

# L'extension *intexgral*

v4.0.0

Valentin Dao\*

6 juin 2026

---

## Résumé

Tandis que la composition d'intégrales est très fréquente en  $\text{\LaTeX}$ , cette dernière s'avère souvent peu pratique. Lorsque l'expression se complexifie, il devient alors difficile d'en modifier les différents éléments dans un code source peu lisible. Pour répondre à ce problème, le package *intexgral* met à disposition une macro centrale dont le seul argument est l'intégrande. Tout le reste (les symboles, les bornes, les variables d'intégration...) pourra aisément être modifié grâce à une interface `\cle=va leur`. Contrairement à la méthode classique, où l'utilisateur choisit l'ordre des bornes avec les caractères actifs `_` et `^`, le package a dû fixer une convention. Ainsi, pour les deux clés gérant les bornes qui seront présentées, l'entrée supposée sera `_(borne inférieure)^(borne supérieure)`. Ce package étant écrit en `expl3`, il dépend donc de *l3kernel*.

---

---

\*E-mail : [vdao.texdev@gmail.com](mailto:vdao.texdev@gmail.com)

# Table des matières

<b>Résumé</b>	<b>1</b>
<b>I Documentation</b>	<b>4</b>
<b>1 Options de package</b>	<b>5</b>
<b>2 Présentation de la macro principale</b>	<b>5</b>
<b>3 Liste des clés</b>	<b>6</b>
3.1 Bornes d'intégration	6
3.1.1 Définir les bornes	6
3.2 Modes d'affichage	7
3.3 Symbole (et encore des bornes)	8
3.3.1 Sélectionner le symbole	8
3.3.2 Générer beaucoup d'intégrales	8
3.3.3 Définir des bornes ponctuellement	9
3.3.4 Motifs récurrents de bornes	9
3.4 Différentielles	10
3.4.1 Spécifier les variables d'intégration	10
3.4.2 Inclure le jacobien	10
3.4.3 Modifier le symbole des différentielles	10
3.4.4 Réaliser une intégrale curviligne	11
3.4.5 Ajouter un exposant	11
<b>4 Macros annexes</b>	<b>11</b>
4.1 Emplacements personnalisés des différentielles	12
4.2 Retour sur la clé <i>limits</i>	12
4.3 Retour sur la clé <i>variables</i>	14
4.4 Retour sur la clé <i>symbol</i>	15
<b>5 Syntaxe <i>spéciale</i></b>	<b>16</b>
5.1 Présentation de la syntaxe	16
5.2 Fonctionnement de la syntaxe	17
<b>6 Paramètres annexes</b>	<b>18</b>
<b>Historique des modifications</b>	<b>20</b>
<b>II Implémentation</b>	<b>21</b>
<b>1 Initialisation</b>	<b>22</b>
1.1 Déclaration du package	22
1.2 Vérifications préliminaires	22
<b>2 Options de package</b>	<b>23</b>
2.1 Définition des variables	23
2.2 Déclaration des options	23
<b>3 Messages</b>	<b>24</b>

<b>4</b>	<b>Variantes utiles</b>	<b>26</b>
<b>5</b>	<b>Variables internes</b>	<b>26</b>
5.1	Tokens	26
5.2	Booléens	27
5.3	Séquences et clist	28
5.4	Liste de propriétés	28
5.5	Musksip	29
5.6	String	29
<b>6</b>	<b>Macros et séquences auxiliaires</b>	<b>29</b>
6.1	Séquences principales	30
6.2	Séquences subsidiaires	31
<b>7</b>	<b>Bornes d'intégration</b>	<b>31</b>
<b>8</b>	<b>Variables</b>	<b>34</b>
<b>9</b>	<b>Syntaxe spéciale</b>	<b>35</b>
<b>10</b>	<b>Composition de l'intégrale</b>	<b>37</b>
10.1	Mode <i>default</i>	38
10.2	Mode <i>nested</i>	39
10.3	Mode <i>product</i>	40
10.4	Composition finale	41
<b>11</b>	<b>Déclaration des clés</b>	<b>42</b>
<b>12</b>	<b>Macros d'interface utilisateur</b>	<b>45</b>
12.1	Mots-clés pour les bornes	45
12.2	Mots-clés pour les variables	46
12.3	Mots-clés pour les symboles	47
12.4	Macros de configuration	48
<b>13</b>	<b>Configuration du package</b>	<b>49</b>
13.1	Symbole	49
13.2	Bornes	49
13.3	Variables	50
13.4	Paramètres par défaut	50
	<b>Index</b>	<b>51</b>

## Licence

© 2025 Valentin Dao, publié sous la ~~La~~TeX Project Public License (LPPL) 1.3c

## Polices

*Chronicle Text G3 Roman*

© 2002, 2007 Hoefler & Frere-Jones.

*Whitney-Medium*

© 1996, 2009 Hoefler & Frere-Jones.

*Monospace Argon Medium*

© 2023, GitHub

## Dépôt

 [Voir dépôt GitHub.](#)

## Remerciements

Un grand merci à Plante et Slurpy de m’avoir accompagné dans cette aventure que fut l’apprentissage de  $\text{\TeX}$ , vos conseils ont été précieux. Je tiens également à remercier Anthony pour avoir toujours gentiment révisé les nouvelles versions et m’avoir donné des idées pour celles à venir.

# PREMIÈRE PARTIE

# DOCUMENTATION

## 1 Options de package

`\intexgralsetup` `\intexgralsetup` {<options de package>}

nouveau: v2.0.0

Ces options peuvent être déclarées de façon classique avec `\usepackage` ou bien avec `\intexgralsetup` dans le préambule.

`invert-limits` `invert-limits`=<true | false>

Cette clé permet d'inverser la convention d'ordre des limites. Le document ne peut suivre qu'une seule convention.

`invert-diff` `invert-diff`=<true | false>

mise à jour: v3.0.0

En sciences physiques, il est courant de voir les variables d'intégration placées avant l'intégrande. Cette clé intervertit donc leurs positionnements.

`limits-mode` `limits-mode`=<limits | nolimits>

nouveau: v3.0.0

Applique `\limits` ou `\nolimits` à toutes les intégrales.

`italic`  
`upright` `italic`=<true | false>  
`upright`=<true | false>

mise à jour: v4.0.0

Applique un «d» droit ou italique pour les différentielles.

*Remarque :* depuis la version v4.0.0, le package *derivative* n'est plus utilisé pour gérer les différentielles. Le nombre d'option y est donc plus restreint mais devrait convenir à la plupart des cas d'usage dans le cadre des intégrales.

## 2 Présentation de la macro principale

`\integral` `\integral` [(liste de clés)] {<intégrande>}

mise à jour: v3.0.0

Cette macro est celle qui permet de composer les intégrales. Elle doit naturellement être utilisée en mode mathématique uniquement. En voici un premier exemple sous sa forme la plus simple :

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

$$\int x \, dx \tag{1}$$

Puisque cette macro est axée autour de l'utilisation de clés, la première partie de cette documentation les présentera en les regroupant par domaines. La seconde partie quant à elle introduira les quelques macros annexes qui viennent compléter l'usage de certaines clés. Le reste de ce document exposera les autres fonctionnalités du package.

## 3 Liste des clés

### 3.1 Bornes d'intégration

#### 3.1.1 Définir les bornes

**limits** `limits= {⟨liste-mixte⟩}, ⟨mot-clé⟩`  
**limits\*** `limits*= {⟨liste-mixte⟩}, ⟨mot-clé⟩`

Cette clé détermine les bornes d'intégration à utiliser en suivant la convention édictée dans le résumé. Sous sa forme la plus simple, la valeur de la clé prend comme argument une liste d'éléments séparés par une virgule (*comma-separated values* ou `⟨csv-list⟩`).

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

$$\int_1^{10} f(x) \, dx \quad (2)$$

L'avantage notable de la clé **limits** est qu'on peut lui spécifier les bornes d'intégration de plusieurs intégrales, et le package compose automatiquement les symboles correspondants. Pour séparer les paires de bornes, il faut utiliser le point-virgule (*semicolon-separated values* ou `⟨ssv-list⟩`<sup>1</sup>).

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

$$\int_1^2 \int_3^4 f(x, y) \, dx \, dy \quad (3)$$

Il est également possible de renseigner des bornes prédéfinies à l'aide de mots-clés (voir sous-section 4.2). Enfin, la variante étoilée permet d'exprimer les bornes de l'intégrale sous forme d'intervalle. Le cas échéant, le sens des crochets s'adapte automatiquement à la présence de  $\pm\infty$

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

$$\int_{[1,10]} f(x) \, dx \quad (4)$$

1. On entend donc par `⟨liste-mixte⟩` le fait que **limits** puisse accepter un ensemble de `⟨csv-list⟩` dans une `⟨ssv-list⟩` plus globale.

## 3.2 Modes d'affichage

**mode** `mode=<default | nested | product>`  
**nouveau: v3.0.0** Uniquement lorsque la clé `limits` est employée, il est possible d'alterner entre trois styles d'affichage distincts. Chacun d'entre eux est entièrement compatible avec l'option `invert-diff`.

**default** Compose l'ensemble des symboles, puis l'intégrande, puis les variables d'intégration. La macro opérant par défaut dans ce mode, il n'est donc pas nécessaire d'utiliser la clé avec cette valeur.

**nested** Compose une intégrale *imbriquée* où symbole et intégrande alternent avant que toutes les variables d'intégration soient écrits.

**product** Compose un produit d'intégrales où symbole, intégrande et variables d'intégration alternent.

Bien sûr, rien ne vous empêche pour le mode produit d'utiliser plusieurs fois la macro `\integral` d'affilée pour produire le même résultat que dans l'exemple ci-dessous. Pour celles et ceux qui préféreraient cependant obtenir le résultat avec une seule macro, cette méthode est permise.

**Important:** pour les deux derniers modes, l'intégrande sera, d'une certaine manière, *découpée*. Afin de correctement effectuer cette action, il faut avoir recours à la même méthode qu'avec `limits`, c'est-à-dire en employant le point-virgule.

MODE **default**

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

$$\int_1^2 \int_3^4 \int_5^6 xyz \, dx \, dy \, dz \quad (5)$$

MODE **nested**

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

$$\int_1^2 x \int_3^4 y \int_5^6 z \, dz \, dy \, dx \quad (6)$$

MODE **product**

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

$$\int_1^2 x \, dx \int_3^4 y \, dy \int_5^6 z \, dz \quad (7)$$

Pour bien fonctionner, il est évident que les clés `limits` et `variables` (voir 3.4.1), ainsi que l'intégrande, doivent avoir le même nombre d'éléments.

### 3.3 Symbole (et encore des bornes)

Les clés `limits(*)` se révèlent très utiles lorsque l'on souhaite composer une intégrale définie. Néanmoins, cela couvre seulement les expressions finales où les bornes de chaque variable ont été spécifiées. Pour des cas plus généraux (les intégrales indéfinies donc) elles sont relativement malcommodes. Pour composer une intégrale double sur une surface  $S$ , on devrait écrire `limits={, ; S, }`. Il va sans dire que c'est assez mal adapté pour l'utilisateur et peu optimal pour le package. Par ailleurs, le glyphe ne serait pas correct. L'ensemble des clés à suivre propose donc une façon de facilement modifier le symbole ainsi que les bornes.

#### 3.3.1 Sélectionner le symbole

`symbol`

`symbol=<séquence de contrôle>`

mise à jour : v3.0.0

Cette clé accepte une macro désignant un symbole d'intégration. Tout symbole préalablement défini par une séquence de contrôle est recevable. Si vous tentez d'en utiliser une qui n'est pas définie, elle sera substituée à `\iint` et un message d'avertissement sera émis.

*Remarque :* à part la vérification de l'existence de la macro, aucun contrôle particulier n'est effectué et la valeur de la clé est utilisée telle quelle pour composer le symbole. Ceci implique notamment que le symbole va dépendre de la façon dont il est défini. Par exemple, `amsmath` et `unicode-math` ne définissent pas `\iint` de la même manière. Le résultat sera donc naturellement différent : `amsmath` colle deux `\iint` ensemble avec un certain crénage pour définir `\iint`, tandis qu'`unicode-math` définit le vrai glyphe U+222C.

