L'extension intexgral*

Valentin Dao†

Publiée le 26-07-2025

Résumé

Tandis que la composition d'intégrales est très fréquente en 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 ⟨`clé=valeur`⟩. 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 des packages LaTeX3 *l3kernel* et *l3package*. De plus, *intexgral* utilise le package *derivative*[1] afin de faciliter la manipulation de différentielles, ainsi que le package *pkginfograb*[2].

*Ce fichier décrit la version v3.0.0, mise à jour le 24-12-2025

†E-mail : vdao.texdev@gmail.com

1. Disponible sur CTAN sous : https://ctan.org/pkg/derivative
2. Disponible sur CTAN sous : https://ctan.org/pkg/pkginfograb

Table des matières

Licence

Polices

Chronicle Text G3 Roman

Whitney-Medium

Monaspace Argon Medium

Dépôt

Voir dépôt GitHub.

Remerciements

Un grand merci à Plante et Slurpy de m'avoir accompagné dans cette aventure que fut l'apprentissage de 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.

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 la macro ici présente 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

Il est courant de voir les variables d'intégration placées avant l'intégrande dans les articles de physique. 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 `italic=⟨true|false⟩`
upright `upright=⟨true|false⟩`

Ce sont les deux clés du package *derivative*.

2 Présentation de la macro principale

\integral `\integral [⟨list 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}
```

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

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`⟩[1]).

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

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

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`⟩[1]).

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

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

Il est également possible de renseigner des bornes prédéfinies à l'aide de mots-clés (voir 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 ±`\infty`

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

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

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 tous les variables d'intégrations 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}
```

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

MODE `nested`

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

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

MODE `product`

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

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

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. Si ce n'est pas le cas, la compilation n'échouera pas, mais l'expression de l'intégrale risque de ne plus avoir aucun sens. De nombreux messages d'avertissement seront là pour vous en informer.

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 `\int` et un message d'avertissement sera émis.

> ***remarque :*** À part la vérification de l'exis tence 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 deux choses. Tout d'abord, 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 différent. De plus, si vous renseignez comme argument de la clé une macro *définie* mais ne correspondant pas à un symbole intégral, le package n'en tiendra pas compte et utilisera quand même le symbole donné. Outre l'incohérence mathématique que cela peut représenter, il est susceptible que cela mène, sous certaines circonstances, à des erreurs de bas niveau.

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

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

3.3.2 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}
```

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

3.3.3 Définir des bornes ponctuellement

`llimit` `llimit=`⟨borne inférieure⟩
`ulimit` `ulimit=`⟨borne supérieure⟩

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}
```

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

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`.

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

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={r*r}, variables={x, y}]{xy}
\end{equation}
```

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

`boundary` `boundary=` {⟨borne inférieure⟩}

Cette clé place simplement le symbole ∂ avant la borne inférieure.

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

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

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 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 «$\mathrm{d}x$» (ou «$\mathrm{d}\vec{r}$» 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}
```

