

Le package `tnsmath` :
des formules plus sémantiques

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

Version **1.7.0-beta** développée et testée sur Mac OS X.

Christophe BAL

2020-08-06

Table des matières

A. Généralités	11
I. Introduction	11
II. Beta-dépendance	11
III. Comment lire cette documentation ?	11
IV. A propos des macros	12
1. Règles de nommage	12
i. Les macros de même « type »	12
ii. Les formes « négatives » des macros	12
iii. Les macros en mode <code>displaystyle</code>	12
iv. Les macros standards redéfinies	12
v. Casse utilisée pour les lettres	12
2. Les arguments : deux conventions à connaître	13
i. Avec un nombre fixé d'arguments	13
ii. Avec un nombre variable d'arguments	13
iii. Des arguments pas toujours affichés	13
V. Couleurs	13
VI. Packages utilisés	13
B. Théorie générale des ensembles	15
I. Ensembles	15
1. Ensembles versus accolades	15
2. Ensembles pour la géométrie	15
3. Ensembles probabilistes	16
4. Ensembles pour l'algèbre générale	16
5. Ensembles classiques en mathématiques et en informatique théorique	16
i. La liste complète	16
ii. Ensembles classiques suffixés	17
iii. Des suffixes à la carte	17
II. Intervalles	17
1. Intervalles réels - Notation française (?)	17
2. Intervalles réels – Notation américaine	18
3. Intervalles discrets d'entiers	18
III. Unions et intersections en mode ligne	18
IV. Applications	19
1. Cardinal, image et compagnie	19
2. Application totale, partielle, injective, surjective et/ou bijective	20
3. Fonction identité	20
4. Fonction caractéristique	21
5. Définition explicite d'une fonction	21
6. Composition	22

C. Méthodes formelles en logique	23
I. Espace après la négation logique	23
II. Différents types de comparaisons « standard »	23
1. Définir quelque chose	23
2. Indiquer une identité	23
3. Une égalité à vérifier ou non, une hypothèse, une condition	24
4. Une égalité indiquant le choix d'une valeur ou l'application d'une relation	24
5. Une égalité indiquant l'équation d'une courbe	24
6. Différents types d'inéquations	24
7. Des formes négatives aussi pour les inéquations	24
8. Une table récapitulative	25
9. Textes utilisés	25
III. Équivalences et implications	25
1. Des symboles logiques supplémentaires	25
2. Une table récapitulative	25
3. Équivalences et implications verticales	26
4. Tables des décorations possibles des opérateurs	26
IV. Des versions alternatives du quantificateur existentiel	27
1. Quantifier l'existence	27
2. Versions négatives	27
V. Détailler un raisonnement simple	27
1. Version pour le lycée et après	27
2. Version pour les collégiens	31
3. De courts commentaires	33
4. Un mini hack très utile pour des « <i>étapes alignées</i> »	35
5. Un conseil de mise en forme	35
VI. Détailler un « vrai » raisonnement	36
1. Un tableau pour le post-bac	36
2. Un tableau sur plusieurs pages	38
3. Un tableau pour le collège et le lycée	38
4. Un tableau sur plusieurs pages	39
D. Géométrie	41
I. Ensembles géométriques	41
II. Utiliser des unités S.I.	41
III. Points et lignes	41
1. Points	41
2. Lignes	42
3. Droites parallèles ou non	42
IV. Vecteurs	43
1. Les écrire	43
2. Norme	43
3. Produit scalaire	43
4. 3D – Produit vectoriel	44
i. Écriture symbolique	44
ii. Explication du mode de calcul	45
iii. Les coordonnées	46
5. 2D – Critère de colinéarité de deux vecteurs	47
6. 2D – Déterminant de deux vecteurs	47
V. Géométrie cartésienne	49

1.	Coordonnées	49
2.	Nommer un repère	50
VI.	Arcs circulaires	51
VII.	Angles	52
1.	Angles géométriques « intérieurs »	52
2.	Angles orientés de vecteurs	52
E.	Analyse	55
I.	Constantes et paramètres	55
1.	Constantes classiques ajoutées	55
2.	Constantes latines personnelles	55
II.	Une variable « symbolique »	55
III.	La fonction valeur absolue	56
IV.	Fonctions nommées spéciales	56
1.	Sans paramètre	56
2.	Avec un paramètre	56
3.	Toutes les fonctions nommées en plus	56
V.	Limite	56
VI.	Calcul différentiel	57
1.	Les opérateurs ∂ et d	57
2.	Dérivations totales d'une fonction – Version longue avec une variable	57
3.	Dérivations totales d'une fonction – Version courte sans variable	59
4.	L'opérateur de dérivation totale	59
5.	Dérivations partielles	60
6.	L'opérateur de dérivation partielle	60
VII.	Tableaux de variation et de signe	61
1.	Les bases de tkz-tab	61
2.	Décorer facilement un tableau	67
i.	Motivation	67
ii.	Ajouter une couleur de fond à une ligne	69
iii.	Commenter une ligne	69
iv.	Graphiques pour expliquer des signes	69
v.	Quelques exemples	70
VIII.	Calcul intégral	72
1.	Le symbole standard revisité	72
2.	Un opérateur d'intégration clés en main	73
3.	L'opérateur « crochet »	73
F.	Suites	75
I.	Des notations complémentaires pour des suites spéciales	75
II.	Sommes et produits en mode ligne	75
III.	Comparaison asymptotique de suites et de fonctions	75
1.	Les notations \mathcal{O} et \mathcal{o}	75
2.	La notation Ω	76
3.	La notation Θ	76
G.	Probabilité	77
I.	Ensembles classiques de nombres	77
II.	Généralités	77
1.	Probabilité « simple »	77
2.	Probabilité conditionnelle	77

3.	Évènement contraire	78
4.	Espérance, variance et écart-type	78
III.	Arbres pondérés	78
1.	Au commencement était la forêt...	78
2.	Les bases	79
3.	Commenter les racines	81
4.	Avec des cadres	82
5.	Mettre en valeur des chemins	83
6.	Utiliser les noms automatiques donnés par <code>forest</code>	85
H. Arithmétique		87
I.	Ensembles classiques	87
II.	Opérateurs de base	87
III.	Fonctions nommées spéciales	87
IV.	Fractions continuées	88
1.	Fractions continuées standard	88
2.	Fractions continuées généralisées	88
3.	Comme une fraction continuée isolée	88
4.	L'opérateur \mathcal{K}	88
I. Algèbre linéaire		91
I.	Matrices via <code>nicematrix</code>	91
1.	Quelques exemples pour bien démarrer	91
2.	Calculs expliqués des déterminants 2×2	94
J. Polynômes et séries formelles		95
I.	Ensembles classiques de nombres	95
II.	Polynômes	95
III.	Séries formelles classiques	95
IV.	Polynômes et séries formelles de Laurent	96
K. Historique		97
L. Toutes les fiches techniques		107
I.	Théorie générale des ensembles	107
1.	Ensembles	107
i.	Ensembles versus accolades	107
2.	Ensembles	107
i.	Ensembles pour la géométrie	107
3.	Ensembles	107
i.	Ensembles probabilistes	107
4.	Ensembles	108
i.	Ensembles pour l'algèbre générale	108
5.	Ensembles	108
i.	Ensembles classiques en mathématiques et en informatique théorique	108
6.	Ensembles	109
i.	Ensembles classiques en mathématiques et en informatique théorique	109
7.	Intervalles	109
i.	Intervalles réels - Notation française (?)	109
8.	Intervalles	109
i.	Intervalles réels - Notation américaine	109

9.	Intervalles	110
i.	Intervalles discrets d'entiers	110
10.	Unions et intersections en mode ligne	111
11.	Applications	111
i.	Cardinal, image et compagnie	111
12.	Applications	111
i.	Application totale, partielle, injective, surjective et/ou bijective . . .	111
13.	Applications	111
i.	Fonction identité	111
14.	Applications	111
i.	Fonction caractéristique	111
15.	Applications	111
i.	Définition explicite d'une fonction	111
16.	Applications	112
i.	Composition	112
II.	Méthodes formelles en logique	112
1.	Espace après la négation logique	112
2.	Différents types de comparaisons « standard »	112
i.	Opérateurs décorés – Les textes	112
ii.	Opérateurs de comparaison supplémentaires	112
3.	Équivalences et implications	113
i.	Des symboles logiques supplémentaires	113
4.	Équivalences et implications	114
i.	Équivalences et implications verticales	114
5.	Des versions alternatives du quantificateur existentiel	114
6.	Détailler un raisonnement simple	114
i.	Détailler un raisonnement simple	114
ii.	Détailler un raisonnement simple – Mise en forme du texte	115
7.	Détailler un « vrai » raisonnement	116
i.	Détailler un « vrai » raisonnement via un tableau	116
ii.	Détailler un « vrai » raisonnement via un tableau - Textes utilisés . .	117
III.	Géométrie	117
1.	Points et lignes	117
i.	Points	117
2.	Points et lignes	117
i.	Lignes	117
3.	Points et lignes	118
i.	Droites parallèles ou non	118
4.	Vecteurs	118
i.	Les écrire	118
5.	Vecteurs	118
i.	Norme	118
6.	Vecteurs	118
i.	Produit scalaire	118
7.	Vecteurs	119
i.	3D – Produit vectoriel	119
8.	Vecteurs	119
i.	3D – Produit vectoriel	119
9.	Vecteurs	120
i.	3D – Produit vectoriel	120

10.	Vecteurs	120
i.	2D – Critère de colinéarité de deux vecteurs	120
11.	Vecteurs	121
i.	2D – Déterminant de deux vecteurs	121
12.	Géométrie cartésienne	121
i.	Coordonnées	121
13.	Géométrie cartésienne	122
i.	Nommer un repère	122
14.	Arcs circulaires	123
15.	Angles	123
i.	Angles géométriques « intérieurs »	123
16.	Angles	123
i.	Angles orientés de vecteurs	123
IV.	Analyse	124
1.	Constantes et paramètres	124
i.	Constantes classiques ajoutées	124
2.	Constantes et paramètres	124
i.	Constantes latines personnelles	124
3.	Une variable « symbolique »	124
4.	La fonction valeur absolue	124
5.	Fonctions nommées spéciales	124
i.	Sans paramètre	124
6.	Fonctions nommées spéciales	125
i.	Avec un paramètre	125
7.	Limite	125
8.	Calcul différentiel	125
i.	Les opérateurs ∂ et d	125
ii.	Dérivations totales d'une fonction – Version longue avec une variable	126
iii.	Dérivations totales d'une fonction – Version courte sans variable	126
iv.	L'opérateur de dérivation totale	126
v.	Dérivations partielles	126
vi.	L'opérateur de dérivation partielle	127
9.	Tableaux de variation et de signe	127
i.	Coloriser le fond	127
ii.	Commentaires	127
iii.	Graphiques explicatifs	127
10.	Calcul intégral	128
i.	Le symbole standard revisité	128
11.	Calcul intégral	128
i.	Un opérateur d'intégration clés en main	128
12.	Calcul intégral	128
i.	L'opérateur « crochet »	128
V.	Suites	129
1.	Des notations complémentaires pour des suites spéciales	129
2.	Sommes et produits en mode ligne	129
3.	Comparaison asymptotique de suites et de fonctions	129
i.	Les notations \mathcal{O} et \mathcal{o}	129
ii.	La notation Ω	129
iii.	La notation Θ	130
VI.	Probabilité	130

1.	Généralités	130
i.	Probabilité « simple »	130
2.	Généralités	130
i.	Probabilité conditionnelle	130
3.	Généralités	130
i.	Évènement contraire	130
4.	Généralités	130
i.	Espérance, variance et écart-type	130
5.	Arbres pondérés	131
VII.	Arithmétique	131
1.	Opérateurs de base	131
2.	Fonctions nommées spéciales	132
3.	Fractions continuées	132
i.	Fractions continuées standard	132
4.	Fractions continuées	132
i.	Fractions continuées généralisées	132
5.	Fractions continuées	132
i.	Comme une fraction continuée isolée	132
6.	Fractions continuées	132
i.	L'opérateur \mathcal{K}	132
VIII.	Algèbre linéaire	133
1.	Matrices via <code>nicematrix</code>	133
i.	Calculs expliqués des déterminants 2×2	133
IX.	Polynômes et séries formelles	133
1.	Polynômes	133
2.	Séries formelles classiques	133
3.	Polynômes et séries formelles de Laurent	133

Chapitre A.

Généralités

I. Introduction

L^AT_EX est un excellent langage, pour ne pas dire le meilleur, pour rédiger des documents contenant des formules mathématiques. Malheureusement toute la puissance de L^AT_EX permet d'écrire des codes très peu sémantiques. Le modeste but du package `tnsmath` est de fournir quelques macros sémantiques¹ pour la rédaction de documents mathématiques élémentaires. Considérons le code L^AT_EX suivant.

```
Sachant que  $f'(x) = \cos(x^2)$  sur  $[a ; b]$  , nous avons :  

$$\int_a^b \cos(x^2) dx = \left[ f(x) \right]_{x=a}^{x=b}.$$

```

Avec `tnsmath`, vous pouvez écrire le code suivant.

```
Sachant que  $\sder{f}(x) = \cos(x^2)$  sur  $\intervalC{a}{b}$ , nous avons :  

$$\dintegrate*\cos(x^2){x}{a}{b} = \hook{f(x)}{x}{a}{b}.$$

```

Même si certaines commandes sont plus longues à écrire que ce que permet L^AT_EX, il y a des avantages à utiliser des commandes sémantiques.

1. La mise en forme du document devient consistante.
2. Il est facile de changer une mise en forme sur l'ensemble d'un document ou localement via certaines options.
3. `tnsmath` résout certains problèmes « complexes » pour vous.

II. Beta-dépendance

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

III. Comment lire cette documentation ?

Le choix a été fait de fournir des exemples comme documentation du package et de donner des fiches techniques des macros-commandes dans le chapitre dédié L.. Les exemples se présentent comme ci-dessous et sont généralement très courts.

1. En fait l'aspect sémantique est l'objectif commun de tous les packages de la suite `tns`.

```
\dintegrate*{cos(x^2)}{x}{a}{b}
=
\hook*{f(x)}{x}{a}{b}$
```

$$\int_a^b \cos(x^2) dx = [f(x)]_a^b$$

IV. A propos des macros

1. Règles de nommage

i. Les macros de même « type »

Les macros partageant une même fonctionnalité mathématique suivent les règles suivantes.

1. Un nom de base explicite est choisi comme par exemple `dotproduct` pour « *produit scalaire* » en anglais ou `set` pour « *ensemble* » en anglais.
2. Si besoin on spécialise du point de vue sémantique avec un préfixe et/ou un suffixe. Voici deux exemples.
 - (a) Dans `\vdotproduct`, le préfixe `v` est pour *v*-ecteur car cette macro s'utilise avec des noms de vecteurs `u` et `v` et non directement des vecteurs `\vect{u}` et `\vect{v}`, autrement dit c'est `\vdotproduct` qui se charge d'appliquer `\vect` à `u` et `v`.
 - (b) Dans `\setproba`, le suffixe `proba` est pour *proba*-bilité car cette macro sert à écrire des ensembles munis d'une probabilité².
3. Si l'on propose différentes mises en forme pour une même signification sémantique alors ceci se fera via des versions étoilées et/ou par le biais d'option(s) comme dans `\dotproduct[r]` pour obtenir des produits scalaires utilisant des chevrons `<` et `>` (*r* est pour *r*-after soit « *chevron* » en anglais).

ii. Les formes « négatives » des macros

Les formes « négatives » des macros auront un nom préfixé par la lettre `n` en référence à `n`-ot. C'est l'usage dans le monde L^AT_EX comme par exemple pour `\neq`.

iii. Les macros en mode `displaystyle`

Les macros évitant d'avoir à taper `\displaystyle` auront un nom préfixé par la lettre `d` comme par exemple pour `\dintegrate`.

iv. Les macros standards redéfinies

Certaines macros comme `\frac` sont un peu revues par `tnsmath`. Dans ce cas, les versions standard restent accessibles en utilisant le préfixe `std` ce qui donne ici la macro `\stdfrac`.

v. Casse utilisée pour les lettres

Les macros à usage graphique utiliseront une casse en bosses de chameau comme c'est le cas par exemple pour `\ptreeFrame` qui trace des cadres sur des arbres de probabilité, ou pour `\graphSign` qui ajoute des graphiques de signe près d'une ligne d'un tableau de signe³.

2. Ce choix est assumé même si on obtient un nom faisant penser à « *régler ...* » au lieu de « *ensemble de type ...* ».

3. La raison qui a poussé à ce choix est expliqué dans la présentation des outils de décoration des tableaux de signe : voir la présentation de la macro `\backLine`.

Les macros non graphiques n'utiliseront que des minuscules. L'auteur de `tnsmath` préfère cette convention car elle est plus efficace à utiliser lors de la saisie et qu'elle impose aussi aux concepteurs de ne pas proposer des noms de macro à rallonge⁴.

2. Les arguments : deux conventions à connaître

i. Avec un nombre fixé d'arguments

Dans ce cas, c'est la syntaxe L^AT_EX usuelle qui sera à utiliser comme dans `\dotprod{u}{v}`.

ii. Avec un nombre variable d'arguments

Certaines macros offrent la possibilité de fournir un nombre variable d'arguments comme dans `\coord{x | y | z | t}` et `\coord{x | y}`. Ceci se fait en utilisant un seul argument, au sens de L^AT_EX, dont le contenu est formé de morceaux séparés par des traits verticaux `|`. Ainsi dans `\coord{x | y | z | t}` l'unique argument `x | y | z | t`, au sens de L^AT_EX, sera analysé par `tnsmath` comme étant formé des quatre arguments `x`, `y`, `z` et `t`.

iii. Des arguments pas toujours affichés

Certaines macros pourront demander des arguments non utilisés pour la mise en forme. Pourquoi cela? Tout simplement pour permettre des copier-coller lors de la rédaction : voir par exemple le calcul du déterminant de deux vecteurs du plan pour vérifier leur colinéarité (*ceci est présenté dans la section 6.*).

V. Couleurs

Certaines macros utilisent de la couleur. Les choix faits sont tels que l'impression en noir et blanc ne soit pas impactée.

VI. Packages utilisés

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

- | | | | |
|-----------------------------|---------------------------|----------------------------|---------------------------|
| • <code>amssymb</code> | • <code>forest</code> | • <code>pgfplots</code> | • <code>upgreek</code> |
| • <code>bm</code> | • <code>forloop</code> | • <code>relsize</code> | • <code>witharrows</code> |
| • <code>centernot</code> | • <code>ifmtarg</code> | • <code>simplekv</code> | • <code>xstring</code> |
| • <code>circledsteps</code> | • <code>longtable</code> | • <code>stackengine</code> | • <code>yhmath</code> |
| • <code>commado</code> | • <code>mathrsfs</code> | • <code>stmaryrd</code> | |
| • <code>dsfont</code> | • <code>mathtools</code> | • <code>tkz-tab</code> | |
| • <code>esvect</code> | • <code>nicematrix</code> | • <code>trimspaces</code> | |

4. Cette pratique n'est pas inutile en interne pour autant par exemple pour nommer de façon compréhensible des macros privées.

Chapitre B.

Théorie générale des ensembles

I. Ensembles

1. Ensembles versus accolades

Exemple 1

<code>\$\setgene{1 ; 3 ; 5}\$.</code>	$\{1;3;5\} .$
--	---------------

Exemple 2

Dans l'exemple suivant on utilise l'option **sb** pour **s**-mall **b**-races soit « *petites accolades* » en anglais.

<code>\$\setgene {\dfrac{1}{3} ; \dfrac{5}{7} ; \dfrac{9}{11}}\$</code>	$\left\{ \frac{1}{3} ; \frac{5}{7} ; \frac{9}{11} \right\}$
<code>\$\setgene*{\dfrac{1}{3} ; \dfrac{5}{7} ; \dfrac{9}{11}}\$</code>	$\left\{ \frac{1}{3} ; \frac{5}{7} ; \frac{9}{11} \right\}$

2. Ensembles pour la géométrie

Exemple 1

<code>\$\setgeo{C}\$, \$\setgeo{D}\$ ou \$\setgeo{d}\$</code>	$\mathcal{C} , \mathcal{D} \text{ ou } d$
--	---

Remarque. Pour le moment, il n'est pas possible de taper `\setgeo{ABC}` avec plusieurs lettres.

Exemple 2 – Avec des indices

<code>\$\setgeo*{C}{1}\$ ou \$\setgeo*{C}{2}\$</code>	$\mathcal{C}_1 \text{ ou } \mathcal{C}_2$
---	---

3. Ensembles probabilistes

Exemple 1

<code>\setproba{E}</code> ou <code>\setproba{G}</code>	\mathcal{E} ou \mathcal{G}
---	--------------------------------

Remarque. Pour le moment, il n'est pas possible de taper `\setproba{ABC}` avec plusieurs lettres.

Exemple 2 – Avec des indices

<code>\setproba*{E}{1}</code> ou <code>\setproba*{E}{2}</code>	\mathcal{E}_1 ou \mathcal{E}_2
---	------------------------------------

4. Ensembles pour l'algèbre générale

Exemple 1

<code>\setalge{A}</code> , <code>\setalge{K}</code> , <code>\setalge{h}</code> ou <code>\setalge{k}</code>	\mathbb{A} , \mathbb{K} , \mathbb{h} ou \mathbb{k}
---	--

Remarque. Pour le moment, il n'est pas possible de taper `\setalge{ABC}` avec plusieurs lettres.

Exemple 2 – Avec des indices

<code>\setalge*{k}{1}</code> ou <code>\setalge*{k}{2}</code>	\mathbb{k}_1 ou \mathbb{k}_2
--	----------------------------------

5. Ensembles classiques en mathématiques et en informatique théorique

i. La liste complète

Dans l'exemple suivant, \mathbb{P} désigne l'ensemble des nombres premiers, \mathbb{H} celui des quaternions, \mathbb{O} celui des octonions et \mathbb{F} un ensemble de nombres flottants (*notation à préciser suivant le contexte*).

<code>\nullset</code>	\emptyset
<code>\NN</code> , <code>\ZZ</code> , <code>\PP</code>	\mathbb{N} , \mathbb{Z} , \mathbb{P}
<code>\DD</code> , <code>\QQ</code> , <code>\RR</code> , <code>\CC</code>	\mathbb{D} , \mathbb{Q} , \mathbb{R} , \mathbb{C}
<code>\HH</code> , <code>\OO</code>	\mathbb{H} , \mathbb{O}
<code>\FF</code>	\mathbb{F}

ii. Ensembles classiques suffixés

L'ensemble \mathbb{R} nous permet de voir tous les cas possibles.

$\$ \backslash \text{RRn} \$$, $\$ \backslash \text{RRp} \$$, $\$ \backslash \text{RRs} \$$	\mathbb{R}_- , \mathbb{R}_+ , \mathbb{R}^*
$\$ \backslash \text{RRsn} \$$, $\$ \backslash \text{RRsp} \$$	\mathbb{R}_-^* , \mathbb{R}_+^*

Nous avons utilisé les suffixes **n** pour **n**-égatif, **p** pour **p**-ositif et **s** pour **s**-tar soit « étoile » en anglais. Il y a aussi les suffixes composites **sn** et **sp**.

Notez qu'il est interdit d'utiliser $\$ \backslash \text{CCn} \$$ pour \mathbb{C}_- car l'ensemble \mathbb{C} ne possède pas de structure ordonnée standard. Jetez un oeil à la section suivante pour apprendre à taper \mathbb{C}_- si vous en avez besoin. L'interdiction est ici purement sémantique !

Remarque. La table B..1 de la présente page montre les associations autorisées entre ensembles classiques et suffixes.

