

Le package `tnsana` : de l'analyse élémentaire

Code source disponible sur <https://github.com/typensee-latex/tnsana.git>.

Version 0.6.0-beta développée et testée sur Mac OS X.

Christophe BAL

2020-07-30

Table des matières

1. Introduction	3
2. Beta-dépendance	3
3. Packages utilisés	3
4. Constantes et paramètres	3
a. Constantes classiques ajoutées	3
b. Constantes latines personnelles	3
5. Une variable « symbolique »	3
6. La fonction valeur absolue	4
7. Fonctions nommées spéciales	4
a. Sans paramètre	4
b. Avec un paramètre	4
c. Toutes les fonctions nommées en plus	4
8. Définition explicite d'une fonction	5
9. Limite	6
10. Calcul différentiel	6
a. Les opérateurs ∂ et d	6
b. Dérivations totales d'une fonction – Version longue avec une variable	6
c. Dérivations totales d'une fonction – Version courte sans variable	8
d. L'opérateur de dérivation totale	8
e. Dérivations partielles	9
f. L'opérateur de dérivation partielle	9

11.	Tableaux de variation et de signe	10
a.	Les bases de <code>tkz-tab</code>	10
b.	Décorer facilement un tableau	16
i.	Motivation	16
ii.	Ajouter une couleur de fond à une ligne	18
iii.	Commenter une ligne	18
iv.	Graphiques pour expliquer des signes	18
v.	Quelques exemples	19
12.	Calcul intégral	21
a.	Le symbole standard revisité	21
b.	Un opérateur d'intégration clés en main	22
c.	L'opérateur « crochet »	22
13.	Historique	24
14.	Toutes les fiches techniques	26
a.	Constantes et paramètres	26
i.	Constantes classiques ajoutées	26
ii.	Constantes latines personnelles	26
b.	Une variable « symbolique »	26
c.	La fonction valeur absolue	26
d.	Fonctions nommées spéciales	26
i.	Sans paramètre	26
ii.	Avec un paramètre	27
e.	Définition explicite d'une fonction	27
f.	Limite	27
g.	Calcul différentiel	28
i.	Les opérateurs ∂ et d	28
ii.	Dérivations totales d'une fonction – Version longue avec une variable	28
iii.	Dérivations totales d'une fonction – Version courte sans variable	28
iv.	L'opérateur de dérivation totale	28
v.	Dérivations partielles	29
vi.	L'opérateur de dérivation partielle	29
h.	Tableaux de variation et de signe	29
i.	Coloriser le fond	29
ii.	Commentaires	29
iii.	Graphiques explicatifs	30
i.	Calcul intégral	30
i.	Le symbole standard revisité	30
ii.	Un opérateur d'intégration clés en main	30
iii.	L'opérateur « crochet »	30

1. Introduction

Le package `tnsana` propose des macros pour écrire de l'analyse mathématique basique via un codage sémantique simple.

2. Beta-dépendance

`tnscom` qui est disponible sur <https://github.com/typensee-latex/tnscom.git> est un package utilisé en coulisse.

3. Packages utilisés

La roue ayant déjà été inventée, le package `tnsana` réutilise les packages suivants sans aucun scrupule.

- | | | | |
|-----------------------------|---------------------------|-------------------------|------------------------|
| • <code>circledsteps</code> | • <code>ifmtarg</code> | • <code>pgfplots</code> | • <code>upgreek</code> |
| • <code>commado</code> | • <code>mathtools</code> | • <code>relsize</code> | • <code>xstring</code> |
| • <code>forloop</code> | • <code>nicematrix</code> | • <code>tkz-tab</code> | |

4. Constantes et paramètres

a. Constantes classiques ajoutées

Les macros ci-dessous ne nécessitent pas d'accolades.

```
\ggamma$ , \ppi$ , \tttau$ ,  
\ee$ , \ii$ , \jj$  
et \kk$ où \tttau = 2 \ppi$
```

$\gamma, \pi, \tau, e, i, j$ et k où $\tau = 2\pi$

Remarque. Faites attention car `\Large $\ppi \neq \pi$` produit $\pi \neq \pi$. Comme vous le constatez, les symboles ne sont pas identiques. Ceci est vraie pour toutes les constantes grecques.

b. Constantes latines personnelles

La macro `\param` est surtout là pour une utilisation pédagogique.

```
\param{a} x^2 + \param{b} x + \param{c}$  
ou  
$a x^2 + b x + c$
```

$ax^2 + bx + c$ ou $ax^2 + bx + c$

5. Une variable « symbolique »

Le package `tnscom`, chargé en coulisse, propose la macro `\symvar` qui produit différents symboles donnés ci-dessous.

```
\symvar = \symvar[1]$ ou  
\symvar[2]$ ou  
\symvar[3]$
```

$\bullet = \bullet$ ou \circ ou \blacksquare

Le nom `symvar` vient de `symb-olic var-iable` soit « *variable symbolique* » en anglais. Ces symboles sont utiles pour indiquer un argument symboliquement sans faire référence précisément à une ou des variables nommées.

6. La fonction valeur absolue

Exemple

```
\abs{2}$ ,
\abs {\dfrac{3}{5}}$ ou
\abs*{\dfrac{3}{5}}$
```

$$|2|, \left| \frac{3}{5} \right| \text{ ou } \left| \frac{3}{5} \right|$$

Remarque. Le code L^AT_EX vient directement de ce poste : <https://tex.stackexchange.com/a/43009/6880>.

7. Fonctions nommées spéciales

a. Sans paramètre

Quelques fonctions nommées ont été ajoutées en plus de celles fournies par le package `amsmath` qui est chargé en coulisse et propose par exemple $\lg x$ et $\ln x$ via les macros `\lg` et `\ln`. Ci-dessous le `f` dans `fch` est pour `f`-rench soit « *français* » en anglais (*ce choix sert à éviter des incompatibilités avec d'autres packages*). La liste complète des fonctions nommées est donnée plus bas dans la section c.

```
\fch x = \cosh x$ ou $\acos x$
```

$\operatorname{ch} x = \cosh x$ ou $\operatorname{acos} x$

b. Avec un paramètre

Pour le moment il y a juste deux fonctions utilisables avec un paramètre optionnel. Les voici.

```
\log[2] x = \lg x$ ou
\exp[6] y = 6^y$
```

$\log_2 x = \lg x$ ou $\exp_6 y = 6^y$

c. Toutes les fonctions nommées en plus

`\acos x` : $\operatorname{acos} x$

`\asin x` : $\operatorname{asin} x$

`\atan x` : $\operatorname{atan} x$

`\arccosh x` : $\operatorname{arccosh} x$

`\arcsinh x` : $\operatorname{arcsinh} x$

`\arctanh x` : $\operatorname{arctanh} x$

`\acosh x` : $\operatorname{acosh} x$

`\asinh x` : $\operatorname{asinh} x$

`\atanh x` : $\operatorname{atanh} x$

`\fch x` : $\operatorname{ch} x$

`\fsh x` : $\operatorname{sh} x$

`\fth x` : $\operatorname{th} x$

`\afch x` : $\operatorname{ach} x$

`\afsh x` : $\operatorname{ash} x$

`\afth x` : $\operatorname{ath} x$

`\exp x` : $\exp x$

`\exp[p] x` : $\exp_p x$

`\log x` : $\log x$

`\log[p] x` : $\log_p x$

8. Définition explicite d'une fonction

Exemple 1 – Écriture par défaut

<code>\funcdef{f}{x}{x^2}% {I}{J}\$</code>	$f: \begin{array}{l} I \rightarrow J \\ x \mapsto x^2 \end{array}$
--	--

Remarque. Même si cela est peu utile, vous pouvez utiliser la mise en forme dans du texte pour obtenir $f: \begin{array}{l} I \rightarrow J \\ x \mapsto x^2 \end{array}$ mais c'est un peu affreux.

Exemple 2 – Écriture alternative

On peut cacher le trait vertical via l'option `s` pour `s`-hort soit « *court* » en anglais.

<code>\funcdef[s]{f}{x}{x^2}% {I}{J}\$</code>	$f: I \rightarrow J \\ x \mapsto x^2$
---	---------------------------------------

Exemple 3 – Écriture en ligne

Pour avoir tout sur une ligne, ce qui est l'idéal pour une insertion dans du texte, il suffit d'utiliser l'option `h` pour `h`-orizontal.