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

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}
```

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

Comme évoqué plus tôt, *derivative* est utilisé par le package pour composer les différentielles. L'ensemble des clés à suivre, dont le nom portera le préfixe «`diff-`»[1], vous permettent ainsi d'appliquer les fonctionnalités de la macro `\odif`.

1. Quoique la clé `diff-vec` ne soit pas liée à *derivative*.

3.4.3 Modifier le symbole des différentielles

diff-symb `diff-symb=` ⟨macro de différentielle⟩

Cette clé permet de modifier le style des différentielles parmi ceux définis par `\NewDifferential`.

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

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

3.4.4 Réaliser une intégrale curviligne

diff-vec `diff-vec`

Cette clé applique un vecteur à chaque variable d'intégration en plus d'un point médian. Elle n'est compatible qu'avec le mode `default`.

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

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

3.4.5 Opter pour la variante étoilée

diff-star `diff-star`

Activer cette clé revient à utiliser `\odif*` ou ses variantes.

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

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

3.4.6 Désigner des options

diff-options `diff-options=` {⟨liste clé-valeur⟩}

Cette clé accepte une liste de clés telle qu'elle aurait été écrite comme argument optionnel de `\odif` ou ses variantes. Pour illustrer, `diff-options={order={2, 3}, var=all}` sera interprété comme `\odif[order={2, 3}, var=all]{...}`.

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

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

4 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`. Pour les modes `default` et `nested`, la macro compose toutes les différentielles d'un coup. Avec `product`, il faudra répéter la macro entre chaque point-virgule.

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

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

4.2 Retour sur la clé *limits*

\NewLimitsKeyword
\RenewLimitsKeyword
\ProvideLimitsKeyword
\DeclareLimitsKeyword

`\NewLimitsKeyword {⟨mot-clé⟩} {⟨bornes⟩}`

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.

Voici la liste des mots-clés déjà définis par le package et les bornes qu'ils contiennent :

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

> ***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.
>
> ***important :*** Dû à l'implémentation de `\NewLimitsKeyword` et ses variantes, l'utilisateur *doit* suivre la convention d'ordre des bornes du package, même si `invert-limits` est fixé à `true`

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

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

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

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

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

$$\int_{0}^{T}\int_{34}^{40}\int_{0}^{H} h\omega t \,\mathrm{d}h \,\mathrm{d}\omega \,\mathrm{d}t \tag{21}$$

4.3 Retour sur la clé *variables*

\NewVariableKeyword
\RenewVariableKeyword
\ProvideVariableKeyword
\DeclareVariableKeyword

`\NewVariableKeyword {⟨mot-clé⟩} {⟨variables⟩} [⟨jacobien⟩]`

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)
`spherical`	r, θ, ϕ $(r^2, \sin\theta)$

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

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

4.4 Retour sur la clé *symbol*

\NewSymbolKeyword
\RenewSymbolKeyword
\ProvideSymbolKeyword
\DeclareSymbolKeyword

\NewSymbolKeyword {⟨clé⟩} {⟨symbole⟩}

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 :

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

> ***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[1] 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}
```

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

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}
```

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

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}
```

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

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}
```

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

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}
```

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

Vous pouvez aussi toujours profiter du placement automatique du « $\mathrm{d}x$ » avec cette syntaxe en faisant abstraction de `variables`.

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

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

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[1] de l'implémentation, il est porté à votre intention que le package tente d'extraire la première clé de l'argument optionnel[2] 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.

6 Paramètres annexes

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

nouveau: v3.0.0

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

1. Les plus courageux d'entre vous sont tout de même invités à lire le code source.
2. Ou du moins, le groupe de tokens qu'il pense correspondre à une clé.

defaultvar
defaultvar*

`defaultvar=` {⟨variables⟩}

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`.

vectorstyle

`vectorstyle=`⟨séquence de contrôle⟩

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.

domainstyle

`domainstyle=`⟨séquence de contrôle⟩

Semblablement, on peut changer la macro à appliquer pour les clés `domain(*)`.

symbolskip

`symbolskip=`⟨muexpr⟩

Lorsque le mode `default` est en vigueur, le package insère un crénage entre les symboles pour légèrement les rapprocher. La dimension par défaut (−6mu) peut ainsi être ajustée selon la police mathématique en vigueur.

hide-diff

`hide-diff=`⟨`true` | `false`⟩

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 `variables=none`.

Historique des modifications

3.0.0 (09-12-2025)

Ajouté

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

Supprimé

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

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 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 controlant symbole et borne (clés `single`, `contour` etc).

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

Index

S

T

V

Implémentation

L'implémentation sera expliquée plus en détails dans une version ultérieure...

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

% Package declaration

\pkginfograbProvidesExplPackage{intexgral}
  {
    name         = {intexgral}   ,
    author       = {Valentin Dao},
    date         = {2025-12-24}  ,
    creation     = {2025-07-26}  ,
    version      = {3.0.0}       ,
    description  = {A LaTeX package for typesetting integrals.}
  }

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

% expl3 version verification

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

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

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

% Definition of data types for package options

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

% Package options

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

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

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

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

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

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

% Loads the package options and "derivative" package

\ProcessKeyOptions[intexgral]

\RequirePackage{derivative}

% Warning/Error/Info messages

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

% Argument specifiers variants
```

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

% DATA TYPES

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

\tl_set_eq:NN \l__intexgral_integral_symbol_tl \int

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

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

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

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

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

\str_new:N \l__intexgral_display_mode_str

\muskip_new:N \l__intexgral_inline_symbol_muskip

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

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

% TeX macros adapted to expl3 syntax

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

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

% SYMBOL

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

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

% Generates the symbols after <limits> is executed

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

```
% LIMITS

% Inverts the limit order convention

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

% Creates a keyword containing predefined limits

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

% Verifies if the argument corresponds to a valid keyword

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

% Splits the argument of the key to prepare for parsing

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

% Limits under regular form

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

% Limits under interval form
```

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

% VARIABLES

% Creates a keyword containing predefined variables

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