TABLE B..1 – Suffixes

	n	p	s	sn	sp
$\backslash \text{NN}$			×		
$\backslash \text{PP}$					
$\backslash \text{ZZ}$					
$\backslash \text{DD}$	×	×	×	×	×
$\backslash \text{QQ}$					
$\backslash \text{RR}$					
$\backslash \text{CC}$					
$\backslash \text{HH}$			×		
$\backslash \text{OO}$					
$\backslash \text{FF}$	×	×	×	×	×

iii. Des suffixes à la carte

Dans l'exemple suivant, il faut savoir que le 2^e argument ne peut prendre que les valeurs **n**, **p**, **s**, **sn** ou **sp**.

$\$ \backslash \text{setspecial} \{ \backslash \text{CC} \} \{ n \} \$$, $\$ \backslash \text{setspecial} \{ \backslash \text{HH} \} \{ sp \} \$$ ou $\$ \backslash \text{setspecial} * \{ \backslash \text{setproba} \{ P \} \} \{ n \} \$$	\mathbb{C}_- , \mathbb{H}_+^* ou $\mathcal{P}_{\leq 0}$
---	---

II. Intervalles

1. Intervalles réels - Notation française (?)

Exemple 1

Dans cet exemple, la syntaxe fait référence à **O**-pened et **C**-losed pour « ouvert et fermé » en anglais. Nous verrons que **CC** et **OO** sont contractés en **C** et **O**. Notez au passage que la macro utilisée résout un problème d'espacement vis à vis du signe = .

$$\text{\$I =]a ; b] = \intervalOC{a}{b}\$}$$

$$I =]a ; b] =]a ; b]$$

Exemple 2

Les crochets s'étendent verticalement automatiquement. Pour empêcher cela, il suffit d'utiliser la version étoilée de la macro. Dans ce cas, les crochets restent tout de même un peu plus grands que des crochets utilisés directement. Voici un exemple.

```
\displaystyle
\intervalC{ \frac{1}{2} }{ 1^{2^3} }
=
[ \frac{1}{2} ; 1^{2^3} ]
=
\intervalC*{ \frac{1}{2} }{ 1^{2^3} }$
```

$$\left[\frac{1}{2} ; 1^{2^3} \right] = \left[\frac{1}{2} ; 1^{2^3} \right] = \left[\frac{1}{2} ; 1^{2^3} \right]$$

2. Intervalles réels – Notation américaine

Dans l'exemple suivant la syntaxe fait référence à P-arenthèse. Cette notation est utilisée aux États Unis.

```
\intervalPC{a}{b} = \intervalOC{a}{b}$
et
\intervalP{a}{b} = \intervalO{a}{b}$.
```

$$(a ; b] =]a ; b] \text{ et } (a ; b) =]a ; b[.$$

3. Intervalles discrets d'entiers

Dans l'exemple suivant la syntaxe fait référence à \mathbb{Z} l'ensemble des entiers relatifs.

```
\ZintervalC{-1}{4} =
\{ -1 ; 0 ; 1 ; 2 ; 3 ; 4 \}$

\ZintervalC{-1}{4} =
\ZintervalO{-2}{5}$.
```

$$\begin{aligned} \llbracket -1 ; 4 \rrbracket &= \{-1 ; 0 ; 1 ; 2 ; 3 ; 4\} \\ \llbracket -1 ; 4 \rrbracket &= \llbracket -2 ; 5 \rrbracket. \end{aligned}$$

III. Unions et intersections en mode ligne

L^AT_EX permet d'afficher sans souci $\cup_{k=1}^n$ mais ne propose pas $\bigcup_{k=1}^n$. Les macros `\dcap`, `\dcup` et `\dsqcup` donnent accès à ce type de fonctionnalité pour \cap , \cup et \sqcup respectivement. Voici des exemples d'utilisation.

Exemple 1 – Les symboles « seuls »

$$\text{\$A \dcap B = C \cap D\$}$$

$$\text{\$A \dcup B = C \cup D\$}$$

$$\text{\$A \dsqcup B = C \sqcup D\$}$$

$$A \cap B = C \cap D$$

$$A \cup B = C \cup D$$

$$A \sqcup B = C \sqcup D$$

Exemple 2 – Des intersections indicées

Ci-dessous est utilisée la macro `\bigcap` proposée par le package `amssymb`.

<pre> $\cap_{k=1}^n A_k$, $\bigcap_{k=1}^n B_k$, $\bigcap_{k=1}^n C_k$, $\bigcap_{k=1}^n D_k$ </pre>	$\cap_{k=1}^n A_k , \bigcap_{k=1}^n B_k , \bigcap_{k=1}^n C_k , \bigcap_{k=1}^n D_k$
---	--

Exemple 3 – Des unions indicées

Ci-dessous est utilisée la macro `\bigcup` proposée par le package `amssymb`.

<pre> $\cup_{k=1}^n A_k$, $\bigcup_{k=1}^n B_k$, $\bigcup_{k=1}^n C_k$, $\bigcup_{k=1}^n D_k$ </pre>	$\cup_{k=1}^n A_k , \bigcup_{k=1}^n B_k , \bigcup_{k=1}^n C_k , \bigcup_{k=1}^n D_k$
---	--

Exemple 4 – Des unions disjointes indicées

Ci-dessous sont utilisées les macros `\sqcup` et `\bigsqcup` proposée par le package `amssymb`.

<pre> $\sqcup_{k=1}^n A_k$, $\bigsqcup_{k=1}^n B_k$, $\bigsqcup_{k=1}^n C_k$, $\bigsqcup_{k=1}^n D_k$ </pre>	$\sqcup_{k=1}^n A_k , \bigsqcup_{k=1}^n B_k , \bigsqcup_{k=1}^n C_k , \bigsqcup_{k=1}^n D_k$
---	--

IV. Applications

1. Cardinal, image et compagnie

Exemple 1 – Cardinal

<pre> $\#E = \text{card } E$ </pre>	$\#E = \text{card } E$
--	------------------------

Exemple 2 – Image et compagnie

Ci-dessous se trouve la macro `\ker` proposée par `amsmath` qui est importé par `tnssets`.

<pre> $\ker f$, $\text{dom } f$, $\text{im } f$ ou $\text{codom } f$ </pre>	$\ker f , \text{dom } f , \text{im } f \text{ ou codom } f$
---	---

2. Application totale, partielle, injective, surjective et/ou bijective

Voici des symboles qui, bien que très techniques, facilitent la rédaction de documents à propos des applications totales ou partielles¹ (on parle aussi d'applications, sans qualificatif, et de fonctions).

Exemple 1 – Applications totales

$f: A \rightarrow B$ est une application totale, c'est à dire définie sur A tout entier.

$i: C \hookrightarrow D$ est une application totale injective.

$s: E \twoheadrightarrow F$ est une application totale surjective.

$b: G \xrightarrow{\sim} H$ est une application totale bijective.

$f: A \rightarrow B$ est une application totale, c'est à dire définie sur A tout entier.

$i: C \hookrightarrow D$ est une application totale injective.

$s: E \twoheadrightarrow F$ est une application totale surjective.

$b: G \xrightarrow{\sim} H$ est une application totale bijective.

Exemple 2 – Applications partielles

$f: A \rightarrowtail B$ est une application partielle, c'est à dire définie sur un sous-ensemble de A .

$i: C \hookrightarrowtail D$ est une application partielle injective.

$s: E \twoheadrightarrowtail F$ est une application partielle surjective.

$b: G \xrightarrow{\sim}tail H$ est une application partielle bijective.

$f: A \rightarrowtail B$ est une application partielle, c'est à dire définie sur un sous-ensemble de A .

$i: C \hookrightarrowtail D$ est une application partielle injective.

$s: E \twoheadrightarrowtail F$ est une application partielle surjective.

$b: G \xrightarrow{\sim}tail H$ est une application partielle bijective.

3. Fonction identité

id_A vérifie par définition
 $\forall x \in A, \text{id}(x) = x$.

Id_A vérifie par définition $\forall x \in A, \text{Id}(x) = x$.

1. $a: E \rightarrow F$ est une application totale si $\forall x \in E, \exists ! y \in F$ tel que $y = a(x)$. Plus généralement, $f: E \rightarrow F$ est une application partielle si $\forall x \in E, \exists \leq 1 y \in F$ tel que $y = f(x)$, autrement dit soit $f(x)$ existe dans F , soit f n'est pas définie en x .

4. Fonction caractéristique

Exemple 1 – Version très utilisée

$\text{\texttt{\$}\charact_A\$}$ vérifie par définition $\text{\texttt{\$}\forall x \in A, \text{\texttt{\$}\charact_A(x) = 1\$}}$ et $\text{\texttt{\$}\forall x \notin A, \text{\texttt{\$}\charact_A(x) = 0\$}}$.	χ_A vérifie par définition $\forall x \in A, \chi_A(x) = 1$ et $\forall x \notin A, \chi_A(x) = 0$.
--	--

Exemple 2 – Version alternative

$\text{\texttt{\$}\caractone_A = \charact_A\$}$	$\mathbb{1}_A = \chi_A$
---	-------------------------

5. Définition explicite d'une fonction

Exemple 1 – Écriture par défaut

$\text{\texttt{\$}\funcdef{f}{x}{x^2}\%$ $\text{\texttt{\{I\}\{J\}\$}}$	$f: \left. \begin{array}{l} I \rightarrow J \\ x \mapsto x^2 \end{array} \right $
--	---

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

$\text{\texttt{\$}\funcdef[s]{f}{x}{x^2}\%$ $\text{\texttt{\{I\}\{J\}\$}}$	$f: I \rightarrow J$ $x \mapsto x^2$
---	---

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.

$\text{\texttt{Soit \texttt{\$}\funcdef[h]{f}{x}{x^2}\%$ $\text{\texttt{\{I\}\{J\}\$}} \dots$	$\text{Soit } f : x \in I \mapsto x^2 \in J \dots$
--	--

Exemple 4 – Écriture en ligne incomplète

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

$\text{\texttt{\$}\funcdef[h]{f}{x}{x^2}\{\}\{J\}\$}$	$f : x \mapsto x^2 \in J$
$\text{\texttt{\$}\funcdef[h]{f}{x}{x^2}\{I\}\{\}\$}$	$f : x \in I \mapsto x^2$
$\text{\texttt{\$}\funcdef[h]{f}{x}{x^2}\{\}\{\}\$}$	$f : x \mapsto x^2$

Exemple 5 – Écriture textuelle

On peut enfin obtenir une version « *textuelle* » via la macro `\txtfuncdef` où `txt` est pour `texte`. Cette macro ne s'utilise pas en mode mathématique et elle accepte l'omission des ensembles.

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

6. Composition**Exemple 1 – Opérateur**

La macro `\compo` est juste une version un peu plus petite de `\circ`.

<code>\$f \compo g\$ et non</code> <code>\$f \circ g\$</code>	$f \circ g$ et non $f \circ g$
--	--------------------------------

Exemple 2 – Compositions successives

La macro `\multicompo` sert à indiquer la composition d'une application plusieurs fois de suite par elle-même². Voici toutes les mises en forme disponibles où l'option `exp` nécessite que le nombre d'applications composées soit un naturel non nul connu. Vous noterez que l'écriture par défaut, qui n'est pas standard, n'est pas $f^{(p)}$ car cette notation est traditionnellement utilisée pour indiquer la dérivée p^e d'une application.

<code>\$\multicompo {g}{5}</code> <code>= \multicompo[exp]{g}{5}\$</code>	$g^{\langle 5 \rangle} = g \circ g \circ g \circ g \circ g$
<code>\$\multicompo {f}{p}</code> <code>= \multicompo[dot]{f}{p}\$</code>	$f^{\langle p \rangle} = f \circ \dots \circ f$

Remarque. La convention retenue est analogue à ce que l'on fait avec les puissances de nombres réels. En particulier, $f^{\langle 1 \rangle} = f$ et $f^{\langle 0 \rangle} = \text{Id}_{\text{dom } f}$.

2. Une telle fonction f doit vérifier $\text{im } f \subseteq \text{dom } f$.

Chapitre C.

Méthodes formelles en logique

I. Espace après la négation logique

`\neg` a été redéfinie pour ajouter un peu d'espace après le symbole. Le comportement par défaut se retrouve en utilisant la macro `\stdneg`. Voici un exemple.

`\neg A = \stdneg A`

$$\neg A = \neg A$$

`\neg\neg A = \stdneg \stdneg A`

$$\neg\neg A = \neg\neg A$$

II. Différents types de comparaisons « standard »

D'un point de vue pédagogique, il peut être intéressant de disposer de différentes façon d'écrire une égalité, une non égalité ou une inégalité. Bien entendu on tord les règles de typographie avec ce type de pratique mais c'est pour le bien de la communauté éducative.

1. Définir quelque chose

L'exemple suivant montre deux façons de rédiger une égalité signifiant une définition (*la section 9. explique comment est défini le texte « déf »*).

`f(x) \eqdef x^3 + 1`

$$f(x) \stackrel{\text{déf}}{=} x^3 + 1$$

`f(x) \eqdef* x^3 + 1`

$$f(x) := x^3 + 1$$

2. Indiquer une identité

L'exemple suivant montre deux façons de rédiger des identités avec une notation symbolique non standard (*la section 9. explique comment est défini le texte « id »*).

`(a + b)^2 \eqid a^2 + b^2 + 2 a b`

$$(a + b)^2 \stackrel{\text{id}}{=} a^2 + b^2 + 2ab$$

`(a + b)^2 \eqid* a^2 + b^2 + 2 a b`

$$(a + b)^2 \rightleftharpoons a^2 + b^2 + 2ab$$

3. Une égalité à vérifier ou non, une hypothèse, une condition

Se reporter à la section 9. pour savoir comment sont définis les textes « *cons* », « *cond* » et « *hyp* ».

<code>\$(a + b)^3 \eqtest a^3 + b^3 + 3 a b\$</code>	$(a + b)^3 \stackrel{?}{=} a^3 + b^3 + 3ab$
<code>\$(a + b)^3 \neqid a^3 + b^3 + 3 a b\$</code>	$(a + b)^3 \stackrel{id}{\neq} a^3 + b^3 + 3ab$
<code>\$x \neqhyp 0\$ ou</code> <code>\$x \neqcond 0\$ ou</code> <code>\$x \eqcons 0\$</code>	$x \stackrel{hyp}{\neq} 0$ ou $x \stackrel{cond}{\neq} 0$ ou $x \stackrel{cons}{=} 0$

4. Une égalité indiquant le choix d'une valeur ou l'application d'une relation

La section 9. permet de savoir comment les textes « *choix* » et « *appli* » sont définis.

<code>\$x \geqcond 4\$ implique</code> <code>\$x^2 \geqcons 16\$.</code>	$x \stackrel{cond}{\geq} 4$ implique $x^2 \stackrel{cons}{\geq} 16$.
<code>Donc \$x \eqchoice 123\$ donne</code> <code>\$123^2 \geqappli 16\$.</code>	Donc $x \stackrel{choix}{=} 123$ donne $123^2 \stackrel{appli}{\geq} 16$.

5. Une égalité indiquant l'équation d'une courbe

La section 9. permet de savoir comment les texte « *graph* » est défini.

<code>\$M \in C: y \eqplot x^2 + 3\$</code> <code>donne</code> <code>\$y_M \eqappli x_M^2 + 3\$.</code>	$M \in C : y \stackrel{graph}{=} x^2 + 3$ donne $y_M \stackrel{appli}{=} x_M^2 + 3$.
---	---

6. Différents types d'inéquations

Le principe reste le même pour les symboles d'équations excepté qu'il n'y a ici aucune écriture purement symbolique. Voici un code « fourre-tout » montrant quelques exemples.

<code>\$x \leqtest x^2\$ ou \$x \lesscons x^2\$ ou</code> <code>\$x \geqhyp 1\$ ou \$x \gtrcond 2\$.</code>	$x \stackrel{?}{\leq} x^2$ ou $x \stackrel{cons}{<} x^2$ ou $x \stackrel{hyp}{\geq} 1$ ou $x \stackrel{cond}{>} 2$.
--	--

7. Des formes négatives aussi pour les inéquations

Tous les opérateurs de comparaison ont une forme négative qui s'obtient en préfixant le nom de l'opérateur par n. Voici quelques exemples d'utilisation.

<code>\$x \nlesshyp 3\$ ou</code> <code>\$y \nleqtest 4\$ ou</code> <code>\$z \ngeqcons 5\$</code>	$x \stackrel{hyp}{\not\leq} 3$ ou $y \stackrel{?}{\not\leq} 4$ ou $z \stackrel{cons}{\not\geq} 5$
--	---

8. Une table récapitulative

La table C..1 page suivante fournit toutes les associations autorisées entre opérateurs de comparaison et décorations.

9. Textes utilisés

Voici les macros définissant les textes utilisés qui tiennent compte de l'utilisation ou non de l'option `french` de `babel`. Nous ne donnons que les versions françaises.

<code>\textopappli{}</code>	donne « <i>appli</i> »	<code>\textopchoice{}</code>	donne « <i>choix</i> »
<code>\textopcond{}</code>	donne « <i>cond</i> »	<code>\textopcons{}</code>	donne « <i>cons</i> »
<code>\textopdef{}</code>	donne « <i>déf</i> »	<code>\textophyp{}</code>	donne « <i>hyp</i> »
<code>\textopid{}</code>	donne « <i>id</i> »	<code>\textopplot{}</code>	donne « <i>graph</i> »
<code>\textoptest{}</code>	donne « ? »		

III. Équivalences et implications

1. Des symboles logiques supplémentaires

Exemple 1 – Implication réciproque

En plus des opérateurs `\iff` et `\implies` proposés par L^AT_EX, il a été ajouté l'opérateur `\liesimp`, où l'on a inversé les groupes syllabiques de `\implies`, un opérateur pour obtenir \Leftarrow ¹, ainsi que des versions négatives. Voici un exemple d'utilisation.

<code>\$(A \implies B)</code> <code>\iff (B \liesimp A)\$</code>	$(A \Rightarrow B) \iff (B \Leftarrow A)$
<code>\$(A \implies B)</code> <code>\niff (A \nimplies B)\$</code>	$(A \Rightarrow B) \niff (A \nRightarrow B)$

Exemple 2 – Des opérateurs décorés

Tout comme pour les comparaisons, il existe des versions décorées de type test, hypothèse, condition ... Elles sont toutes présentes dans l'exemple suivant.

<code>\$(A \iffappli B \niffchoice C)\$</code>	$A \overset{\text{appli}}{\iff} B \overset{\text{choix}}{\niff} C$
<code>\$(A \impliescond B \nimpliescons C)\$</code>	$A \overset{\text{cond}}{\implies} B \overset{\text{cons}}{\nimplies} C$
<code>\$(A \liesimphyp B \nliesimptest C)\$</code>	$A \overset{\text{hyp}}{\Leftarrow} B \overset{?}{\nLeftarrow} C$

2. Une table récapitulative

La table C..1 page suivante montre toutes les associations autorisées entre opérateurs logiques et décorations.

1. Penser aussi aux preuves d'équivalence par double implication.

IV. Des versions alternatives du quantificateur existentiel

1. Quantifier l'existence

Voici deux versions, l'une classique, et l'autre beaucoup moins, permettant de préciser le quantificateur \exists .

$\$ \backslash \text{existsone } x \backslash \text{in } E \$ \text{ pour}$ $\backslash \text{og il existe un seul } \$x\$ \backslash \text{fg.}$	$\exists ! x \in E$ pour « il existe un seul x ».
$\$ \backslash \text{existmulti}\{\leq 1\} y \backslash \text{in } F \$ \text{ pour}$ $\backslash \text{og il existe au plus un } \$y\$ \backslash \text{fg.}$	$\exists_{\leq 1} y \in F$ pour « il existe au plus un y ». $\exists_{=1} := \exists !$
$\$ \backslash \text{existmulti}\{=1\} \backslash \text{eqdef* } \backslash \text{existsone} \$$	

2. Versions négatives

$\$ \backslash \text{nexistsone} \$ \text{ pour}$ $\backslash \text{og il n'existe pas un unique } \backslash \text{fg.}$	$\nexists !$ pour « il n'existe pas un unique ».
$\$ \backslash \text{existmulti}\{\neq 1\} \backslash \text{eqdef* } \backslash \text{nexistsone} \$$	$\exists_{\neq 1} := \nexists !$
$\$ \backslash \text{nexistmulti}\{>4\} \$ \text{ pour}$ $\backslash \text{og il n'existe pas plus de quatre } \backslash \text{fg.}$	$\nexists_{>4}$ pour « il n'existe pas plus de quatre ». \nexists vient de <code>amssymb</code> .
$\$ \backslash \text{nexists} \$ \text{ vient de } \backslash \text{verb+amssymb+}.$	

V. Détailler un raisonnement simple

1. Version pour le lycée et après

Exemple 1 – Avec les réglages par défaut

L'environnement `explain` permet de détailler les étapes principales d'un calcul ou d'un raisonnement simple en s'appuyant sur la macro `\explnext` dont le nom vient de « *expl-ain next step* » soit « *expliquer la prochaine étape* » en anglais². On dispose aussi de `\explnext*` pour des explications descendantes et/ou montantes³.

Ci-dessous se trouve un exemple, très farfelu vers la fin, où l'on utilise les réglages par défaut. Notons au passage que ce type de présentation n'est sûrement pas bien adaptée à un jeune public pour lequel une 2^e façon de détailler des calculs et/ou un raisonnement simple est proposée plus bas dans la section 2..

2. Cet environnement utilise aussi le package `witharrows` qui est très sympathique pour expliquer des étapes de calcul.

3. Les explications données ne doivent pas être trop longues car ce serait contre-productif.

```

\begin{explain}
  (a + b)^2
  \explnext{On utilise  $x^2 = x \cdot x$ .}
  (a + b) (a + b)
  \explnext*{Double développement depuis la parenthèse gauche.}%
  {Double factorisation pas facile.}
  a^2 + a b + b a + b^2
  \explnext*{}%
  {Commutativité du produit.}
  a^2 + 2 a b + b^2
  \explnext*{Commutativité de l'addition.}%
  {}
  a^2 + b^2 + 2 a b
\end{explain}

```

$$\begin{aligned}
 &(a + b)^2 \\
 = &\quad \{ \textit{On utilise } x^2 = x \cdot x. \} \\
 &(a + b)(a + b) \\
 = &\quad \left\{ \begin{array}{l} \downarrow \textit{Double développement depuis la parenthèse gauche.} \downarrow \\ \uparrow \quad \quad \quad \textit{Double factorisation pas facile.} \quad \quad \uparrow \end{array} \right\} \\
 &a^2 + ab + ba + b^2 \\
 = &\quad \{ \uparrow \textit{Commutativité du produit.} \uparrow \} \\
 &a^2 + 2ab + b^2 \\
 = &\quad \{ \downarrow \textit{Commutativité de l'addition.} \downarrow \} \\
 &a^2 + b^2 + 2ab
 \end{aligned}$$

Remarque. Il faut savoir que la mise en forme est celle d'une formule ce qui peut rendre service comme dans l'exemple suivant.

```

Un calcul avec un placement pouvant être
utile :
\begin{explain}
  (a + b)^2
  \explnext{Identité remarquable.}
  a^2 + b^2 + 2 a b
\end{explain}

```

Un calcul avec un placement pouvant être
utile : $(a + b)^2$
 $= \quad \{ \textit{Identité remarquable.} \}$
 $a^2 + b^2 + 2ab$

Avec un retour à la ligne, il faudra donc si besoin gérer l'espacement vertical.

```

Mon calcul pas trop proche.

\medskip
\begin{explain}
  (a + b)^2
  \explnext{Identité remarquable.}
  a^2 + b^2 + 2 a b
\end{explain}

```

Mon calcul pas trop proche.
 $(a + b)^2$
 $= \quad \{ \textit{Identité remarquable.} \}$
 $a^2 + b^2 + 2ab$

Remarque. Voici des petites choses à connaître sur les macros `\explnext` et `\explnext*`.

