, 679, 680, 685, 700, 701, 706, 716
\l__intexgral_lower_limits_seq 201, 222, 253, 285, 287, 292
\l__intexgral_manual_differential_bool . . 167, 438, 439, 443, 460, 468, 479, 493, 510, 531, 547, 553, 565
__intexgral_mathchoice:nnnn 183, 663
__intexgral_mkern:N . . 181,

448, 452, 453, 457, 463, 473, 477, 478, 482, 484, 498, 504, 509, 513, 521, 524, 542, 546, 567, 570
__intexgral_new_limits_-group:nn 764, 777, 782, 788, 793
__intexgral_new_variables_-group:nnn 795, 809, 817, 824, 830
__intexgral_parse_integral_-keys:n 360, 906
__intexgral_parse_limits:n 202, 221, 233
__intexgral_parse_variables: 298, 435
__intexgral_print_default_-integral: 441, 593, 597
__intexgral_print_default_-integral_inv: . 466, 584, 588
__intexgral_print_integral: 576, 909
__intexgral_print_nested_-integral: 491, 594
__intexgral_print_nested_-integral_inv: 516, 585
__intexgral_print_product_-integral: 529, 595
__intexgral_print_product_-integral_inv: 551, 586
__intexgral_retrieve_key_-name:w 345, 355
\c__intexgral_right_bracket_-tl 165, 257, 265
\l__intexgral_separate_-integral_bool . . 169, 415, 605, 610, 615
__intexgral_set_limits:nn 269, 291
__intexgral_set_limits_-regular: 216, 623
__intexgral_set_limits_-starred: 228, 630
\l__intexgral_symbol_-inner_sep_muskip 176, 452, 477, 761
\l__intexgral_symbol_post_-sep_muskip . 177, 453, 478, 504, 521, 542, 567, 756
\l__intexgral_upper_limit_-tl 156, 192, 274, 278, 636
\l__intexgral_upper_limits_-seq . . . 200, 224, 252, 286, 293
\l__intexgral_variable_-sep_muskip . 178, 457, 484, 509, 524, 546, 570, 757
\l__intexgral_variables_-seq 198, 302, 728, 739
\l__intexgral_vectorial_-differential_bool . . 170, 316, 336, 458, 485, 746
\l__intexgral_vectorial_-differential_style_tl 160, 317, 338, 759
__intexgral_warning_msg_-header: . . 83, 118, 125, 132, 139
\intexgralsetup 897, 899
iow commands:
\iow_newline: 19, 20, 89

K

keys commands:
\keys_define:nn 33, 600, 751, 839, 855, 873, 885
\keys_if_exist:nnTF 146, 362, 836, 853, 871
\keys_set:nn 363, 386, 896, 898

M

\mathbb 27, 946
\mathchoice 183
\mathnormal 56, 75
\mathrm 61, 70
msg commands:
\msg_critical:nn 25
\msg_error:nnn 776, 783, 808, 819, 837, 865
\msg_line_context: 87

\msg_new:nnn . 16, 115, 123, 130, 137
\msg_new:nnnn . 91, 95, 99, 103, 107, 111
\msg_see_documentation_text:n 21
\msg_warning:nn 651
\msg_warning:nnn . . 643, 737, 835, 852, 870
muskip commands:
\muskip_new:N 176, 177, 178, 179

N

\NeedsTeXFormat 3
\NewDocumentCommand 773, 779, 785, 790, 805, 811, 821, 826, 832, 849, 867, 883, 895, 897, 900
\NewLimitsKeyword 113, 118, 773, 919, 920, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930
\NewSymbolKeyword 97, 832, 912, 913, 914, 915, 916, 917, 918
\NewVariableKeyword 805, 931, 932, 933, 934, 935, 936
\NewVarKeyword 105

O

\oiiint 918
\oiint 917
\oint 916

P

\partial 716
\phi 936
\pi 924, 925, 926
\ProcessKeyOptions 81
prop commands:
\prop_get:NnNTF 724
\prop_gput:Nnn . 149, 766, 797
\prop_if_in:NnTF . . 204, 775, 787, 807, 823, 834, 851, 869
\prop_item:Nn 209, 212
\prop_new:N 174, 175
\prop_pop:NnNTF . . . 781, 813
\prop_remove:Nn . . . 792, 828
\ProvideLimitsKeyword 785
\ProvidesExplPackage 11
\ProvideSymbolKeyword 867
\ProvideVariableKeyword . . 821