```
  }

% expl3 equivalent of \odif(*)

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

% Extracts the whole "order" <keyval>

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

% Extracts the whole "var" <keyval>

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

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

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

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

% Reassembles the variables

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

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

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

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

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

% SPECIAL SYNTAX

% Cuts off the value of a key

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

% Extracts first key name of optional argument

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

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

\cs_new_protected:Npn \__intexgral_parse_macro_keys:n #1
  {
    \keys_if_exist:nVTF { integral } \l__intexgral_key_name_tl
      { \keys_set:nn { integral } { #1 } }
      {
        \regex_split:nnN { : } { #1 } \l_tmpa_seq
        \int_compare:nNnT { \seq_count:N \l_tmpa_seq } < { 2 }
          {
            \seq_put_right:NV
              \l_tmpa_seq
              \l__intexgral_default_differential_tl
          }
        \int_compare:nNnT { \seq_count:N \l_tmpa_seq } < { 3 }
          { \seq_put_right:Nn \l_tmpa_seq { d } }
        \seq_set_split:Nne \l_tmpb_seq
          { + }
          { \seq_item:Nn \l_tmpa_seq { 2 } }
        \keys_set:ne { integral }
          {
            limits    = { \seq_item:Nn \l_tmpa_seq { 1 } },
            variables = { \seq_item:Nn \l_tmpb_seq { 1 } },
            mode      =
              {
                \str_case:enF { \seq_item:Nn \l_tmpa_seq { 3 } }
```

```
                {
                  { d       } { default }
                  { n       } { nested  }
                  { p       } { product }
                  { default } { default }
                  { nested  } { nested  }
                  { product } { product }
                }
              { default }
            },
          \bool_if:nT
            {
              \int_compare_p:nNn { \seq_count:N \l_tmpb_seq } > { 1 }
              &&
              \str_if_eq_p:en { \seq_item:Nn \l_tmpb_seq { 2 } } { j }
            }
            { jacobian }
        }
    }
  }

% PRELIMINARY CONFIGURATIONS

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

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

% Performs preparatory actions before executing the main macro

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

    \bool_if:NT \l__intexgral_separate_integral_bool
```

```
      {
        \seq_set_split:NnV
          \l__intexgral_integrand_seq
          { ; }
          \l__intexgral_integrand_tl
      }

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

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

    \bool_if:NF \l__intexgral_deactivate_differentials_bool
      { \__intexgral_parse_differentials: }

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

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

% Default mode

\cs_new_protected:Nn \__intexgral_print_default_integral:
  {
    \bool_if:NT \l__intexgral_manual_differentials_bool
      {
        \cs_set_protected:Npn \differentials
          { \seq_use:Nn \l__intexgral_differential_seq {  } }
      }
    \seq_use:Nn \l__intexgral_integral_symbol_seq
      { \__intexgral_mkern:N \l__intexgral_inline_symbol_muskip }
    \bool_if:nT
      {
        \l__intexgral_invert_differentials_bool
        &&
        !\l__intexgral_manual_differentials_bool
      }
      {
        \seq_use:Nn \l__intexgral_differential_seq {  } \tex_mathop:D{} \!
        \bool_if:NT \l__intexgral_vectorial_differential_bool { \cdot \: }
      }
    \seq_use:Nn \l__intexgral_integrand_seq {  }
    \bool_if:NT \l__intexgral_jacobian_bool
      { \seq_use:Nn \l__intexgral_jacobian_seq {  } }
    \bool_if:nT
      {
        !\l__intexgral_invert_differentials_bool
```

```
        &&
        !\l__intexgral_manual_differentials_bool
      }
      {
        \bool_if:NT \l__intexgral_vectorial_differential_bool { \cdot }
        \seq_use:Nn \l__intexgral_differential_seq {  }
      }
  }

% Nested mode

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

% Product mode

\cs_new_protected:Nn \__intexgral_print_product_integral:
  {
    \bool_if:NT \l__intexgral_manual_differentials_bool
```

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

% General macro for typesetting

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

% KEYS

\keys_define:nn { integral } {

  % LIMITS

  mode .choice:,
```