1. `\expltxt` est utilisée par `\explnext` pour mettre en forme le texte d'explication.
2. `\expltxtup` et `\expltxtdown` sont utilisées par `\explnext*` décorer les textes d'explication juste avant leur mise en forme finale via `\expltxtupdown`.
3. `\explnext` et `\explnext*` utilisent la macro constante `\expltxtspacein` pour l'espacement entre le symbole et la courte explication. Par défaut, cette macro vaut `2em`.

Exemple 2 – Utiliser un autre symbole globalement

L'environnement `explain` possède plusieurs options dont l'une est `ope` qui vaut `{=}` par défaut. Ceci permet de faire ce qui suit sans effort.

<pre>\begin{explain}[ope = \viff] x^2 + 10 x + 25 = 0 \explnext{Identité remarquable.} (x + 5)^2 = 0 \explnext{\$P^2 = 0\$ si et seulement si \$P = 0\$.} x = -5 \end{explain}</pre>	$x^2 + 10x + 25 = 0$ $\Downarrow \quad \{ \textit{Identité remarquable.} \}$ $(x + 5)^2 = 0$ $\Downarrow \quad \{ P^2 = 0 \textit{ si et seulement si } P = 0. \}$ $x = -5$
---	---

Exemple 3 – Juste utiliser des symboles

Si l'argument obligatoire de la macro `\explnext` est vide alors seul le symbole est affiché (*ne pas oublier les accolades vides*). Voici un court exemple de ceci.

<pre>\begin{explain}[ope = \viff] a^2 = b^2 \explnext{} a = \pm b \end{explain}</pre>	$a^2 = b^2$ \Downarrow $a = \pm b$
---	--------------------------------------

Exemple 4 – Utiliser un autre symbole localement

La macro `\explnext` possède un argument optionnel qui utilise par défaut celui de l'environnement. En utilisant cette option, on choisit alors localement le symbole à employer. Voici un exemple d'utilisation complètement farfelu bien que correct.

<pre>\begin{explain}[ope = \viff] 0 \leq a < b \explnext[\vimplies]{% {Croissance de \$x^2\$ sur \$\mathbb{R}_+\$}.} a^2 < b^2 \explnext{} a^2 - b^2 < 0 \explnext{Identité remarquable.} (a - b)(a + b) < 0 \explnext[\vimplies]{} a \neq b \end{explain}</pre>	$0 \leq a < b$ $\Downarrow \quad \{ \textit{Croissance de } x^2 \textit{ sur } \mathbb{R}_+. \}$ $a^2 < b^2$ \Downarrow $a^2 - b^2 < 0$ $\Downarrow \quad \{ \textit{Identité remarquable.} \}$ $(a - b)(a + b) < 0$ \Downarrow $a \neq b$
---	--

Exemple 5 – Choisir la mise en forme des explications

Pour la mise en forme des explications à double sens, la macro `\explnext` fait appel à la macro `\expltxt`. Par défaut, le package utilise la définition suivante.

```
\newcommand\expltxt[1]{%
  \text{\color{blue}\footnotesize \{\,\,\itshape #1\,\,\}}%
}
```

Pour la mise en forme des explications à sens unique, la macro `\explnext*` fait appel aux macros `\expltxtup`, `\expltxtdown` et `\expltxtupdown`. Par défaut le package utilise les définitions suivantes.

```
\newcommand\expltxtup[1]{%
  $\uparrow$ #1 $\uparrow$%
}

\newcommand\expltxtdown[1]{%
  $\downarrow$ #1 $\downarrow$%
}

\newcommand\expltxtupdown[2]{%
  \displaystyle\footnotesize\color{blue}%
  \left\{\,\,%
    \genfrac{}{}{0pt}{}{%
      \text{\itshape\expltxtdown{\samesizeas{#1}{#2}}}%
    }{%
      \text{\itshape\expltxtup{\samesizeas{#2}{#1}}}%
    }%
  \,\right\}%
}
```

Nous allons expliquer comment obtenir l'affreux exemple ci-dessous montrant que l'on peut adapter si besoin la mise en forme.

```
\begin{explain}
  (a + b) (a + b)
  \explnext{Se souvenir de $P\cdot P$}
  P = P^2$.}%
  (a + b)^2
  \explnext*{Id. Rm. - Dév.}%
  {Id. Rm. - Facto.}%
  a^2 + 2 a b + b^2
\end{explain}
```

$$\begin{aligned}
 & (a + b)(a + b) \\
 &= \quad \Downarrow \textit{Se souvenir de } P \cdot P = P^2. \quad \Uparrow \\
 & (a + b)^2 \\
 &= \quad \left\langle \begin{array}{c} \Downarrow \textit{Id. Rm. - Dév.} \Downarrow \\ \hline \Uparrow \textit{Id. Rm. - Facto.} \Uparrow \end{array} \right\rangle \\
 & a^2 + 2ab + b^2
 \end{aligned}$$

La mise en forme a été obtenue en utilisant le code L^AT_EX suivant où la macro `\samesizeas{#1}{#2}` rend le texte `#1` aussi large que `#2` en ajoutant des espaces supplémentaires tout en centrant le résultat final si besoin (*ne pas oublier de passer en mode texte via `\text`*).

```

\newcommand\myexpltxt[2]{%
  \text{\color{#1} \footnotesize \itshape \bfseries #2}%
}

\renewcommand\expltxt[1]{%
  \myexpltxt{gray}{\Downarrow$ #1 $\Uparrow$}%
}

\renewcommand\expltxtup[1]{%
  \myexpltxt{orange}{\Uparrow$ #1 $\Uparrow$}%
}

\renewcommand\expltxtdown[1]{%
  \myexpltxt{red}{\Downarrow$ #1 $\Downarrow$}%
}

\renewcommand\expltxtupdown[2]{%
  \displaystyle\color{blue!20!black!30!green}%
  \genfrac{\langle}{\rangle}{1pt}{}{%
    \expltxtdown{\samesizeas{#1}{#2}}%
  }{%
    \expltxtup{\samesizeas{#2}{#1}}%
  }%
}

```

2. Version pour les collégiens

L'environnement `explain` avec l'option `style = ar`⁴ utilise des flèches pour indiquer les explications (*ar est pour ar-row soit « flèche » en anglais*). Dans ce cas d'utilisation, la macro `\explnext*` permet d'avoir une flèche unidirectionnelle, vers le haut ou le bas au choix, ou bien d'écrire deux indications dont l'une est montante et l'autre descendante.

Il existe aussi l'option `style = sar` lorsque la toute 1^{re} étape n'est pas expliquée (*s est pour s-short soit « court » en anglais*). Attention car forcément ceci nécessite au tout début de l'environnement l'usage de la macro `\explnext` sans aucun contenu !

Exemple 1 – Des flèches à double sens

<pre> \begin{explain}[style = ar] (a + b)^2 \explnext{Identité remarquable} a^2 + 2 a b + b^2 \explnext{} a^2 + b^2 + 2 a b \end{explain} </pre>	$ \begin{aligned} &(a + b)^2 \\ &= a^2 + 2ab + b^2 \\ &= a^2 + b^2 + 2ab \end{aligned} $ <p style="color: blue; margin-top: 10px;"> $\left. \begin{array}{l} \\ \end{array} \right\} \text{Identité remarquable}$ </p>
--	---

Exemple 2 – Des flèches unidirectionnelles

Ce qui suit est juste là comme démo. car les explications y sont un peu farfelues.

4. Cet environnement utilise aussi le package `witharrows`.

```

\begin{explain}[style = ar]
  (a + b)^2
  \explnext*{Via  $P^2 = P \cdot P$ .}
  {Via  $P \cdot P = P^2$ .}
  (a + b) (a + b)
  \explnext*{Double développement.}%
  {Double factorisation (pas simple).}
  a^2 + a b + b a + b^2
  \explnext*{Commutativité du produit.}%
  {}
  a^2 + 2 a b + b^2
  \explnext*{}%
  {Commutativité de l'addition.}
  a^2 + b^2 + 2 a b
\end{explain}

```

$$\begin{array}{ll}
 (a + b)^2 & \\
 = (a + b)(a + b) & \left. \begin{array}{l} \text{Via } P^2 = P \cdot P. \\ \text{Via } P \cdot P = P^2. \end{array} \right\} \\
 = a^2 + ab + ba + b^2 & \left. \begin{array}{l} \text{Double développement.} \\ \text{Double factorisation (pas simple).} \end{array} \right\} \\
 = a^2 + 2ab + b^2 & \left. \begin{array}{l} \text{Commutativité du produit.} \\ \text{Commutativité de l'addition.} \end{array} \right\} \\
 = a^2 + b^2 + 2ab &
 \end{array}$$

Exemple 3 – Ne pas expliquer le tout début

L'environnement étoilé `explain` avec l'option `style = sar` débute différemment la mise en forme. Bien entendu ici le tout premier `\explnext` doit avoir un argument vide !

```

\begin{explain}[style = sar]
  (a + b) (a + b)
  \explnext{}
  (a + b)^2
  \explnext{Identité remarquable.}
  a^2 + b^2 + 2 a b
\end{explain}

```

$$\begin{array}{ll}
 (a + b)(a + b) = (a + b)^2 & \\
 = a^2 + b^2 + 2ab & \left. \begin{array}{l} \text{Identité remarquable.} \end{array} \right\}
 \end{array}$$

Exemple 4 – Choisir son symbole

Voici comment faire où l'implication finale est juste là pour la démonstration (*on notera une petite bidouille un peu sale à faire pour avoir un alignement à peu près correct*).


```

\begin{explain}[style = ar, ope = \iff]
  a^2 + 2 a b + b^2 = 0
  \explnext{}
  (a + b)^2 = 0
  \explnext[\:\implies]%
    {$P^2 = 0$ ssi $P = 0$.}

  a + b = 0
\end{explain}

```

$$\begin{aligned}
 & a^2 + 2ab + b^2 = 0 \\
 \iff & (a + b)^2 = 0 \quad \left. \vphantom{(a + b)^2 = 0} \right\} P^2 = 0 \text{ ssi } P = 0. \\
 \implies & a + b = 0
 \end{aligned}$$

Avec la version courte, on obtient ce qui suit.

```

\begin{explain}[style = sar, ope = \iff]
  a^2 + 2 a b + b^2 = 0
  \explnext{}
  (a + b)^2 = 0
  \explnext[\:\implies]%
    {$P^2 = 0$ ssi $P = 0$.}

  a + b = 0
\end{explain}

```

$$\begin{aligned}
 a^2 + 2ab + b^2 = 0 & \iff (a + b)^2 = 0 \\
 & \implies a + b = 0 \quad \left. \vphantom{(a + b)^2 = 0} \right\} P^2 = 0 \text{ ssi } P = 0.
 \end{aligned}$$

3. De courts commentaires

Exemple 1 – Sans alignement

Il est possible d'ajouter de petits commentaires via `\comthis` où `comthis` est pour `com-ment this` soit « *commenter ceci* » en anglais.

```

\begin{explain}
  (a + b)^2
  \comthis{Forme facto.}
  \explnext*[Id.Rq. -- Dév.]%
    {Id.Rq. -- Facto.}

  a^2 + 2 a b + b^2
  \comthis{Forme dév.}
\end{explain}

```

$$\begin{aligned}
 & (a + b)^2 \quad [\textit{Forme facto.}] \\
 = & \left\{ \begin{array}{l} \downarrow \textit{Id.Rq.} - \textit{Dév.} \downarrow \\ \uparrow \textit{Id.Rq.} - \textit{Facto.} \uparrow \end{array} \right\} \\
 & a^2 + 2ab + b^2 \quad [\textit{Forme dév.}]
 \end{aligned}$$

Remarque. La mise en forme du texte des commentaires est fait via la macro personnalisable `\explcom`. Quant à l'espacement ajouté entre le texte et son commentaire il est défini par la macro `\expltxtspacein` qui est égale à 2em par défaut.

Exemple 2 – Tout aligner

Il peut être utile d'aligner tous les commentaires. Ceci s'obtient via l'option `com = al` où `al` est pour `al-igné` (par défaut `com = nal` avec le préfixe `n` pour `n-on`).

<pre> \begin{explain}[com = al] (a + b)^2 \comthis{Forme facto.} \explnext*{Id.Rq - Dév.}% {Id.Rq - Facto.} a^2 + 2 a b + b^2 \comthis{Forme dév.} \end{explain} </pre>	$(a + b)^2$ <p style="text-align: right;">[<i>Forme facto.</i>]</p> $= \left\{ \begin{array}{l} \downarrow \text{Id.Rq - Dév.} \downarrow \\ \uparrow \text{Id.Rq - Facto.} \uparrow \end{array} \right\}$ $a^2 + 2ab + b^2$ <p style="text-align: right;">[<i>Forme dév.</i>]</p>
---	--

Exemple 3 – Le meilleur des deux mondes

Dans d'autres situations, utiliser les deux types d'alignement peut faire sens. Ceci s'obtient via l'option `com = al` et l'emploi de la macro étoilée `\comthis*` à chaque fois que l'on souhaite "coller" un commentaire le plus à gauche possible.

<pre> \begin{explain}[com = al] (a + b) (a + b) \comthis{Forme facto.} \explnext{Via \$x^2 = x \cdot x\$.} (a + b)^2 \comthis*{Au passage...} \explnext*{Id.Rq - Dév.}% {Id.Rq - Facto.} a^2 + 2 a b + b^2 \comthis{Forme dév.} \end{explain} </pre>	$(a + b)(a + b)$ <p style="text-align: right;">[<i>Forme facto.</i>]</p> $= \{ \text{Via } x^2 = x \cdot x. \}$ $(a + b)^2$ <p style="text-align: right;">[<i>Au passage...</i>]</p> $= \left\{ \begin{array}{l} \downarrow \text{Id.Rq - Dév.} \downarrow \\ \uparrow \text{Id.Rq - Facto.} \uparrow \end{array} \right\}$ $a^2 + 2ab + b^2$ <p style="text-align: right;">[<i>Forme dév.</i>]</p>
--	---

Remarque. Si l'alignement n'est pas activé, les macros `\comthis*` et `\comthis` auront toutes les deux le même effet.

Exemple 4 – Ceci marche aussi avec le style « fléché »

Voici ce que donne le mode mixte lorsque des flèches sont utilisées pour les explications. Il semble moins pertinent ici de mixer les modes « alignement » et « non alignement » mais chacun pris séparément peut avoir son utilité.

<pre> \begin{explain}[style = ar, com = al] (a + b) (a + b) \comthis{Forme facto.} \explnext{Via \$x^2 = x \cdot x\$.} (a + b)^2 \comthis*{Au passage...} \explnext*{Id.Rq - Dév.}% {Id.Rq - Facto.} a^2 + 2 a b + b^2 \comthis{Forme dév.} \end{explain} </pre>	$(a + b)(a + b)$ <p style="text-align: right;">[<i>Forme facto.</i>]</p> $= (a + b)^2$ <p style="text-align: right;">[<i>Au passage...</i>]</p> $= a^2 + 2ab + b^2$ <p style="text-align: right;">[<i>Forme dév.</i>]</p>
<p style="text-align: center;"> $\left. \begin{array}{l} \text{[Forme facto.]} \\ \text{[Au passage...]} \end{array} \right\} \text{Via } x^2 = x \cdot x.$ $\left. \begin{array}{l} \text{[Forme dév.]} \end{array} \right\} \text{Id.Rq - Dév.} \quad \left. \begin{array}{l} \text{Id.Rq - Facto.} \end{array} \right\}$ </p>	

Remarque. Bien entendu il est impossible de commenter le tout début en mode fléché court.

4. Un mini hack très utile pour des « étapes alignées »

Vous pouvez écrire très facilement des calculs ou raisonnement simples alignés comme suit sans trop vous fatiguer.

<pre> \begin{explain}[style = sar] (a + b) (a + b) \explnext{} (a + b)^2 \explnext{} a^2 + b^2 + 2 a b \comthis{Pourquoi ?} \explnext{} a^2 + 2 a b + b^2 \end{explain} </pre>	$ \begin{aligned} (a + b)(a + b) &= (a + b)^2 \\ &= a^2 + b^2 + 2ab \quad [\text{Pourquoi ?}] \\ &= a^2 + 2ab + b^2 \end{aligned} $
--	---

On a accès à une autre mise en forme (*ceci peut rendre aussi service*).

<pre> \begin{explain}[style = ar] (a + b) (a + b) \explnext{} (a + b)^2 \explnext{Pourquoi ?} a^2 + b^2 + 2 a b \explnext{} a^2 + 2 a b + b^2 \end{explain} </pre>	$ \begin{aligned} (a + b)(a + b) \\ &= (a + b)^2 \\ &= a^2 + b^2 + 2ab \quad \text{Pourquoi ?} \\ &= a^2 + 2ab + b^2 \end{aligned} $
--	---

Enfin dans le cadre de calculs à faire expliquer par des élèves, ce qui suit peut être utile.

<p>Donner les justifications J1, J2 et J3.</p> <pre> \medskip \begin{explain} (a + b) (a + b) \explnext{J1} (a + b)^2 \explnext{J2} a^2 + b^2 + 2 a b \explnext{J3} a^2 + 2 a b + b^2 \end{explain} </pre>	<p>Donner les justifications J1, J2 et J3.</p> $ \begin{aligned} (a + b)(a + b) \\ &= \{ J1 \} \\ (a + b)^2 \\ &= \{ J2 \} \\ a^2 + b^2 + 2ab \\ &= \{ J3 \} \\ a^2 + 2ab + b^2 \end{aligned} $
--	---

5. Un conseil de mise en forme

Voici un style de codage que nous trouvons très facile à relire et maintenir.

<pre> \begin{explain}[com = al] (a + b) (a + b) \comthis{Forme facto.} \explnext{Via $x^2 = x \cdot x$.} % (a + b)^2 \comthis*{Au passage...} \explnext*{Id.Rq - Dév.}% {Id.Rq - Facto.} % a^2 + 2 a b + b^2 \comthis{Forme dév.} \end{explain} </pre>	$ \begin{aligned} & (a + b)(a + b) && [\textit{Forme facto.}] \\ = & \{ \textit{Via } x^2 = x \cdot x. \} \\ & (a + b)^2 && [\textit{Au passage...}] \\ = & \left\{ \begin{array}{l} \downarrow \textit{Id.Rq - Dév.} \downarrow \\ \uparrow \textit{Id.Rq - Facto.} \uparrow \end{array} \right\} \\ & a^2 + 2ab + b^2 && [\textit{Forme dév.}] \end{aligned} $
---	--

VI. Détailler un « vrai » raisonnement

1. Un tableau pour le post-bac

Exemple 1 – Le minimum avec les réglages par défaut

Prenons un exemple utile à la logique formelle en informatique théorique mais qui a complètement sa place en mathématiques plus classiques (*voir la section 3. pour un autre type de présentation plus adapté à un public de collège ou de lycée*). Ci-dessous l'environnement `demoexplain` facilite la mise en page⁵ et la macro étoilée `\explref*` permet d'indiquer une référence interne au raisonnement⁶. Dans cet exemple en deux morceaux, pour montrer au passage comment continuer la numérotation là où elle s'était arrêtée, on utilise « *m.p.* » comme abréviation de « *modus ponens* ».

```

\begin{demoexplain}
  \demostep
    Hypothèse & $A$
  \demostep
    Axiome 1 & $A \implies B$
  \demostep
    m.p. sur
      \explref*{1} et \explref*{2}
    & $B$
  \demostep
    \explref*{1} et \explref*{3}
    & $A \wedge B$
\end{demoexplain}

```

1	Hypothèse	A
2	Axiome 1	$A \implies B$
3	m.p. sur 1 et 2	B
4	1 et 3	$A \wedge B$

Il est possible de couper sa démonstration en morceaux en indiquant à l'environnement la valeur du 1^{er} numéro de justification via la clé `start` : la valeur spéciale `last` indique de continuer la numérotation à la suite.

5. En coulisse est utilisé l'environnement `longtable` du package éponyme.

6. Les indications peuvent être numérotées jusqu'à 99 ce qui est bien au-delà des besoins pratiques.

```

\begin{demoexplain}[start = last]
  \demostep
    Axiome 3
    & $(A \wedge B) \implies C$
  \demostep
    m.p. sur
    \explref*{4} et \explref*{5}
    & $C$
\end{demoexplain}

```

5	Axiome 3	$(A \wedge B) \implies C$
6	m.p. sur 4 et 5	C

Exemple 2 – Référencer une indication

L'argument optionnel de `\demostep` permet de définir un label qui ensuite facilitera le référencement d'une justification de façon pérenne via la macro non étoilée `\explref`.

```

\begin{demoexplain}
  \demostep[demo-my-hyp]
    Hypothèse & $A$
  \demostep[demo-use-axiom-1]
    Axiome 1 & $A \implies B$
  \demostep
    m.p. sur
    \explref{demo-my-hyp}
    et
    \explref{demo-use-axiom-1}
    & $B$
\end{demoexplain}

```

1	Hypothèse	A
2	Axiome 1	$A \implies B$
3	m.p. sur 1 et 2	B

Remarque. Prendre bien garde au fait que ce mécanisme utilise les macros `\label` et `\ref` de L^AT_EX. On travaille donc avec des références globalement au document compilé.

Exemple 3 – Indiquer ce que l'on cherche à faire

Les clés optionnelles `hyps` pour plusieurs hypothèses, `hyp` pour une seule hypothèse et `ccl` pour la conclusion permettent d'expliquer ce que l'on démontre et sous quel contexte.

```

\begin{demoexplain}[hyp = $A$, ccl = $B$]
  \demostep
    Hypothèse & $A$
  \demostep
    Axiome 1 & $A \implies B$
  \demostep
    m.p. sur
    \explref*{1} et \explref*{2}
    & $B$
\end{demoexplain}

```

Démonstration sous l'hypothèse : A

1	Hypothèse	A
2	Axiome 1	$A \implies B$
3	m.p. sur 1 et 2	B

Conclusion : B

Remarque. Aucune des clés `hyps`, `hyp` et `ccl` n'est obligatoire. Par contre il n'est pas possible d'utiliser à la fois les clés `hyps` et `hyp`.

2. Un tableau sur plusieurs pages

Un tableau devant utiliser plusieurs pages sera scindé comme ci-dessous sans perte d'information ⁷.

1	Hypothèse	A
2	Axiome 1	$A \implies B$
Suite de la démo. page suivante...		
1		
3	m.p. sur 1 et 2	B
4	1 et 3	$A \wedge B$

3. Un tableau pour le collège et le lycée

Exemple 1 – Avec les réglages par défaut

L'environnement étoilé `demoexplain*` est différent de l'environnement `demoexplain` puisqu'il sert à indiquer trois choses et non juste deux comme le montre l'exemple suivant ⁸. Par contre, la syntaxe est très similaire. Notez au passage la possibilité d'utiliser `\newline` pour forcer un retour à la ligne dans une cellule et aussi la nécessité d'écrire les accolades de la macro sans argument `\demostep` lorsque la 1^{re} case est vide (*ceci est inutile lorsque l'argument optionnel est renseigné comme nous allons le vérifier dans l'exemple juste après*).

```
\begin{demoexplain*}
  \demostep
    $ABC$ est un triangle \newline équilatéral
    & Définition d'un triangle \newline équilatéral.
    & $AB = BC = AC$
  \demostep{} % --> Ne pas oublier ici !
    & Voir l'énoncé.
    & $AB = 10 \text{ \, cm}$
  \demostep
    Voir les conséquences \newline \explref*{1} et \explref*{2} .
    & Simple calcul.
    & $ABC$ a pour périmètre $30 \text{ \, cm}$.
\end{demoexplain*}
```

Réf.	Je sais que...	Propriété ou fait utilisé	Conséquence
1	ABC est un triangle équilatéral	Définition d'un triangle équilatéral.	$AB = BC = AC$
2		Voir l'énoncé.	$AB = 10 \text{ cm}$
3	Voir les conséquences 1 et 2 .	Simple calcul.	ABC a pour périmètre 30 cm .