<code>Soit \funcdef[h]{f}{x}{x^2}% {I}{J}\$...</code>	Soit $f : x \in I \mapsto x^2 \in J \dots$
--	--

Exemple 4 – Écriture en ligne incomplète

En mode horizontal, les ensembles peuvent être de valeur vide pour ne pas les indiquer.

<code>\funcdef[h]{f}{x}{x^2}{}{J}\$</code>	$f : x \mapsto x^2 \in J$
<code>\funcdef[h]{f}{x}{x^2}{I}{}\$</code>	$f : x \in I \mapsto x^2$
<code>\funcdef[h]{f}{x}{x^2}{}{}\$</code>	$f : x \mapsto x^2$

Exemple 5 – Écriture textuelle

On peut enfin obtenir une version « *textuelle* » via la macro `\txtfuncdef` qui ne s'utilise pas en mode mathématique où `txt` est pour `texte`. Ici aussi on peut omettre des ensembles.

<code>\txtfuncdef{f}{x}{x^2}{I}{J}</code>	$f(x) = x^2 \text{ pour } x \in I \text{ (on sait que } f(x) \in J)$
<code>\txtfuncdef{f}{x}{x^2}{}{J}</code>	$f(x) = x^2 \text{ (on sait que } f(x) \in J)$
<code>\txtfuncdef{f}{x}{x^2}{I}{}\$</code>	$f(x) = x^2 \text{ pour } x \in I$
<code>\txtfuncdef{f}{x}{x^2}{}{}\$</code>	$f(x) = x^2$

9. Limite

Exemple 1 – Cas minimaliste

Notez ci-dessous que le rendu de la limite via `\limit` se fait toujours en mode `\displaystyle` pour l'opérateur de limite.

$\text{\texttt{\$}\texttt{\backslash limit}\texttt{\{f(x)\}\{x\}\{0\}} = \texttt{\backslash frac}\texttt{\{1\}\{2\}}\texttt{\$}$	$\lim_{x \rightarrow 0} f(x) = \frac{1}{2}$
--	---

Remarque. `\lim\limits_{x \rightarrow 0} f(x)` produit $\lim_{x \rightarrow 0} f(x)$ contre $\lim_{x \rightarrow 0} f(x)$ ci-dessus. La version `\limit` utilise un petit peu plus d'espace sous le mot `lim`.

Exemple 2 – Avec des conditions

Ce 2^e exemple devrait rendre intéressante la macro `\limit` qui permet d'ajouter facilement plusieurs conditions¹ comme le montre la 2^e limite ci-après.

$\text{\texttt{\$}\texttt{\backslash limit}\texttt{\{f(x)\}\{x\}\{0 \mid x > 0\}}\texttt{\$}} \text{ ou } \text{\texttt{\$}\texttt{\backslash limit}\texttt{\{f(z)\}\{z\}\{0 \mid z \in \Omega \mid \texttt{\backslash abs}\{z\} < 3\}}\texttt{\$}}$	$\lim_{\substack{x \rightarrow 0 \\ x > 0}} f(x) \text{ ou } \lim_{\substack{z \rightarrow 0 \\ z \in \Omega \\ z < 3}} f(z)$
--	---

Exemple 3 – Ajout de parenthèses

Les options `p` et `sp` permettent d'ajouter facilement des parenthèses extensibles ou non autour de la fonction. Indiquons que `sp` est pour `s`-mall `p`-arenthesis soit « *petites parenthèses* » en anglais.

$\text{\texttt{\$}\texttt{\backslash displaystyle}\texttt{\backslash limit}\texttt{[p]}\texttt{\{\frac\{1\}\{x\}\}\{x\}}\texttt{\{0 \mid x > 0\}}\texttt{\$}}$ <p>ou</p> $\text{\texttt{\$}\texttt{\backslash displaystyle}\texttt{\backslash limit}\texttt{[sp]}\texttt{\{\frac\{1\}\{x\}\}\{x\}}\texttt{\{0 \mid x > 0\}}\texttt{\$}}$	$\lim_{\substack{x \rightarrow 0 \\ x > 0}} \left(\frac{1}{x} \right) \text{ ou } \lim_{\substack{x \rightarrow 0 \\ x > 0}} \left(\frac{1}{x} \right)$
--	---

10. Calcul différentiel

a. Les opérateurs ∂ et d

Voici deux opérateurs utiles aussi bien pour du calcul différentiel que du calcul intégral.

$\text{\texttt{\$}\texttt{\ddot{t}}} = \texttt{\backslash dd}\texttt{[1]}\texttt{\{t\}} \text{ ou } \texttt{\backslash dd}\texttt{[n]}\texttt{\{x\}}$ $\text{\texttt{\$}\texttt{\pp{t}}} = \texttt{\backslash pp}\texttt{[1]}\texttt{\{t\}} \text{ ou } \texttt{\backslash pp}\texttt{[n]}\texttt{\{x\}}$	$dt = d^1t \text{ ou } d^n x$ $\partial t = \partial^1t \text{ ou } \partial^n x$
---	---

b. Dérivations totales d'une fonction – Version longue avec une variable

Exemple 1 – Différentes écritures possibles

La macro `\der` est stricte du point de vue sémantique car on doit lui fournir la fonction, l'ordre de dérivation et la variable de dérivation (*voir la section iii. qui présente la macro `\sder` permettant une*

1. En pratique, on utilise généralement au maximum une condition.

rédaction efficace pour obtenir $f^{(1)}$ ou f'). Voici plusieurs mises en forme faciles à taper via l'option de `\der`. Attention bien entendu à n'utiliser l'option par défaut `u` ou l'option `d` qu'avec un ordre de dérivation de valeur naturelle connue !

$\begin{aligned} &\$ \backslash \text{der} \quad \{f\}\{x\}\{3\} \\ &= \backslash \text{der}[e]\{f\}\{x\}\{3\} \\ &= \backslash \text{der}[d]\{f\}\{x\}\{3\}\$ \end{aligned}$	$f''' = f^{(3)} = \ddot{f}$
$\begin{aligned} &\$ \backslash \text{der}[i] \{u\}\{x\}\{k\} \\ &= \backslash \text{der}[f] \{u\}\{x\}\{k\} \\ &= \backslash \text{der}[sf]\{u\}\{x\}\{k\}\$ \end{aligned}$	$d_x^k u = \frac{d^k u}{dx^k} = \frac{d^k u}{dx^k}$

On peut aussi ajouter autour de la fonction des parenthèses extensibles ou non sauf même si cela n'est pas utile pour le mode `d` (voir juste après le mode `bd`). Ci-dessous on montre au passage une écriture du type « *opérateur fonctionnel* » : voir la section iv. page 28 à ce sujet.

$\begin{aligned} &\$ \backslash \text{der}[\text{osf}, \text{sp}]\{\frac{1}{2} \quad uv\}\{x\}\{k\} \\ &= \backslash \text{der}[\text{of}, \text{p}] \quad \{\backslash \text{dfrac}\{1\}\{2\} \quad uv\}\{x\}\{k\}\$ \end{aligned}$	$\frac{d^k}{dx^k}(\frac{1}{2}uv) = \frac{d^k}{dx^k} \left(\frac{1}{2}uv \right)$
--	---

Avec l'option `d` les parenthèses seules sont sans utilité car on peut obtenir des choses non souhaitées comme $(u + v)$. À la place on utilisera l'option `bd` où le `b` est pour `b-racket` soit « *crochet* » en anglais. Notez au passage que `p` et `sp` restent utilisables.

$\begin{aligned} &\$ \backslash \text{der}[\text{bd}] \quad \{\frac{1}{2} \quad uv\}\{x\}\{2\} \\ &= \backslash \text{der}[\text{bd}, \text{p}] \quad \{\frac{1}{2} \quad uv\}\{x\}\{2\} \\ &= \backslash \text{der}[\text{bd}, \text{sp}]\{\backslash \text{dfrac}\{1\}\{2\} \quad uv\}\{x\}\{2\}\$ \end{aligned}$	$\overline{\overline{\frac{1}{2}uv}} = \overline{\overline{\left(\frac{1}{2}uv\right)}} = \overline{\overline{\left(\frac{1}{2}uv\right)}}$
---	---

Remarque. Expliquons les valeurs des options.