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

$$\iint_S f(x, y) \, dx \, dy \quad (8)$$

#### 3.3.2 Générer beaucoup d'intégrales

`nint` `nint=<entier>`

Cette clé accepte un entier  $n$  qui permet de composer  $n$  intégrales. Il est conseillé d'avoir recours à cette clé seulement si le nombre de symboles dépasse 4 afin de privilégier les glyphes définis par la police mathématique utilisée.



```
\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}
```

$$\iiint\limits_{\Omega} f(x_1, x_2, x_3, x_4, x_5) dx_1 dx_2 dx_3 dx_4 dx_5 \quad (9)$$

### 3.3.3 Définir des bornes ponctuellement

<code>llimit</code>	<code>llimit=&lt;borne inférieure&gt;</code>
<code>ulimit</code>	<code>ulimit=&lt;borne supérieure&gt;</code>

mise à jour: v3.0.0

Ces deux clés permettent de spécifier les bornes inférieures et supérieures respectivement. Elles ne sont adaptées que si un seul symbole est affiché. Si vous avez besoin de spécifier les deux bornes, la clé `limits` devra être privilégiée.

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

$$\int_{x^2+y^2 \leq 1} f(x, y) dx dy \quad (10)$$

### 3.3.4 Motifs récurrents de bornes

Les bornes suivent parfois des schémas courants qui peuvent se généraliser avec des clés, évitant ainsi de surcharger l'argument de `llimit`.

<code>domain</code>	<code>domain= {(*-liste)}</code>
<code>domain*</code>	<code>domain*= {(*-liste)}</code>

mise à jour: v3.0.0

Ces clés acceptent une liste dont le délimiteur est un astérisque. Ensuite, chaque élément de la liste est analysé de la façon suivante :

- Le premier token de l'item se voit passer `\mathbb` (précédé d'`\uppercase` au cas où la touche SHIFT serait trop loin pour vos doigts).
- Les tokens restants, qui peuvent être vides, sont placés comme exposant (ou en indice pour la variante étoilée de la clé).

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

$$\iint_{\mathbb R \times \mathbb R} xy dx dy \quad (11)$$

<code>boundary</code>	<code>boundary= {(borne inférieure)}</code>
-----------------------	---

Cette clé place simplement le symbole  $\partial$  avant la borne inférieure.

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

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

## 3.4 Différentielles

### 3.4.1 Spécifier les variables d'intégration

variables `variables=` {<liste-csv>}, <mot-clé>, `none`

mise à jour: v3.0.0

Cette clé permet de définir les variables d'une intégrale sous forme d'une <csv-list>. La clé peut, tout comme `limits`, accepter un mot-clé comme argument. Ce comportement est aussi expliqué plus tard (voir sous-section 4.3)

*Remarque :* si aucune variable d'intégration n'est renseignée, c'est-à-dire que `variables` n'est pas appelée, le package place automatiquement «dx» (ou «d $\vec{r}$ » si `diff-vec` est actif). De plus, si `none` est passé comme valeur, aucune variable d'intégration ne sera affichée.

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

$$\int t^2 dt \quad (13)$$

### 3.4.2 Inclure le jacobien

jacobian `jacobian`

Cette clé active l'affichage du jacobien lorsque défini par `\NewVariableKeyword`.

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

$$\int_0^R r^2 dr \int_0^{2\pi} \sin \theta d\theta \int_0^\pi d\phi \quad (14)$$

### 3.4.3 Modifier le symbole des différentielles

diff-symb `diff-symb=` <séquence de contrôle>

Cette clé permet de modifier le symbole ou bien le style des différentielles.

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

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

### 3.4.4 Réaliser une intégrale curviligne

#### diff-vec diff-vec

Cette clé applique un vecteur à la variable d'intégration en plus d'un point médian. Elle n'est compatible qu'avec le mode **default**. L'utiliser avec les options **nested** ou **product** n'aura aucun effet.

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

$$W = \int_C \mathbf{F} \cdot d\mathbf{s} \quad (16)$$

### 3.4.5 Ajouter un exposant

#### diff-order diff-order=<liste-csv>

Cette clé place un exposant à chaque différentielle.

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

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

## 4 Macros annexes

En plus de la large gamme de clés définies par le package, quelques macros viennent également améliorer leurs usages.

## 4.1 Emplacements personnalisés des différentielles

### `\differentials` `\differentials`

Bien que l'option `invert-diff` existe, il est souvent souhaitable de pouvoir placer les différentielles où l'on veut. Par exemple, il est fréquent de les voir au numérateur d'une fraction lorsque celui-ci vaut 1. Pour répondre à ce besoin, le package met à disposition la macro `\differentials`, dont le fonctionnement varie légèrement d'un mode d'affichage à un autre :

- **default** : compose toutes les différentielles d'un coup. Fonctionne avec `invert-diff`.
- **nested** : compose toutes les différentielles d'un coup. La macro n'est *pas* définie si `invert-diff` est actif.
- **product** : compose une seule différentielle à la fois. Il faudra donc la répéter entre chaque point-virgule. Fonctionne aussi avec `invert-diff`.

```
\begin{gather}
\integral{\frac{\differentials}{x}}{\mid 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}
```

$$\int \frac{dx}{x} \quad (18)$$

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

## 4.2 Retour sur la clé *limits*

`\NewLimitsKeyword` `\NewLimitsKeyword {<mot-clé>} {<bornes>}`  
`\RenewLimitsKeyword`  
`\ProvideLimitsKeyword`  
`\DeclareLimitsKeyword`

mise à jour : v4.0.0

Il est courant de devoir renseigner les mêmes bornes dans une intégrale. Pour faciliter leur écriture, les clés `limits(*)` acceptent, en plus des bornes explicites, un mot-clé en désignant une paire prédéfinie par l'une de ces quatre macros :

- `\NewLimitsKeyword` Crée un nouveau mot-clé et émet une erreur s'il existe déjà.
- `\RenewLimitsKeyword` Redéfinit un mot-clé et émet une erreur s'il n'existe pas déjà.
- `\ProvideLimitsKeyword` Ne crée un nouveau mot-clé que s'il n'existe pas déjà. Aucun message ne sera émis dans le cas échéant.
- `\DeclareLimitsKeyword` Crée un nouveau mot-clé quoi qu'il en soit, écrasant toute définition préalable.

La saisie d'un nouveau mot-clé devra se faire dans le respect de la convention d'ordre des bornes. Vous pouvez ainsi écrire :

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

**key1** et **key2** produiront alors la même intégrale. Voici la liste des mots-clés déjà définis par le package et les bornes qu'ils contiennent :

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

***Remarque :*** les mots-clés contiennent les *deux* bornes d'une intégrale en même temps. Ils doivent donc être précédés et/ou suivis de point-virgule.

On pourrait donc modifier l'expression d'une l'intégrale de la façon suivante :

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

$$\int_0^H \int_0^{2\pi} \int_0^R \rho \, d\rho \, d\theta \, dz \quad (20)$$

Il est tout à fait possible de mélanger ces mots-clés avec la syntaxe plus explicite de **limits**.

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

$$\int_0^H \int_0^W \int_0^L \kappa(T) \nabla T \cdot \mathbf{n} \, dx \, dy \, dz \quad (21)$$

## 4.3 Retour sur la clé *variables*

---

```

\NewVariableKeyword \NewVariableKeyword {<mot-clé>} {<variables>} [<jacobien>]
\RenewVariableKeyword
\ProvideVariableKeyword
\DeclareVariableKeyword

```

---

mise à jour : v4.0.0

De façon similaire, les mêmes groupes de variables réapparaissent souvent. On peut donc également définir des mots-clés contenant un ensemble de variables d'intégration préenregistrés. Les variantes de cette macro ont le même comportement que pour `\NewLimitsKeyword`. La seule différence est qu'il est ici possible de définir un jacobien, sous forme d'une `<csv-list>` si besoin. Son affichage est ensuite contrôlé par la clé `jacobian` expliquée précédemment. Voici la liste des mots-clés de variables déjà définis par le package, avec le jacobien pour certains :

```

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 (r2, sin θ)

```

```

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

```

$$\iiint_V f(r, \theta, \phi) \, dr \, d\theta \, d\phi \quad (22)$$

## 4.4 Retour sur la clé *symbol*

---

```

\NewSymbolKeyword \NewSymbolKeyword {<clé>} {<symbole>}
\RenewSymbolKeyword
\ProvideSymbolKeyword
\DeclareSymbolKeyword

```

---

nouveau: v3.0.0

Afin de composer des intégrales indéfinies, le package offre essentiellement deux clés : `symbol` et `llimit`. Bien qu'elles soient simples d'utilisation, il demeure toujours laborieux de les écrire toutes les deux avec leurs valeurs. C'est pourquoi `intexgral` met à disposition des clés dites *raccourcies*, dont le but est de combiner l'action de sélection du symbole et des bornes. Contrairement aux deux macros précédentes, il s'agit ici de créer de toutes nouvelles clés, et non simplement des valeurs spécifiques attribuées à `symbol`. Toutes seules, ces clés modifient le symbole intégral. La valeur de ces clés correspondra elle à la borne inférieure. Voici la liste de l'ensemble des *clé-symbole* définie par le package :

<code>single</code>	symbole utilisé = <code>\int</code>
<code>double</code>	symbole utilisé = <code>\iint</code>
<code>triple</code>	symbole utilisé = <code>\iiint</code>
<code>quadruple</code>	symbole utilisé = <code>\iiiiint</code>
<code>contour</code>	symbole utilisé = <code>\oint</code>
<code>surface</code>	symbole utilisé = <code>\oiint</code>
<code>volume</code>	symbole utilisé = <code>\oiint</code>

**Important :** offrir la possibilité de créer soi-même une clé à la macro `\integral` représente un risque qui sera mieux expliqué dans la prochaine section. Retenez pour l'instant que toutes ces macros émettront un message d'avertissement si vous tentez de créer une nouvelle *clé-symbole* portant le même nom qu'un groupe de bornes définies.

```

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

```

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

## 5 Syntaxe spéciale

### 5.1 Présentation de la syntaxe

`\integral` `\integral` [(limits:variables(+j):mode)] {(intégrande)}

nouveau: v3.0.0

En ce qui concerne les intégrales définies, l'utilisateur sera amené à employer au maximum quatre clés pour les composer : `limits`, `variables`, `mode` et `jacobian`. On peut néanmoins leur reprocher la même chose qu'avant ; écrire ces quatre clés, pouvant recevoir des arguments assez longs, va à l'encontre de l'objectif principal de ce package. Ainsi, l'utilisateur peut désigner comme argument optionnel d'`\integral`, une syntaxe dite *spéciale* qui n'est propre qu'à `intexgral`. La logique est la suivante :

- Vous spécifiez un argument *valide* de la clé `limits`
- Vous spécifiez un argument *valide* de la clé `variables`
- Vous spécifiez un argument *valide* de la clé `mode`

Et vous séparez le tout d'un deux points. Voyons un exemple pour mieux comprendre.

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

$$\int_1^2 x \int_4^5 y \, dy \, dx \quad (24)$$

On entend par *argument valide* le fait que toutes les formes de valeurs acceptées par les quatre clés peuvent être pareillement adoptées par la syntaxe spéciale. Cela comprend donc les mots-clés.

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

$$\int_0^R r \, dr \int_0^{2\pi} \theta \, d\theta \int_0^\pi \phi \, d\phi \quad (25)$$

Afin d'inclure le jacobien, il suffit d'écrire `+j` après l'argument de `variables`. Le mode peut aussi être indiqué à l'aide d'une initiale seulement.

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

$$\int_0^R r^2 \, dr \int_0^{2\pi} \sin \theta \, d\theta \int_0^\pi d\phi \quad (26)$$

Il est important de répéter que dans la configuration classique de l'argument optionnel, il n'est bien sûr pas obligatoire de renseigner les quatre clés citées. Il en va de même pour cette syntaxe. Puisque la clé `mode` n'est pas nécessaire pour `default`, cette partie peut être omise.



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

$$\int_1^2 \int_3^4 xy \, dx \, dy \quad (27)$$

Vous pouvez aussi toujours profiter du placement automatique du « $dx$ » avec cette syntaxe en faisant abstraction de [variables](#).

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

$$\int_1^{10} x \, dx \quad (28)$$

Un cas particulier notable de cette syntaxe spéciale est que si vous souhaitez modifier la variable sans spécifier les bornes, vous pouvez effectuer plus rapidement cette action de la façon suivante :

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

$$\int f(z) \, dz \quad (29)$$

Même si elle n'est pas recommandée.