7. Tout le travail est fait par l'environnement `longtable` du package éponyme.

8. C'est pour cela qu'est proposé une version étoilée de l'environnement et non l'utilisation d'une option de l'environnement non étoilé.

Exemple 2 – Avec toutes les options

Le système de référence marche ici aussi. Par contre `demoexplain*` ne propose que `start` comme clé optionnelle avec le même fonctionnement que pour `demoexplain`.

```
\begin{demoexplain*}[start = last]
  \demostep[demo-first-geo-fact]
    $ABC$ est un triangle \newline équilatéral
    & Définition d'un triangle \newline équilatéral.
    & $AB = BC = AC$
  \demostep[known-data]
    & Voir l'énoncé.
    & $AB = 10 \$, cm$
  \demostep
    Voir les conséquences \newline
    \explref{demo-first-geo-fact} et \explref{known-data} .
    & Simple calcul.
    & $ABC$ a pour périmètre $30 \$, cm$.
\end{demoexplain*}
```

Réf.	Je sais que...	Propriété ou fait utilisé	Conséquence
4	ABC est un triangle équilatéral	Définition d'un triangle équilatéral.	$AB = BC = AC$
5		Voir l'énoncé.	$AB = 10\text{ cm}$
6	Voir les conséquences <div>4</div> et <div>5</div> .	Simple calcul.	ABC a pour périmètre 30 cm .

4. Un tableau sur plusieurs pages

Un tableau devant utiliser plusieurs pages sera scindé comme ci-dessous sans perte d'information ⁹.

9. Tout le travail est fait par l'environnement `longtable` du package éponyme.

Réf.	Je sais que...	Propriété ou fait utilisé	Conséquence
1	ABC est un triangle équilatéral	Définition d'un triangle équilatéral.	$AB = BC = AC$
2		Voir l'énoncé.	$AB = 10\text{ cm}$
3	Voir les conséquences 1 et 2 .	Simple calcul.	ABC a pour périmètre 30 cm .
Suite de la démo. page suivante...			
1			
Réf.	Je sais que...	Propriété ou fait utilisé	Conséquence
4	ABC est un triangle équilatéral	Définition d'un triangle équilatéral.	$AB = BC = AC$
5		Voir l'énoncé.	$AB = 10\text{ cm}$
6	Voir les conséquences 4 et 5 .	Simple calcul.	ABC a pour périmètre 30 cm .

Chapitre D.

Géométrie

I. Ensembles géométriques

Se reporter à la section 2. page 15 où est présentée la macro `\setgeo` pour indiquer des ensembles géométriques.

II. Utiliser des unités S.I.

Comme l'excellent package `siunitx` est chargé en coulisse, il devient facile de travailler avec des unités de mesure tout en respectant **les conventions d'écriture sont françaises**¹ dès lors que vous aurez chargé `babel` avec l'option `french` comme c'est le cas pour cette documentation. Notez que les espaces dans `\num{123 000}` et `\num{1230 * 100}` sont inutiles mais ils facilitent la relecture du code.

```
$\ang{180} = \SI{\pi}{\radian}$ ,  
$\SI{1}{km} = \SI{1e3}{m}$ avec aussi  
$\num{123 000} = \num{1230 * 100}$
```

$180^\circ = \pi \text{ rad}$, $1 \text{ km} = 1 \times 10^3 \text{ m}$ avec aussi
 $123\,000 = 1230 \times 100$

III. Points et lignes

1. Points

Exemple 1 – Sans indice

```
$\pt{I}$
```

I

Exemple 2 – Avec un indice

```
$\pt*{I}{1}$ ou  
$\pt*{I}{2}$
```

I_1 ou I_2

1. Notez dans l'exemple l'écriture de 1230 n'utilise pas d'espace contrairement à celle de 12 300.

2. Lignes

Exemple 1 – Les droites

Dans l'exemple suivant, le préfixe `g` est pour `g`-éometrie tandis que `p` est pour `p`-oint.

$\backslash\mathrm{gline}\{A\}\{B\}$, $\backslash\mathrm{gline}\{\mathrm{pt}\{A\}\}\{\mathrm{pt}\{B\}\}$ ou $\backslash\mathrm{pgline}\{A\}\{B\}$	(AB) , (AB) ou (AB)
--	---------------------------

Exemple 2 – Les segments

Les macros `\segment` et `\psegment` ont un comportement similaire à `\gline` et `\pgline`.

$\backslash\mathrm{segment}\{A\}\{B\}$, $\backslash\mathrm{segment}\{\mathrm{pt}\{A\}\}\{\mathrm{pt}\{B\}\}$ ou $\backslash\mathrm{psegment}\{A\}\{B\}$	$[AB]$, $[AB]$ ou $[AB]$
--	---------------------------

Exemple 3 – Les demi-droites

Dans l'exemple suivant, le préfixe `h` est pour `h`-alf soit « *moitié* » en anglais.

$\backslash\mathrm{hgline}\{A\}\{B\}$, $\backslash\mathrm{hgline}\{\mathrm{pt}\{A\}\}\{\mathrm{pt}\{B\}\}$ ou $\backslash\mathrm{phgline}\{A\}\{B\}$	$[AB)$, $[AB)$ ou $[AB)$
---	---------------------------

Exemple 4 – D'autres demi-droites

Ce qui suit nécessite d'utiliser l'argument optionnel de `\gline` et `\pgline`. La valeur `OC` provient de `O`-pened – `C`-losed soit « *ouvert* – *fermé* » en anglais.

$\backslash\mathrm{gline}[OC]\{A\}\{B\}$, $\backslash\mathrm{gline}[OC]\{\mathrm{pt}\{A\}\}\{\mathrm{pt}\{B\}\}$ ou $\backslash\mathrm{pgline}[OC]\{A\}\{B\}$	$(AB]$, $(AB]$ ou $(AB]$
--	---------------------------

Remarque. Les segments utilisent en fait l'option `C` et les demi-droites standard l'option `CO`. La valeur par défaut est `O`.

3. Droites parallèles ou non

Les opérateurs `\parallel` et `\nparallel` utilisent des obliques au lieu de barres verticales comme le montre l'exemple qui suit où `\stdnparallel` est un alias de `\nparallel` fourni par le package `amssymb`, et `\stdparallel` est un alias de la version standard de `\parallel` proposée par L^AT_EX.

```

 $\pgline{A}{B} \parallel \pgline{C}{D}$ 
au lieu de
 $\pgline{A}{B}$ 
 $\stdparallel \pgline{C}{D}$ 

 $\pgline{E}{F} \nparallel \pgline{G}{H}$ 
au lieu de
 $\pgline{E}{F}$ 
 $\stdnparallel \pgline{G}{H}$ 

```

$(AB) \parallel (CD)$ au lieu de $(AB) \parallel (CD)$
 $(EF) \nparallel (GH)$ au lieu de $(EF) \nparallel (GH)$

IV. Vecteurs

1. Les écrire

Exemple 1

```

 $\vect{ABCDEFG}$  ,
 $\vect*{e}_{rot}$  ou
 $\vect{e_{rot}}$ 

```

$\overrightarrow{ABCDEFG}$, \vec{e}_{rot} ou $\overrightarrow{e_{rot}}$

Exemple 2

```

 $\vect{i}$  ou
 $\vect*{j}{2}$ 

```

\vec{i} ou \vec{j}_2

2. Norme

Ci-dessous l'argument optionnel de `\vnorm` vaut **b** par défaut pour **b**-ig soit « *gros* » en anglais mais l'on peut aussi utiliser **s** pour **s**-mall soit « *petit* ». Par contre `\vnorm` n'a pas d'option.

```

 $\norm{\vect{i}} = \vnorm{i}$ 

 $\norm{\dfrac{2}{7} \vect*{e}{k}}$ 
 $= \norm[s]{\dfrac{2}{7} \vect*{e}{k}}$ 

```

$\|\vec{i}\| = \|\vec{i}\|$
 $\left\| \frac{2}{7} \vec{e}_k \right\| = \left\| \frac{2}{7} \vec{e}_k \right\|$

Remarque. Le code L^AT_EX pour des doubles barres extensibles ou non vient directement de ce message : <https://tex.stackexchange.com/a/43009/6880>.

3. Produit scalaire

Les 1^{ers} exemples utilisent une syntaxe longue mais adaptables à toutes les situations. Voir l'exemple 5 un peu plus bas pour une écriture rapide utilisable dans certains cas.

Exemple 1 – Version classique

```

 $\dotprod{\dfrac{1}{2} \vect{u}}{\vect{v}}$ 

```

$\frac{1}{2} \vec{u} \cdot \vec{v}$

Exemple 2 – Version « pédagogique mais pas écolo. »

Dans l'exemple suivant l'option **b** est pour **b**-ullet soit « *puce* » en anglais. Cette écriture peut être utile avec des débutants mais elle est peu pratique pour une écriture manuscrite.

$\dot{\text{prod}}[b]{\frac{1}{2} \text{vect}\{u\}}{\text{vect}\{v\}}$	$\frac{1}{2} \vec{u} \cdot \vec{v}$
--	-------------------------------------

Exemple 3 – Écriture « universitaire »

Dans l'exemple suivant l'option **p** est pour **p**-arenthèse et dans **sp** le **s** est pour **s**-mall soit « *petit* » en anglais. On rencontre souvent cette écriture dans les cursus mathématiques universitaires.

$\dot{\text{prod}}[p]{\frac{1}{2} \text{vect}\{u\}}{\text{vect}\{v\}}$	$\left(\frac{1}{2} \vec{u} \mid \vec{v} \right)$
$\dot{\text{prod}}[sp]{\frac{1}{2} \text{vect}\{u\}}{\text{vect}\{v\}}$	$(\frac{1}{2} \vec{u} \mid \vec{v})$

Exemple 4 – Écriture « à la physicienne »

Dans l'exemple suivant **r** est pour **r**-after soit « *chevron* » en anglais. Les physiciens aiment bien cette notation.

$\dot{\text{prod}}[r]{\frac{1}{2} \text{vect}\{u\}}{\text{vect}\{v\}}$	$\left\langle \frac{1}{2} \vec{u} \mid \vec{v} \right\rangle$
$\dot{\text{prod}}[sr]{\frac{1}{2} \text{vect}\{u\}}{\text{vect}\{v\}}$	$\langle \frac{1}{2} \vec{u} \mid \vec{v} \rangle$

Exemple 5 – Version courte mais restrictive

Dans l'exemple suivant le préfixe **v** est pour **v**-ecteur. Notons que dans ce cas les options **sp** et **sr** n'apportent rien de nouveau.

$\text{vdotprod} \{u\}\{v\}$ $= \dot{\text{prod}}[b]{u}\{v\}$	$\vec{u} \cdot \vec{v} = \vec{u} \bullet \vec{v}$
$\text{vdotprod}[r]\{u\}\{v\}$ $= \dot{\text{prod}}[p]{u}\{v\}$	$\langle \vec{u} \mid \vec{v} \rangle = (\vec{u} \mid \vec{v})$

4. 3D – Produit vectoriel**i. Écriture symbolique****Exemple 1 – Version classique en France**

$\text{crossprod}\{\frac{1}{2} \text{vect}\{i\}}{\text{vect}\{j\}}$	$\frac{1}{2} \vec{i} \wedge \vec{j}$
---	--------------------------------------

Exemple 2 – Version alternative

La macro `\crossprod` possède un argument optionnel que l'on peut utiliser pour obtenir la mise en forme suivante.

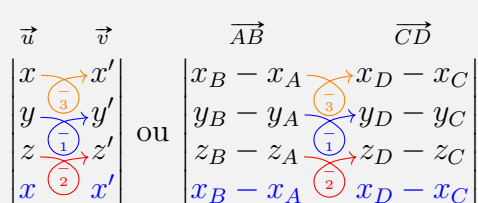
<code>$\backslash\text{crossprod}[t]{\dfrac{1}{2} \ \text{vect}\{i\}}\%$</code> <code>$\{\text{vect}\{j\}\}\\$</code>	$\frac{1}{2} \vec{i} \times \vec{j}$
---	--------------------------------------

Exemple 3 – Version courte mais restrictive

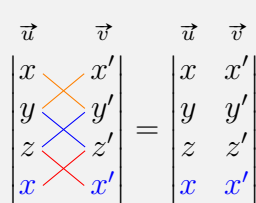
<code>$\backslash\text{vcrossprod} \ \{i\}\{j\}\\$</code> ou <code>$\backslash\text{vcrossprod}[t]{i}\{j\}\\$</code>	$\vec{i} \wedge \vec{j}$ ou $\vec{i} \times \vec{j}$
---	--

ii. Explication du mode de calcul

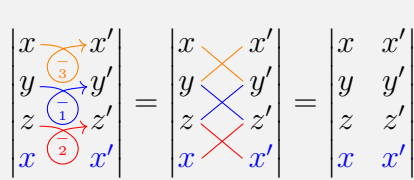
Dans l'exemple suivant, le préfixe `calc` est pour calculer et `v` pour v-ecteur.

<code>$\backslash\text{calccrossprod}\{\text{vect}\{u\}\{x\}\{y\}\{z\}\%$</code> <code>$\{\text{vect}\{v\}\{x'\}\{y'\}\{z'\}\}\\$</code> ou <code>$\backslash\text{vcalccrossprod}\{AB\}\{x_B - x_A\}\%$</code> <code>$\{y_B - y_A\}\%$</code> <code>$\{z_B - z_A\}\%$</code> <code>$\{CD\}\{x_D - x_C\}\%$</code> <code>$\{y_D - y_C\}\%$</code> <code>$\{z_D - z_C\}\%$</code>	
---	---

Avec un public averti on peut juste proposer des croix voir juste les coordonnées sans les décorations comme ci-après via les versions simplement et doublement étoilées de `\vcalccrossprod` (*ceci fonctionne aussi avec `\calccrossprod`*).

<code>$\backslash\text{vcalccrossprod*} \{u\}\{x\}\{y\}\{z\}\%$</code> <code>$\{v\}\{x'\}\{y'\}\{z'\}\%$</code> = <code>$\backslash\text{vcalccrossprod**}\{u\}\{x\}\{y\}\{z\}\%$</code> <code>$\{v\}\{x'\}\{y'\}\{z'\}\}\\$</code>	
--	--

Enfin si les vecteurs vous gênent il suffira d'utiliser l'option `novec` pour no vec-tor soit « pas de vecteur » en anglais comme ci-après. Ceci fonctionne aussi pour la macro `\calccrossprod`. Il peut sembler un peu lourd d'avoir des arguments pour des vecteurs non affichés mais ce choix permet à l'usage de faire des copier-coller redoutables d'efficacité!

<code>$\backslash\text{vcalccrossprod}[novec] \ \{u\}\{x\}\{y\}\{z\}\%$</code> <code>$\{v\}\{x'\}\{y'\}\{z'\}\%$</code> = <code>$\backslash\text{vcalccrossprod*}[novec] \ \{u\}\{x\}\{y\}\{z\}\%$</code> <code>$\{v\}\{x'\}\{y'\}\{z'\}\%$</code>	
--	--

iii. Les coordonnées

Exemple 1 – Les coordonnées « développées »

Pour avoir le détail directement dans des coordonnées vous pouvez faire appel à `\coordcrossprod` où le préfixe `coord` fait référence à `coord`-onnée². On peut utiliser des options pour choisir certains paramètres de mise en forme.

<code>\$\coordcrossprod{\dfrac{1}{2}x}{y}{z}\%</code> <code>{x'}{y'}{z'}\$</code>	$\left(y z' - z y' ; z x' - \frac{1}{2} x z' ; \frac{1}{2} x y' - y x' \right)$
<code>\$\coordcrossprod[vb]\%</code> <code>{\dfrac{1}{2}x}{y}{z}\%</code> <code>{x'}{y'}{z'}\$</code>	$\begin{bmatrix} y z' - z y' \\ z x' - \frac{1}{2} x z' \\ \frac{1}{2} x y' - y x' \end{bmatrix}$
<code>\$\coordcrossprod[sp,c]\%</code> <code>{\dfrac{1}{2}x}{y}{z}\%</code> <code>{x'}{y'}{z'}\$</code>	$(y \cdot z' - z \cdot y' ; z \cdot x' - \frac{1}{2} x \cdot z' ; \frac{1}{2} x \cdot y' - y \cdot x')$

Voici les options disponibles. Nous expliquons ensuite comment les utiliser.

1. `p` vient de `p`-arenthèses. Ceci donnera une écriture horizontale.
2. `b` vient de `b`-rackets soit « *crochets* » en anglais. Ceci donnera une écriture horizontale.
3. `sp` et `sb` produisent des délimiteurs non extensibles en mode horizontal. Ici `s` vient de `s`-mall soit « *petit* » en anglais.
4. `vp` et `vb` produisent des écritures verticales. Ici `v` vient de `v`-ertical.
5. `s` tout seul demande d'utiliser un espace pour séparer les facteurs de chaque produit.
6. `t` tout seul demande d'utiliser `\times` comme opérateur de multiplication.
7. `c` tout seul demande d'utiliser `\cdot` comme opérateur de multiplication.

On peut indiquer des options vis à vis du mode vertical ou horizontal avec des délimiteurs extensibles ou non éventuellement, ou bien sur le symbole pour les produits. On peut aussi combiner deux de ces types de choix en les séparant par une virgule ce qui fait un total de $6 \times 3 = 18$ combinaisons possibles. La valeur par défaut est `p,s`.

Attention ! Les produits sont rédigés stupidement. Autrement dit ce sera à vous d'ajouter des parenthèses là où il y en aura besoin sinon vous obtiendrez des horreurs comme celle ci-dessous.

<code>\$\coordcrossprod[vb]\%</code> <code>{x_B - x_A}{y_B - y_A}\%</code> <code>{z_B - z_A}{x_D - x_C}\%</code> <code>{y_D - y_C}{z_D - z_C}\$</code>	$\begin{bmatrix} y_B - y_A z_D - z_C - z_B - z_A y_D - y_C \\ z_B - z_A x_D - x_C - x_B - x_A z_D - z_C \\ x_B - x_A y_D - y_C - y_B - y_A x_D - x_C \end{bmatrix}$
---	---

Ici nous n'avons pas d'autre choix que de corriger le tir nous-même. Ceci étant indiqué, ce genre de situation est très rare dans la vraie vie mathématique où l'on évite d'avoir à calculer un produit vectoriel avec des expressions compliquées.

<code>\$\coordcrossprod[vb]\%</code> <code>{(x_B - x_A)}{(y_B - y_A)}\%</code> <code>{(z_B - z_A)}{(x_D - x_C)}\%</code> <code>{(y_D - y_C)}{(z_D - z_C)}\$</code>	$\begin{bmatrix} (y_B - y_A)(z_D - z_C) - (z_B - z_A)(y_D - y_C) \\ (z_B - z_A)(x_D - x_C) - (x_B - x_A)(z_D - z_C) \\ (x_B - x_A)(y_D - y_C) - (y_B - y_A)(x_D - x_C) \end{bmatrix}$
---	---

2. En coulisse on utilise la macro `\coord` présentée dans la section 1. page 49.

5. 2D – Critère de colinéarité de deux vecteurs

Exemple 1 – Version complète

Dans l'exemple suivant, le préfixe coli est pour colin-éarité et criteria signifie « critère » en anglais.

<pre> $\backslash\text{colicriteria}\{\text{vect}\{u\}\{x\}\{y\}\%$ $\{\text{vect}\{v\}\{x'\}\{y'\}\\$ ou $\backslash\text{colicriteria}\{\text{vect}\{AB\}\}\%$ $\{x_B - x_A\}\{y_B - y_A\}\%$ $\{\text{vect}\{CD\}\}\%$ $\{x_D - x_C\}\{y_D - y_C\}\\$ </pre>	
---	--

Exemple 2 – Rédaction raccourcie pour les vecteurs

Dans l'exemple suivant, le préfixe v est pour v-ecteur.

<pre> $\backslash\text{vcolicriteria}\{u\}\{x\}\{y\}\%$ $\{v\}\{x'\}\{y'\}\\$ </pre>	
---	--

Exemple 3 – Versions sans les vecteurs

Dans l'exemple suivant, on utilise la valeur novect pour l'argument optionnel de $\backslash\text{vcolicriteria}$ qui par défaut est vect pour pour vecteur. À l'usage ceci permet des copier-coller très efficaces !

<pre> $\backslash\text{vcolicriteria}[\text{novect}]\{u\}\{x\}\{y\}\%$ $\{v\}\{x'\}\{y'\}\\$ </pre>	
--	--

6. 2D – Déterminant de deux vecteurs

Exemple 1 – Version décorée

Dans l'exemple suivant, le préfixe calc est pour calc-uler.

<pre> $\backslash\text{calcdetplane}\{\text{vect}\{u\}\{x\}\{y\}\%$ $\{\text{vect}\{v\}\{x'\}\{y'\}\\$ ou $\backslash\text{calcdetplane}\{\text{vect}\{AB\}\}\%$ $\{x_B - x_A\}\{y_B - y_A\}\%$ $\{\text{vect}\{CD\}\}\%$ $\{x_D - x_C\}\{y_D - y_C\}\\$ </pre>	
---	--

Exemple 2 – Version non décorée

<pre> $\backslash\text{calcdetplane}*\{\text{vect}\{u\}\{x\}\{y\}\%$ $\{\text{vect}\{v\}\{x'\}\{y'\}\\$ </pre>	
---	--

Exemple 3 – Rédaction raccourcie pour les vecteurs

$\begin{aligned} & \$\backslash\text{vcaldetplane}\{u\}\{x\}\{y\}\% \\ & \qquad \qquad \{v\}\{x'\}\{y'\}\% \\ & = \\ & \backslash\text{vcaldetplane}*\{u\}\{x\}\{y\}\% \\ & \qquad \qquad \{v\}\{x'\}\{y'\}\% \end{aligned}$	$\begin{array}{cc} \vec{u} & \vec{v} \\ \left \begin{array}{cc} x & x' \\ y & y' \end{array} \right & = \left \begin{array}{cc} x & x' \\ y & y' \end{array} \right \end{array}$
--	--

Exemple 4 – Versions sans les vecteurs

$\begin{aligned} & \$\backslash\text{vcaldetplane}[\text{novec}] \{u\}\{x\}\{y\}\% \\ & \qquad \qquad \{v\}\{x'\}\{y'\}\% \\ & = \backslash\text{vcaldetplane}*\{u\}\{x\}\{y\}\% \\ & \qquad \qquad \{v\}\{x'\}\{y'\}\% \end{aligned}$	$\begin{array}{cc} \left \begin{array}{cc} x & x' \\ y & y' \end{array} \right & = \left \begin{array}{cc} x & x' \\ y & y' \end{array} \right \end{array}$
--	---

Remarque. Ce qui précède marche aussi avec les macros `\caldetplane` et `\caldetplane*`.

Exemple 5 – Calcul développé

Grâce à l'argument optionnel de `\caldetplane` ou de `\vcaldetplane` il est aussi possible d'obtenir le résultat développé du calcul comme ci-après où `exp` est pour `exp-and` soit « *développer* » en anglais, `c` pour `\cdot` et enfin `t` pour `\times`. Même si les vecteurs ne sont pas utilisés pour la mise en forme, on obtient ici une méthode très pratique à l'usage car elle permet de faire des copier-coller.