1. `u`, la valeur par défaut, est pour `u-suel` soit l'écriture avec les primes. Cette option ne marchera pas avec un nombre symbolique de dérivations.
2. `e` est pour `e-xposant`.
3. `i` est pour `i-ndice`.
4. `d` est pour `d-ot` soit « *point* » en anglais.
5. `bd` est pour `b-racket d-ot` où « *bracket* » est pour « *crochet* » en anglais.
6. `f` est pour `f-raction` avec aussi `sf` pour une écriture réduite où `s` est pour `s-mall` soit « *petit* » en anglais.
7. `of` et `osf` utilisent le préfixe `o` pour `o-pérateur`.
8. `p` est pour `p-arenthèse` : dans ce cas les parenthèses seront extensibles. Le fonctionnement est différent avec l'option `d` comme nous l'avons vu avant.
9. `sp` est pour des parenthèses non extensibles. Là aussi le fonctionnement est différent avec l'option `d`.

Exemple 2 – Pas de uns inutiles

$\begin{aligned} &\$ \backslash \text{der}[i] \{u\}\{x\}\{1\} \\ &= \backslash \text{der}[f] \{u\}\{x\}\{1\} \\ &= \backslash \text{der}[sf]\{u\}\{x\}\{1\} \\ &= \backslash \text{der}[\text{of}]\{u\}\{x\}\{1\}\$ \end{aligned}$	$d_x u = \frac{du}{dx} = \frac{du}{dx} = \frac{d}{dx} u$
--	--

Remarque. Voici comment forcer les exposants 1 si besoin. Fonctionnel mais très moche...

```
\der[i]{u}{x}{\,\!\!1}
= \der[f]{u}{x}{\,\!\!1}
= \der[sf]{u}{x}{\,\!\!1}
= \der[of]{u}{x}{\,\!\!1}
```

$$d_x^1 u = \frac{d^1 u}{dx^1} = \frac{d^1 u}{dx^1} = \frac{d^1}{dx^1} u$$

c. Dérivations totales d'une fonction – Version courte sans variable

Dans l'exemple suivant le code manque de sémantique car on n'indique pas la variable de dérivation. Ceci étant dit à l'usage la macro `\sder` rend de grands services. Ici le préfixe **s** est pour **s**-imple voire **s**-impliste... Voici des exemples où de nouveau l'option par défaut **u** et l'option **d** ne seront fonctionnelles qu'avec un ordre de dérivation de valeur naturelle connue !

Exemple 1

```
\sder{f}{1} = \der{f}{x}{1}$

\sder {f}{3}
= \sder[e]{f}{3}
= \sder[d]{f}{3}$
```

$$f' = f' \\ f''' = f^{(3)} = \ddot{f}$$

Exemple 2

```
\sder[sp]{\dfrac{1}{2} uv}{2}
= \sder[e,p]{\dfrac{1}{2} uv}{2}
= \sder[bd]{\dfrac{1}{2} uv}{2}$
```

$$\left(\frac{1}{2}uv\right)'' = \left(\frac{1}{2}uv\right)^{(2)} = \overline{\frac{1}{2}}'' uv$$

Remarque. Ici les seules options disponibles sont **u**, **e**, **b**, **bd**, **p** et **sp**.

d. L'opérateur de dérivation totale

Ce qui suit peut rendre service au niveau universitaire. Les options possibles sont **f**, valeur par défaut, **sf** et **i** avec les mêmes significations que pour la macro `\der`.

```
\derope {x}{k}
= \derope[sf]{x}{k}
= \derope[i]{x}{k}$
```

$$\frac{d^k}{dx^k} = \frac{d^k}{dx^k} = d_x^k$$

Ici non plus il n'y a pas de uns inutiles mais l'astuce `\,\!\!1` reste utilisable.

```
\derope {x}{1}
= \derope[sf]{x}{1}
= \derope[i]{x}{1}$
```

$$\frac{d}{dx} = \frac{d}{dx} = d_x$$

e. Dérivations partielles

Exemple 1 – Différentes écritures

La macro `\pder`² avec `p` pour `p`-artielle permet de rédiger des dérivées partielles en utilisant facilement plusieurs mises en forme via une option qui vaut `f` par défaut. Cette macro attend une fonction, les dérivées partielles effectuées et l'ordre total de dérivation. Voici deux types de mise en forme classiques où vous noterez comment `x | y^2` est interprété.

$\begin{aligned} &\$ \backslash \text{pder} \quad \{f\}\{x y^2\}\{3\} \\ &= \backslash \text{pder}[\text{sf}]\{f\}\{x y^2\}\{3\}\$ \end{aligned}$	$\frac{\partial^3 f}{\partial x \partial y^2} = \frac{\partial^3 f}{\partial x \partial y^2}$
---	---

Il existe en plus deux notations indicielles données en exemple ci-dessous. Notez qu'avec l'option `ei` l'exposant total n'est pas imprimé et que les exposants partiels doivent être des naturels connus.

$\begin{aligned} &\$ \backslash \text{pder}[\text{i}] \{f\}\{x y^2\}\{3\} \\ &= \backslash \text{pder}[\text{ei}]\{f\}\{x y^2\}\{3\}\$ \end{aligned}$	$\partial_{x,y(2)}^3 f = f'_{x y y}$
---	--------------------------------------

Les options `i` et `ei` marchent avec des variables indicées à condition de bien mettre les dites variables entre des accolades.

$\begin{aligned} &\$ \backslash \text{pder}[\text{i}] \{f\}\{x_1 \{x_2\}^2\}\{3\} \\ &= \backslash \text{pder}[\text{ei}]\{f\}\{x_1 \{x_2\}^2\}\{3\}\$ \end{aligned}$	$\partial_{x_1, x_2(2)}^3 f = f'_{x_1 x_2 x_2}$
---	---

On peut aussi ajouter autour de la fonction à différencier des parenthèses extensibles ou non via `p` et `sp` respectivement. Ci-dessous on montre aussi une écriture du type « *opérateur fonctionnel* » : voir la section vi. page 29 à ce sujet.

$\begin{aligned} &\$ \backslash \text{pder}[\text{of}, \text{sp}] \{u + v\}\{x y^2\}\{\} \\ &= \backslash \text{pder}[\text{osf}, \text{sp}]\{u + v\}\{x y^2\}\{\}\$ \end{aligned}$	$\frac{\partial}{\partial x \partial y^2}(u + v) = \frac{\partial}{\partial x \partial y^2}(u + v)$
$\begin{aligned} &\$ \backslash \text{pder}[\text{i}, \text{sp}] \{u + v\}\{x y^2\}\{\} \\ &= \backslash \text{pder}[\text{ei}, \text{sp}] \{u + v\}\{x y^2\}\{\}\$ \end{aligned}$	$\partial_{x,y(2)}(u + v) = (u + v)'_{x y y}$

Remarque. Les options disponibles sont `f`, `sf`, `of`, `osf`, `p` et `sp` avec des significations similaires à celles pour la macro `\der` auxquelles s'ajoutent `i` et `ei` pour les écritures indicielles où le `e` dans `ei` est pour `e`-xpand soit « *développer* » en anglais.

Exemple 2 – Pas de uns inutiles

$\begin{aligned} &\$ \backslash \text{pder} \quad \{u\}\{x\}\{1\} \\ &= \backslash \text{pder}[\text{sf}]\{u\}\{x\}\{1\} \\ &= \backslash \text{pder}[\text{i}] \{u\}\{x\}\{1\}\$ \end{aligned}$	$\frac{\partial u}{\partial x} = \frac{\partial u}{\partial x} = \partial_x u$
--	--

Remarque. Rappelons que pour obtenir $\partial_x^1 u$ on peut taper `\pder[i]\{u\}\{x\}\{\,\backslash!\,1\}`.

f. L'opérateur de dérivation partielle

Ce qui suit peut rendre service au niveau universitaire. Les options possibles sont `f`, valeur par défaut, `sf` et `i` avec les mêmes significations que pour la macro `\pder`.