## 5.2 Fonctionnement de la syntaxe

Du point de vue de l'utilisateur, il est relativement simple de choisir quelle syntaxe adopter. Toutefois, le package doit pouvoir distinguer les deux. Sans trop rentrer dans les détails<sup>2</sup> de l'implémentation, il est porté à votre intention que le package tente d'extraire la première clé de l'argument optionnel<sup>3</sup> et vérifie si elle existe. C'est pourquoi, si vous créez une clé avec `\NewSymbolKeyword` dont le nom pourra probablement servir de bornes d'intégrations, `intexrgal` peut faussement évaluer la nature de l'argument optionnel. Dans ce genre de situation, l'interprétation des clés sera erronée et le résultat incorrect.

2. Les plus courageux d'entre vous sont tout de même invités à lire le code source.

3. Ou du moins, le groupe de tokens qu'il pense correspondre à une clé.

## 6 Paramètres annexes

**\IntegralSetup** `\IntegralSetup {⟨liste de paramètres⟩}`

nouveau : v3.0.0  
mise à jour : v4.0.0

Cette macro permet, sous la syntaxe `⟨clé=valeur⟩`, de modifier des paramètres liés à certaines clés. Toutes les assignations effectuées sont locales et la macro peut donc être placée dans un groupe si besoin. Voici un exemple de la macro avec les assignations par défaut du package :

```
\IntegralSetup{
  diff-symb    = \l_@@_default_differential_symbol_tl, % voir au début de l
  'implémentation
  defaultvar   = {x},
  defaultvar*  = {r},
  varsep       = \thinmuskip,
  diffsep      = \thinmuskip,
  innersymbsep = {-5mu},
  postsymbsep  = {-1mu},
  vectorstyle  = \vec,
  domainstyle  = \mathbb,
  novar        = false
}
```

**diff-symb** `diff-symb=⟨séquence de contrôle⟩`

nouveau : v4.0.0

Cette clé contrôle le symbole utilisée pour les différentielles. Si l'option **italic** est utilisée, `\mathnormal` est appliqué (voir au début de l'implémentation). Contrairement à la clé homonyme d'`\integral`, celle-ci agit à un niveau plus global.

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

**defaultvar\***

nouveau : v3.0.0

Cette clé modifie la variable d'intégration insérée automatiquement lorsque **variables** n'est pas utilisée. L'argument peut tout à fait correspondre à un mot clé défini par `\NewVariableKeyword`. La variante étoilée de la clé contrôle la variable d'intégration à composer avec **diff-vec**.

**varsep** `varsep=⟨muexpr⟩`

nouveau : v4.0.0

Cette clé modifie l'espacement entre l'intégrande (ou le jacobien s'il est affiché) et les variables.

**diffsep** `diffsep=⟨muexpr⟩`

nouveau : v4.0.0

Cette clé modifie l'espacement entre les différentielles elles-mêmes.

**innersymbsep** `innersymbsep=⟨muexpr⟩`

nouveau : v4.0.0

Lorsque le mode **default** est en vigueur, c'est-à-dire que la clé **limits** est utilisée, le package insère un crénage entre les symboles pour légèrement les rapprocher. Cette clé permet donc de régler cet espacement.

**postsymbsep** `postsymbsep=⟨muexpr⟩`

nouveau : v4.0.0

Cette clé contrôle l'espacement entre le symbole et l'élément qui suit (intégrande ou différentielle selon **invert-limits**).

**vectorstyle** `vectorstyle=⟨séquence de contrôle⟩`

nouveau : v3.0.0

Cette clé modifie la macro à appliquer aux variables quand **diff-vec** est en action. Il est donc possible d'altérer le style du vecteur : gras, souligné, ou même un autre style d'un package particulier, **esvect** par exemple.

<u>domainstyle</u>	<code>domainstyle=</code> $\langle$ séquence de contrôle $\rangle$
<u>nouveau: v3.0.0</u>	Semblablement, on peut changer la macro à appliquer pour les clés <code>domain(*)</code> .
<u>novar</u>	<code>novar=</code> $\langle$ <code>true</code>   <code>false</code> $\rangle$
<u>nouveau: v3.0.0</u> <u>mise à jour: v4.0.0</u>	Lorsqu'une large partie du document nécessite de ne pas inclure les différentielles, il est possible de prolonger dans la durée l'action plus ponctuelle de <code>variables=none</code> .

# Historique des modifications

## 4.0.0 (06-06-2026)

### Ajouté

- ▶ Implémentation détaillée.
- ▶ Clés `varsep`, `diffsep`, `postsymbsep` et `diff-order`.
- ▶ Mot-clé de variable `cylindrical*`.

### Modifié

- ▶ Le package `derivative` n'est plus une dépendance. Les différentielles sont maintenant gérées de façon interne.
- ▶ Clé `symbolskip`, renommée en `innersymbsep`.
- ▶ Clé `hide-diff`, renommée en `novar`.

### Supprimé

- ▶ Clés `diff-star` et `diff-options`.

## 3.0.1 (02-01-2026)

### Corrigé

- ▶ Bug avec le jacobien dans la syntaxe spéciale ([problème #3](#)).
- ▶ Documentation anglaise et française ([problème #4](#), [problème #6](#) et [problème #7](#)).

### Modifié

- ▶ Les mots-clés `positive` et `real` contiennent maintenant un signe + ([problème #5](#)).

## 3.0.0 (24-12-2025)

### Ajouté

- ▶ Syntaxe spéciale.
- ▶ Clés `domain*` et `mode`.
- ▶ Macros `\IntegralSetup` et `\NewSymbolKeyword`.

### Supprimé

- ▶ Macros `\defaultdiff`, `\defaultdiff`, `\defaultvdiff` et `\vdiffstyle` en faveur de `\IntegralSetup`.
- ▶ Clés contrôlant symbole et borne, maintenant gérées à un plus haut niveau avec `\NewSymbolKeyword`.
- ▶ Clé `int-split`, en faveur de `mode`.
- ▶ Macro `\NewIntegralSymbol`.

### Modifié

- ▶ Les noms de quelques clés (`variable` en `variables`, `lower-lim` et `upper-lim` en `llimit` et `ulimit`, `int-symb` en `symbol`, `invert-differentials` et `hide-differentials` en `invert-diff` et `hide-diff`).

- ▶ `jacobian` et `diff-star` ne nécessitent plus de valeur booléenne. Il suffit maintenant de les écrire pour activer leurs fonctionnalités.
- ▶ `hide-diff` attribué à une option locale plutôt qu'à une option de package.
- ▶ `limits-mode` attribué à une option de package plutôt qu'à une clé de macro.

**v2.0.1** (13-09-2025)

#### Corrigé

- ▶ Problème de compatibilité entre `unicode-math` et `amssymb` selon l'ordre de chargement ([problème #2](#)).

**2.0.0** (09-09-2025)

#### Ajouté

- ▶ Macros `\intexgralsetup`, `\defaultdiff`, `\defaultvdiff` et `\vdiffstyle`.

#### Modifié

- ▶ Messages d'avertissement liés aux symboles non-existants. Ils ne sont maintenant provoqués que lorsque l'intégrale est composée.

#### Supprimé

- ▶ Clé `diff-vec-style` en faveur de `\vdiffstyle`.
- ▶ Message d'avertissement lié à la mauvaise utilisation des points-virgules en conjonction de la clé `int-split`.

#### Corrigé

- ▶ Bug où l'intégrande n'était pas réinitialisée lorsque `\integral` était employée successivement dans le même groupe  $\text{\TeX}$  ([détails ici](#)).
- ▶ Des restes de `log expl3` qui n'auraient pas dû être là.

**1.1.0** (29-07-2025)

#### Ajouté

- ▶ Variantes étoilées pour les clés contrôlant symbole et borne (clés `single`, `contour` etc).

**1.0.0** (26-07-2025) — Version initiale.

# PARTIE II

# IMPLÉMENTATION

## 1 Initialisation

```
1 <*package>
2 <@@=intexgral>
```

### 1.1 Déclaration du package

```
3 \NeedsTeXFormat{LaTeX2e}
4 \RequirePackage{expl3}[2025-05-14]
5
6 \def\intexgral@module{intexgral}
7 \def\intexgral@version{v4.0.0}
8 \def\intexgral@date{2026-05-26}
9 \def\intexgral@description{A LaTeX package for typesetting integrals}
10
11 \ProvidesExplPackage
12   \intexgral@module
13   \intexgral@date
14   \intexgral@version
15   \intexgral@description
```

### 1.2 Vérifications préliminaires

On vérifie que la version d'expl3 soit suffisamment récente avant de poursuivre. Le package requiert la macro `\regex_if_match:nnTF`, précédemment nommée `\regex_match:nnTF` dans les versions antérieures à mai 2025.

```
16 \msg_new:nnn { intexgral } { expl3-too-old }
17 {
18   Your~expl3~installation~is~too~old,~
19   "intexgral"~requires~expl3~dated~2025-05-14~or~later.\iow_newline:
20   Package~loading~has~been~aborted.\iow_newline:
21   \msg_see_documentation_text:n { intexgral }
22 }
23
24 \@ifpackagelater{expl3}{2025-05-14}{}
25 { \msg_critical:nn { intexgral } { expl3-too-old } }
```

Puisqu'*intexgral* utilise `\mathbb` pour les clés `domain(*)`, on doit s'assurer que la macro existe. La vérification est placée dans un *hook* exécuté au début du document au cas où *amsfonts* ou tout package définissant `\mathbb` serait chargé après *intexgral*. Cela sert notamment à éviter les conflits entre packages (*unicode-math* et *amssymb* par exemple).

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

## 2 Options de package

### 2.1 Définition des variables

`\l_@@_invert_limits_bool`

Booléen servant à inverser la convention d'ordre des bornes.

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

`\l_@@_deactivate_variables_bool`

Détermine si les différentielles doivent être affichées ou non.

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

`\l_@@_invert_differential_bool`

Détermine l'ordre d'affichage des différentielles.

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

`\l_@@_default_differential_symbol_tl`

Contient le symbole par défaut pour les différentielles.

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

`\l_@@_limits_style_tl`

Contient la primitive  $\TeX$  à utiliser pour le style des bornes.

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

### 2.2 Déclaration des options

On peut dès à présent définir les options de package avant de les charger.

```
33 \keys_define:nn { intexgral }
34 {
35   invert-limits .bool_set:N = \l__intexgral_invert_limits_bool,
36   invert-limits .usage:n   = { preamble },
37   invert-limits .initial:n  = { false },
38
39   invert-diff .bool_set:N = \l__intexgral_invert_differential_bool,
40   invert-diff .usage:n   = { preamble },
41   invert-diff .initial:n  = { false },
42
43   limits-mode .choice:,
44   limits-mode / limits .code:n =
45     { \tl_set_eq:NN \l__intexgral_limits_style_tl \tex_limits:D },
46   limits-mode / nolimits .code:n =
47     { \tl_set_eq:NN \l__intexgral_limits_style_tl \tex_nolimits:D },
48
49   limits-mode .usage:n   = { preamble },
50   limits-mode .initial:n = { nolimits },
51
52   italic .choice:,
53   italic / true .code:n =
54     {
55       \tl_set:Nn \l__intexgral_default_differential_symbol_tl
```

```

56         { \mathnormal{d} }
57     },
58     italic / false .code:n =
59     {
60         \tl_set:Nn \l__intexgral_default_differential_symbol_tl
61         { \mathrm{d} }
62     },
63     italic .usage:n = { preamble },
64     italic .initial:n = { false },
65
66     upright .choice:,
67     upright / true .code:n =
68     {
69         \tl_set:Nn \l__intexgral_default_differential_symbol_tl
70         { \mathrm{d} }
71     },
72     upright / false .code:n =
73     {
74         \tl_set:Nn \l__intexgral_default_differential_symbol_tl
75         { \mathnormal{d} }
76     },
77     upright .usage:n = { preamble },
78     upright .initial:n = { true }
79 }
80
81 \ProcessKeyOptions[intexgral]

```

### 3 Messages

La partie suivante de l'implémentation définit l'ensemble des messages d'avertissement et d'erreur.