$\begin{aligned} & \$\backslash\text{vcaldetplane}[\text{exp}]\{u\}\{x\}\{y\}\% \\ & \qquad \qquad \{v\}\{x'\}\{y'\}\% \end{aligned}$	
$\begin{aligned} & \$\backslash\text{vcaldetplane}[\text{cexp}]\{u\}\{x\}\{y\}\% \\ & \qquad \qquad \{v\}\{x'\}\{y'\}\% \end{aligned}$	$\begin{aligned} & x y' - x' y \\ & x \cdot y' - x' \cdot y \\ & x \times y' - x' \times y \end{aligned}$
$\begin{aligned} & \$\backslash\text{vcaldetplane}[\text{texp}]\{u\}\{x\}\{y\}\% \\ & \qquad \qquad \{v\}\{x'\}\{y'\}\% \end{aligned}$	

Remarque. Ce qui précède marche aussi avec les versions étoilées.

Attention ! Le développement effectué est stupide. Autrement dit ce sera à vous d'ajouter des parenthèses là où il y en aura besoin sinon vous obtiendrez des horreurs comme celle qui suit.

$\begin{aligned} & \$\backslash\text{vcaldetplane}[\text{exp}]\{AB\}\% \\ & \qquad \{x_B - x_A\}\% \\ & \qquad \{y_B - y_A\}\% \\ & \qquad \{CD\}\% \\ & \qquad \{x_D - x_C\}\% \\ & \qquad \{y_D - y_C\}\% \end{aligned}$	$x_B - x_A y_D - y_C - x_D - x_C y_B - y_A$
--	---

Ici nous n'avons pas d'autre choix que de régler le problème à la main. Ce genre de situation n'est pas rare dans la vraie vie mathématique.

$\begin{aligned} &\$ \backslash \text{vcaldetplane}[\text{exp}]\{AB\}\% \\ &\quad \{(x_B - x_A)\}\% \\ &\quad \{(y_B - y_A)\}\% \\ &\quad \{CD\}\% \\ &\quad \{(x_D - x_C)\}\% \\ &\quad \{(y_D - y_C)\}\$ \end{aligned}$	$(x_B - x_A)(y_D - y_C) - (x_D - x_C)(y_B - y_A)$
---	---

V. Géométrie cartésienne

1. Coordonnées

Exemple 1 – Des coordonnées seules

`tnsgeo` propose, via un argument optionnel, six façons différentes de rédiger des coordonnées seules (*nous verrons après des macros pour les coordonnées d'un point et celles d'un vecteur afin de produire un code $L^A T_E X$ plus sémantique*). Commençons par les écritures horizontales où vous noterez l'utilisation de `|` pour séparer les coordonnées dont le nombre peut être quelconque.

$\begin{aligned} &\$ \backslash \text{coord} \quad \{\backslash \text{dfrac}\{1\}\{3\} \mid -4 \mid 0\}\$ \text{ ou} \\ &\$ \backslash \text{coord}[\text{sp}]\{\backslash \text{dfrac}\{1\}\{3\} \mid -4 \mid 0\}\$ \end{aligned}$	$\left(\frac{1}{3}; -4; 0\right) \text{ ou } \left(\frac{1}{3}; -4; 0\right)$
$\begin{aligned} &\$ \backslash \text{coord}[\text{b}] \{\backslash \text{dfrac}\{1\}\{3\} \mid -4 \mid 0\}\$ \text{ ou} \\ &\$ \backslash \text{coord}[\text{sb}]\{\backslash \text{dfrac}\{1\}\{3\} \mid -4 \mid 0\}\$ \end{aligned}$	$\left[\frac{1}{3}; -4; 0\right] \text{ ou } \left[\frac{1}{3}; -4; 0\right]$

Il existe en plus deux versions verticales.

$\begin{aligned} &\$ \backslash \text{coord}[\text{vp}]\{3 \mid -4\}\$ \text{ ou} \\ &\$ \backslash \text{coord}[\text{vb}]\{3 \mid -4\}\$ \end{aligned}$	$\begin{pmatrix} 3 \\ -4 \end{pmatrix} \text{ ou } \begin{bmatrix} 3 \\ -4 \end{bmatrix}$
---	---

Voici d'où viennent les noms des options.

1. `p`, qui est aussi la valeur par défaut, vient de `p`-arenthèses.
2. `b` vient de `b`-rackets soit « *crochets* » en anglais.
3. `s` pour `s`-mall soit « *petit* » en anglais permet d'avoir des délimiteurs non extensibles en mode horizontal car par défaut ils le sont.
4. `v` pour `v`-ertical demande de produire une écriture verticale.

Exemple 2 – Coordonnées d'un point

La macro `\pcoord` avec `p` pour `p`-oint prend un argument supplémentaire avant les coordonnées qui est le nom d'un point qui sera mis en forme par la macro `\pt`. Si vous ne souhaitez pas que `\pt` soit appliquée, il suffit de passer via la version étoilée `\pcoord*`.

$\begin{aligned} &\$ \backslash \text{pcoord}\{A\}\{3 \mid -4 \mid 0 \mid -1\}\$ \text{ ou} \\ &\$ \backslash \text{pcoord}*\{\backslash \text{Sigma}\}\{7 \mid 9 \mid 8\}\$ \end{aligned}$	$A(3; -4; 0; -1) \text{ ou } \Sigma(7; 9; 8)$
---	---

Toutes les options disponibles avec `\coord` le sont aussi avec `\pcoord`.

$\begin{aligned} &\$ \backslash \text{pcoord}[\text{b}]\{A\}\{3 \mid -4 \mid 0 \mid -1\}\$ \text{ ou} \\ &\$ \backslash \text{pcoord}*\{b\}\{\backslash \text{Sigma}\}\{7 \mid 9 \mid 8\}\$ \end{aligned}$	$A[3; -4; 0; -1] \text{ ou } \Sigma[7; 9; 8]$
--	---

Exemple 3 – Coordonnées d'un vecteur

Le fonctionnement de `\vcoord` est similaire à celui de `\pcoord` si ce n'est que c'est la macro `\vect` qui sera appliquée si besoin.

$\begin{aligned} &\$ \backslash \text{vcoord}\{u\}\{3 \mid -4\} \$ \text{ ou} \\ &\$ \backslash \text{vcoord}\{* \{ \dfrac{1}{2} \} \backslash \text{vect}\{u\}\} \% \\ &\quad \{3 \mid -4\} \$ \end{aligned}$	$\vec{u}(3; -4) \text{ ou } \frac{1}{2} \vec{u}(3; -4)$
$\begin{aligned} &\$ \backslash \text{vcoord}\{vp\}\{u\}\{3 \mid -4\} \$ \text{ ou} \\ &\$ \backslash \text{vcoord}\{*\{vp\}\{ \dfrac{1}{2} \} \backslash \text{vect}\{u\}\} \% \\ &\quad \{3 \mid -4\} \$ \end{aligned}$	$\vec{u}\left(\begin{smallmatrix} 3 \\ -4 \end{smallmatrix}\right) \text{ ou } \frac{1}{2} \vec{u}\left(\begin{smallmatrix} 3 \\ -4 \end{smallmatrix}\right)$

2. Nommer un repère**Exemple 1 – La méthode basique**

Commençons par la manière la plus basique d'écrire un repère (*nous verrons d'autres méthodes qui peuvent être plus efficaces*).

$\begin{aligned} &\$ \backslash \text{axes}\{ \backslash \text{pt}\{0\} \% \\ &\quad \mid \backslash \text{pt}\{I\} \mid \backslash \text{pt}\{J\} \} \$ \end{aligned}$	$(O; I, J)$
---	-------------

Exemple 2 – La méthode basique en version étoilée

Dans l'exemple ci-dessous, on voit que la version étoilée produit des petites parenthèses.

$\begin{aligned} &\$ \backslash \text{axes}\{ \backslash \text{pt}\{0\} \% \\ &\quad \mid \backslash \text{dfrac}\{7\}\{3\} \backslash \text{vect}\{i\} \% \\ &\quad \mid \backslash \text{vect}\{j\} \} \$ \\ \text{ou} \\ &\$ \backslash \text{axes}\{* \{ \backslash \text{pt}\{0\} \% \\ &\quad \mid \backslash \text{dfrac}\{7\}\{3\} \backslash \text{vect}\{i\} \% \\ &\quad \mid \backslash \text{vect}\{j\} \} \$ \end{aligned}$	$\left(O; \frac{7}{3} \vec{i}, \vec{j}\right) \text{ ou } \left(O; \frac{7}{3} \vec{i}, \vec{j}\right)$
---	---

Exemple 3 – La méthode basique en dimension quelconque

Il faut au minimum deux "morceaux" séparés par des barres `|`, cas de la dimension 1, mais il n'y a pas de maximum, cas d'une dimension quelconque $n > 0$.

$\begin{aligned} &\$ \backslash \text{axes}\{ \backslash \text{pt}\{0\} \% \\ &\quad \mid \backslash \text{vect}\{* \{i\}\{1\} \% \\ &\quad \mid \backslash \text{vect}\{* \{i\}\{2\} \% \\ &\quad \mid \backslash \text{vect}\{* \{i\}\{3\} \% \\ &\quad \mid \backslash \text{dots} \% \\ &\quad \mid \backslash \text{vect}\{* \{i\}\{9\} \% \\ &\quad \mid \backslash \text{vect}\{* \{i\}\{10\} \% \\ &\quad \mid \backslash \text{vect}\{* \{i\}\{11\} \% \\ &\quad \mid \backslash \text{vect}\{* \{i\}\{12\} \} \$ \end{aligned}$	$(O; \vec{i}_1, \vec{i}_2, \vec{i}_3, \dots, \vec{i}_9, \vec{i}_{10}, \vec{i}_{11}, \vec{i}_{12})$
---	--

Exemple 4 – Repère affine

Dans l'exemple suivant, le préfixe p est pour p-oint.

$\$ \backslash \text{paxes} \{ 0 \mid I \mid J \mid K \} \$$ au lieu de $\$ \backslash \text{axes} \{ \backslash \text{pt} \{ 0 \} \mid \backslash \text{pt} \{ I \} \mid \backslash \text{pt} \{ J \} \mid \backslash \text{pt} \{ K \} \} \$$	$(O ; I, J, K)$ au lieu de $(O ; I, J, K)$
---	--

Exemple 5 – Repère vectoriel (méthode 1)

Dans l'exemple suivant, le préfixe v est pour v-ecteur.

$\$ \backslash \text{vaxes} \{ \backslash \text{pt} \{ 0 \} \mid i \mid j \} \$$ au lieu de $\$ \backslash \text{axes} \{ \backslash \text{pt} \{ 0 \} \mid \backslash \text{vect} \{ i \} \mid \backslash \text{vect} \{ j \} \} \$$	$(O ; \vec{i}, \vec{j})$ au lieu de $(O ; \vec{i}, \vec{j})$
---	--

Exemple 6 – Repère vectoriel (méthode 2)

Dans l'exemple suivant, le préfixe pv permet de combiner ensemble les fonctionnalités proposées par les préfixes p et v.

$\$ \backslash \text{pvaxes} \{ 0 \mid i \mid j \} \$$ au lieu de $\$ \backslash \text{axes} \{ \backslash \text{pt} \{ 0 \} \mid \backslash \text{vect} \{ i \} \mid \backslash \text{vect} \{ j \} \} \$$	$(O ; \vec{i}, \vec{j})$ au lieu de $(O ; \vec{i}, \vec{j})$
---	--

VI. Arcs circulaires

Exemple 1

$\$ \backslash \text{circarc} \{ ABCDEF \} \$$, $\$ \backslash \text{circarc} * \{ A \} \{ \text{rot} \} \$$ ou $\$ \backslash \text{circarc} \{ A_{\text{rot}} \} \$$	\widehat{ABCDEF} , \widehat{A}_{rot} ou $\widehat{A_{rot}}$
---	---

Exemple 2

$\$ \backslash \text{circarc} \{ i \} \$$ ou $\$ \backslash \text{circarc} * \{ j \} \{ 2 \} \$$	\widehat{i} ou \widehat{j}_2
---	----------------------------------

VII. Angles

1. Angles géométriques « intérieurs »

Exemple 1

<code>\$\anglein {ABCDEF}\$</code>	\widehat{ABCDEF}
<code>\$\anglein* {A}{rot}\$</code>	$\widehat{A_{rot}}$
<code>\$\anglein {A_{rot}}\$</code>	$\widehat{A_{rot}}$

Exemple 2 – Cacher les points du i et du j

<code>\$\anglein{i}\$ et \$\anglein*{j}{2}\$</code>	\widehat{i} et $\widehat{j_2}$
---	----------------------------------

2. Angles orientés de vecteurs

Sans chapeau - Version longue

L'option par défaut est p pour p-arenthèse. Dans sp le s est pour s-mall soit « *petit* » en anglais.

<code>\$\angleorient {\dfrac{1}{2} \vect{i}}% {\vect{j}}\$</code>	$\left(\frac{1}{2} \vec{i} ; \vec{j} \right)$
<code>\$\angleorient[sp]{\dfrac{1}{2} \vect{i}}% {\vect{j}}\$</code>	$\left(\frac{1}{2} \vec{i} ; \vec{j} \right)$

Sans chapeau - Version courte mais restrictive

Dans l'exemple suivant, le préfixe v est pour v-ecteur qui permet de simplifier la saisie quand l'on a juste des vecteurs nommés avec des lettres (*notez que l'option sp n'apporte rien de nouveau*).

<code>\$\vangleorient {i}{j}\$ comme \$\vangleorient[sp]{i}{j}\$</code>	$(\vec{i} ; \vec{j})$ comme $(\vec{i} ; \vec{j})$
---	---

Avec un chapeau

Dans l'exemple suivant, h est pour h-at soit « *chapeau* » en anglais. Notez au passage que sh produit juste des parenthèses petites mais ce choix de nom simplifie l'utilisation de la macro (*c'est mieux que hsp par exemple*).

$\$ \backslash \text{angleorient}[h] \{ \dfrac{1}{2} \ \text{vect}\{i\} \} \% \{ \text{vect}\{j\} \} \$$	$\widehat{\left(\frac{1}{2} \vec{i} ; \vec{j} \right)}$
$\$ \backslash \text{angleorient}[sh] \{ \dfrac{1}{2} \ \text{vect}\{i\} \} \% \{ \text{vect}\{j\} \} \$$	$\overline{\left(\frac{1}{2} \vec{i} ; \vec{j} \right)}$
$\$ \backslash \text{vangleorient}[h] \{ i \} \{ j \} \$$ comme $\$ \backslash \text{vangleorient}[sh] \{ i \} \{ j \} \$$	$\widehat{(\vec{i} ; \vec{j})}$ comme $\overline{(\vec{i} ; \vec{j})}$

Chapitre E.

Analyse

I. Constantes et paramètres

1. Constantes classiques ajoutées

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

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

γ , π , τ , 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.

2. 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 $ax^2 + bx + c$

II. 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.

III. 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>.

IV. Fonctions nommées spéciales

1. 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 3.

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

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

2. 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`

$$\log_2 x = \lg x \text{ ou } \exp_6 y = 6^y$$

3. Toutes les fonctions nommées en plus

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

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

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

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

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

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

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

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

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

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

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

`\exp x` : $\exp x$

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

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

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

`\log x` : $\log x$

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

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

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

V. 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.

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

$$\lim_{x \rightarrow 0} f(x) = \frac{1}{2}$$

Remarque. $\backslash\text{lim}\backslash\text{limits}_{\{x \rightarrow 0\}} f(x)$ produit $\lim_{x \rightarrow 0} f(x)$ contre $\lim_{x \rightarrow 0} f(x)$ ci-dessus. La version $\backslash\text{limit}$ 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 $\backslash\text{limit}$ qui permet d'ajouter facilement plusieurs conditions¹ comme le montre la 2^e limite ci-après.

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

$$\lim_{\substack{x \rightarrow 0 \\ x > 0}} f(x) \text{ ou } \lim_{\substack{z \rightarrow 0 \\ z \in \Omega \\ |z| < 3}} f(z)$$

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.

$$\text{\$}\backslash\text{displaystyle}\backslash\text{limit}[\text{p}] \left\{ \text{\frac{1}{x}} \right\} \{x\} \{0 \mid x > 0\} \text{\$}$$

ou

$$\text{\$}\backslash\text{displaystyle}\backslash\text{limit}[\text{sp}] \left\{ \text{\frac{1}{x}} \right\} \{x\} \{0 \mid x > 0\} \text{\$}$$

$$\lim_{\substack{x \rightarrow 0 \\ x > 0}} \left(\frac{1}{x} \right) \text{ ou } \lim_{\substack{x \rightarrow 0 \\ x > 0}} \left(\frac{1}{x} \right)$$

VI. Calcul différentiel

1. Les opérateurs ∂ et d

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

$$\text{\$}\backslash\text{dd}\{t\} = \backslash\text{dd}[1]\{t\} \text{\$} \text{ ou } \text{\$}\backslash\text{dd}[n]\{x\} \text{\$}$$

$$\text{\$}\backslash\text{pp}\{t\} = \backslash\text{pp}[1]\{t\} \text{\$} \text{ ou } \text{\$}\backslash\text{pp}[n]\{x\} \text{\$}$$

$$dt = d^1 t \text{ ou } d^n x$$

$$\partial t = \partial^1 t \text{ ou } \partial^n x$$

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

Exemple 1 – Différentes écritures possibles

La macro $\backslash\text{der}$ 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 3. qui présente la macro $\backslash\text{sder}$ permettant une rédaction efficace pour obtenir $f^{(1)}$ ou f'*). Voici plusieurs mises en forme faciles à taper via l'option de $\backslash\text{der}$. 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 !

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

```

 $\backslash\text{der}$  {f}{x}{3}
=  $\backslash\text{der}[e]{f}{x}{3}$ 
=  $\backslash\text{der}[d]{f}{x}{3}$ 

```

```

 $\backslash\text{der}[i]{u}{x}{k}$ 
=  $\backslash\text{der}[f]{u}{x}{k}$ 
=  $\backslash\text{der}[sf]{u}{x}{k}$ 

```

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

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

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 4, page 59 à ce sujet.

```

 $\backslash\text{der}[osf,sp]{\frac{1}{2} uv}{x}{k}$ 
=  $\backslash\text{der}[of,p]{\frac{1}{2} uv}{x}{k}$ 

```

$$\frac{d^k}{dx^k} \left(\frac{1}{2} uv \right) = \frac{d^k}{dx^k} \left(\frac{1}{2} uv \right)$$

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.

```

 $\backslash\text{der}[bd]{\frac{1}{2} uv}{x}{2}$ 
=  $\backslash\text{der}[bd,p]{\frac{1}{2} uv}{x}{2}$ 
=  $\backslash\text{der}[bd,sp]{\frac{1}{2} uv}{x}{2}$ 

```

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

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

```

 $\backslash\text{der}[i]{u}{x}{1}$ 
=  $\backslash\text{der}[f]{u}{x}{1}$ 
=  $\backslash\text{der}[sf]{u}{x}{1}$ 
=  $\backslash\text{der}[of]{u}{x}{1}$ 

```

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

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

```

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

```

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

3. 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

```

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

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

```

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

Exemple 2

```

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

```

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

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

4. 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`.

```

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

```

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

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

```

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

```

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

5. 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.

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

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.

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

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

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

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 6. page 60 à ce sujet.

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

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\}`.

6. 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 ∂ .

$\$ \backslash pderope \quad \{x \mid y^2\}^{\{3\}}$ $= \backslash pderope[sf] \{x \mid y^2\}^{\{3\}}$ $= \backslash pderope[i] \{x \mid y^2\}^{\{3\}} \$$	$\frac{\partial^3}{\partial x \partial y^2} = \frac{\partial^3}{\partial x \partial y^2} = \partial^3_{x,y(2)}$
--	---

VII. Tableaux de variation et de signe

1. 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.

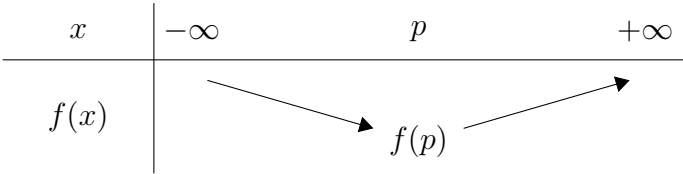
Exemple 1 – Avec des signes

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

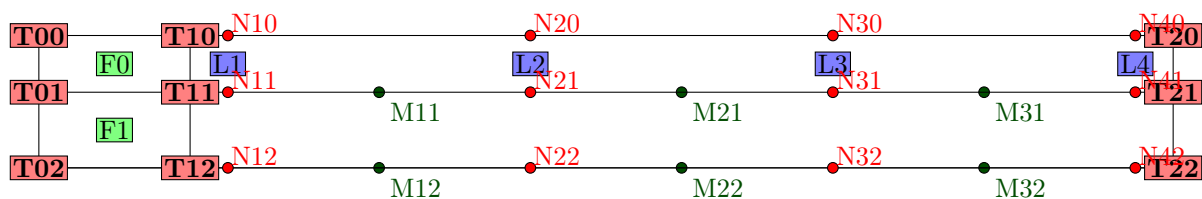
Voici le code du 1^{er} tableau où le placement de la valeur 3 au milieu entre 2 et $+\infty$ va nous simplifier le travail pour le 2^e tableau.


```

\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 maison³.



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[
    espc1 = 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}
```

2. Décorer facilement un tableau

i. Motivation

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

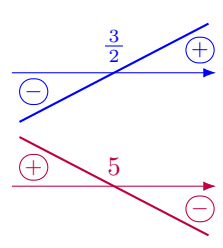
x	$-\infty$	$\frac{3}{2}$	5	$+\infty$
Signe de $2x - 3$	$-$	0	$+$	$+$
Signe de $-x + 5$	$+$	$+$	0	$-$
Signe de $(2x - 3)(-x + 5)$	$-$	0	$+$	$-$

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 72.

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 72 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}{\$\$}`.

- *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 $	$\exp x$	$\ln x$

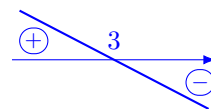
v. Quelques exemples

Exemple 1 – Avec une parabole

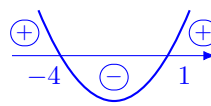
Il devient très facile de proposer un tableau décoré comme le suivant.

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	+

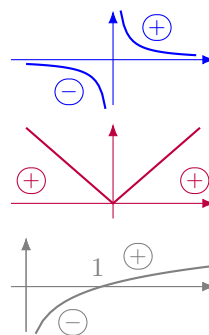
Le code correspondant est le suivant.

```
\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 -
$f(x)$		$-\infty$ \nearrow e \searrow 0	

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

Les limites sont hors programme pour cette année.

VIII. Calcul intégral

1. Le symbole standard revisité

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


```


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


```

```


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


```

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

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

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.

2. 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.

```


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


```

```


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


```

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

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

Exemple 2 – Le mode `displaystyle`

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

```


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


```

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

3. L'opérateur « crochet »

Exemple 1

```


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


```

```


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


```

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

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

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*`.

```

 $\hook*\{\dfrac{x - 1}{5 + x^2}\}{x}\%$ 
 $\hook*[sb]\{\dfrac{x - 1}{5 + x^2}\}{x}\%$ 

```

$$\left[\frac{x - 1}{5 + x^2} \right]_a^b = \left[\frac{x - 1}{5 + x^2} \right]_a^b$$

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.