2. `\partial` existe déjà pour obtenir ∂ .

```

 $\pderope$  {x | y^2}{3}
= \pderope[sf]{x | y^2}{3}
= \pderope[i] {x | y^2}{3}$

```

$$\frac{\partial^3}{\partial x \partial y^2} = \frac{\partial^3}{\partial x \partial y^2} = \partial_{x,y(2)}^3$$

11. Tableaux de variation et de signe

a. Les bases de tkz-tab

Comment ça marche ?

Pour les tableaux de variation et de signe non décorés, tout le boulot est fait par le package `tkz-tab`. Ce package est utilisé avec les réglages par défaut « *maison* » suivants via l'utilisation de `\tkzTabSetup`.

1. `arrowstyle = triangle 60` permet d'obtenir des pointes de flèche plus visibles.
2. `doubledistance = 3pt` améliore la visibilité des doubles barres pour les valeurs interdites.

Nous donnons quelques exemples classiques d'utilisation proches ou identiques de certains proposés dans la documentation de `tkz-tab` (les codes ont été mis en forme pour faciliter la compréhension de la syntaxe à suivre). Reportez vous à la documentation de `tkz-tab` pour des compléments d'information : vous y trouverez des réglages très fins.

Exemple 1 – Avec des signes

x	0	$\frac{\pi}{2}$	π
$\cos(x)$	+	0	−

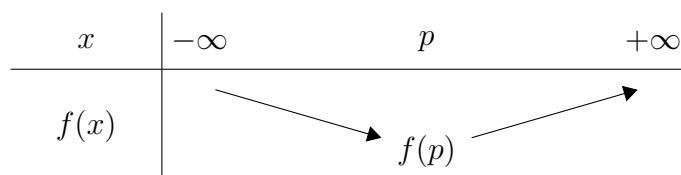
Le rendu précédent a été obtenu via le code suivant.

```

\begin{tikzpicture}
  \tkzTabInit{
    $x$ / 0.75 , % Facteur d'échelle de 0.75 pour la hauteur de la 1er ligne.
    $\cos(x)$ / 1 % Facteur d'échelle de 1 pour la hauteur de la 2e ligne.
  }{
    $0$ , $\frac{\pi}{2}$ , $\pi$ % Valeurs utiles de x.
  }
  \tkzTabLine{ , + , z , - , } % Signes et zéro.
\end{tikzpicture}

```

Exemple 2 – Avec des variations (sans cadre)



Le rendu précédent a été obtenu via le code suivant.

```

\begin{tikzpicture}
  \tkzTabInit[nocadre]{
    $x$ / 0.75 ,
    $f(x)$ / 1.5
  }{
    $-\infty$ , $p$ , $+\infty$
  }
  \tkzTabVar{+ / , - / $f(p)$ , + / } % Variations via position / valeur.
\end{tikzpicture}

```

Exemple 3 – Variations via une dérivée (sans cadre)

x	0	$\frac{\pi}{2}$	π
$\cos(x)$	+	0	−
$\sin(x)$	0	1	0

Le rendu précédent a été obtenu via le code suivant.

```

\begin{tikzpicture}
  \tkzTabInit[nocadre]{
    $x$ / 0.75 ,
    $\cos(x)$ / 1 ,
    $\sin(x)$ / 1.5
  }{
    $0$ , $\frac{\pi}{2}$ , $\pi$
  }
  \tkzTabLine{ , + , z , - , }
  \tkzTabVar {- / 0 , + / 1 , - / 0 }
\end{tikzpicture}

```

Exemple 4 – Une image intermédiaire avec une seule flèche

x	$-\infty$	0	$+\infty$
$3x^2$	+	0	+
x^3	$-\infty$	0	$+\infty$

Le rendu précédent a été obtenu via le code suivant.

```

\begin{tikzpicture}
  \tkzTabInit{
    $x$ / 0.75 ,
    $3 x^2$ / 1 ,
    $x^3$ / 1.5
  }{
    $-\infty$ , $0$ , $+\infty$
  }
  \tkzTabLine{ , + , 0 , + , }
  \tkzTabVar {- / $-\infty$ , R , + / $+\infty$}
  %
  \tkzTabIma{1}{3}{2} % Position entre les 1re et 3e valeurs puis rang relatif.
                     {$0$} % Valeur de l'image.
\end{tikzpicture}

```

Exemple 5 – Valeurs interdites et valeurs supplémentaires

x	0	1	e	$+\infty$
$\frac{1}{x}$			+	
ln				

Le rendu précédent a été obtenu via le code suivant.

```

\begin{tikzpicture}
  \tkzTabInit[espc1 = 6]{ % Largeur entre les valeurs du tableau.
    $x$ / 0.75 ,
    $\frac{1}{x}$ / 1.25 ,
    $\ln$ / 1.75
  }{
    $0$ , $+\infty$
  }
  \tkzTabLine{d , + , }
  \tkzTabVar {D- / $-\infty$ , + / $+\infty$}
  %
  \tkzTabVal{1}{2}{0.35} % Position entre les 1re et 2e valeurs puis en proportion.
    {1}{0} % x_1 et f(x_1)
  \tkzTabVal{1}{2}{0.65} % Position entre les 1re et 2e valeurs puis en proportion.
    {$\ee$}{1} % x_2 et f(x_2)
\end{tikzpicture}

```

Voici un autre exemple pour comprendre comment utiliser `\tkzTabVal` avec en plus l'option `draw` qui peut rendre service.

x	0	1	e	e^2	$+\infty$	
$f'(x)$		+	0	-		
$f(x)$		$-\infty$	$\frac{1}{e}$	e	1	0

Le rendu précédent a été obtenu via le code suivant.

```
\begin{tikzpicture}
  \tkzTabInit[espc1 = 4]{
    $x$ / 0.75 ,
    $f'(x)$ / 1 ,
    $f(x)$ / 1.5
  }{
    $0$ , $e$ , $+\infty$
  }
  \tkzTabLine{d , + , 0 , - , }
  \tkzTabVar {D- / $-\infty$ , + / $e$ , - / $0$ }
  %
  \tkzTabVal[draw]{1}{2}{0.5} % Position entre les 1re et 2e valeurs au milieu.
    {$1$}{$\frac{1}{e}$}
  \tkzTabVal[draw]{2}{3}{0.5} % Position entre les 2e et 3e valeurs au milieu.
    {$e^2$}{$1$}
\end{tikzpicture}
```

Exemple 6 – Signe à partir des variations (un peu de pédagogie...)

Il est assez facile de produire des choses très utiles pédagogiquement comme ce qui suit en se salissant un peu les mains avec du code TikZ.

x	$-\infty$	-1	2	3	$+\infty$
$f(x)$		-9		0	

On déduit du tableau précédent le signe de la fonction f .

x	$-\infty$		3	$+\infty$
$f(x)$		-	0	+

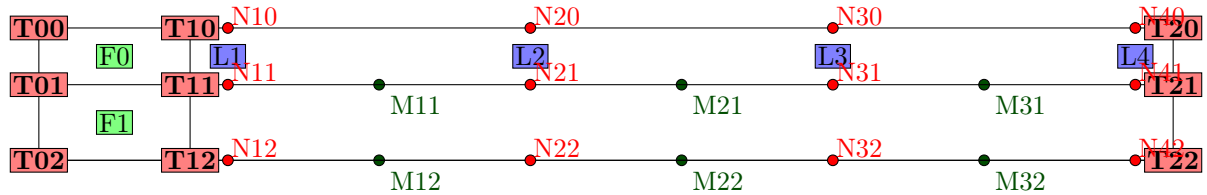
Voici le code du 1^{er} tableau où le placement de la valeur 3 au milieu entre 2 et $+\infty$ va nous simplifier le travail pour le 2^e tableau.

```

\begin{tikzpicture}
  \tkzTabInit[espc1 = 4]{
    $x$      / 0.75 ,
    $f(x)$   / 1.5
  }{
    $-\infty$ , $-1$      , $2$ , $+\infty$
  }
  \tkzTabVar{- /      , + / $-9$ , - / , + /      }
  %
  \tkzTabVal[draw]{3}{4}{0.5} % Position entre les 3e et 4e valeurs au milieu.
    {$3$}{$0$}
\end{tikzpicture}

```

Pour produire le code du 2^e tableau il a fallu utiliser au préalable ce qui suit en activant l'option `help` qui demande à `tkz-tab` d'afficher les noms de noeuds au sens TikZ qui ont été créés. Ceci permet alors d'utiliser ces noeuds pour des dessins TikZ faits maison ³.