Q

quark commands:
\q_stop . 184, 210, 345, 355, 769

R

regex commands:
\regex_if_match:nnTF . . 143, 437
\regex_split:nnN 365
\RenewLimitsKeyword . . . 109, 779
\RenewSymbolKeyword 93, 849
\RenewVariableKeyword 811
\RenewVarKeyword 101
\RequirePackage 4, 27
\rho 935

S

scan commands:
\scan_stop: 182, 664, 665, 666, 667
seq commands:
\seq_count:N 286, 287, 374, 381, 405, 501, 518, 539, 561
\seq_gclear:N 816, 829
\seq_gset_split:Nnn 150, 802
\seq_if_empty:NTF . 285, 428
\seq_if_exist:NTF 731
\seq_item:Nn 223, 225, 238, 240, 246, 248, 255, 261, 292, 293, 349, 367, 385, 388, 389, 392, 407, 503, 505, 507, 520, 522, 541, 543, 545, 548, 563, 568, 571, 573
\seq_map_indexed_inline:Nn 230, 302
\seq_map_inline:Nn . 218, 677, 698
\seq_new:N . . 172, 194, 195, 196, 197, 198, 199, 200, 201, 801
\seq_pop_left:NN . . 535, 557
\seq_put_right:Nn 222, 224, 252, 253, 294, 304, 323, 376, 382, 423, 430
\seq_set_eq:NN 733
\seq_set_item:Nnn 370
\seq_set_split:Nnn 151, 220, 232, 348, 383, 417, 622, 629, 676, 697, 727, 739
\seq_use:Nn 147, 447, 451, 454, 456, 462, 472, 476, 481, 487, 489, 497, 512, 525, 527

\l_tmpa_seq ... 220, 223, 225, 232, 238, 240, 246, 248, 255, 261, 348, 349, 365, 367, 370, 374, 377, 381, 382, 385, 388, 392
\l_tmpb_seq 383, 389, 405, 407
\sin ... 936
str commands:
\str_case:nnTF . 392, 582, 591
\str_case_e:nnTF .. 255, 261
\str_if_eq:nnTF 144, 145, 366, 721
\str_if_eq_p:nn ... 148, 407
\str_new:N ... 180
\str_set:Nn 606, 611, 616
\str_uppercase:n .. 682, 703

T

TEX and LATEX 2ε commands:
\@ifpackagelater 24
\@onlypreamble 899
\intexgral@date 8, 13
\intexgral@description 9, 15
\intexgral@module 6, 12
\intexgral@version 7, 14
tex commands:
\tex_left:D 257, 259
\tex_limits:D 45
\tex_mathpunct:D .. 239, 247
\tex_mkern:D 182, 664, 665, 666, 667
\tex_nolimits:D 47
\tex_right:D 263, 265
\tex_unkern:D 459, 486
\theta 933, 934, 935, 936
\thinmuskip 941, 942
\times 680, 701
tl commands:
\c_space_tl 86
\tl_clear:N 653
\tl_const:Nn 164, 165
\tl_head:n 682, 703
\tl_if_blank:nTF 799
\tl_if_empty:NTF .. 679, 700
\tl_if_empty:nTF 903
\tl_if_in:NnTF 350
\tl_new:N . 31, 32, 152, 153, 155, 156, 157, 158, 159, 160, 161, 162, 163, 166
\tl_put_right:Nn .. 657, 661, 680, 685, 701, 706
\tl_set:Nn 55, 60, 69, 74, 236, 244, 273, 274, 277, 278, 349, 352, 675, 681, 683, 696, 702, 704, 716, 908
\tl_set_eq:NN 45, 47, 154, 358, 641, 644
\tl_tail:n 684, 705
\tl_trim_spaces:n 92, 96, 100, 104, 108, 112, 126
\tl_use:N 260, 536, 558
\l_tmpa_tl 349, 350, 355, 358, 535, 536, 557, 558, 675, 676, 696, 697, 725, 730, 781, 814
token commands:
\c_math_subscript_token 189, 710
\c_math_superscript_token 191, 309, 329, 689
\token_to_str:N . 93, 97, 101, 105, 109, 118, 127

V

\vec 945