```

 $\hook[r] \{\dfrac{x - 1}{5 + x^2}\}{x}\%$ 
 $\hook*[sr]\{\dfrac{x - 1}{5 + x^2}\}{x}\%$ 

```

$$\frac{x - 1}{5 + x^2} \Big|_{x=a}^{x=b} = \frac{x - 1}{5 + x^2} \Big|_a^b$$

Chapitre F.

Suites

I. Des notations complémentaires pour des suites spéciales

Voici trois types de suites avec deux ou quatre indices.

<code>$\backslash\mathrm{seqplus}\{F\}\{1\}\{2\}$</code>	F_1^2
<code>$\backslash\mathrm{seqhypergeo}\{F\}\{1\}\{2\}$</code>	${}_1F_2$
<code>$\backslash\mathrm{seqsuprgeo}\{F\}\{1\}\{2\}\{3\}\{4\}$</code> <code>pour les fous\ldots :-)</code>	${}_1F_2^3$ pour les fous... :-)

II. Sommes et produits en mode ligne

Pour limiter l'espace, L^AT_EX affiche $\sum_{k=0}^n$ et non $\sum_{k=0}^n$ sauf si l'on utilise la commande `\displaystyle`.

Les macros `\dsu` et `\dpr` permettent de se passer de `\displaystyle`. Voici un exemple.

<code>$\backslash\mathrm{dsu}_{\{k=0\}}^{\{n\}} 2^k$</code> <code>= $\backslash\mathrm{sum}_{\{k=0\}}^{\{n\}} 2^k$</code>	$\sum_{k=0}^n 2^k = \sum_{k=0}^n 2^k$
<code>$\backslash\mathrm{dpr}_{\{k=1\}}^{\{n\}} k$</code> <code>= $\backslash\mathrm{prod}_{\{k=1\}}^{\{n\}} k$</code>	$\prod_{k=1}^n k = \prod_{k=1}^n k$

Remarque. On peut taper $\sum_{k=0}^n \frac{1}{n}$ où la fraction n'est pas en mode `\displaystyle`.

III. Comparaison asymptotique de suites et de fonctions

1. Les notations \mathcal{O} et \mathcal{o}

Exemple 1

Les notations suivantes sont dues à Landau.

<code>$\backslash\mathrm{bigO}\{\}$</code> ou <code>$\backslash\mathrm{smallO}\{\}$</code>	\mathcal{O} ou \mathcal{o}
--	--------------------------------

Exemple 2

`$\bigO{x} \neq \smallO{x}$` ou
`$e^{t + \smallO{t}} = e^{\bigO{t}}$`

$\mathcal{O}(x) \neq \mathcal{o}(x)$ ou $e^{t+\mathcal{o}(t)} = e^{\mathcal{O}(t)}$

2. La notation Ω **Exemple 1**

La notation suivante est due à Hardy et Littlewood.

`$\bigomega{}$`

Ω

Exemple 2

Dans l'exemple suivant, $f(n) = \Omega(g(n))$ signifie : $\exists(m, n_0)$ tel que $n \geq n_0$ implique $f(n) \geq mg(n)$.

`$f(n) = \bigomega{g(n)}$`

$f(n) = \Omega(g(n))$

3. La notation Θ **Exemple 1**

`$\bigtheta{}$`

Θ

Exemple 2

Dans l'exemple suivant, $f(n) = \Theta(g(n))$ signifie : $\exists(m, M, n_0)$ tel que $mg(n) \leq f(n) \leq Mg(n)$ dès que $n \geq n_0$.

`$f(n) = \bigtheta{g(n)}$`

$f(n) = \Theta(g(n))$

Chapitre G.

Probabilité

I. Ensembles classiques de nombres

Se reporter à la section 3. page 16 où est présentée la macro `\setproba` pour indiquer des ensembles de type probabiliste.

II. Généralités

1. Probabilité « simple »

Exemple 1

<code>$\backslash\proba{A}$</code>	$p(A)$
---	--------

Exemple 2 – Choisir le nom de la probabilité

<code>$\backslash\proba[P]{A}$</code>	$P(A)$
--	--------

2. Probabilité conditionnelle

Exemple 1 – Les deux écritures classiques

La 1^{re} notation, qui est devenue standard, permet de comprendre l'ordre des arguments.

<code>$\backslash\probacond{B}{A}$ <code>= $\backslash\probacond*{B}{A}$</code></code>	$p_B(A) = p(A \mid B)$
--	------------------------

Exemple 2 – Obtenir la formule de définition

Le préfixe `e` est pour `e-xpand` soit « *développer* » en anglais¹.

<code>$\backslash\eprobacond{B}{A}$ <code>= $\backslasheprobacond*{B}{A}$</code></code>	$\frac{p(A \cap B)}{p(B)} = \frac{p(A \cap B)}{p(B)}$
---	---

1. Pour ne pas alourdir l'utilisation de `\probacond`, il a été choisi d'utiliser un préfixe au lieu d'un système de multi-options.

Exemple 3 – Choisir le nom de la probabilité

```

 $\backslash$ probacond [P]{B}{A}
=  $\backslash$ probacond* [P]{B}{A}
=  $\backslash$ eprobacond*[P]{B}{A}
=  $\backslash$ eprobacond [P]{B}{A}$

```

$$P_B(A) = P(A \mid B) = \frac{P(A \cap B)}{P(B)} = \frac{P(A \cap B)}{P(B)}$$

3. Évènement contraire

\backslash nevent vient de not event qui est une pseudo-traduction de « *évènement contraire* » en anglais.

```
 $\backslash$ nevent{A}$
```

$$\overline{A}$$

4. Espérance, variance et écart-type**Exemple 1 – Espérance**

\backslash expval vient de expected val-ue soit « *espérance* » en anglais.

```
 $\backslash$ expval{X}$
```

$$E(X)$$

Exemple 2 – Choisir le nom de l'espérance

```
 $\backslash$ expval[E_1]{X}$
```

$$E_1(X)$$

Exemple 3 – Variance

```

 $\backslash$ var {X}$ ou
 $\backslash$ var[v]{X}$

```

$$V(X) \text{ ou } v(X)$$

Exemple 4 – Écart-type

\backslash stddev vient de standard deviation soit « *écart-type* » en anglais.

```

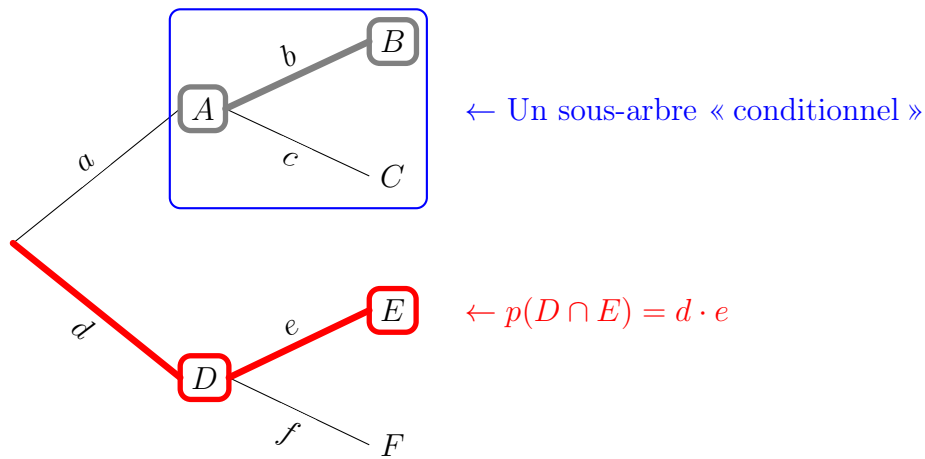
 $\backslash$ stddev {X}$ ou
 $\backslash$ stddev[s]{X}$

```

$$\sigma(X) \text{ ou } s(X)$$

III. Arbres pondérés**1. Au commencement était la forêt...**

Le gros du travail est fait par le package `forest` qui s'appuie TikZ dont on peut utiliser toute la machinerie afin d'obtenir des choses sympathiques comme ci-dessous et ceci à moindre coût neuronal comme vont le montrer les explications données dans les sections suivantes.



Le rendu précédent a été obtenu via le code suivant.

```
\begin{probatree}
  [{}, name = nU
    [$A$, apweight = $a$,
      name      = nA,
      pframe    = blue
    [$B$, name      = nB,
      apweight = $b$]
    [$C$, bpweight = $c$]
    ]
    [$D$, bpweight = $d$,
      name      = nD
    [$E$, apweight = $e$,
      name      = nE]
    [$F$, bpweight = $f$]
    ]
  ]
  %
  \ptreeFocus[gray]{nA | nB}
  %
  \ptreeComment[blue]{nA}%
                        {\$ \leftarrow$ Un sous-arbre \og conditionnel \fg}

  %
  \ptreeFocus*[red]{nU | nD | nE}
  \ptreeComment[red]{nE}%
                        {\$ \leftarrow \proba{D \cap E} = d \cdot e$}
\end{probatree}
```

Remarque. Jusqu'à la section 6. page 85, nous nommerons à la main les noeuds des arbres via `name = ...` lorsque cela sera nécessaire. Dans la section indiquée nous verrons comment utiliser les noms automatiques donnés par le package `forest`.

2. Les bases

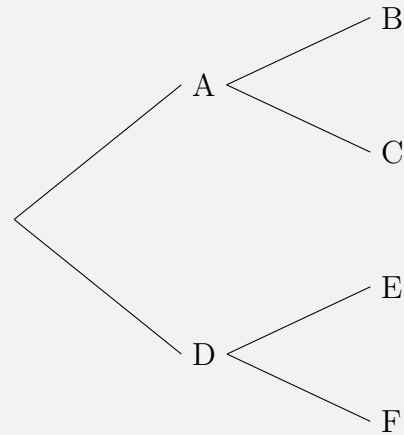
Exemple 1 – Le cas type

Commençons par un arbre nu pour voir comment utiliser l'environnement `probatree` qui s'appuie en coulisse sur celui nommé `forest` du package éponyme. L'exemple qui suit utilise juste les réglages spécifiques de mise en forme de l'arbre qui sont propres à `probatree`.

```

\begin{probatree}
  [ % Noeud racine sans texte
    [A % Sous-noeud nommé
      [B] % Sous-sous-noeud nommé
      [C] % Sous-sous-noeud nommé
    ]
    [D % Sous-noeud nommé
      [E] % Sous-sous-noeud nommé
      [F] % Sous-sous-noeud nommé
    ]
  ]
\end{probatree}

```



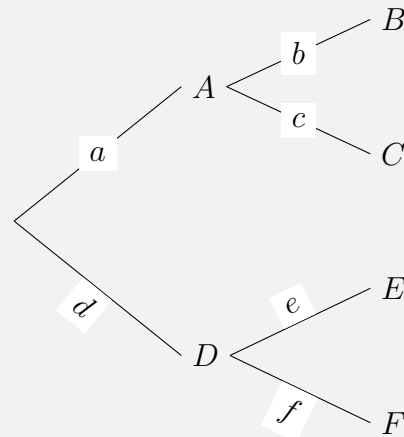
Exemple 2 – Ajouter des pondérations

Dans le code suivant, ce sont les clés² `pweight`, `apweight` et `bpweight` qui facilitent l'écriture des pondérations sur les branches. Indiquons que `pweight` vient de `p`-robability et `weight` soit « *probabilité* » et « *poids* » en anglais. Quant au `a` et au `b` au début de `apweight` et `bpweight` respectivement, ils viennent de `a`-bove et `b`-elow soit « *dessus* » et « *dessous* » en anglais.

```

\begin{probatree}
  [
    [A$, pweight = $a$
      [B$, pweight = $b$]
      [C$, pweight = $c$]
    ]
    [D$, bpweight = $d$
      [E$, apweight = $e$]
      [F$, bpweight = $f$]
    ]
  ]
\end{probatree}

```



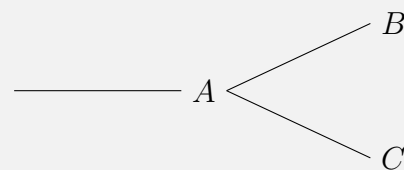
Exemple 3 – Des poids cachés partout

On peut cacher tous les poids via l'environnement étoilé `probatree*` sans avoir à les effacer partout dans le code L^AT_EX (ceci peut être utile lors de la rédaction d'exercices).

```

\begin{probatree*}
  [
    [A$, pweight = $a$
      [B$, apweight = $b$]
      [C$, bpweight = $c$]
    ]
  ]
\end{probatree*}

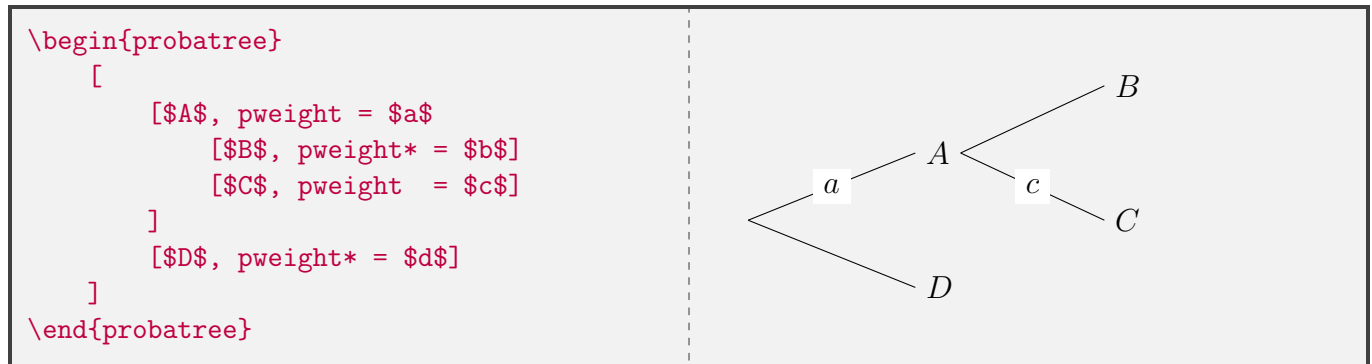
```



2. En fait du point de vue de TikZ, ce sont des styles.

Exemple 4 – Des poids cachés localement

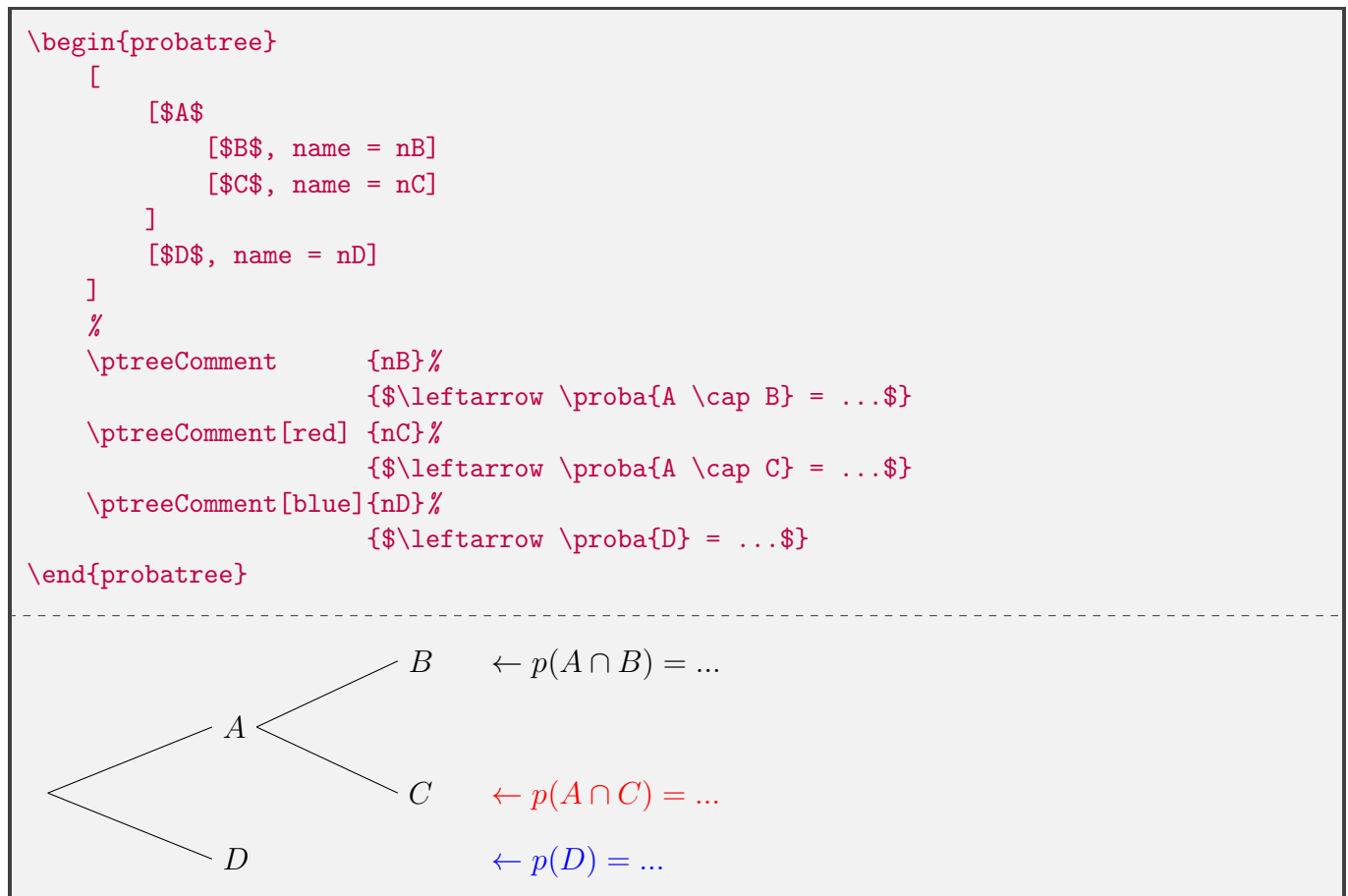
Pour ne cacher que certains poids afin de produire par exemple un arbre à compléter, il faudra utiliser localement le style `pweight*` comme dans l'exemple ci-dessous (*ceci aussi peut servir à rédiger des exercices*).



3. Commenter les racines

Exemple 1 – Tout aligner

Que ce soit pour expliquer un arbre de probabilité, ou bien pour raisonner sur ce dernier, l'effet suivant est très utile³.

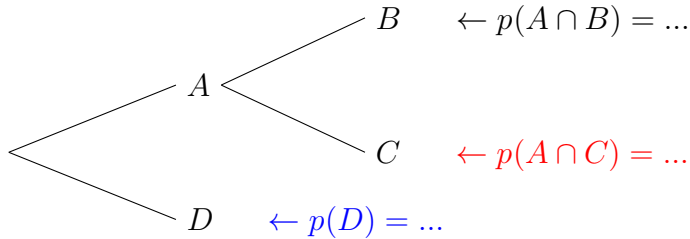


Remarque. Commenter un noeud interne ne provoquera pas d'erreur même si `\ptreeComment` n'a pas été conçu pour ceci. Ceci a été utilisé dans l'exemple d'introduction mais ça reste un petit hack.

3. Le package `forest` permet d'indiquer directement des mises en forme dans le code de l'arbre. L'auteur du présent package trouve bien plus efficace à l'usage de ne pas toucher au code minimal d'un arbre. Ceci explique donc le choix retenu de donner les décorations supplémentaires après le code de l'arbre.

Exemple 2 – Coller au plus près

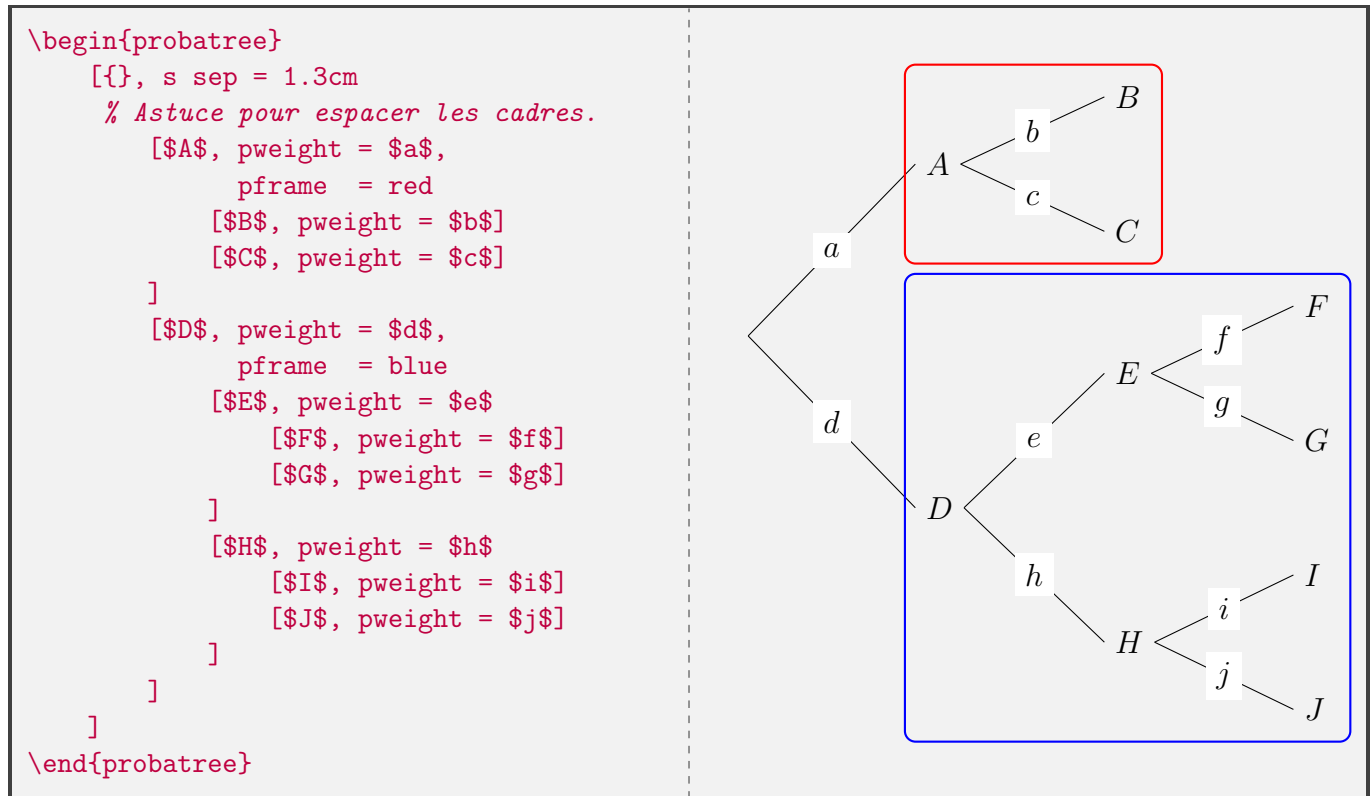
En utilisant `\ptreeComment*` au lieu de `\ptreeComment`, les commentaires seront proches des noeuds et donc non alignés verticalement. Avec l'exemple précédent on obtient la mise en forme qui suit.



4. Avec des cadres

Exemple 1 – Des cadres finaux

Via la clé `pframe` il est très aisé d'encadrer un sous-arbre final⁴ comme le montre l'exemple suivant⁵. Dans l'exemple ci-après nous utilisons la bidouille `\s sep = 1.3cm` qui évite que les cadres se superposent.



Remarque. La clé `pframe` est un cas particulier car tous les autres décorations se font en dehors de la définition de l'arbre

Exemple 2 – Des cadres non finaux

La macro `\ptreeFrame` permet facilement d'encadrer un sous-arbre non final. Ceci nécessite d'utiliser des noms de noeuds. Voici un exemple où la macro `\ptreeFrame` attend les noms de la racine et des deux noeuds finaux le plus haut et le plus bas.

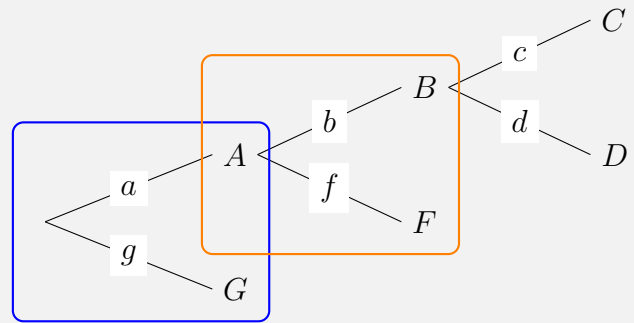
4. Un sous-arbre sera dit final si toutes ses feuilles correspondent à des feuilles de l'arbre initial.

5. Ce type de cadre est très utile d'un point de vue pédagogique.