Le rendu précédent a été obtenu via le code suivant.

```

\begin{tikzpicture}
  \tkzTabInit[
    espc1 = 4,
    help      % <-- Pour voir les noms des noeuds.
  ]{
    $x$      / 0.75 ,
    $f(x)$   / 1
  }{
    $-\infty$ , $-1$ , $2$ , $+\infty$
  }
\end{tikzpicture}

```

Maintenant que les noms des noeuds sont connus, il devient facile de produire le code ci-après. Bien noter l'usage de valeurs utiles « vides » de x ainsi que les mystiques `\node at ($(A)!0.5!(B)$)` permettant de placer un noeud au milieu entre les deux noeuds A et B.

3. C'est grâce à ce mécanisme que `tnsana` peut proposer des outils explicatifs des tableaux de signe : voir la section suivante.

```

\begin{tikzpicture}
  \tkzTabInit[
    espcl = 4,
    %      help      % <-- Pour voir les noms des noeuds.
  ]{
    $x$      / 0.75 ,
    $f(x)$ / 1
  }{
    $-\infty$ , , , $+\infty$ % ATTENTION ! Ne pas oublier de virgules.
  }
  %
  \node at ($(N31)!0.5!(N40)$){$3$};
  \node at ($(M31)!0.5!(M32)$){$0$};
  \node at ($(M31)!0.5!(N42)$){$+$};
  \node at ($(N11)!0.5!(M32)$){$-$};
\end{tikzpicture}

```

Exemple 7 – Convexité et concavité symbolisées dans les variations

Voici un autre exemple s'utilisant la machinerie TikZ afin d'indiquer dans les variations la convexité et la concavité via des flèches incurvées (*cette convention est proposée dans la sous-section « Exemple utilisant l'option `\help` » de la section « Galerie » de la documentation de `tkz-tab`*).

x	$-\infty$	x_1	0	x_2	$+\infty$
f''		$-$	0	$+$	
f'	$+\infty$	0	-2	0	$+\infty$
f					

Le code utilisé est le suivant.

```

\begin{tikzpicture}
  \tkzTabInit[
    espcl = 3,
    % help % <-- Pour voir les noms des noeuds.
  ]{
    $x$ / 0.75 ,
    $f'$ / 1 ,
    $f'$ / 2 ,
    $f$ / 3
  }{
    $-\infty$ , $0$ , $+\infty$
  }
  \tkzTabLine{ , - , z , + , }
  \tkzTabVar {+ / $+\infty$ , - / $-2$ , + / $+\infty$}

  \tkzTabVal[draw]{1}{2}{.5}{$x_1$}{$0$}
  \tkzTabVal[draw]{2}{3}{.5}{$x_2$}{$0$}

  \begin{scope}[->, > = triangle 60]
    \coordinate (Middle1) at ($(N13)!0.5!(N14)$);
    \coordinate (Middle2) at ($(N23)!0.5!(N24)$);
    \coordinate (Middle3) at ($(N33)!0.5!(N34)$);
    \path (N13) -- (N23) node[midway,below=6pt](N){};
    %
    \draw ([above=6pt]Middle1)
      to [bend left=45] ([left=1pt]N);
    \draw ([right=3pt]N)
      to [bend left=45] ([above=6pt]Middle2) ;
    \draw ([below right=3pt]Middle2)
      to [bend left=-45] ([above=6pt]M24) ;
    \draw ([above right=6pt]M24)
      to [bend right=40] ([below left=6pt]Middle3);
  \end{scope}
\end{tikzpicture}

```

b. Décorer facilement un tableau

i. Motivation

Considérons le tableau suivant et imaginons que nous voulions l'expliquer à un débutant.

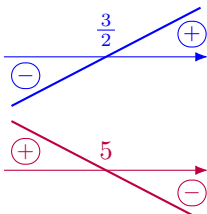
x	$-\infty$	$\frac{3}{2}$	5	$+\infty$	
Signe de $2x - 3$	$-$	0	$+$	$+$	
Signe de $-x + 5$	$+$	$+$	0	$-$	
Signe de $(2x - 3)(-x + 5)$	$-$	0	$+$	0	$-$

Deux options s'offrent à nous pour justifier comment a été rempli le tableau.

1. Classiquement on résout par exemple juste les deux inéquations $2x - 3 > 0$ et $-x + 5 > 0$ puis on complète les deux premières lignes⁴ pour en déduire la dernière via la règle des signes d'un produit.
2. On peut proposer une méthode moins sujette à la critique qui s'appuie sur la représentation graphique d'une fonction affine en produisant le tableau suivant.

x	$-\infty$	$\frac{3}{2}$	5	$+\infty$
Signe de $2x - 3$	-	0	+	+
Signe de $-x + 5$	+	+	0	-
Signe de $(2x - 3)(-x + 5)$	-	0	+	-

← Valeurs utiles de x



← Signe du produit.

Pour produire le 2^e tableau, en plus du code `tkz-tab` pour le tableau de signe⁵, il a fallu ajouter les lignes données ci-dessous où sont utilisées les macros `\backLine`, `\graphSign` et `\comLine` proposées par `tnsana` (la syntaxe simple à suivre sera expliquée dans les trois sections suivantes). Indiquons que les lignes pour les signes doivent utiliser un coefficient minimal de 1.5 pour la hauteur afin d'éviter la superposition des graphiques.

```
\begin{tikzpicture}
% ----- %
% -- Code tkz-tab pour les signes non reproduit ici -- %
% ----- %

\backLine{0, 3}

\comLine[gray]{0}{\leftarrow Valeurs utiles de $x$}

\graphSign      {1}{ax+b, ap}{\frac{3}{2}}
\graphSign[purple]{2}{ax+b, an}{5}

\comLine[gray]{3}{\leftarrow Signe du produit.}
\end{tikzpicture}
```

Remarque. Il est aussi possible de décorer une ligne de variation comme cela sera montré dans l'exemple 3 page 21.

4. Notons que cette approche est un peu scandaleuse car il faudrait en toute rigueur aussi résoudre $2x - 3 < 0$, $-x + 5 < 0$, $2x - 3 = 0$ et $-x + 5 = 0$. Personne ne le fait car l'on pense aux variations d'une fonction affine. Dans ce cas pourquoi ne pas juste utiliser ce dernier argument ? C'est ce que propose la 2^e méthode.

5. Nous avons utilisé les réglages optionnels `lgt = 3.5` et `espcl = 2.5` de `\tkzTabInit` pour avoir de la place dans la 1^{re} colonne pour le dernier produit et aussi réduire la largeur des colonnes pour les signes.

ii. Ajouter une couleur de fond à une ligne

La modification de la couleur de fond d'une ligne se fait via la macro `\backLine`⁶ pour `back-ground of the line` soit « *fond de la ligne* » en anglais. Cette macro possède un argument optionnel et un obligatoire.

- *L'argument optionnel : choix de la couleur de fond.*

Ci-dessus nous avons utilisé la couleur par défaut qui est `gray!30`.

- *L'argument obligatoire : les numéros de ligne séparés par des virgules.*

La numérotation des lignes commence à 0 comme en informatique. Ainsi `\backLine{0,3}` ajoute une couleur de fond à la ligne des valeurs utiles de x et à la 3^e ligne de signes, ou moins intuitivement à la $(3+1)$ ^e ligne du tableau.

iii. Commenter une ligne

L'ajout de commentaires courts se fait via la macro `\comLine` pour `com-ment a line` soit « *commenter une ligne* » en anglais. Cette macro possède un argument optionnel et deux obligatoires.

- *L'argument optionnel : choix de la couleur du texte.*

Ci-dessus nous avons utilisé `\comLine[gray]{0}{...}` pour avoir un texte en gris.