#### \g\_@@\_integral\_number\_int

On commence par créer un compteur global servant à numéroté les intégrales composées dans le document.

```

82 \int_gzero_new:N \g__intexgral_integral_number_int

```

#### \@@\_warning\_msg\_header: ★

On peut dès lors définir une macro servant d'en-tête aux messages d'avertissement. Celle-ci inclut le compteur vu juste avant.

```

83 \cs_new:Nn \__intexgral_warning_msg_header: {
84     (
85         integral~no.~
86         \int_use:N\g__intexgral_integral_number_int\c_space_tl
87         \msg_line_context:
88     )
89     \iow_newline:
90 }

```

Tous les messages sont maintenant définis.

```

91 \msg_new:nnnn { intexgral } { symb-key-alr-def }
92 { Symbol~key~"\tl_trim_spaces:n{#1}"~is~already~defined. }
93 { Use~\token_to_str:N \RenewSymbolKeyword\ instead. }
94

```



```

95 \msg_new:nnnn { intexgral } { symb-key-not-def }
96 { Symbol~key~"\tl_trim_spaces:n{#1}"~is~not~defined. }
97 { Use~\token_to_str:N \NewSymbolKeyword\ instead. }
98
99 \msg_new:nnnn { intexgral } { diff-group-alr-def }
100 { Differential~group~"\tl_trim_spaces:n{#1}"~is~already~defined. }
101 { Use~\token_to_str:N \RenewVarKeyword\ instead. }
102
103 \msg_new:nnnn { intexgral } { diff-group-not-def }
104 { Differential~group~"\tl_trim_spaces:n{#1}"~is~not~defined. }
105 { Use~\token_to_str:N \NewVarKeyword\ instead. }
106
107 \msg_new:nnnn { intexgral } { lim-group-alr-def }
108 { Limits~group~"\tl_trim_spaces:n{#1}"~is~already~defined. }
109 { Use~\token_to_str:N \RenewLimitsKeyword\ instead. }
110
111 \msg_new:nnnn { intexgral } { lim-group-not-def }
112 { Limits~group~"\tl_trim_spaces:n{#1}"~is~not~defined. }
113 { Use~\NewLimitsKeyword\ instead. }
114
115 \msg_new:nnn { intexgral } { key-exists-for-lim }
116 {
117   Symbol~key~already~defined~with~
118   \token_to_str:N \NewLimitsKeyword\ \__intexgral_warning_msg_header:
119   Key~"#1"~already~exists~for~predefined~limits.~
120   This~can~disrupt~the~functioning~of~the~special~syntax.
121 }
122
123 \msg_new:nnn { intexgral } { unknown-symb }
124 {
125   Unknown~integral~symbol~\__intexgral_warning_msg_header:
126   The~symbol~"\tl_trim_spaces:n{#1}"~has~been~replaced~with~
127   \token_to_str:N\int.
128 }
129
130 \msg_new:nnn { intexgral } { no-jacobian }
131 {
132   Jacobian~unavailable~\__intexgral_warning_msg_header:
133   A~Jacobian~was~requested~for~the~"#1"~keyword,
134   ~but~none~was~declared~for~the~latter.
135 }
136
137 \msg_new:nnn { intexgral } { use-glyph-instead }
138 {
139   Consider~using~the~dedicated~glyph~\__intexgral_warning_msg_header:
140   The~key~"nint"~was~used~with~fewer~than~
141   5~regular~integral~signs.
142 }

```

## 4 Variantes utiles

---

```
\regex_if_match:nVTF
\str_if_eq:neTF
\str_if_eq:neF
\keys_if_exist:nVTF
\seq_use:Ne
\str_if_eq_p:en
\prop_gput:Nne
\seq_gset_split:cnn
\seq_set_split:Nnf
```

---

```
143 \cs_generate_variant:Nn \regex_if_match:nnTF { nVTF }
144 \cs_generate_variant:Nn \str_if_eq:nnTF      { neTF }
145 \cs_generate_variant:Nn \str_if_eq:nnF       { neF }
146 \cs_generate_variant:Nn \keys_if_exist:nnTF  { nVTF }
147 \cs_generate_variant:Nn \seq_use:Nn         { Ne }
148 \cs_generate_variant:Nn \str_if_eq_p:nn      { en }
149 \cs_generate_variant:Nn \prop_gput:Nnn       { Nne }
150 \cs_generate_variant:Nn \seq_gset_split:Nnn  { cnn }
151 \cs_generate_variant:Nn \seq_set_split:Nnn   { Nnf }
```

## 5 Variables internes

### 5.1 Tokens

---

```
\l_@@_integrand_tl
```

---

Token contenant l'intégrande. Il est utilisé tel quel pour le mode **default** tandis que pour les autres modes, il est séparé ultérieurement en une séquence.

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

---

```
\l_@@_integral_symbol_tl
```

---

Token contenant le symbole de l'intégrale. Lorsque non modifié par la clé **symbol**, il faut que le token soit tout de même initialisé à **\int**.

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

---

```
\l_@@_lower_limit_tl
\l_@@_upper_limit_tl
```

---

Tokens contenant les bornes d'intégration.

```
155 \tl_new:N \l__intexgral_lower_limit_tl
156 \tl_new:N \l__intexgral_upper_limit_tl
```

---

```
\l_@@_differential_symbol_tl
```

---

Token contenant le symbole des différentielles. Il contient **\l\_@@\_default\_differential\_symbol\_tl**.

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

---

`\l_@@_default_differential_tl`  
`\l_@@_default_vector_differential_tl`

Tokens contenant les différentielles par défaut, utilisés lorsque la clé `variables` n'est pas renseignée.

```
158 \tl_new:N \l__intexgral_default_differential_tl
159 \tl_new:N \l__intexgral_default_vector_differential_tl
```

---

`\l_@@_vectorial_differential_style_tl`  
`\l_@@_domain_style_tl`

Tokens contenant les macros à appliquer aux clés `diff-vec` et `domain(*)`.

```
160 \tl_new:N \l__intexgral_vectorial_differential_style_tl
161 \tl_new:N \l__intexgral_domain_style_tl
```

---

`\l_@@_domain_char_tl`  
`\l_@@_domain_dimension_tl`

Token utilisés pour stocker respectivement le caractère et la dimension d'un domaine d'intégration.

```
162 \tl_new:N \l__intexgral_domain_char_tl
163 \tl_new:N \l__intexgral_domain_dimension_tl
```

---

`\c_@@_left_bracket_tl`  
`\c_@@_right_bracket_tl`

Tokens constants pour la clé `limits*`.

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

---

`\l_@@_key_name_tl`

Token contenant le potentiel nom d'une clé, utilisé pour différencier la syntaxe `<clé=valeur>` de la syntaxe spéciale.

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

## 5.2 Booléens

---

`\l_@@_manual_differential_bool`

Booléen servant à déterminer si les différentielles sont insérées par le biais de `\differentials`.

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

---

`\l_@@_custom_variables_bool`

Booléen servant à déterminer si les variables sont définies avec `variables`.

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

---

`\l_@@_separate_integral_bool`

Booléen servant à déterminer si l'intégrale est *séparée* en plusieurs parties, c'est-à-dire si les modes `nested` ou `product` sont en vigueur.

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

---

\l\_@@\_vectorial\_differential\_bool

Booléen décidant s'il faut appliquer des vecteurs aux différentielles.

170 \bool\_new:N \l\_\_intexgral\_vectorial\_differential\_bool

---

\l\_@@\_jacobian\_bool

Booléen décidant s'il faut afficher le jacobien ou non.

171 \bool\_new:N \l\_\_intexgral\_jacobian\_bool

## 5.3 Séquences et clist

---

\l\_@@\_domain\_seq

Séquence contenant les domaines d'intégration et leurs dimensions (les séquences principales seront expliquées plus tard).

172 \seq\_new:N \l\_\_intexgral\_domain\_seq

---

\l\_@@\_differential\_order\_clist

Liste contenant les puissances des différentielles (clist et non séquence car `.seq_set:N` n'existe pas pour les clés).

173 \clist\_new:N \l\_\_intexgral\_differential\_order\_clist

## 5.4 Liste de propriétés

---

\g\_@@\_limits\_keyword\_prop

\g\_@@\_differential\_group\_keyword\_prop

Une liste de propriété est créée pour stocker d'un côté les mots clés propres aux bornes, et de l'autre, ceux pour les différentielles.

174 \prop\_new:N \g\_\_intexgral\_limits\_keyword\_prop

175 \prop\_new:N \g\_\_intexgral\_differential\_group\_keyword\_prop

## 5.5 Muskip

---

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

---

Muskips internes pour stocker les espacements avec respectivement:

1. L'espacement entre les symboles d'intégration lorsque plusieurs sont générés par `limits(*)`.
2. L'espacement qui suit le symbole.
3. L'espacement entre les différentielles lorsque la clé `variables` est utilisée.
4. L'espacement entre les différentielles et l'intégrande.

```
176 \muskip_new:N \l__intexgral_symbol_inner_sep_muskip  
177 \muskip_new:N \l__intexgral_symbol_post_sep_muskip  
178 \muskip_new:N \l__intexgral_variable_sep_muskip  
179 \muskip_new:N \l__intexgral_differential_sep_muskip
```

## 5.6 String

---

```
\l_@@_display_mode_str
```

---

Variable interne pour stocker le mode d'affichage de l'intégrale. La raison d'utiliser une *string* et non pas un token s'explique par le manque d'un `\tl_case:nnTF`

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

## 6 Macros et séquences auxiliaires

On définit ici les deux primitives  $\TeX$  `\mkern` et `\mathchoice` avec une syntaxe *expl3*.

---

```
\l_@@_mkern:N \l_@@_mkern:N {muskip}
```

---

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

---

```
\l_@@_mathchoice:nnnn \l_@@_mathchoice:nnnn {<display style>} {<inline style>} {<script style>}  
{<scriptscript style>}
```

---

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

---

```
\l_@@_invert_limits:w \l_@@_invert_limits:w {borne inférieure}, {borne supérieure} \q_stop
```

---

Macro inversant les limites pour l'option `invert-limits`. L'action est purement développable et plus efficace qu'un `\seq_inverse:N` dans le contexte du package, étant donné qu'au maximum deux éléments sont présents.

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

### \@@\_general\_integral\_symbol:

On crée une macro qui contient l'expression générale d'une intégrale: symbole, style des bornes, borne inférieure et borne supérieure. Il ne reste plus qu'à assigner les tokens pour composer un symbole complet, et éventuellement répéter la macro pour des intégrales définies sur plusieurs variables.

```
185 \cs_new:Npn \__intexgral_general_integral_symbol:
186 {
187   \l__intexgral_integral_symbol_tl
188   \l__intexgral_limits_style_tl
189   \c_math_subscript_token
190   { \l__intexgral_lower_limit_tl }
191   \c_math_superscript_token
192   { \l__intexgral_upper_limit_tl }
193 }
```

## 6.1 Séquences principales

Au vu du grand nombre de clés existantes et de combinaisons possibles, il faut judicieusement organiser et stocker les tokens afin de composer l'intégrale finale. L'approche adoptée a été de créer des séquences dédiées à chaque élément d'une intégrale:

### \l\_@@\_integral\_symbol\_seq

Une pour le symbole (ou *les* symboles, si la clé **limits\*** est en usage).

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

### \l\_@@\_integrand\_seq

Une pour l'intégrande (ou *les* intégrandes, si découpée en plusieurs parties par **nested** ou **product**).

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

### \l\_@@\_jacobian\_seq

Une pour le jacobien.

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

### \l\_@@\_differential\_seq

Et enfin, une pour les différentielles.

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

De cette façon, on peut construire l'intégrale finale en jouant sur la façon d'extraire les différents éléments de ces séquences. En voici une illustration plus concrète:  
Pour une intégrale indéfinie.

$$\boxed{\int\!\!\int_S} \boxed{f(x,y)} \boxed{dx\,dy}$$

Pour une intégrale définie.

$$\boxed{\int_a^b} \boxed{\int_c^d} \boxed{f(x)} \boxed{g(y)} \boxed{dx} \boxed{dy}$$

## 6.2 Séquences subsidiaires

Quelques séquences supplémentaires seront nécessaires durant les étapes intermédiaires qui consisteront à préparer les quatre séquences principales listées ci-dessus.