```

\begin{probatree}
  [{}], name = nU
  % La racine a un texte vide.
  [A$, pweight = $a$,
    name = nA
    [B$, pweight = $b$,
      name = nB
      [C$, pweight = $c$]
      [D$, pweight = $d$]
    ]
  [F$, pweight = $f$,
    name = nF]
  ]
  [G$, pweight = $g$,
    name = nG]
]
%
\ptreeFrame {nU}{nA}{nG}
\ptreeFrame[orange]{nA}{nB}{nF}
\end{probatree}

```



5. Mettre en valeur des chemins

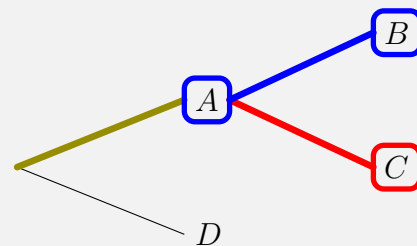
Exemple 1 – Juste avec deux noeuds

Il est relativement aisé de mettre en valeur un chemin particulier comme dans l'exemple ci-après qui est une simple démo. montrant les différences entre `\ptreeFocus`, `\ptreeFocus*` et `\ptreeFocus**`. Notez que les noms des noeuds sont séparés par des barres verticales `|` et qu'il est possible d'utiliser des espaces pour améliorer la lisibilité du code.

```

\begin{probatree}
  [{}], name = nU
  [A$, name = nA
    [B$, name = nB]
    [C$, name = nC]
  ]
  [D$]
]
%
\ptreeFocus* [red] {nA | nC}
\ptreeFocus**[olive]{nU | nA}
\ptreeFocus {nA | nB}
\end{probatree}

```

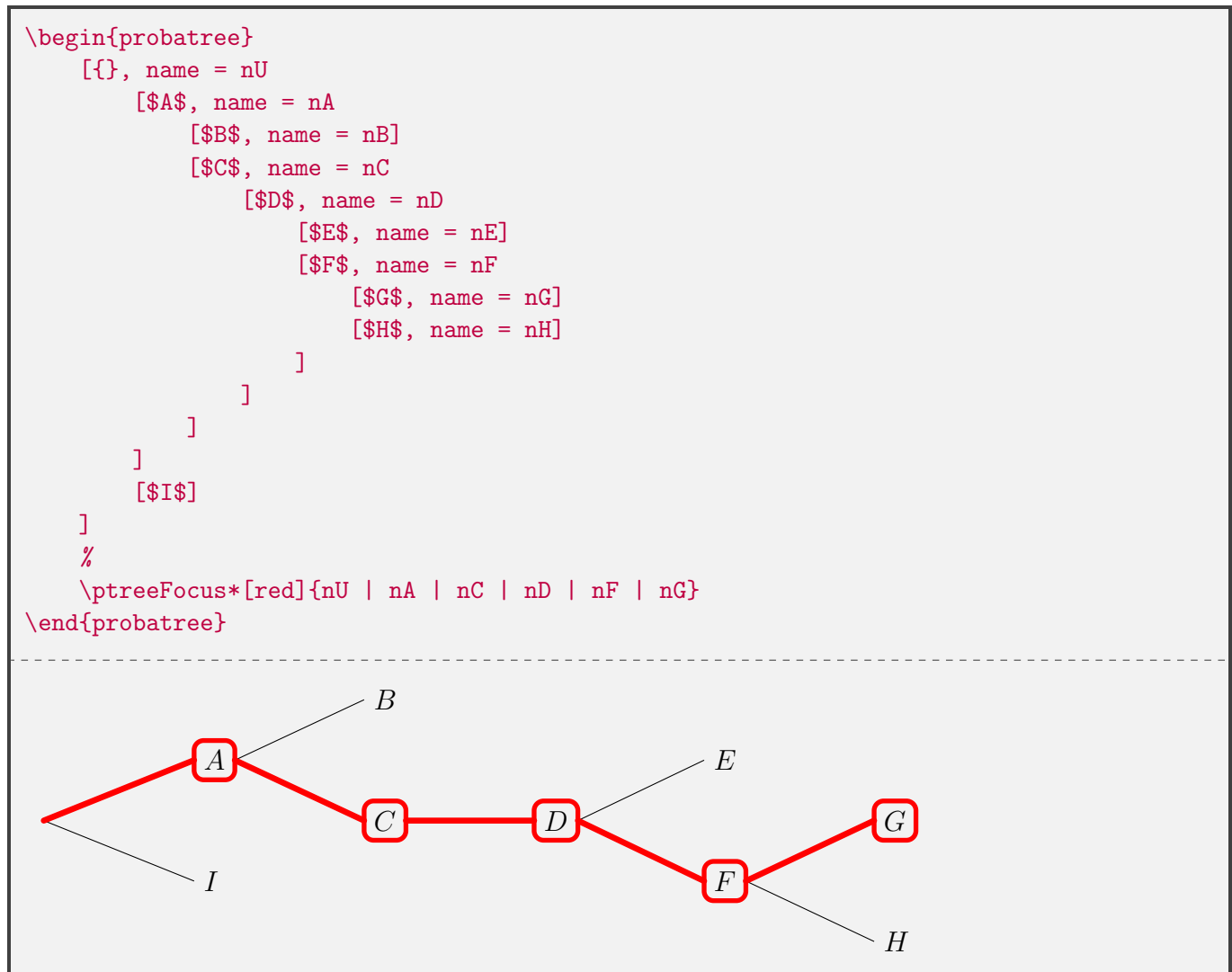


Voici ce qu'il faut retenir.

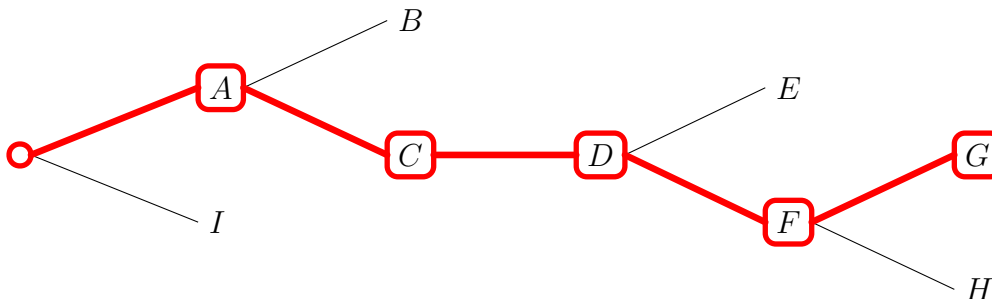
1. `\ptreeFocus` encadre tous les noeuds.
2. `\ptreeFocus*` n'encadre pas le tout premier noeud (*typiquement cela est utile pour un chemin partant de la racine de l'arbre si celle-ci n'est pas nommée comme on le fait très souvent*).
3. `\ptreeFocus**` n'encadre aucun des noeuds.
4. La couleur peut être changée via l'argument optionnel en utilisant les couleurs de type TikZ. Par défaut le bleu est utilisé.

Exemple 2 – Plusieurs noeuds d’un coup

Rien de bien compliqué à condition de bien respecter l’ordre de saisie des noeuds.

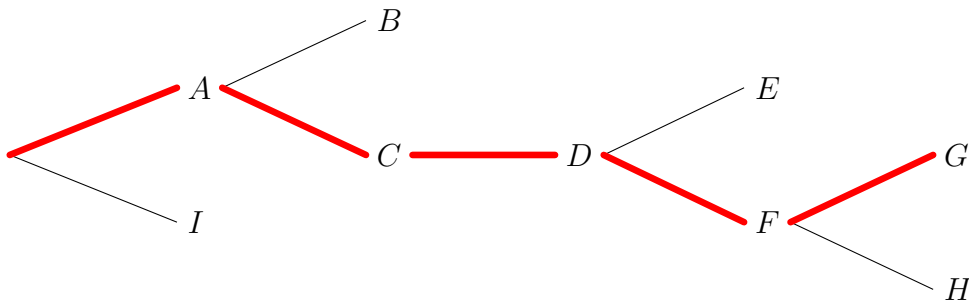


Avec `\ptreeFocus` on obtient l’arbre suivant où le mini disque initial⁶ n’est pas forcément souhaité.



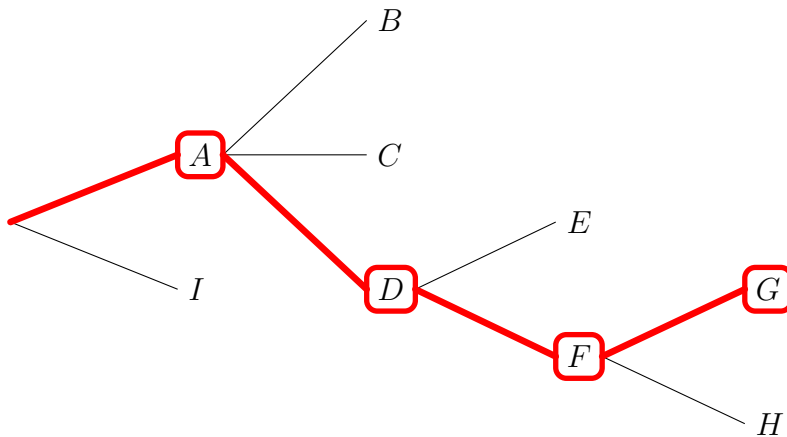
Avec `\ptreeFocus**` on obtient l’arbre ci-dessous.

6. Ce disque est en fait un carré aux coins arrondis autour d’un texte vide. `\ptreeFocus` sera utile si l’univers est indiqué au départ de l’arbre.



6. Utiliser les noms automatiques donnés par forest

Voyons comment obtenir le résultat suivant en indiquant tous les noeuds via les noms automatiques fabriqués par `forest`.



Le rendu précédent a été obtenu via le code suivant.

```
\begin{probatree}
  [{}
    [$A$
      [$B$]
      [$C$]
      [$D$
        [$E$]
        [$F$
          [$G$]
          [$H$]
        ]
      ]
    ]
  ]
  [$I$]
]
%
```

```
\ptreeFocus*[red]{! | !1 | !13 | !132 | !1321}
\end{probatree}
```

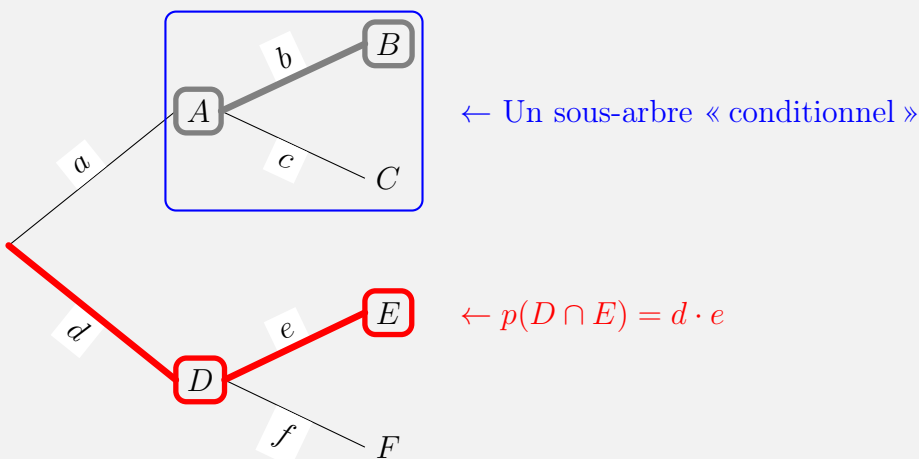
On comprend alors la logique utilisée.

1. Chaque nom automatique commence par `!`.
2. La racine est nommée `!`.
3. Pour voir ce qu'il faut faire pour un noeud autre que la racine, considérons par exemple `!1321`.
On indique en fait le chemin à suivre en partant de la racine `!` pour arriver au noeud voulu.

- Aller d'abord au $\boxed{1}^{\text{er}}$ noeud du niveau 1 qui ici est A .
- Aller ensuite au $\boxed{3}^{\text{e}}$ noeud du niveau 2 qui ici est D .
- Aller après au $\boxed{2}^{\text{e}}$ noeud du niveau 3 qui ici est F .
- Aller enfin au $\boxed{1}^{\text{er}}$ noeud du niveau 4 qui ici est G . C'est notre noeud nommé $\boxed{!1321}$.

Remarque. Utilisez de préférence les noms automatiques car cela facilitera la maintenance de vos arbres sur le long terme. Si on reprend le tout premier exemple d'arbre décoré, il est bien plus simple de faire comme suit car on ne touche pas à la structure minimale du code de l'arbre.

```
\begin{probatree}
[
  [$A$, apweight = $a$,
    pframe = blue
  [$B$, apweight = $b$]
  [$C$, bpweight = $c$]
]
  [$D$, bpweight = $d$
    [$E$, apweight = $e$]
    [$F$, bpweight = $f$]
  ]
]
%
\ptreeFocus[gray]{!1 | !11}
%
\ptreeComment[blue]{!1}%
      {$\leftarrow$ Un sous-arbre \og conditionnel \fg}
%
\ptreeFocus*[red]{! | !2 | !21}
\ptreeComment[red]{!21}%
      {$\leftarrow$ \proba{D \cap E} = d \cdot e$}
\end{probatree}
```



Chapitre H.

Arithmétique

I. Ensembles classiques

Se reporter à la section 5. page 16 où sont présentées les macros `\NN`, `\ZZ`, `\QQ` ainsi que `\PP` pour indiquer l'ensemble des naturels, celui des entiers relatifs, celui des fractions rationnelles et enfin celui des nombres premiers.

II. Opérateurs de base

Pour des raisons d'expressivité des codes L^AT_EX, les opérateurs binaires `\divides`, `\ndivides` et `\modulo` ont été ajoutés comme alias respectifs de `\mid`, `\nmid` et `\bmod` qui sont proposés par le package `amssymb`. Un opérateur `\nequiv` a été aussi ajouté.

`$10 \divides 150$` au lieu de
`$10 \mid 150$`

`$10 \ndivides 154$` au lieu de
`$10 \not\mid 154$`

`$a \nequiv b \modulo p`
`\iff`
`p \ndivides (a - b)$`.

$10 \mid 150$ au lieu de $10|150$
 $10 \nmid 154$ au lieu de $10 \not|154$
 $a \neq b \bmod p \iff p \nmid (a - b).$

III. Fonctions nommées spéciales

Deux fonctions nommées `\pgcd` et `\ppcm` utiles au francophone ont été ajoutées ainsi que la fonction `\lcm` pour les anglophones car cette dernière n'est pas disponible par défaut.

`$\pgcd x = \gcd x$` et `$\ppcm x = \lcm x$`

$\text{pgcd } x = \gcd x$ et $\text{ppcm } x = \text{lcm } x$

IV. Fractions continuées

1. Fractions continuées standard

Dans l'exemple suivant, la notation en ligne semble être due à Alfred Pringsheim. La notation à gauche utilise toujours le maximum d'espace pour améliorer la lisibilité.

```
\contfrac {u_0 | u_1 | u_2 | \dots | u_n}
= \contfrac*{u_0 | u_1 | u_2 | \dots | u_n}$
```

$$u_0 + \frac{1}{u_1 + \frac{1}{u_2 + \frac{1}{\dots + \frac{1}{u_n}}}} = u_0 + \frac{1}{\left| u_1 \right|} + \frac{1}{\left| u_2 \right|} + \frac{1}{\left| \dots \right|} + \frac{1}{\left| u_n \right|}$$

2. Fractions continuées généralisées

Voici comment écrire une fraction continuée généralisée.

```
\displaystyle
\contfracgene {a | b | c | d | e | f | \dots | y | z}
= \contfracgene*{a | b | c | d | e | f | \dots | y | z}$
```

$$a + \frac{b}{c + \frac{d}{e + \frac{f}{\dots + \frac{y}{z}}}} = a + \frac{b}{\left| c \right|} + \frac{d}{\left| e \right|} + \frac{f}{\left| \dots \right|} + \frac{y}{\left| z \right|}$$

3. Comme une fraction continuée isolée

La raison d'être de la macro ci-dessous vient juste de son usage en interne.

```
\singlecontfrac{a}{b}$
pour les fous\dots :-)
```

$$\frac{a}{\left| b \right|} \text{ pour les fous... :-)}$$

4. L'opérateur \mathcal{K}

Exemple 1

La notation suivante est proche de celle qu'utilisait Carl Friedrich Gauss.


```


$$\backslash contfracope_{\{k=1\}^{\{n\}} (b_k:c_k)$$


$$= \cfrac{b_1}{c_1 + \frac{b_2}{c_2 + \frac{b_3}{\dots + \frac{b_n}{c_n}}}}$$


```

$$\mathcal{K}_{k=1}^n(b_k : c_k) = \frac{b_1}{c_1 + \frac{b_2}{c_2 + \frac{b_3}{\dots + \frac{b_n}{c_n}}}}$$

Remarque. La lettre \mathcal{K} vient de "kettenbruch" qui signifie "fraction continuée" en allemand.

Exemple 2

```


$$u_0 + \backslash contfracope_{\{k=1\}^{\{n\}} (1:u_k)$$


$$= \contfrac{u_0 | u_1 | u_2 | \dots | u_n}$$


```

$$u_0 + \mathcal{K}_{k=1}^n(1 : u_k) = u_0 + \frac{1}{u_1 + \frac{1}{u_2 + \frac{1}{\dots + \frac{1}{u_n}}}}$$

Chapitre I.

Algèbre linéaire

I. Matrices via nicematrix

Le gros du boulot est fait par l'excellent package `nicematrix`¹. `tnslinalg` propose en plus une macro à but pédagogique : voir la section 2. page 94. Veuillez vous reporter à la documentation de `nicematrix` pour savoir comment s'y prendre en général.

1. Quelques exemples pour bien démarrer

Exemple 1 – Vu dans la documentation de `nicematrix`

```
$\begin{pmatrix}  
1 & \cdots & \cdots & 1 & \\  
0 & \ddots & & & \vdots \\  
\vdots & \ddots & \ddots & & \vdots \\  
0 & \cdots & 0 & & 1 \\  
\end{pmatrix}
```

$$\begin{pmatrix} 1 & \cdots & \cdots & 1 & \\ 0 & \ddots & & & \vdots \\ \vdots & \ddots & \ddots & & \vdots \\ 0 & \cdots & 0 & & 1 \end{pmatrix}$$

Exemple 2

```
$\begin{vmatrix}  
1 & \cdots & \cdots & 1 & \\  
0 & \ddots & & & \vdots \\  
\vdots & \ddots & \ddots & & \vdots \\  
0 & \cdots & 0 & & 1 \\  
\end{vmatrix}
```

$$\begin{vmatrix} 1 & \cdots & \cdots & 1 & \\ 0 & \ddots & & & \vdots \\ \vdots & \ddots & \ddots & & \vdots \\ 0 & \cdots & 0 & & 1 \end{vmatrix}$$

1. On impose l'option `transparent`.

Exemple 3

```

 $\begin{bmatrix} 1 & \cdots & \cdots & 1 \\ 0 & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 \end{bmatrix}$ 

```

$$\begin{bmatrix} 1 & \cdots & \cdots & 1 \\ 0 & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 \end{bmatrix}$$

Exemple 4 – Vu dans la documentation de nicematrix

```

 $\begin{pNiceMatrix}[name = mymatrix] \\ 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\ \end{pNiceMatrix}$ 
 $\tikz[remember picture, overlay] \\ \draw[red] \\ (mymatrix-2-2) circle (2.5mm);$ 

```

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

Exemple 5 – Vu dans la documentation de nicematrix

```

 $\left( \begin{NiceArray}{cccc:c} \\ 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 10 \\ 11 & 12 & 13 & 14 & 15 \\ \end{NiceArray} \right)$ 

```

$$\left(\begin{array}{cccc:c} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 10 \\ 11 & 12 & 13 & 14 & 15 \end{array} \right)$$

Exemple 6 – Proposition de l’auteur de nicematrix suite à une discussion par mail

```
% Besoin du package ‘‘ifthen’’.
\newcommand\aij{%
  a_{\arabic{iRow}\arabic{jCol}}%
}

$\begin{bNiceArray}{*{5}{>{%
  \ifthenelse{\value{iRow}>0}{\aij}{}%
}c}}[
  first-col,
  first-row,
  code-for-first-row
    = \mathbf{\arabic{jCol}},
  code-for-first-col
    = \mathbf{\arabic{iRow}}
]

& & & & \backslash\backslash
& & & & \backslash\backslash
& & & &
\end{bNiceArray}$
```

$$\begin{matrix} & \mathbf{1} & \mathbf{2} & \mathbf{3} & \mathbf{4} & \mathbf{5} \\ \mathbf{1} & a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\ \mathbf{2} & a_{21} & a_{22} & a_{23} & a_{24} & a_{25} \end{matrix}$$

Exemple 7 – Avec des calculs automatiques

```
\newcounter{cntaij}
\newcommand\aij{%
  \setcounter{cntaij}{\value{iRow}}%
  \addtocounter{cntaij}{\value{jCol}}%
  \addtocounter{cntaij}{-1}%
  \arabic{cntaij}%
}

Si $a_{ij} = i + j - 1$ alors

$(a_{ij})_{1 \leq i \leq 3, 1 \leq j \leq 5}$

=

\begin{bNiceArray}{*{5}{>\aij}c}}
  & & & & \backslash\backslash
  & & & & \backslash\backslash
  & & & &
\end{bNiceArray}$
```

Si $a_{ij} = i + j - 1$ alors

$$(a_{ij})_{1 \leq i \leq 3, 1 \leq j \leq 5} = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 3 & 4 & 5 & 6 \\ 3 & 4 & 5 & 6 & 7 \end{bmatrix}$$

2. Calculs expliqués des déterminants 2×2

Exemple

$\begin{aligned} & \$\backslash\text{calcdettwo}^* \quad \begin{matrix} \{a\}\{c\}\% \\ \{b\}\{d\} \end{matrix} \\ & = \backslash\text{calcdettwo} \quad \begin{matrix} \{a\}\{c\}\% \\ \{b\}\{d\} \end{matrix} \\ & = \backslash\text{calcdettwo}[\text{exp}] \begin{matrix} \{a\}\{c\}\% \\ \{b\}\{d\}\$ \end{matrix} \end{aligned}$	$\begin{vmatrix} a & c \\ b & d \end{vmatrix} = \begin{vmatrix} a & c \\ b & \ominus d \end{vmatrix} = a d - b c$
--	---

Remarque. Il existe deux autres types de développement.

1. $a \cdot d - b \cdot c$ s'obtient via l'option **cexp**.
2. $a \times d - b \times c$ s'obtient via l'option **texp**

exp est pour **exp**-and soit « *développer* » en anglais, **c** pour $\backslash\text{cdot}$ et enfin **t** pour $\backslash\text{times}$.

Chapitre J.

Polynômes et séries formelles

I. Ensembles classiques de nombres

Se reporter aux sections 4. page 16 et 5. page 16 où sont présentées diverses macros pour indiquer des ensembles classiques en algèbre.

II. Polynômes

Exemple 1 – Polynômes

`\setpoly{R}{X}` ou
`\setpoly{R}{X | Y | Z}`

$R[X]$ ou $R[X;Y;Z]$

Exemple 2 – Fractions polynômiales

`\setpolyfrac{Q}{T}` ou
`\setpolyfrac{Q}{%`
`{S_1 | S_2 | \dots | S_k}`

$Q(T)$ ou $Q(S_1;S_2;\dots;S_k)$

III. Séries formelles classiques

Exemple 1 – Séries formelles

`\setserie{C}{X}` ou
`\setserie{C}{T | O | P}`

$C[[X]]$ ou $C[[T;O;P]]$

Exemple 2 – Corps des fractions de séries formelles

`\setseriefrac{Z}{X}` ou
`\setseriefrac{Z}{Z | T | O | P}`

$Z((X))$ ou $Z((Z;T;O;P))$

IV. Polynômes et séries formelles de Laurent

Exemple 1 – Polynômes de Laurent

Ci-dessous, la notation $R\{X_1; X_2\}$ n'est pas standard.