- *Le 1^{er} argument : le numéro de ligne (comme pour \backLine).*

- *Le 2^e argument : texte du commentaire.*

Par défaut aucun retour à la ligne n'est possible. Si besoin se reporter à l'exemple 3, page 21 section v., où est montré comment écrire sur plusieurs lignes.

iv. Graphiques pour expliquer des signes

Pour le moment, la macro `\graphSign` propose différents types de graphiques de fonctions dites de référence. Avant de voir ce qui est proposé rappelons que la convention est de prendre 0 pour numéro de la toute 1^{re} ligne contenant les valeurs utiles de la variable.

1. Fonctions affines non constantes avec une contrainte.

Pour les fonctions du type $f(x) = ax + b$ avec $a \neq 0$, nous devons connaître le signe de a et la racine r de f . Le codage est simple : considérons par exemple `\graphSign{2}{ax+b, an}{5}`.

- *1^{er} argument 2.*

Ceci indique d'ajouter le graphique dans la 2^e ligne de signes.

- *$ax+b$ dans le 2^e argument.*

Ce code sans espace indique une fonction affine.

- *an dans le 2^e argument.*

Ce code venant de **a négatif** ajoute la condition $a < 0$.

- *3^e argument 5.*

Ceci donne la racine.

Donc pour ajouter dans la 4^e ligne de signe le graphique de $f(x) = 3x$, on utilisera dans ce cas `\graphSign{4}{ax+b, ap}{0}` où **ap** pour **a positif** code la condition $a > 0$.

6. L'auteur de `tnsana` n'est absolument pas un fan de la casse en bosses de chameau mais par souci de cohérence avec ce que propose `tkz-tab` le nom `\backLine` a été proposé à la place de `\backline`.

2. Fonctions trinômiales du 2^e degré avec deux contraintes.

Pour les fonctions du type $f(x) = ax^2 + bx + c$ avec $a \neq 0$, en plus du signe de a nous devons connaître celui du discriminant $\Delta = b^2 - 4ac$, ce dernier pouvant être nul, sans oublier les racines réelles éventuelles. Voyons comment coder ce genre de chose via par exemple `\graphSign{5}{ax2+bx+c, an, dp}{r_1}{r_2}`.

- 1^{er} argument 5.

On indique la 5^e ligne de signes.

- `ax2+bx+c` dans le 2^e argument.

Ce code sans espace indique un trinôme du 2^e degré.

- `an` dans le 2^e argument (comme avant).

- `dp` dans le 2^e argument.

Ce code venant de discriminant positif ajoute la condition $\Delta > 0$. En plus de `dn` et `dp` il y a aussi `dz` pour discriminant zéro.

- 3^e et 4^e arguments r_1 et r_2 .

Ceci donne les deux racines réelles avec obligatoirement $r_1 < r_2$.

Ainsi pour indiquer dans la 3^e ligne de signe la courbe relative à $f(x) = -4x^2$, on utilisera `\graphSign{3}{ax2+bx+c, an, dz}{0$}`.

Enfin le graphique associé au trinôme $f(x) = 7x^2 + 3$, qui est sans racine réelle, s'obtiendra dans la 4^e ligne de signe via `\graphSign{4}{ax2+bx+c, ap, dn}`.

3. Fonctions sans contrainte.

Voici ce qui est disponible via `\graphSign{noligne}{codefunc}` où les valeurs possibles de `codefunc` sont les suivantes.

codefunc	x2	sqrt	1/x	abs	exp	ln
$f(x)$	x^2	\sqrt{x}	$\frac{1}{x}$	$ x $	$\exp x$	$\ln x$

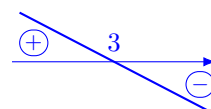
v. Quelques exemples

Exemple 1 – Avec une parabole

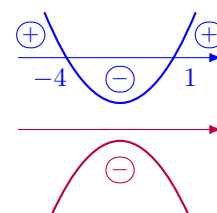
Il devient très facile de proposer un tableau décoré comme le suivant.

x	$-\infty$	-4	1	3	$+\infty$		
Signe de $-x + 3$	$+$	$+$	$+$	0	$-$		
Signe de $f(x)$	$-$	0	$+$	0	$+$	0	$-$
Signe de $x^2 + 3x - 4$	$+$	0	$-$	0	$+$	$+$	
Signe de $-x^2 + x - 4$	$-$	$-$	$-$	$-$	$-$		
Signe du produit	$+$	0	$+$	0	$-$	0	$-$

Schémas



Voir Q.1-a)



← Conclusion

En plus des deux exemples de schémas de paraboles, il faut noter dans le code supplémentaire ajouté l'utilisation de `\kern1.75em` dans `\comLine[gray]{0}{\kern1.75em Schémas}` afin de mettre un espace horizontal précis pour centrer à la main le texte « *Schémas* » (un peu sâle mais ça marche).

```
\begin{tikzpicture}
% ----- %
% -- Code tkz-tab pour les signes non reproduit ici -- %
% ----- %

\backLine{0, 5}

\comLine[gray]{0}{\kern1.75em Schémas}

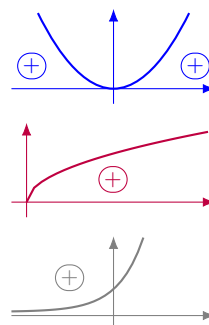
\graphSign      {1}{ax+b, an}{\$3\$}
\comLine[purple]{2}{Voir Q.1-a)}
\graphSign      {3}{ax2+bx+c, ap, dp}{\$-4\$}{\$1\$} % Deux racines réelles.
\graphSign[purple]{4}{ax2+bx+c, an, dn}              % Aucune racine réelle.

\comLine[gray]{5}{\$ \leftarrow Conclusion}
\end{tikzpicture}
```

Exemple 2 – Avec des fonctions sans paramètre

Voici un 1^{er} tableau avec certaines des fonctions sans paramètre.

x	0	$+\infty$
x^2	0	+
\sqrt{x}	0	+
$\exp x$		+
$x^2 \sqrt{x} \exp x$	0	+



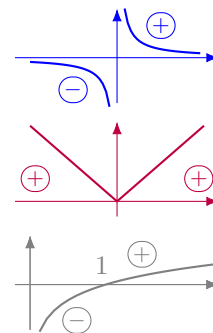
Le code correspondant est le suivant.

```
\begin{tikzpicture}
% ----- %
% -- Code tkz-tab pour les signes non reproduit ici -- %
% ----- %

\graphSign      {1}{x2}
\graphSign[purple]{2}{sqrt}
\graphSign[gray]{3}{exp}
\end{tikzpicture}
```

Voici un 2^e tableau avec les fonctions sans paramètre manquantes ci-dessus.

x	0	1	$+\infty$
$\frac{1}{x}$		+	+
$ x $	0	+	+
$\ln x$		-	+
$\frac{ x }{x \ln x}$		-	+



Le code correspondant est le suivant.

```
\begin{tikzpicture}
% ----- %
% -- Code tkz-tab pour les signes non reproduit ici -- %
% ----- %

\graphSign      {1}{1/x}
\graphSign[purple]{2}{abs}
\graphSign[gray] {3}{ln}
\end{tikzpicture}
```

Exemple 3 – Commenter des variations

Pour finir, indiquons que les outils de décoration marchent aussi pour les tableaux de variation. Voici un exemple possible d'utilisation où les retours à la ligne ont été obtenus affreusement, ou pas, via `\parbox{11.5em}{...}`.

x	0	e	$+\infty$
$f'(x)$	+	0	-
$f(x)$	$-\infty$	e	0

Sur un intervalle le signe de $f'(x)$ implique les variations de $f(x)$.

Les limites sont hors programme pour cette année.

12. Calcul intégral

a. Le symbole standard revisité

Commençons par un point important : le package réduit les espacements entres des symboles \int successifs. Voici un exemple.

```
\displaystyle
\int \int \int
F(x;y;z) \dd{x} \dd{y} \dd{z}
```

$$\iiint F(x;y;z) \, dx \, dy \, dz$$

```
\displaystyle
\int_{a}^{b} \int_{c}^{d} \int_{e}^{f}
F(x;y;z) \dd{x} \dd{y} \dd{z}
```

$$\int_a^b \int_c^d \int_e^f F(x;y;z) \, dx \, dy \, dz$$

Remarque. Par défaut, L^AT_EX affiche $\int \int \int F(x;y;z) \, dx \, dy \, dz$ et $\int_a^b \int_c^d \int_e^f F(x;y;z) \, dx \, dy \, dz$. Nous avons obtenu ce résultat en utilisant `\stdint` qui est l'opérateur proposé de façon standard par L^AT_EX.

b. Un opérateur d'intégration clés en main

Exemple 1 – À quoi bon ?