`\l_@@_variables_seq`

Tout d'abord, une séquence contenant les variables d'intégration.

```
198 \seq_new:N \l__integragl_variables_seq
```

`\l_@@_limits_seq`

Ensuite, une séquence contenant les paires de limites d'intégration, c'est-à-dire les éléments de la clé `limits` séparés en deux par le point-virgule.

```
199 \seq_new:N \l__integragl_limits_seq
```

`\l_@@_upper_limits_seq`

`\l_@@_lower_limits_seq`

Deux séquences contenant respectivement les bornes supérieures et inférieures. Ainsi, si la clé `limits` contient les éléments `a ; b`, `c ; d` et `e ; f`, alors la séquence des bornes inférieures contiendra les éléments `a`, `c` et `e`, tandis que celle des bornes supérieures contiendra les éléments `b`, `d` et `f`.

```
200 \seq_new:N \l__integragl_upper_limits_seq
```

```
201 \seq_new:N \l__integragl_lower_limits_seq
```

## 7 Bornes d'intégration

`@@_parse_limits:n` `@@_parse_limits:n {<borne inférieure>} {< borne supérieure>}, <mot-clé>`

Cette macro vérifie si son argument correspond à un mot-clé défini pour les bornes, auquel cas elle se développe en son contenu en l'inversant si besoin (selon le statut de `\l_@@_invert_limits_bool`). Sinon, la macro se développe simplement en son argument.

```
202 \cs_new:Npn \__integragl_parse_limits:n #1
203 {
204   \prop_if_in:NnTF \g__integragl_limits_keyword_prop { #1 }
205   {
206     \bool_if:NTF \l__integragl_invert_limits_bool
207     {
208       \exp_last_unbraced:Ne \__integragl_invert_limits:w
209       { \prop_item:Nn \g__integragl_limits_keyword_prop { #1 } }
210       \q_stop
211     }
212     { \prop_item:Nn \g__integragl_limits_keyword_prop { #1 } }
213   }
214   { #1 }
215 }
```

### \@@\_set\_limits\_regular:

Une fois que l'argument de la clé `limits` a été converti en une séquence (action effectuée au sein de `.code:n` dans déclaration des clés), on assigne chaque paire de borne à une séquence temporaire. Celle-ci est créée de sorte à ce que si un mot-clé est présent, il est substitué à ses bornes correspondantes en vertu de la macro précédente. L'action est exécutée dans un développement de type `\romannumeral` afin d'éviter le développement des bornes contenant des macros fragiles (type `\mathbb`). On peut alors extraire la borne inférieure et la borne supérieure de cette séquence pour les assigner à leur séquence respective.

```
216 \cs_new_protected:Nn \__intexgral_set_limits_regular:
217 {
218   \seq_map_inline:Nn \l__intexgral_limits_seq
219   {
220     \seq_set_split:Nnf \l_tmpa_seq { , }
221     { \__intexgral_parse_limits:n { ##1 } }
222     \seq_put_right:Ne \l__intexgral_lower_limits_seq
223     { \seq_item:Nn \l_tmpa_seq { 1 } }
224     \seq_put_right:Ne \l__intexgral_upper_limits_seq
225     { \seq_item:Nn \l_tmpa_seq { 2 } }
226   }
227 }
```

### \@@\_set\_limits\_starred:

La macro suivante est similaire à la précédente, mais elle s'occupe des bornes sous forme d'intervalle lorsque la clé `limits*` est utilisée.

```
228 \cs_new_protected:Npn \__intexgral_set_limits_starred:
229 {
230   \seq_map_indexed_inline:Nn \l__intexgral_limits_seq
231   {
232     \seq_set_split:Nnf \l_tmpa_seq { , }
233     { \__intexgral_parse_limits:n { ##2 } }
234     \bool_if:NTF \l__intexgral_invert_limits_bool
235     {
236       \tl_set:Ne \l__intexgral_lower_limit_tl
237       {
238         \seq_item:Nn \l_tmpa_seq { 2 }
239         \tex_mathpunct:D,
240         \seq_item:Nn \l_tmpa_seq { 1 }
241       }
242     }
243     {
244       \tl_set:Ne \l__intexgral_lower_limit_tl
245       {
246         \seq_item:Nn \l_tmpa_seq { 1 }
247         \tex_mathpunct:D,
248         \seq_item:Nn \l_tmpa_seq { 2 }
249       }
250     }
251     \bool_if:NTF \l__intexgral_invert_limits_bool
252     { \seq_put_right:Ne \l__intexgral_upper_limits_seq }
253     { \seq_put_right:Ne \l__intexgral_lower_limits_seq }
254   }
255   \str_case_e:nnF { \seq_item:Nn \l_tmpa_seq { 1 } }
256   {
```



```

257         { -\infty } { \tex_left:D \c__intexgral_right_bracket_tl }
258     }
259     { \tex_left:D \c__intexgral_left_bracket_tl }
260     \tl_use:N \l__intexgral_lower_limit_tl
261     \str_case_e:nnF { \seq_item:Nn \l_tmpa_seq { 2 } }
262     {
263         { +\infty } { \tex_right:D \c__intexgral_left_bracket_tl }
264     }
265     { \tex_right:D \c__intexgral_right_bracket_tl }
266 }
267 }
268 }

```

---

**`\@@_set_limits:nn`** `\@@_set_limits:nn {<borne inférieure>} {<borne supérieure>}`

Selon l'option **`invert-limits`**, cette macro assigne les bornes inférieures et supérieures aux tokens dédiés. Si elle est active, l'assignation devient quelque peu contre-intuitive (la borne inférieure est assignée à `\l__@@_upper_limit_tl` et inversement), mais produira tout de même le résultat attendu. Le développement des bornes est empêché afin d'éviter les problèmes de macros fragiles comme vu précédemment.

```

269 \cs_new_protected:Npn \__intexgral_set_limits:nn #1#2
270 {
271     \bool_if:NTF \l__intexgral_invert_limits_bool
272     {
273         \tl_set:Nc \l__intexgral_lower_limit_tl { \exp_not:n { #2 } }
274         \tl_set:Nc \l__intexgral_upper_limit_tl { \exp_not:n { #1 } }
275     }
276     {
277         \tl_set:Nc \l__intexgral_lower_limit_tl { \exp_not:n { #1 } }
278         \tl_set:Nc \l__intexgral_upper_limit_tl { \exp_not:n { #2 } }
279     }
280 }

```

---

**`\@@_generate_integral_sequence:`**

Une fois les séquences des bornes inférieures et supérieures remplies, on peut utiliser la structure des séquences à notre avantage pour générer automatiquement autant de symbole que nécessaire. On procède à une boucle incrémentée dans laquelle on effectue deux actions:

1. À l'aide de **`\@@_set_limits:nn`**, on assigne les bornes d'intégrations aux tokens `\l__@@_lower_limit_tl` et `\l__@@_upper_limit_tl` à partir du  $i^{\text{e}}$  élément des séquences `\l__@@_lower_limits_seq` et `\l__@@_upper_limits_seq` respectivement.
2. On place à droite (c.-à-d. dans l'ordre) de la séquence `\l__@@_integral_symbol_seq` la macro **`\@@_general_integral_symbol:`**, qui contient la forme générale d'une intégrale. À noter qu'on développe au maximum son contenu afin de récupérer l'assignation immédiate des tokens.

```

281 \cs_new_protected:Nn \__intexgral_generate_integral_sequence:
282 {
283     \int_set:Nn \l_tmpa_int
284     {

```

```

285     \seq_if_empty:NTF \l__intexgral_lower_limits_seq
286     { \seq_count:N \l__intexgral_upper_limits_seq }
287     { \seq_count:N \l__intexgral_lower_limits_seq }
288   }
289   \int_step_inline:nn { \l_tmpa_int }
290   {
291     \__intexgral_set_limits:nn
292     { \seq_item:Nn \l__intexgral_lower_limits_seq { ##1 } }
293     { \seq_item:Nn \l__intexgral_upper_limits_seq { ##1 } }
294     \seq_put_right:Ne \l__intexgral_integral_symbol_seq
295     { \__intexgral_general_integral_symbol: }
296   }
297 }

```

## 8 Variables

---

**`\@@_parse_variables:n`** `\@@_parse_variables:n {<variable d'intégration>}, <mot-clé>`

Cette macro s'applique à chaque élément de l'argument de la clé **variables** pour vérifier s'il s'agit d'un mot-clé ou d'une liste explicite, de la même manière que pour les limites. Il s'agit principalement de transférer les éléments de `\l__@@_variables_seq` (contenant par exemple « $x, y, z$ »») vers `\l__@@_differential_seq` (qui contiendra « $dx, dy, dz$ »); tout en prenant en compte les différentes options qui ont été appliquées. Dans le cas où aucune variable n'est saisie, la macro s'occupe d'y placer  $x$  ou  $\vec{r}$  en fonction de la valeur du booléen `\l__@@_vectorial_differential_bool`.

```

298 \cs_new_protected:Nn \__intexgral_parse_variables:
299 {
300   \bool_if:NTF \l__intexgral_custom_variables_bool
301   {
302     \seq_map_indexed_inline:Nn \l__intexgral_variables_seq
303     {
304       \seq_put_right:Ne \l__intexgral_differential_seq
305       {
306         \exp_not:V \l__intexgral_differential_symbol_tl
307         \clist_if_empty:NF \l__intexgral_differential_order_clist
308         {
309           \c_math_superscript_token
310           {
311             \clist_item:Nn
312             \l__intexgral_differential_order_clist
313             { ##1 }
314           }
315         }
316         \bool_if:NT \l__intexgral_vectorial_differential_bool
317         { \exp_not:V \l__intexgral_vectorial_differential_style_tl }
318         { ##2 }
319       }
320     }
321   }
322   {
323     \seq_put_right:Ne \l__intexgral_differential_seq
324     {
325       \exp_not:V \l__intexgral_differential_symbol_tl
326       \clist_if_empty:NF \l__intexgral_differential_order_clist
327       {

```

```

328         % TODO: permettre de soumettre une séquence en variables par défaut
329         \c_math_superscript_token
330         {
331             \clist_item:Nn
332             \l__intexgral_differential_order_clist
333             { 1 }
334         }
335     }
336     \bool_if:NTF \l__intexgral_vectorial_differential_bool
337     {
338         \exp_not:V \l__intexgral_vectorial_differential_style_tl
339         {\l__intexgral_default_vector_differential_tl}
340     }
341     { \l__intexgral_default_differential_tl }
342 }
343 }
344 }

```

## 9 Syntaxe spéciale

---

\@@\_retrieve\_key\_name:w ★ \@@\_retrieve\_key\_name:w <clé>=<valeur> \q\_stop

Afin de gérer la syntaxe spéciale, on commence par créer une macro délimitée qui extrait la valeur de la clé et son nom, et qui ne renvoie que ce dernier.

```

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

```

---

\@@\_extract\_first\_key\_name:n \@@\_extract\_first\_key\_name:n {<argument optionnel  
d'*\integral*>}

On associe ensuite l'argument optionnel reçu par *\integral* à une *comma list* temporaire, dont on extrait le premier élément; on vérifie alors s'il contient un signe «=». Le cas échéant, on emploie la macro précédente pour ne récupérer que le nom de la clé; sinon, on copie simplement le groupe de tokens récupéré dans *\l\_\_@@\_key\_name\_tl*.

```

346 \cs_new_protected:Npn \__intexgral_extract_first_key_name:n #1
347 {
348     \seq_set_split:Nnn \l_tmpa_seq { : } { #1 }
349     \tl_set:Ne \l_tmpa_tl { \seq_item:Nn \l_tmpa_seq { 1 } }
350     \tl_if_in:NnTF \l_tmpa_tl { = }
351     {
352         \tl_set:Ne \l__intexgral_key_name_tl
353         {
354             \exp_last_unbraced:NV
355             \__intexgral_retrieve_key_name:w \l_tmpa_tl \q_stop
356         }
357     }
358     { \tl_set_eq:NN \l__intexgral_key_name_tl \l_tmpa_tl }
359 }

```

```
\@@_parse_integral_keys:n \@@_parse_integral_keys:n {<première clé de l'argument  
optionnel>}
```