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

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

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

llimit .tl_set:N = \l__intexgral_lower_limit_tl,

ulimit .tl_set:N = \l__intexgral_upper_limit_tl,

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

% SYMBOL SHORTCUT

nint .code:n =
  {
    \int_compare:nNnT { #1 } < { 5 }
      { \msg_warning:nn { intexgral } { use-glyph-instead } }

    \tl_clear:N \l__intexgral_integral_symbol_tl

    \int_step_inline:nn { #1 }
      {
        \tl_put_right:Nn \l__intexgral_integral_symbol_tl
          { \int }
```

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

% LIMITS SHORTCUTS

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

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

```

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

  % VARIABLES

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

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

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

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

\keys_define:nn { IntegralSetup }
  {
    defaultvar .code:n =
      {
        \prop_get:NnNTF
          \g__intexgral_differential_group_keyword_prop
          { #1 }
          \l_tmpa_tl
          {
            \tl_set_eq:NN
              \l__intexgral_default_differential_tl
              \l_tmpa_tl
```

```
          }
          { \tl_set:Nn \l__intexgral_default_differential_tl { #1 } }
      },
    defaultvar* .code:n =
      {
        \prop_get:NnNTF
          \g__intexgral_differential_group_keyword_prop
          { #1 }
          \l_tmpa_tl
          {
            \tl_set_eq:NN
              \l__intexgral_default_vector_differential_tl
              \l_tmpa_tl
          }
          { \tl_set:Nn \l__intexgral_default_vector_differential_tl { #1 } }
      },
    vectorstyle .tl_set:N   = \l__intexgral_vectorial_differential_style_tl,
    domainstyle .tl_set:N   = \l__intexgral_domain_style_tl,
    symbolskip  .tl_set:N   = \l__intexgral_inline_symbol_muskip,
    hide-diff   .bool_set:N = \l__intexgral_deactivate_differentials_bool,
  }

% END USER MACROS

% Limits keywords

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

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

\NewDocumentCommand\ProvideLimitsKeyword{ m m }
  {
```

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

\NewDocumentCommand\DeclareLimitsKeyword{ m m }
  {
    \prop_pop:NnNT \g__intexgral_limits_keyword_prop { #1 } \l_tmpa_tl
      {
        \bool_if:NTF \l__intexgral_invert_limits_bool
          {
            \clist_set:Nn \l_tmpa_clist { #2 }
            \clist_reverse:N \l_tmpa_clist
            \__intexgral_new_limits_group:nV { #1 } \l_tmpa_clist
          }
          { \__intexgral_new_limits_group:nn { #1 } { #2 } }
      }
  }

% Variable keywords

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

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

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

\NewDocumentCommand\DeclareVariableKeyword{ m m o }
  {
    \prop_pop:NnNT \g__intexgral_differential_group_keyword_prop { #1 }
      \l_tmpa_tl
```

```
      {
        \seq_clear:c { g__intexgral_#1_jacobian_seq }
        \__intexgral_new_differential_group:nnn { #1 } { #2 } { #3 }
      }
  }

% Symbol keys

\NewDocumentCommand\NewSymbolKeyword{ m m }
  {
    \prop_if_in:NnT \g__intexgral_limits_keyword_prop { #1 }
      { \msg_warning:nnn { intexgral } { key-exists-for-lim } { #1 } }
    \keys_if_exist:nnTF { integral } { #1 }
      { \msg_error:nnn { intexgral } { symb-key-alr-def } { #1 } }
      {
        \keys_define:nn { integral }
          {
            #1 .meta:n =
              {
                symbol=#2,
                llimit=##1
              },
          }
      }
  }

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

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

```
          }
      }
  }

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

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

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

% Preamble only

\@onlypreamble\intexgralsetup

% Included keywords

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

% Limits keywords
\NewLimitsKeyword{ab}        {a, b}
\NewLimitsKeyword{real}      {-\infty, \infty}
\NewLimitsKeyword{positive}  {0, \infty}
\NewLimitsKeyword{negative}  {-\infty, 0}
\NewLimitsKeyword{unit}      {0, 1}
\NewLimitsKeyword{circle}    {0, 2\pi}
\NewLimitsKeyword{scircle}   {0, \pi}
\NewLimitsKeyword{qcircle}   {0, \frac{\pi}{2}}
\NewLimitsKeyword{height}    {0, H}
```

```
\NewLimitsKeyword{radius}    {0, R}
\NewLimitsKeyword{length}    {0, L}
\NewLimitsKeyword{time}      {0, T}

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

% Default parameters

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

\ExplSyntaxOff
```