Le 1^{er} exemple qui suit semblera être une hérésie pour les habitués de L^AT_EX mais rappelons que le but de **tnsana** est de rendre les documents facilement modifiables globalement ou localement comme le montre le 2^e exemple.

```
\displaystyle
\integrate{f(x)}{x}{a}{b}
= \int_{x=a}^{x=b} f(x) \dd{x}
```

$$\int_{x=a}^{x=b} f(x) \, dx = \int_{x=a}^{x=b} f(x) \, dx$$

```
\displaystyle
\integrate*{f(x)}{x}{a}{b}
= \integrate{f(x)}{x}{a}{b}
```

$$\int_a^b f(x) \, dx = \int_{x=a}^{x=b} f(x) \, dx$$

Exemple 2 – Le mode displaystyle

La macro `\dintegrate*` présentée ci-dessous possède aussi une version non étoilée `\dintegrate`.

```
\dintegrate*{f(x)}{x}{a}{b}
= \integrate*{f(x)}{x}{a}{b}
```

$$\int_a^b f(x) \, dx = \int_a^b f(x) \, dx$$

c. L'opérateur « crochet »

Exemple 1

```
\hook{F(x)}{x}{a}{b}
= F(b) - F(a)
```

$$[F(x)]_{x=a}^{x=b} = F(b) - F(a)$$

```
\dintegrate*{f(x)}{x}{a}{b}
= \hook*{F(x)}{x}{a}{b}
```

$$\int_a^b f(x) \, dx = [F(x)]_a^b$$

Remarque. Il faut savoir que `\hook` signifie « *crochet* » en anglais mais la bonne traduction du terme mathématique est en fait « *square bracket* ». Ceci étant dit l'auteur de **tnsana** trouve plus efficace d'utiliser `\hook` comme nom de macro.

Exemple 2 – Des crochets non extensibles

Dans l'exemple suivant, on utilise l'option `sb` pour `s`-mall `b`-rackets soit « *petits crochets* » en anglais. Les options sont disponibles à la fois pour `\hook` et `\hook*`.

<pre> $\hook*\{\dfrac{x - 1}{5 + x^2}\}{x}\%$ $\hook*[sb]\%$ $\{\dfrac{x - 1}{5 + x^2}\}{x}\%$ $\{a\}{b}\\$ </pre>	$\left[\frac{x - 1}{5 + x^2} \right]_a^b = \left[\frac{x - 1}{5 + x^2} \right]_a^b$
---	---

Exemple 3 – Un trait vertical épuré

Via les options `r` et `sr` pour `s`-mall et `r`-ull soit « *petit* » et « *trait* » en anglais, on obtient ce qui suit.

<pre> $\hook[r] \{\dfrac{x - 1}{5 + x^2}\}{x}\%$ $\hook*[sr]\{\dfrac{x - 1}{5 + x^2}\}{x}\%$ $\{a\}{b}\\$ </pre>	$\frac{x - 1}{5 + x^2} \Big _{x=a}^{x=b} = \frac{x - 1}{5 + x^2} \Big _a^b$
--	---

13. Historique

Nous ne donnons ici qu'un très bref historique récent ⁷ de **tnsana** à destination de l'utilisateur principalement. Tous les changements sont disponibles uniquement en anglais dans le dossier **change-log** : voir le code source de **tnsana** sur **github**.

2020-07-30 Nouvelle version mineure **0.6.0-beta**.

- **DÉFINITION EXPLICITE D'UNE FONCTION.**
 - Une nouvelle macro `\txfuncdef` produit une version textuelle courte.
 - Omission possible des ensembles, via des arguments vides, quand on utilise `\funcdef [h]` ou `\txfuncdef`.
- **FONCTIONS AVEC UN PARAMÈTRE :** les macros `\expb` et `\logb` ont été remplacées par `\exp` et `\log` qui ont un argument optionnel pour indiquer éventuellement une base.
- **DÉRIVATION PARTIELLE :** `\pder [ei]` fonctionne maintenant aussi avec des variables indexées.

2020-07-22 Nouvelle version mineure **0.5.0-beta**.

- **DÉFINITION EXPLICITE D'UNE FONCTION :** ajout de `\funcdef`.

2020-07-21 Nouvelle version mineure **0.4.0-beta**.

- **LIMITE :** ajout de `\limit` pour l'écriture de limites de fonctions à une seule variable.
- **DÉRIVATION :** par souci de cohérence, il faudra taper `\der{f}{x}{n}` au lieu de l'ancien `\der{f}{n}{x}`.
- **INTÉGRATION :** par souci de cohérence, il faudra taper `\integrate{f}{x}{a}{b}` au lieu de l'ancien `\integrate{a}{b}{f}{x}`. Il en va de même pour `\hook`.

2020-07-17 Nouvelle version mineure **0.3.0-beta**.

- **DÉRIVATION.**
 - Dérivation pointée à la physicienne via `d` et `bd` deux nouvelles options de `\der`.
 - La dérivation partielle indexée du type u_{xxy} à la physicienne via `ei` une nouvelle option de `\pder`.
- **TABLEAUX DE SIGNE ET DE VARIATION.**
 - Ajout de `\backLine` pour changer la couleur de fond d'une ou plusieurs lignes.
 - `\graphSign` propose des fonctions de référence (*sans paramètre*).

2020-07-15 Nouvelle version mineure **0.2.0-beta**.

7. On ne va pas au-delà de un an depuis la dernière version.

- **SYMBOLES** : les nouvelles macros `\symvar` et `\symvar*` produisent un disque plein et un carré plein permettant par exemple d'indiquer symboliquement une ou des variables.
-

2020-07-12 Nouvelle version mineure 0.1.0-beta.

- **FONCTIONS NOMMÉES** : `\ppcm` et `\pgcd` ont été déplacées dans `tnsarith` disponible sur <https://github.com/typensee-latex/tnsarith.git>.
-

2020-07-10 Première version 0.0.0-beta.

14. Toutes les fiches techniques

a. Constantes et paramètres

i. Constantes classiques ajoutées

<code>\ggamma</code>	<code>\ii</code>
<code>\ppi</code>	<code>\jj</code>
<code>\tttau</code>	<code>\kk</code>
<code>\ee</code>	

ii. Constantes latines personnelles

`\param{#1}`

— Argument: un texte utilisant l’alphabet latin.

b. Une variable « symbolique »

`\symvar[#opt]`

— Option: le numéro du symbole qui vaut 1 par défaut.

- 1 donne •.
- 2 donne ◦.
- 3 donne ■.

c. La fonction valeur absolue

`\abs {#1}`

`\abs*{#1}`

— Argument: l’expression sur laquelle appliquer la fonction valeur absolue.

d. Fonctions nommées spéciales

i. Sans paramètre

`\acos`

`\asin`

`\atan`

`\arccosh`

`\arcsinh`

`\arctanh`

`\acosh`

`\asinh`

`\atanh`

`\fch`

`\fsh`

f = f-rench

f = f-rench

`\fth`

`f = f-rench`

`\afch`

`\afsh`

`\afth`

ii. Avec un paramètre

`\exp[#opt]`

— Option: la base de l'exponentielle

`\log[#opt]`

— Option: la base du logarithme

e. Définition explicite d'une fonction

`\funcdef[#opt]{#1..#5}`

— Option: la valeur par défaut `u`.

1. `u` : écriture empilée avec un trait vertical.
2. `s` : écriture empilée sans trait vertical.
3. `h` : écriture horizontale en ligne.

— Argument 1: la fonction.

— Argument 2: la variable.

— Argument 3: la formule explicite de définition.

— Argument 4: l'ensemble de départ. Cet argument peut être vide si le mode `h` est activé

— Argument 5: l'ensemble d'arrivée.

`\txtfuncdef{#1..#5}`

— Arguments 1..5: voir les explications données ci-dessus pour la macro `\funcdef`.

f. Limite

`\limit[#opt]{#1..#3}`

— Option: la valeur par défaut `asit` n'a pas vocation à être utilisée.