En se servant du module *l3keys*, on peut évaluer si le groupe de tokens capturé correspond à une clé définie, auquel cas on applique directement `\keys_set:nn`. Si le test se révèle négatif, on assigne manuellement les clés en considérant l'argument optionnel comme une séquence à délimiter avec le deux-point. Des séquences temporaires sont employées pour manipuler les différentes parties de l'argument. On vérifie notamment la présence du token `+j` pour activer la clé *jacobian*, et on ajoute la variable par défaut si elle n'est pas spécifiée. Enfin, on assigne les valeurs aux clés *limits*, *variables* et *mode* en fonction des éléments extraits.

```
360 \cs_new_protected:Npn \__intexgral_parse_integral_keys:n #1
361 {
362   \keys_if_exist:nVTF { integral } \l__intexgral_key_name_tl
363   { \keys_set:nn { integral } { #1 } }
364   {
365     \regex_split:nnN { : } { #1 } \l_tmpa_seq
366     \str_if_eq:eeTF
367     { \seq_item:Nn \l_tmpa_seq { 2 } }
368     { +j }
369     {
370       \exp_args:NNne \seq_set_item:Nnn \l_tmpa_seq { 2 }
371       { \l__intexgral_default_differential_tl+j }
372     }
373     {
374       \int_compare:nNnT { \seq_count:N \l_tmpa_seq } < { 2 }
375       {
376         \seq_put_right:NV
377         \l_tmpa_seq
378         \l__intexgral_default_differential_tl
379       }
380     }
381     \int_compare:nNnT { \seq_count:N \l_tmpa_seq } < { 3 }
382     { \seq_put_right:Nn \l_tmpa_seq { d } }
383     \seq_set_split:Nne \l_tmpb_seq
384     { + }
385     { \seq_item:Nn \l_tmpa_seq { 2 } }
386     \keys_set:ne { integral }
387     {
388       limits    = { \seq_item:Nn \l_tmpa_seq { 1 } },
389       variables = { \seq_item:Nn \l_tmpb_seq { 1 } },
390       mode      =
391       {
392         \str_case:enF { \seq_item:Nn \l_tmpa_seq { 3 } }
393         {
394           { d      } { default }
395           { n      } { nested  }
396           { p      } { product }
397           { default } { default }
398           { nested } { nested }
399           { product } { product }
400         }
401         { default }
402       },
403       \bool_if:nT
404       {
405         \int_compare_p:nNn { \seq_count:N \l_tmpb_seq } > { 1 }
```

```

406          &&
407          \str_if_eq_p:en { \seq_item:Nn \l_tmpb_seq { 2 } } { j }
408        }
409        { jacobian }
410      }
411    }
412  }

```

## 10 Composition de l'intégrale

### \@@\_integral\_preconfiguration:

Avant de pouvoir composer l'intégrale dans le document, un certain nombre de vérifications doit être effectué. En effet, selon la combinaison de clés renseignée dans l'argument optionel, les séquences ne sont pas nécessairement encore remplies. On effectue donc l'ensemble des vérifications suivantes:

- On regarde si l'intégrande doit être séparé en plusieurs parties par les modes **nested** et **product**, auquel cas on remplit la séquence de l'intégrande en séparant les éléments à partir du point-virgule. Sinon, on ajoute simplement l'intégrande (`\l_@@_integrand_tl`) à la séquence.
- On vérifie si la séquence des symboles est vide, signe que la clé **limits\*** n'a pas été appelée. En effet, seul l'utilisation de cette clé engendre l'exécution de toutes les macros présentées en section 7. Dans ce cas là, on ajoute simplement à la séquence la forme générale de l'intégrale, dont le symbole et les bornes seront éventuellement modifiés par les clés **symbol**, **ulimit** et **llimit**.
- On contrôle ensuite la désactivation des différentielles en plus de détecter la présence de **\differentials** à l'aide d'une expression régulière.

```

413 \cs_new_protected:Nn \__intexgral_integral_preconfiguration:
414 {
415   \bool_if:NTF \l__intexgral_separate_integral_bool
416   {
417     \seq_set_split:NnV
418       \l__intexgral_integrand_seq
419       { ; }
420       \l__intexgral_integrand_tl
421   }
422   {
423     \seq_put_right:NV
424       \l__intexgral_integrand_seq
425       \l__intexgral_integrand_tl
426   }
427
428   \seq_if_empty:NT \l__intexgral_integral_symbol_seq
429   {
430     \seq_put_right:Nn \l__intexgral_integral_symbol_seq
431       { \__intexgral_general_integral_symbol: }
432   }
433
434   \bool_if:NF \l__intexgral_deactivate_variables_bool
435   { \__intexgral_parse_variables: }
436

```

```

437 \regex_if_match:nVTF { \c{differentials} } \l__intexgral_integrand_tl
438 { \bool_set_true:N \l__intexgral_manual_differential_bool }
439 { \bool_set_false:N \l__intexgral_manual_differential_bool }
440 }

```

On peut désormais définir l'ensemble des six macros qui serviront à composer une intégrale précise; trois modes autorisés par `mode`, chacun accompagné d'une variante pour les différentielles inversées. On notera par ailleurs que `\differentials` n'est créé que si le booléen `\l__manual_differential_bool` est vrai, afin d'éviter des définitions inutiles.

## 10.1 Mode *default*

### `\@@_print_default_integral:`

Pour le mode par défaut on compose dans l'ordre: symbole intégrale, intégrande, jacobien (si applicable), et différentielles. Puisque les éléments sont successifs, on applique `\seq_use:Nn` pour chacune des séquences en insérant un crénage lorsque c'est nécessaire. À noter que si `\l__vectorial_differential_bool` est vrai, on ajoute un point médian et supprime le crénage précédant (pour des raisons d'espacement propres à `\cdot`).

```

441 \cs_new_protected:Npn \__intexgral_print_default_integral:
442 {
443   \bool_if:NT \l__intexgral_manual_differential_bool
444   {
445     \cs_set_protected:Npn \differentials
446     {
447       \seq_use:Nn \l__intexgral_differential_seq
448       { \__intexgral_mkern:N \l__intexgral_differential_sep_muskip }
449     }
450   }
451   \seq_use:Nn \l__intexgral_integral_symbol_seq
452   { \__intexgral_mkern:N \l__intexgral_symbol_inner_sep_muskip }
453   \__intexgral_mkern:N \l__intexgral_symbol_post_sep_muskip
454   \seq_use:Nn \l__intexgral_integrand_seq { }
455   \bool_if:NT \l__intexgral_jacobian_bool
456   { \seq_use:Nn \l__intexgral_jacobian_seq { } }
457   \__intexgral_mkern:N \l__intexgral_variable_sep_muskip
458   \bool_if:NT \l__intexgral_vectorial_differential_bool
459   { \tex_unkern:D \cdot}
460   \bool_if:NF \l__intexgral_manual_differential_bool
461   {
462     \seq_use:Nn \l__intexgral_differential_seq
463     { \__intexgral_mkern:N \l__intexgral_differential_sep_muskip }
464   }
465 }

```

### `\@@_print_default_integral_inv:`

Pour les différentielles inversées, rien de bien différent, si ce n'est que `\l__integrand_seq` et `\l__differential_seq` sont interverties.

```

466 \cs_new_protected:Npn \__intexgral_print_default_integral_inv:
467 {
468   \bool_if:NT \l__intexgral_manual_differential_bool
469   {

```

```

470 \cs_set_protected:Npn \differentials
471 {
472   \seq_use:Nn \l__intexgral_differential_seq
473   { \__intexgral_mkern:N \l__intexgral_differential_sep_muskip }
474 }
475 }
476 \seq_use:Nn \l__intexgral_integral_symbol_seq
477 { \__intexgral_mkern:N \l__intexgral_symbol_inner_sep_muskip }
478 \__intexgral_mkern:N \l__intexgral_symbol_post_sep_muskip
479 \bool_if:NF \l__intexgral_manual_differential_bool
480 {
481   \seq_use:Nn \l__intexgral_differential_seq
482   { \__intexgral_mkern:N \l__intexgral_differential_sep_muskip }
483 }
484 \__intexgral_mkern:N \l__intexgral_variable_sep_muskip
485 \bool_if:NT \l__intexgral_vectorial_differential_bool
486 { \tex_unkern:D \cdot}
487 \seq_use:Nn \l__intexgral_integrand_seq { }
488 \bool_if:NT \l__intexgral_jacobian_bool
489 { \seq_use:Nn \l__intexgral_jacobian_seq { } }
490 }

```

## 10.2 Mode *nested*

### \@@\_print\_nested\_integral:

Dans le mode imbriqué, on initie une boucle incrémentée où l'on extrait le  $i^{\text{e}}$  élément de chacune des séquences du symbole, de l'intégrande et du jacobien. Après l'itération, on compose toutes les différentielles à nouveau à l'aide de `\seq_use:Nn`.

```

491 \cs_new_protected:Npn \__intexgral_print_nested_integral:
492 {
493   \bool_if:NT \l__intexgral_manual_differential_bool
494   {
495     \cs_set_protected:Npn \differentials
496     {
497       \seq_use:Nn \l__intexgral_differential_seq
498       { \__intexgral_mkern:N \l__intexgral_differential_sep_muskip }
499     }
500   }
501   \int_step_inline:nn { \seq_count:N \l__intexgral_integral_symbol_seq }
502   {
503     \seq_item:Nn \l__intexgral_integral_symbol_seq { ##1 }
504     \__intexgral_mkern:N \l__intexgral_symbol_post_sep_muskip
505     \seq_item:Nn \l__intexgral_integrand_seq { ##1 }
506     \bool_if:NT \l__intexgral_jacobian_bool
507     { \seq_item:Nn \l__intexgral_jacobian_seq { ##1 } }
508   }
509   \__intexgral_mkern:N \l__intexgral_variable_sep_muskip
510   \bool_if:NF \l__intexgral_manual_differential_bool
511   {
512     \seq_use:Nn \l__intexgral_differential_seq
513     { \__intexgral_mkern:N \l__intexgral_differential_sep_muskip }
514   }
515 }

```

\@@\_print\_nested\_integral\_inv:

Ici, c'est la séquence des différentielles qui apparait dans la boucle et celle de l'intégrande qui est utilisée à la fin, après l'itération.

```
516 \cs_new_protected:Npn \__intexgral_print_nested_integral_inv:
517 {
518   \int_step_inline:nn { \seq_count:N \l__intexgral_integral_symbol_seq }
519   {
520     \seq_item:Nn \l__intexgral_integral_symbol_seq { ##1 }
521     \__intexgral_mkern:N \l__intexgral_symbol_post_sep_muskip
522     \seq_item:Nn \l__intexgral_differential_seq { ##1 }
523   }
524   \__intexgral_mkern:N \l__intexgral_variable_sep_muskip
525   \seq_use:Nn \l__intexgral_integrand_seq { }
526   \bool_if:NT \l__intexgral_jacobian_bool
527     { \seq_use:Nn \l__intexgral_jacobian_seq { } }
528 }
```

### 10.3 Mode product

\@@\_print\_product\_integral:

Le mode produit est très similaire au mode imbriqué, dans la mesure où cette fois ci, toutes les séquences apparaissent dans la boucle.

```
529 \cs_new_protected:Npn \__intexgral_print_product_integral:
530 {
531   \bool_if:NT \l__intexgral_manual_differential_bool
532   {
533     \cs_set_protected:Npn \differentials
534     {
535       \seq_pop_left:NN \l__intexgral_differential_seq \l_tmpa_tl
536       \tl_use:N \l_tmpa_tl
537     }
538   }
539   \int_step_inline:nn { \seq_count:N \l__intexgral_integral_symbol_seq }
540   {
541     \seq_item:Nn \l__intexgral_integral_symbol_seq { ##1 }
542     \__intexgral_mkern:N \l__intexgral_symbol_post_sep_muskip
543     \seq_item:Nn \l__intexgral_integrand_seq { ##1 }
544     \bool_if:NT \l__intexgral_jacobian_bool
545       { \seq_item:Nn \l__intexgral_jacobian_seq { ##1 } }
546     \__intexgral_mkern:N \l__intexgral_variable_sep_muskip
547     \bool_if:NF \l__intexgral_manual_differential_bool
548       { \seq_item:Nn \l__intexgral_differential_seq { ##1 } }
549   }
550 }
```

\@@\_print\_product\_integral\_inv:

De même, peu de changements.

```
551 \cs_new_protected:Npn \__intexgral_print_product_integral_inv:
552 {
553   \bool_if:NT \l__intexgral_manual_differential_bool
554   {
555     \cs_set_protected:Npn \differentials
```



```

556     {
557       \seq_pop_left:NN \l__intexgral_differential_seq \l_tmpa_tl
558       \tl_use:N \l_tmpa_tl
559     }
560   }
561 \int_step_inline:nn { \seq_count:N \l__intexgral_integral_symbol_seq }
562 {
563   \seq_item:Nn \l__intexgral_integral_symbol_seq { ##1 }
564
565   \bool_if:NF \l__intexgral_manual_differential_bool
566   {
567     \__intexgral_mkern:N \l__intexgral_symbol_post_sep_muskip
568     \seq_item:Nn \l__intexgral_differential_seq { ##1 }
569   }
570   \__intexgral_mkern:N \l__intexgral_variable_sep_muskip
571   \seq_item:Nn \l__intexgral_integrand_seq { ##1 }
572   \bool_if:NT \l__intexgral_jacobian_bool
573   { \seq_item:Nn \l__intexgral_jacobian_seq { ##1 } }
574 }
575 }

```

## 10.4 Composition finale

### \@@\_print\_integral:

La macro principale composant l'intégrale incrémente tout d'abord le compteur global permettant de numéroté chaque intégrale. Ensuite, elle mène les actions préparatoires expliquées précédemment. Enfin, selon le placement des différentielles et du mode d'affichage demandé, l'une des six macros est appelée.

```

576 \cs_new_protected:Nn \__intexgral_print_integral:
577 {
578   \int_gincr:N \g__intexgral_integral_number_int
579   \__intexgral_integral_preconfiguration:
580   \bool_if:NTF \l__intexgral_invert_differential_bool
581   {
582     \str_case:VnF \l__intexgral_display_mode_str
583     {
584       { default } { \__intexgral_print_default_integral_inv: }
585       { nested } { \__intexgral_print_nested_integral_inv: }
586       { product } { \__intexgral_print_product_integral_inv: }
587     }
588     { \__intexgral_print_default_integral_inv: }
589   }
590   {
591     \str_case:VnF \l__intexgral_display_mode_str
592     {
593       { default } { \__intexgral_print_default_integral: }
594       { nested } { \__intexgral_print_nested_integral: }
595       { product } { \__intexgral_print_product_integral: }
596     }
597     { \__intexgral_print_default_integral: }
598   }
599 }

```

## 11 Déclaration des clés

La création des clés pour `\integral` et `\IntegralSetup` est relativement triviale. Les quelques actions élémentaires pour les clés `limits` et `variables` qui n'ont pas été couvertes par les macros sont effectuées ici.

```
600 \keys_define:nn { integral }
601 {
602   mode .choice:,
603   mode / default .code:n =
604   {
605     \bool_set_false:N \l__intexgral_separate_integral_bool
606     \str_set:Nn \l__intexgral_display_mode_str { default }
607   },
608   mode / nested .code:n =
609   {
610     \bool_set_true:N \l__intexgral_separate_integral_bool
611     \str_set:Nn \l__intexgral_display_mode_str { nested }
612   },
613   mode / product .code:n =
614   {
615     \bool_set_true:N \l__intexgral_separate_integral_bool
616     \str_set:Nn \l__intexgral_display_mode_str { product }
617   },
618   mode .default:n = { default },
619
620   limits .code:n =
621   {
622     \seq_set_split:Nnn \l__intexgral_limits_seq { ; } { #1 }
623     \__intexgral_set_limits_regular:
624     \__intexgral_generate_integral_sequence:
625   },
626
627   limits* .code:n =
628   {
629     \seq_set_split:Nnn \l__intexgral_limits_seq { ; } { #1 }
630     \__intexgral_set_limits_starred:
631     \__intexgral_generate_integral_sequence:
632   },
633
634   llimit .tl_set:N = \l__intexgral_lower_limit_tl,
635
636   ulimit .tl_set:N = \l__intexgral_upper_limit_tl,
637
638   symbol .code:n =
639   {
640     \cs_if_exist:NTF #1
641     { \tl_set_eq:NN \l__intexgral_integral_symbol_tl #1 }
642     {
643       \msg_warning:nnn { intexgral } { unknown-symb } { #1 }
644       \tl_set_eq:NN \l__intexgral_integral_symbol_tl \int
645     }
646   },
647
648   nint .code:n =
649   {
650     \int_compare:nNnT { #1 } < { 5 }
651     { \msg_warning:nn { intexgral } { use-glyph-instead } }
```

```

652
653 \tl_clear:N \l__intexgral_integral_symbol_tl
654
655 \int_step_inline:nn { #1 }
656 {
657   \tl_put_right:Nn \l__intexgral_integral_symbol_tl
658   { \int }
659   \int_compare:nNnT { ##1 } < { #1 }
660   {
661     \tl_put_right:Nn \l__intexgral_integral_symbol_tl
662     {
663       \__intexgral_mathchoice:nnnn
664       { \tex_mkern:D -12mu \scan_stop: }
665       { \tex_mkern:D -8mu \scan_stop: }
666       { \tex_mkern:D -4mu \scan_stop: }
667       { \tex_mkern:D -2mu \scan_stop: }
668     }
669   }
670 }
671 },
672
673 domain .code:n =
674 {
675   \tl_set:Nn \l_tmpa_tl { #1 }
676   \seq_set_split:NnV \l__intexgral_domain_seq { * } \l_tmpa_tl
677   \seq_map_inline:Nn \l__intexgral_domain_seq
678   {
679     \tl_if_empty:NF \l__intexgral_lower_limit_tl
680     { \tl_put_right:Nn \l__intexgral_lower_limit_tl { \times } }
681     \tl_set:Ne \l__intexgral_domain_char_tl
682     { \exp_args:Ne \str_uppercase:n { \tl_head:n { ##1 } } }
683     \tl_set:Ne \l__intexgral_domain_dimension_tl
684     { \tl_tail:n { ##1 } }
685     \tl_put_right:Ne \l__intexgral_lower_limit_tl
686     {
687       \exp_not:V \l__intexgral_domain_style_tl
688       { \l__intexgral_domain_char_tl }
689       \c_math_superscript_token { \l__intexgral_domain_dimension_tl }
690     }
691   }
692 },
693
694 domain* .code:n =
695 {
696   \tl_set:Nn \l_tmpa_tl { #1 }
697   \seq_set_split:NnV \l__intexgral_domain_seq { * } \l_tmpa_tl
698   \seq_map_inline:Nn \l__intexgral_domain_seq
699   {
700     \tl_if_empty:NF \l__intexgral_lower_limit_tl
701     { \tl_put_right:Nn \l__intexgral_lower_limit_tl { \times } }
702     \tl_set:Ne \l__intexgral_domain_char_tl
703     { \exp_args:Ne \str_uppercase:n { \tl_head:n { ##1 } } }
704     \tl_set:Ne \l__intexgral_domain_dimension_tl
705     { \tl_tail:n { ##1 } }
706     \tl_put_right:Ne \l__intexgral_lower_limit_tl
707     {
708       \exp_not:V \l__intexgral_domain_style_tl
709       { \l__intexgral_domain_char_tl }

```

```

710         \c_math_subscript_token { \l__intexgral_domain_dimension_tl }
711     }
712 },
713 },
714
715 boundary .code:n =
716     { \tl_set:Nn \l__intexgral_lower_limit_tl { \partial #1 } },
717
718 variables .code:n =
719     {
720         \bool_set_true:N \l__intexgral_custom_variables_bool
721         \str_if_eq:nnTF { #1 } { none }
722         { \bool_set_true:N \l__intexgral_deactivate_variables_bool }
723         {
724             \prop_get:NnNTF
725             \g__intexgral_differential_group_keyword_prop { #1 } \l_tmpa_tl
726             {
727                 \seq_set_split:NnV
728                 \l__intexgral_variables_seq
729                 { , }
730                 \l_tmpa_tl
731                 \seq_if_exist:cTF { g__intexgral_#1_jacobian_seq }
732                 {
733                     \seq_set_eq:Nc
734                     \l__intexgral_jacobian_seq
735                     { g__intexgral_#1_jacobian_seq }
736                 }
737                 { \msg_warning:nnn { intexgral } { no-jacobian } { #1 } }
738             }
739             { \seq_set_split:Nnn \l__intexgral_variables_seq { , } { #1 } }
740         }
741     },
742
743 jacobian .code:n =
744     { \bool_set_true:N \l__intexgral_jacobian_bool },
745 diff-vec .code:n =
746     { \bool_set_true:N \l__intexgral_vectorial_differential_bool },
747 diff-symb .tl_set:N = \l__intexgral_differential_symbol_tl,
748 diff-order .clist_set:N = \l__intexgral_differential_order_clist,
749 }
750
751 \keys_define:nn { IntegralSetup }
752 {
753     diff-symb .tl_set:N = \l__intexgral_differential_symbol_tl,
754     defaultvar .tl_set:N = \l__intexgral_default_differential_tl,
755     defaultvar* .tl_set:N = \l__intexgral_default_vector_differential_tl,
756     postsymbsep .muskip_set:N = \l__intexgral_symbol_post_sep_muskip,
757     varsep .muskip_set:N = \l__intexgral_variable_sep_muskip,
758     diffsep .muskip_set:N = \l__intexgral_differential_sep_muskip,
759     vectorstyle .tl_set:N = \l__intexgral_vectorial_differential_style_tl,
760     domainstyle .tl_set:N = \l__intexgral_domain_style_tl,
761     innersymbsep .tl_set:N = \l__intexgral_symbol_inner_sep_muskip,
762     novar .bool_set:N = \l__intexgral_deactivate_variables_bool,
763 }

```

## 12 Macros d'interface utilisateur

### 12.1 Mots-clés pour les bornes

`\@@_new_limits_group:nn` `\@@_new_limits_group:nn {<mot-clé>} {<bornes>}`

Pour le fonctionnement de `\NewLimitsKeyword`, on enregistre le mot clé et sa valeur dans une liste de propriétés. Une particularité qui se doit néanmoins d'être expliquée est qu'il y a un double inversement des bornes: au moment de la création du mot-clé, et au moment d'en extraire son contenu (cf. `\@@_parse_limits:nn`). La raison est qu'il est difficile de rattacher à un mot clé le statut de *borne inversée*. Ainsi, quelque soit la convention choisie, les bornes sont stockées *dans le même ordre*. Ceci permet également aux mots-clés définis par le package d'être dans le bon ordre, même si la convention est changée après son chargement.

```
764 \cs_new_protected:Npn \__intexgral_new_limits_group:nn #1#2
765 {
766   \prop_gput:Nne \g__intexgral_limits_keyword_prop { #1 }
767   {
768     \bool_if:NTF \l__intexgral_invert_limits_bool
769       { \__intexgral_invert_limits:w #2\q_stop }
770       { #2 }
771   }
772 }
```

`\NewLimitsKeyword`

*Cette macro est documentée en sous-section 4.2.*

```
773 \NewDocumentCommand\NewLimitsKeyword{ m m }
774 {
775   \prop_if_in:NnTF \g__intexgral_limits_keyword_prop { #1 }
776     { \msg_error:nnn { intexgral } { lim-group-alr-def } { #1 } }
777     { \__intexgral_new_limits_group:nn { #1 } { #2 } }
778 }
```

`\RenewLimitsKeyword`

*Cette macro est documentée en sous-section 4.2.*

```
779 \NewDocumentCommand\RenewLimitsKeyword{ m m }
780 {
781   \prop_pop:NnNTF \g__intexgral_limits_keyword_prop { #1 } \l_tmpa_tl
782     { \__intexgral_new_limits_group:nn { #1 } { #2 } }
783     { \msg_error:nnn { intexgral } { lim-group-not-def } { #1 } }
784 }
```

`\ProvideLimitsKeyword`

*Cette macro est documentée en sous-section 4.2.*

```
785 \NewDocumentCommand\ProvideLimitsKeyword{ m m }
786 {
787   \prop_if_in:NnF \g__intexgral_limits_keyword_prop { #1 }
788     { \__intexgral_new_limits_group:nn { #1 } { #2 } }
789 }
```

### \DeclareLimitsKeyword

*Cette macro est documentée en sous-section 4.2.*

```
790 \NewDocumentCommand\DeclareLimitsKeyword{ m m }
791 {
792   \prop_remove:Nn \g__intexgral_limits_keyword_prop { #1 }
793   \__intexgral_new_limits_group:nn { #1 } { #2 }
794 }
```

## 12.2 Mots-clés pour les variables

\@@\_new\_variables\_group:nnn \@@\_new\_variables\_group:nnn {<mot-clé>} {<variable>}  
[<jacobien>]

Tout comme pour les bornes, on enregistre les groupes de différentielles dans une liste de propriétés, avec leur nom, leur valeur et éventuellement leur jacobien associé.

```
795 \cs_new_protected:Npn \__intexgral_new_variables_group:nnn #1#2#3
796 {
797   \prop_gput:Nnn \g__intexgral_differential_group_keyword_prop
798   { #1 } { #2 }
799   \tl_if_blank:nF { #3 }
800   {
801     \seq_new:c { g__intexgral_#1_jacobian_seq }
802     \seq_gset_split:cnn { g__intexgral_#1_jacobian_seq } { , } { #3 }
803   }
804 }
```

### \NewVariableKeyword

*Cette macro est documentée en sous-section 4.3.*

```
805 \NewDocumentCommand\NewVariableKeyword{ m m o }
806 {
807   \prop_if_in:NnTF \g__intexgral_differential_group_keyword_prop { #1 }
808   { \msg_error:nnn { intexgral } { diff-group-alr-def } { #1 } }
809   { \__intexgral_new_variables_group:nnn { #1 } { #2 } { #3 } }
810 }
```

### \RenewVariableKeyword

*Cette macro est documentée en sous-section 4.3.*

```
811 \NewDocumentCommand\RenewVariableKeyword{ m m o }
812 {
813   \prop_pop:NnNTF \g__intexgral_differential_group_keyword_prop { #1 }
814   \l_tmpa_tl
815   {
816     \seq_gclear:c { g__intexgral_#1_jacobian_seq }
817     \__intexgral_new_variables_group:nnn { #1 } { #2 } { #3 }
818   }
819   { \msg_error:nnn { intexgral } { diff-group-not-def } { #1 } }
820 }
```

### \ProvideVariableKeyword

*Cette macro est documentée en sous-section 4.3.*

```

821 \NewDocumentCommand\ProvideVariableKeyword{ m m o }
822 {
823   \prop_if_in:NnF \g__intexgral_differential_group_keyword_prop { #1 }
824   { \__intexgral_new_variables_group:nnn { #1 } { #2 } { #3 } }
825 }

```

### \DeclareVariableKeyword

*Cette macro est documentée en sous-section 4.3.*

```

826 \NewDocumentCommand\DeclareVariableKeyword{ m m o }
827 {
828   \prop_remove:Nn \g__intexgral_differential_group_keyword_prop { #1 }
829   \seq_gclear:c { g__intexgral_#1_jacobian_seq }
830   \__intexgral_new_variables_group:nnn { #1 } { #2 } { #3 }
831 }

```

## 12.3 Mots-clés pour les symboles

Cette section ne nécessite pas de macro intermédiaire, puisqu'on modifie directement les clés du module *integral*.

### \NewSymbolKeyword

*Cette macro est documentée en sous-section 4.4.*

```

832 \NewDocumentCommand\NewSymbolKeyword{ m m }
833 {
834   \prop_if_in:NnT \g__intexgral_limits_keyword_prop { #1 }
835   { \msg_warning:nnn { intexgral } { key-exists-for-lim } { #1 } }
836   \keys_if_exist:nnTF { integral } { #1 }
837   { \msg_error:nnn { intexgral } { symb-key-alr-def } { #1 } }
838   {
839     \keys_define:nn { integral }
840     {
841       #1 .meta:n =
842       {
843         symbol=#2,
844         llimit=##1
845       },
846     }
847   }
848 }

```

### \RenewSymbolKeyword

*Cette macro est documentée en sous-section 4.4.*

```

849 \NewDocumentCommand\RenewSymbolKeyword{ m m }
850 {
851   \prop_if_in:NnT \g__intexgral_limits_keyword_prop { #1 }
852   { \msg_warning:nnn { intexgral } { key-exists-for-lim } { #1 } }
853   \keys_if_exist:nnTF { integral } { #1 }
854   {
855     \keys_define:nn { integral }
856     {
857       #1 .undefine:
858       #1 .meta:n =

```

```

859         {
860             symbol=#2,
861             llimit=##1
862         }
863     }
864 }
865 { \msg_error:nnn { intexgral } { symb-key-not-def } }
866 }

```

### \ProvideSymbolKeyword

*Cette macro est documentée en sous-section 4.4.*

```

867 \NewDocumentCommand\ProvideSymbolKeyword{ m m }
868 {
869     \prop_if_in:NnT \g__intexgral_limits_keyword_prop { #1 }
870     { \msg_warning:nnn { intexgral } { key-exists-for-lim } { #1 } }
871     \keys_if_exist:nnF { integral } { #1 }
872     {
873         \keys_define:nn { integral }
874         {
875             #1 .meta:n =
876             {
877                 symbol=#2,
878                 llimit=##1
879             },
880         }
881     }
882 }

```

### \DeclareSymbolKeyword

*Cette macro est documentée en sous-section 4.4.*

```

883 \NewDocumentCommand\DeclareSymbolKeyword{ m m }
884 {
885     \keys_define:nn { integral }
886     {
887         #1 .undefine:
888         #1 .meta:n =
889         {
890             symbol=#2,
891             llimit=##1
892         },
893     }
894 }

```

## 12.4 Macros de configuration

### \IntegralSetup

*Cette macro est documentée en section 6.*

```

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

```



## \intexgralsetup

De même, mais sans oublier de restreindre l'utilisation au préambule.

*Cette macro est documentée en section 1.*

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

## \integral

La macro finale ne tient qu'en quelques lignes. Si un argument optionnel est fourni, elle effectue les actions nécessaires pour juger de la nature de celui-ci (syntaxe spéciale ou non), et assigne les clés en conséquences. Elle assigne ensuite l'argument principal au token correspondant, avant d'appeler la macro centrale qui compose l'intégrale. Le tout est exécuté dans un groupe afin que toutes les modifications demeurent locales.

*Cette macro est documentée en section 2.*

```
900 \NewDocumentCommand\integral{ 0{} m }
901 {
902   \group_begin:
903   \tl_if_empty:nF { #1 }
904   {
905     \__intexgral_extract_first_key_name:n { #1 }
906     \__intexgral_parse_integral_keys:n { #1 }
907   }
908   \tl_set:Nn \l__intexgral_integrand_tl { #2 }
909   \__intexgral_print_integral:
910   \group_end:
911 }
```

# 13 Configuration du package

La dernière partie du package s'occupe de définir les mots-clés par défaut pour les symboles, les bornes et les variables, ainsi que les paramètres par défaut pour la composition des intégrales.

## 13.1 Symbole

```
912 \NewSymbolKeyword{single}      {\int}
913 \NewSymbolKeyword{double}      {\iint}
914 \NewSymbolKeyword{triple}      {\iiint}
915 \NewSymbolKeyword{quadruple}   {\iiiiint}
916 \NewSymbolKeyword{contour}     {\oint}
917 \NewSymbolKeyword{surface}     {\oiint}
918 \NewSymbolKeyword{volume}      {\oiint}
```

## 13.2 Bornes

```
919 \NewLimitsKeyword{ab}          {a, b}
920 \NewLimitsKeyword{real}        {-\infty, +\infty}
921 \NewLimitsKeyword{positive}    {0, +\infty}
922 \NewLimitsKeyword{negative}    {-\infty, 0}
923 \NewLimitsKeyword{unit}        {0, 1}
924 \NewLimitsKeyword{circle}      {0, 2\pi}
925 \NewLimitsKeyword{scircle}     {0, \pi}
926 \NewLimitsKeyword{qcircle}     {0, \pi/2}
```

```

927 \NewLimitsKeyword{height}    {0, H}
928 \NewLimitsKeyword{radius}    {0, R}
929 \NewLimitsKeyword{length}    {0, L}
930 \NewLimitsKeyword{time}      {0, T}

```

### 13.3 Variables

```

931 \NewVariableKeyword{cartesian} {x, y, z}
932 \NewVariableKeyword{planar}    {x, y}
933 \NewVariableKeyword{polar}     {r, \theta} [r]
934 \NewVariableKeyword{cylindrical} {r, \theta, z} [r]
935 \NewVariableKeyword{cylindrical*} {\rho, \theta, z} [\rho]
936 \NewVariableKeyword{spherical} {r, \theta, \phi} [r^2, \sin\theta]

```

### 13.4 Paramètres par défaut

```

937 \IntegralSetup{
938   diff-symb    = \l__intexgral_default_differential_symbol_tl,
939   defaultvar   = {x},
940   defaultvar*  = {r},
941   varsep       = \thinmuskip,
942   diffsep      = \thinmuskip,
943   innersymbsep = {-5mu},
944   postsymbsep  = {-1mu},
945   vectorstyle  = \vec,
946   domainstyle  = \mathbb,
947   novar        = false
948 }
949 \</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

`\oiint` ..... 918  
`\oint` ..... 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

<code>\l_tmpa_seq</code> . . . . .	220, 223, 225, 232, 238, 240, 246, 248, 255, 261, 348, 349, 365, 367, 370, 374, 377, 381, 382, 385, 388, 392
<code>\l_tmpb_seq</code>	383, 389, 405, 407
<code>\sin</code> . . . . .	936
str commands:	
<code>\str_case:nnTF</code> .	392, 582, 591
<code>\str_case_e:nnTF</code> ..	255, 261
<code>\str_if_eq:nnTF</code>	144, 145, 366, 721
<code>\str_if_eq_p:nn</code> ...	148, 407
<code>\str_new:N</code> . . . . .	180
<code>\str_set:Nn</code> . . . .	606, 611, 616
<code>\str_uppercase:n</code> ..	682, 703
<b>T</b>	
T <sub>E</sub> X and L <sup>A</sup> T <sub>E</sub> X 2 <sub>ε</sub> commands:	
<code>\@ifpackagelater</code> . . . . .	24
<code>\@onlypreamble</code> . . . . .	899
<code>\intexgral@date</code> . . . . .	8, 13
<code>\intexgral@description</code>	9, 15
<code>\intexgral@module</code> . . . .	6, 12
<code>\intexgral@version</code> . . . .	7, 14
tex commands:	
<code>\tex_left:D</code> . . . . .	257, 259
<code>\tex_limits:D</code> . . . . .	45
<code>\tex_mathpunct:D</code> ..	239, 247
<code>\tex_mkern:D</code>	182, 664, 665, 666, 667
<code>\tex_nolimits:D</code> . . . . .	47
<code>\tex_right:D</code> . . . . .	263, 265
<code>\tex_unkern:D</code> . . . .	459, 486
<code>\theta</code> . . . . .	933, 934, 935, 936
<code>\thinmuskip</code> . . . . .	941, 942
<code>\times</code> . . . . .	680, 701
tl commands:	
<code>\c_space_tl</code> . . . . .	86
<code>\tl_clear:N</code> . . . . .	653
<code>\tl_const:Nn</code> . . . . .	164, 165
<code>\tl_head:n</code> . . . . .	682, 703
<code>\tl_if_blank:nTF</code> . . . . .	799
<code>\tl_if_empty:NTF</code> ..	679, 700
<code>\tl_if_empty:nTF</code> . . . . .	903
<code>\tl_if_in:NnTF</code> . . . . .	350
<code>\tl_new:N</code> .	31, 32, 152, 153, 155, 156, 157, 158, 159, 160, 161, 162, 163, 166
<code>\tl_put_right:Nn</code> ..	657, 661, 680, 685, 701, 706
<code>\tl_set:Nn</code> . . . . .	55, 60, 69, 74, 236, 244, 273, 274, 277, 278, 349, 352, 675, 681, 683, 696, 702, 704, 716, 908
<code>\tl_set_eq:NN</code>	45, 47, 154, 358, 641, 644
<code>\tl_tail:n</code> . . . . .	684, 705
<code>\tl_trim_spaces:n</code>	92, 96, 100, 104, 108, 112, 126
<code>\tl_use:N</code> . . . . .	260, 536, 558
<code>\l_tmpa_tl</code> . . . .	349, 350, 355, 358, 535, 536, 557, 558, 675, 676, 696, 697, 725, 730, 781, 814
token commands:	
<code>\c_math_subscript_token</code>	189, 710
<code>\c_math_superscript_token</code> . . . . .	191, 309, 329, 689
<code>\token_to_str:N</code> .	93, 97, 101, 105, 109, 118, 127
<b>V</b>	
<code>\vec</code> . . . . .	945