1. `asit` : rien n'est appliqué sur la fonction qui reste donc tel quelle.
2. `p` : ajout de parenthèses extensibles autour de la fonction.
3. `sp` : ajout de parenthèses non extensibles autour de la fonction.

— Argument 1: la fonction dont on indique la limite.

— Argument 2: la variable qui va tendre vers quelque chose.

— Argument 3: la valeur limite suivie éventuellement de conditions en utilisant la barre verticale `|` pour séparer ces différentes informations.

g. Calcul différentiel

i. Les opérateurs ∂ et d

`\dd [#opt] {#1}`

`\pp [#opt] {#1}`

— **Option**: utilisée, cette option sera mise en exposant du symbole ∂ ou d .

— **Argument**: la variable de différentiation à droite du symbole ∂ ou d .

ii. Dérivations totales d'une fonction – Version longue avec une variable

`\der [#opt] {#1..#3}`

— **Option**: la valeur par défaut est `u`.

1. `u` : écriture usuelle avec des primes (*ceci nécessite d'avoir une valeur entière naturelle connue du nombre de dérivations successives*).
2. `e` : écriture via un exposant entre des parenthèses.
3. `i` : écriture via un indice.
4. `d` : écriture pointée à la physicienne (*cf. la dérivation par rapport au temps*).
5. `bd` : écriture pointée avec un crochet entre les points et la fonction.
6. `f` : écriture via une fraction en mode display.
7. `sf` : écriture via une fraction en mode non display.
8. `of` : écriture via une fraction en mode display sous la forme d'un opérateur (*la fonction est à côté de la fraction*).
9. `osf` : écriture via une fraction en mode non display sous la forme d'un opérateur (*la fonction est à côté de la fraction*).
10. `p` : ajout de parenthèses extensibles autour de la fonction.
11. `sp` : ajout de parenthèses non extensibles autour de la fonction.

— **Argument 1**: la fonction à dériver.

— **Argument 2**: la variable de dérivation.

— **Argument 3**: l'ordre de dérivation.

iii. Dérivations totales d'une fonction – Version courte sans variable

`\sder [#opt] {#1..#2}`

`s = s-imple`

— **Option**: la valeur par défaut est `u`. Les options disponibles sont `u`, `e`, `d`, `bd`, `p` et `sp` : voir la fiche technique de `\sder` ci-dessus.

— **Argument 1**: la fonction à dériver.

— **Argument 2**: l'ordre de dérivation.

iv. L'opérateur de dérivation totale

`\derope [#opt] {#1..#2}`

`ope = ope-rator`

— **Option**: la valeur par défaut est `f`. Les options disponibles sont `f`, `sf` et `i` : voir la fiche technique de `\der` donnée un peu plus haut.

— **Argument 1**: la fonction à dériver.

— **Argument 2**: l'ordre de dérivation.

v. Dérivations partielles

`\pder [#opt] {#1..#2}`

p = p-artial

— Option: la valeur par défaut est f.

1. f : écriture via une fraction en mode display.
2. sf : écriture via une fraction en mode non display.
3. of : écriture via une fraction en mode display sous la forme d'un opérateur (*la fonction est à côté de la fraction*).
4. osf : écriture via une fraction en mode non display sous la forme d'un opérateur (*la fonction est à côté de la fraction*).
5. i : écriture via un indice.
6. ei : écriture via un indice mais en « *développant* ».
7. p : ajout de parenthèses extensibles autour de la fonction.
8. sp : ajout de parenthèses non extensibles autour de la fonction.

— Argument 1: la fonction à dériver.

— Argument 2: les variables utilisées avec leur ordre de dérivation pour la dérivation partielle en utilisant une syntaxe du type `x | y^2 | ...` qui indique de dériver suivant x une fois, puis suivant y deux fois... etc.

— Argument 3: l'ordre total de dérivation.

vi. L'opérateur de dérivation partielle

`\pderope [#opt] {#1..#2}`

p = p-artial et ope = ope-rator

— Option: la valeur par défaut est f. Les options disponibles sont f, sf et i : voir la fiche technique de `\pder` juste avant.

— Argument 1: les variables utilisées avec leur ordre de dérivation via la syntaxe indiquée ci-dessus.

— Argument 2: l'ordre total de dérivation.

h. Tableaux de variation et de signe

i. Coloriser le fond

`\backLine [#opt] {#1}`

back = back-ground

— Option: couleur au format TikZ. La valeur par défaut est gray!30.

— Argument 1: les numéros de ligne séparés par des virgules, 0 étant le 1^{er} numéro.

ii. Commentaires

`\comLine [#opt] {#1..#2}`

com = com-ment

— Option: couleur au format TikZ. La valeur par défaut est blue.

— Argument 1: le numéro de ligne, 0 étant le 1^{er} numéro.

— Argument 2: le texte du commentaire.

iii. Graphiques explicatifs

`\graphSign [#opt] {#1..#4}`

— **Option**: couleur au format TikZ. La valeur par défaut est `blue`.

— **Argument 1**: le numéro de ligne, 0 étant le 1^{er} numéro.

— **Argument 2**: le type de fonctions avec des contraintes éventuelles en utilisant la virgule comme séparateur d'informations.

1. `x2` sans espace indique $f(x) = x^2$.
2. `sqrt` sans espace indique $f(x) = \sqrt{x}$.
3. `1/x` sans espace indique $f(x) = \frac{1}{x}$.
4. `abs` sans espace indique $f(x) = |x|$.
5. `exp` sans espace indique $f(x) = \exp x$.
6. `ln` sans espace indique $f(x) = \ln x$.
7. `ax+b` sans espace indique $f(x) = ax + b$ avec $a \neq 0$ à caractériser.
8. `ax2+bx+c` sans espace indique $f(x) = ax^2 + bx + c$ avec $a \neq 0$ et le discriminant d à caractériser.
9. `ap` et `an` indiquent respectivement les conditions $a > 0$ et $a < 0$.
10. `dp`, `dz` et `dn` indiquent respectivement les conditions $d > 0$, $d = 0$ et $d < 0$.

— **Argument 3** supplémentaire pour `ax+b`: la racine réelle de $ax + b$.

— **Arguments** supplémentaires éventuels pour `ax2+bx+c`: si $ax^2 + bx + c$ admet une ou deux racines réelles, on donnera toutes les racines de la plus petite à la plus grande⁸.

i. Calcul intégral

i. Le symbole standard revisité

`\stdint` (pour retrouver le comportement par défaut)

ii. Un opérateur d'intégration clés en main

`\integrate {#1..#4}`

`\integrate* {#1..#4}`

`\dintegrate {#1..#4}`

`d = d-isplaystyle`

`\dintegrate*{#1..#4}`

— **Argument 1**: la fonction intégrée.

— **Argument 2**: la variable d'intégration.

— **Argument 3**: valeur initiale d'intégration qui apparaît en bas du symbole \int_{\bullet} .

— **Argument 4**: valeur finale d'intégration qui apparaît en haut du symbole \int^{\bullet} .

iii. L'opérateur « crochet »

`\hook [#opt] {#1..#4}`

`\hook* [#opt] {#1..#4}`

— **Option**: la valeur par défaut est `b`. Voici les différentes valeurs possibles.

8. Notant $\Delta = b^2 - 4ac$, si $\Delta < 0$ il n'y aura pas d'argument supplémentaire, si $\Delta = 0$ il y en aura un seul et enfin si $\Delta > 0$ il faudra en donner deux, le 1^{er} étant le plus petit.

1. **b** : des crochets extensibles sont utilisés.
2. **sb** : des crochets non extensibles sont utilisés.
3. **r** : un unique trait vertical extensible est utilisé à droite.
4. **sr** : un unique trait vertical non extensible est utilisé à droite.

— **Argument 1**: la fonction sur laquelle effectuer le calcul.

— **Argument 2**: la variable à affecter pour les calculs.

— **Argument 3**: valeur initiale à substituer qui apparait en bas du crochet fermant ou de la barre verticale.

— **Argument 4**: valeur finale à substituer qui apparait en haut du crochet fermant ou de la barre verticale.