```
$\setpolylaurent{R}{X} =  
\setpoly{R}{X | X^{-1}}$
```

```
$\setpolylaurent{R}{X_1 | X_2} =  
\setpoly{R}{X_1 | X_1^{-1} %  
    | X_2 | X_2^{-1}}$
```

$$R\{X\} = R[X; X^{-1}]$$

$$R\{X_1; X_2\} = R[X_1; X_1^{-1}; X_2; X_2^{-1}]$$

Exemple 2 – Séries formelles de Laurent

Ci-dessous, la notation $Q\{\{X_1; X_2\}\}$ n'est pas standard.

```
$\setserielaurent{Q}{X} =  
\setserie{Q}{X | X^{-1}}$
```

```
$\setserielaurent{Q}{X_1 | X_2} =  
\setserie{Q}{X_1 | X_1^{-1} %  
    | X_2 | X_2^{-1}}$
```

$$Q\{\{X\}\} = Q[[X; X^{-1}]]$$

$$Q\{\{X_1; X_2\}\} = Q[[X_1; X_1^{-1}; X_2; X_2^{-1}]]$$

Chapitre K.

Historique

Nous ne donnons ici qu'un très bref historique récent ¹ de `tnsmath` ² à destination de l'utilisateur principalement. Tous les changements sont disponibles uniquement en anglais dans le dossier `change-log` : voir le code source de `tnsmath` sur `github`.

2020-08-06 Nouvelle version mineure `1.7.0-beta`.

ANALYSE [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.

ANALYSE [0.7.0-BETA]

- **DÉFINITION EXPLICITE D'UNE FONCTION :** les macros `\funcdef` et `\txtfuncdef` ont été déplacées dans `tnssets` qui est disponible sur <https://github.com/typensee-latex/tnssets.git>.

GÉOMÉTRIE [0.2.0-BETA]

- **CRITÈRE DE COLINÉARITÉ :** ajout de la macro `\colicriteria`.
- **PRODUIT VECTORIEL :** changement de l'API.
 - `\vcalccrossprod*` devient `\vcalccrossprod**`.
 - `\vcalccrossprod*` dessine des produits en croix à la place des boucles.

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

2. Ce package remplace l'ancien `lymath`.

PROBABILITÉ [0.4.0-BETA]

- **ARBRE** : possibilité de mettre en valeur un chemin via `\ptreeFocus`, `\ptreeFocus*` ou `\ptreeFocus**`.

PROBABILITÉ [0.5.0-BETA]

- **ARBRE DE PROBABILITÉS.**
 - `\ptreeFocus`, `\ptreeFocus*` et `\ptreeFocus**` fonctionnent avec un multi-argument pour pouvoir indiquer un chemin sur plusieurs noeuds.
 - Suppression de la clé `\pcomment`.
 - Ajout des macros `\ptreeComment` et `\ptreeComment*` qui simplifient la saisie.
-

THÉORIE GÉNÉRALE DES ENSEMBLES [0.2.0-BETA]

- **COMPOSITION D'APPLICATIONS.**
 - `\compo` est un opérateur de composition de deux applications.
 - `\multicomp` permet d'indiquer des compositions successives d'une application par elle-même.

THÉORIE GÉNÉRALE DES ENSEMBLES [0.3.0-BETA]

- **DÉFINITION EXPLICITE D'UNE FONCTION** : intégration de deux macros proposées avant par `tnsana` disponible sur <https://github.com/typensee-latex/tnsana.git>.
 - `\funcdef` peut produire trois versions symboliques.
 - `\txtfuncdef` produit une version textuelle courte.
 - **FONCTIONS SPÉCIALES.**
 - Ajout de `\id` pour la fonction identité.
 - Ajout de `\caract` et `\caractone` pour deux versions de la fonction caractéristique d'un ensemble.
-

2020-07-25 Nouvelle version mineure 1.6.0-beta.

PROBABILITÉ [0.3.0-BETA]

- **ARBRE.**
 - Ajout du style `pcomment` pour placer du texte à la droite d'une feuille.
 - Le style `frame` a été renommé `pframe`.
-

2020-07-23 Nouvelle version mineure 1.5.0-beta.

PROBABILITÉ [0.2.0-BETA]

- **ARBRE** : ajout de la macro `\ptreeFrame` pour tracer facilement des sous cadres non « finaux ».

2020-07-22 Nouvelle version mineure 1.4.0-beta.

ANALYSE [0.5.0-BETA]

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

PROBABILITÉ [0.1.0-BETA]

- **PROBABILITÉ CONDITIONNELLE** : `\probacondexp` renommée en `\eprobacond`.
- **ÉVÈNEMENT CONTRAIRE** : ajout de `\nevent`.
- **VARIANCE ET ÉCART-TYPE** : ajout de `\var` et `\stddev`.

2020-07-21 Nouvelle version mineure 1.3.0-beta : passage de `lymath` à `tnsmath`.

ANALYSE [0.0.0-BETA]

- Simple migration depuis l'ancien code de `lymath`.

ANALYSE [0.1.0-BETA]

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

ANALYSE [0.2.0-BETA]

- **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.

ANALYSE [0.3.0-BETA]

- **DÉRIVATION.**
 - Dérivation pointée à la physicienne via `d` et `bd` deux nouvelles options de `\der`.
 - 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*).

ANALYSE [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`.

ARITHMÉTIQUE [0.0.0-BETA]

- Simple migration depuis l'ancien code de `lymath`.

ARITHMÉTIQUE [0.1.0-BETA]

- **FONCTIONS NOMMÉES** : ajout de `\pgcd`, `\ppcm` et `\lcm`.

GÉOMÉTRIE [0.0.0-BETA]

- Simple migration depuis l'ancien code de `lymath`.

GÉOMÉTRIE [0.1.0-BETA]

- **PRODUIT SCALAIRE** : trois nouvelles options pour `\dotprod` et `\vdotprod`.
 - `p` et `sp` donnent une écriture parenthésée.
 - `b` utilise une puce au lieu d'un point.
- **PRODUIT VECTORIEL** : un nouvel argument optionnel pour `\crossprod` et `\vcrossprod` afin d'obtenir aussi une mise en forme avec le symbole \times .

ALGÈBRE LINÉAIRE [0.0.0-BETA]

- Simple migration depuis l'ancien code de `lymath`.

MÉTHODE FORMELLE EN LOGIQUE [0.0.0-BETA]

- Simple migration depuis l'ancien code de `lymath`.

POLYNÔMES ET SÉRIES FORMELLES [0.0.0-BETA]

- Simple migration depuis l'ancien code de `lymath`.

PROBABILITÉ [0.0.0-BETA]

- Simple migration depuis l'ancien code de `lymath`.

SUITES [0.0.0-BETA]

- Simple migration depuis l'ancien code de `lymath`.
-

THÉORIE GÉNÉRALE DES ENSEMBLES [0.0.0-BETA]

- Simple migration depuis l'ancien code de `lymath`.
-

2020-07-05 Nouvelle version mineure 1.2.0-beta.

- **TABLEAUX DE SIGNE ET DE VARIATION** : il est maintenant possible d'ajouter des graphiques expliquant le signe d'une fonction affine ou d'une fonction trinômiale du 2^e degré.

2020-06-27 Nouvelle version mineure 1.1.0-beta.

- **LOGIQUE.**
 - Suppression des environnements `aexplain` et `aexplain*`.
Leurs mises en forme restent accessibles respectivement via les options `style = ar` et `style = sar` de l'environnement `explain`.
 - L'environnement `explain` propose des options de type clé-valeur.
 - Ajout de petits commentaires pour les étapes via `\comthis` et si besoin `\comthis*`.
 - Modification de `\explnext*` pour le mode universitaire sans flèche via l'ajout de `\exptxtupdown` qui gère la mise en forme de deux explications non vides. Ceci a pour effet l'alignement du rendu avec l'opérateur.
 - Les environnements `demoexplain` et `demoexplain*` utilisent `longtable` en coulisse afin de pouvoir écrire un tableau sur plusieurs pages.

2020-06-21 Nouvelle version majeure 1.0.0-beta.

- Le changement vers une nouvelle version majeure se justifie par de nouvelles règles strictes pour définir les signatures des macros. À l'avenir ces règles seront appliquées tout le temps sauf dans de très rares cas.

Ceci a créé beaucoup de nouvelles façons de rédiger.

- **ALGÈBRE LINÉAIRE** : `\calcdettwo` est un outil pédagogique pour expliquer le calcul d'un déterminant 2×2 .
-

- **ANALYSE.**
 - Calcul intégral.
→ `\integrate` et `\dintegrate` servent à rédiger des intégrales simples.

- `\hook` s'utilise différemment : on doit taper `\hook{a}{b}{F(x)}{x}` avec l'obligation de donner la variable.
- `\hook` marche avec des options. Du coup `\vhook` and `\vhook*` ont été supprimées mais les mises en forme correspondantes existent toujours via `\hook[r]` et `\hook[sr]`.
- Dérivées totales.
 - Il ne reste plus que trois macros : `\sder`, `\der` et `\derope`.
 - `\sder` et `\sder[e]` remplacent `\derpow*` et `\derpow` avec en plus la possibilité d'ajout automatique de parenthèses pour faire comme avec `\derpar*`, `\derpar`, `\sderpar*` et `\sderpar` avant.
 - `\der` s'utilise en indiquant la variable de dérivation. Cette macro propose différentes options pour différentes mises en forme (*on peut toujours obtenir la même chose que ce que proposaient `\derfrac`, `\derfrac*` et `\dersub`*).
 - `\derope` sert à écrire un opérateur fonctionnel.
- Dérivées partielles.
 - Il ne reste plus que deux macros : `\pder` et `\pderope`.
 - `\pder` possède des options permettant d'obtenir le même résultat qu'avec les anciennes macros `\partialfrac` et `\partialsub`.
 - La mise en forme proposée par `\partialprime` n'a pas été gardée.
 - `\pderope` sert à écrire un opérateur fonctionnel.

• ARITHMÉTIQUE.

- `\notdivides` a été renommée `\ndivides`.
 - Ajout de `\nequiv`.
-

• GÉOMÉTRIE.

- `\calcdetplane`, `\calcdetplane*`, `\vcalcdetplane` et `\vcalcdetplane*` permettent de détailler le calcul du déterminant de deux vecteurs en dimension 2 (*utile pour le critère de colinéarité*).
- `\calccrossprod`, `\calccrossprod*`, `\vcalccrossprod` et `\vcalccrossprod*` permettent de détailler le calcul du produit vectoriel de deux vecteurs en dimension 3.
- `\coordcrossprod` permet d'obtenir formellement les coordonnées d'un produit vectoriel avec des formats du type $(yz' - zy', zx' - xz', xy' - yx')$.
- Angles orientés.
 - `\angleorient` devient l'unique macro pour rédiger des angles orientés de différentes façons via des options.
 - `\angleorient*` a été remplacée par `\angleorient[sp]`.
 - `\hangleorient` a été remplacée par `\dotprod[h]`.
 - `\hangleorient*` a été remplacée par `\dotprod[sh]`.

- Produit scalaire.
 - `\dotprod` devient l'unique macro pour rédiger des produits scalaires de différentes façons grâce à des options.
 - `\adotprod` a été remplacée par `\dotprod[a]`.
 - `\adotprod*` a été remplacée par `\dotprod[sa]`.
- Coordonnées.
 - Il faut passer via l'une des macros : `\coord`, `\pcoord`, `\pcoord*`, `\vcoord` et `\vcoord*`. Toutes ces macros proposent des options pour choisir la mise en forme : des parenthèse en mode horizontal, des crochets en mode vertical...
 - `\coord` est pour des coordonnées seules.
 - `\pcoord` et `\pcoord*` sont pour un point avec ses coordonnées.
 - `\vcoord` et `\vcoord*` sont pour un vecteur avec ses coordonnées.
 - La version étoilée `\coord*` a été supprimée.
- Norme.
 - `\norm` fonctionne maintenant avec des options. Du coup `\norm*` a été supprimée mais la mise en forme correspondante existe toujours via `\norm[s]`.
 - `\vnorm` évite d'avoir à utiliser `\vect` pour des vecteurs juste nommés.

• LOGIQUE.

- La macro `\explain` a été supprimée pour être remplacée par l'environnement `explain` qui est redoutable d'efficacité pour détailler un calcul ou un raisonnement simple.
- `explain` est complété par les deux environnements `aexplain` et `aexplain*` qui utilisent des flèches pour les indications.
- Les environnements `demoexplain` et `demoexplain*` permettent de rédiger de « vraies » démonstrations via des tableaux efficaces.
- Toutes les macros négatives avec pour préfixe `not` auparavant utilisent maintenant juste le préfixe `n`.
- Tous les opérateurs de comparaison ont une version négative.

• PROBABILITÉS.

- `\probacond**` et `\dprobacond**` sont devenues `\probacondexp*` et `\probacondexp` respectivement.
- `\expval` est une nouvelle macro pour l'écriture symbolique de l'espérance d'une variable aléatoire.

2020-06-08 Nouvelle version mineure 0.7.0-beta.

• ANALYSE.

- Pour éviter des conflits avec d'autres packages les renommages suivants ont dû être faits.
 - `\ch` et `\ach` sont devenus `\fch` et `\afch` où `f` est pour `f`-rench.

- `\sh` et `\ash` sont devenus `\fsh` et `\afsh`.
- `\th` et `\ath` sont devenus `\fth` et `\afth`.
- Les macros `\acosh`, `\asinh` et `\atanh` ont été ajoutées.
- `\derpar` et `\derpar*` servent à rédiger des dérivées avec des parenthèses extensibles. En coulisse, `\derpow` et `\derpow*` sont appelées. Pour utiliser des parenthèses non extensibles, on passera par `\sderpar` et `\sderpar*` où `s` est pour `s-mall`.
- Ajout de `\stdint` pour rendre public l'opérateur intégral proposé par défaut par L^AT_EX.
- **GÉOMÉTRIE.**
 - La macro `\pts` a été supprimée car sans signification sémantique puisqu'un point peut être nommé avec deux lettres.
 - Les macros `\gline` et `\pgline` servent à indiquer des droites définies par deux points.
 - La macro `\hgline`, avec `h` pour `h-alf`, est pour les demi-droites définies par deux points.
 - La macro `\segment` est utile pour les segments définis par deux points.
- **LOGIQUE.**
 - Pour les inégalités, on peut maintenant utiliser le décorateur `plot`.
 - Ajout des versions négatives des opérateurs logiques verticaux.
- **PROBABILITÉS.**
 - Ajout de `\proba` pour écrire des probabilités.
 - Le comportement de `\probacond` a été modifié pour le rendre plus logique.

2019-10-21 Nouvelle version sous-mineure 0.6.3-beta.

- **ANALYSE.**
 - `\hypergeo` est devenu `\seqhypergeo`.
 - `\suprageo` est devenu `\seqsuprageo`.
- **ENSEMBLES :** `\CSinterval` a été déplacée dans le package `lyalgo` disponible à l'adresse <https://github.com/bc-latex/ly-algo>.
- **GÉOMÉTRIE :** `\notparallel` est devenu `\nparallel`.
- **LOGIQUE.**
 - `\eqdef**` a été supprimé. Voir la macro `\Store*` du package `lyalgo`.
 - Différentes versions de l'opérateur \exists via `\existssone` et `\existmulti` avec leurs versions négatives `\nexistsone` et `\nexistmulti`.
 - Deux nouvelles macros `\eqplot` et `\eqappli` pour indiquer une équation de courbe et l'application d'une identité à des variables. Ceci s'accompagne de l'ajout des macros `\textopplot` et `\textopappli`.
 - Ajout des formes négatives `\niff`, `\nimplies` et `\nliesimp`.
 - Les décorations `cons`, `appli` et `choice` sont utilisables avec les opérateurs `\iff`, `\implies` et `\liesimp` et leurs formes négatives.

- Une macro `\textoptest` a été ajoutée afin de rendre personnalisable tous les textes décorant les symboles.

2019-10-14 Nouvelle version sous-mineure 0.6.2-beta.

- **ALGÈBRE.**

- `\polyset` est devenu `\setpoly`.
- `\polyfracset` est devenu `\setpolyfrac`.
- `\serieset` est devenu `\setserie`.
- `\seriefracset` est devenu `\setseriefrac`.
- `\polylaurentset` est devenu `\setpolylaurent`.
- `\serielaurentset` est devenu `\setserielaurent`.

2019-10-13 Nouvelle version sous-mineure 0.6.1-beta.

- **ENSEMBLES.**

- `\algeset` est devenu `\setalge`.
- `\geoset` est devenu `\setgeo`.
- `\geneset` est devenu `\setgene`.
- `\proba` est devenu `\setproba`.
- `\specialset` est devenu `\setspecial`.

- **LOGIQUE** : la macro `\explain` possède maintenant un argument optionnel pour indiquer l'espacement avant le symbole. Ceci s'accompagne de la suppression des macros obsolètes `\explain*` et `\textexplainspacebefore`.

- **PROBABILITÉ.**

- Les macros `\probacond` et `\probacond*` n'ont plus d'argument optionnel. Pour obtenir l'écriture fractionnaire, il faut utiliser `\probacond**` ou `\dprobacond**`.
- Les environnements `probatree` et `probatree*` ont trois nouvelles clés. La clé `frame` permet d'encadrer un sous-arbre, et les clés `apweight` et `bpweight` permettent d'écrire des poids dessus/dessous une branche.

2019-10-10 Nouvelle version mineure 0.6.0-beta.

- **ENSEMBLES** : pour l'informatique théorique la macro `\CSinterval` permet d'obtenir quelque chose comme $a..b$.
- **GÉOMÉTRIE** : la macro `\notparallel` a été rajoutée.
- **LOGIQUE** : il y a deux nouvelles macros sémantiques `\neqid` et `\eqchoice`.
- **PROBABILITÉS.**
 - Les macros `\probacond` et `\probacond*` servent à écrire des probabilités conditionnelles.
 - Les environnements `probatree` et `probatree*` simplifient la production d'arbres probabilistes pondérés ou non.

2019-09-27 Nouvelle version mineure 0.5.0-beta.

- **ARITHMÉTIQUE** : ajout des opérateurs `\divides`, `\notdivides` et `\modulo`.
- **DIVERS** : ajout des macros `\dsum` et `\dprod` qui sont vis à vis de `\sum` et `\prod` des équivalents de `\dfrac` pour `\frac`.

- **GÉOMÉTRIE.**

- `\pts` permet d'indiquer plusieurs points.
- `\parallel` utilise des obliques pour symboliser le parallélisme au lieu de barres verticales.

- **LOGIQUE.**

- La version doublement étoilée `\eqdef**` donne une deuxième écriture symbolique d'un symbole égal de type définition (*cette notation vient du langage B*).
- Ajout de `\liesimp` comme alias de `\Longleftarrow`.
- Les macros `\vimplies`, `\viff` et `\vliesimp` sont des versions verticales de `\implies`, `\iff` et `\liesimp`.
- Comme pour les égalités, il existe les macros `\impliestest`, `\iffhyp` ... etc.

2019-09-06 Nouvelle version mineure 0.4.0-beta.

- **ALGÈBRE LINÉAIRE** : intégration du package `nicematrix` pour écrire des matrices.
- **ANALYSE** : intégration du package `tkz-tab` pour rédiger des tableaux de variations et de signes.
- **LOGIQUE ET FONDEMENTS** : différents types de signes d'inéquation et de non égalité pour des cas de test, d'hypothèse faite et de condition à vérifier.

Chapitre L.

Toutes les fiches techniques

I. Théorie générale des ensembles

1. Ensembles

i. Ensembles versus accolades

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

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

1. `b` : on utilise des accolades extensibles.
2. `sb` : on utilise des accolades non extensibles.

— Argument: la définition de l'ensemble.

— Argument: la définition de l'ensemble.

2. Ensembles

i. Ensembles pour la géométrie

`\setgeo{#1}`

— Argument: un seul caractère ASCII indiquant un ensemble géométrique.

`\setgeo*{#1..#2}`

— Argument 1: un seul caractère ASCII indiquant \mathcal{U} dans le nom \mathcal{U}_d d'un ensemble géométrique.

— Argument 2: un texte donnant d dans le nom \mathcal{U}_d d'un ensemble géométrique.

3. Ensembles

i. Ensembles probabilistes

`\setproba{#1}`

— Argument: un seul caractère ASCII majuscule indiquant un ensemble probabiliste.

`\setproba*{#1..#2}`

— **Argument 1**: un seul caractère ASCII majuscule indiquant \mathcal{U} dans le nom \mathcal{U}_d d'un ensemble probabiliste.

— **Argument 2**: un texte donnant d dans le nom \mathcal{U}_d d'un ensemble probabiliste.

4. Ensembles

i. Ensembles pour l'algèbre générale

`\setalge{#1}`

— **Argument**: soit l'une des lettres **h** et **k**, soit un seul caractère ASCII majuscule indiquant un ensemble de type anneau ou corps.

`\setalge*{#1..#2}`

— **Argument 1**: un seul caractère ASCII indiquant \mathbb{U} dans le nom \mathbb{U}_d d'un ensemble de type anneau ou corps.

— **Argument 2**: un texte donnant d dans le nom \mathbb{U}_d d'un ensemble de type anneau ou corps.

5. Ensembles

i. Ensembles classiques en mathématiques et en informatique théorique

Ensembles classiques suffixés

`\NN` `\NNs`

`\PP`

`\ZZ` `\ZZn` `\ZZp` `\ZZs` `\ZZsn` `\ZZsp`

`\DD` `\DDn` `\DDp` `\DDs` `\DDsn` `\DDsp`

`\QQ` `\QQn` `\QQp` `\QQs` `\QQsn` `\QQsp`

`\RR` `\RRn` `\RRp` `\RRs` `\RRsn` `\RRsp`

`\CC` `\CCs`

`\HH` `\HHs`

`\OO` `\OOs`

6. Ensembles

i. Ensembles classiques en mathématiques et en informatique théorique

Des suffixes à la carte

`\setspecial {#1..#2}`

`\setspecial*{#1..#2}`

— Argument 1: l'ensemble à "suffixer".

— Argument 2: l'un des suffixes `n`, `p`, `s`, `sn` ou `sp`.

7. Intervalles

i. Intervalles réels - Notation française (?)

Pour toutes les macros ci-dessous, la version non étoilée produit des délimiteurs qui s'étirent si besoin verticalement, tandis que la version étoilée ne le fait pas.

`\intervalC0 {#1..#2}`

`\intervalC0*{#1..#2}`

— Argument 1: borne inférieure a de l'intervalle $[a ; b[$.

— Argument 2: borne supérieure b de l'intervalle $[a ; b[$.

`\intervalC {#1..#2}`

`\intervalC*{#1..#2}`

— Argument 1: borne inférieure a de l'intervalle $[a ; b]$.

— Argument 2: borne supérieure b de l'intervalle $[a ; b]$.

`\interval0 {#1..#2}`

`\interval0*{#1..#2}`

— Argument 1: borne inférieure a de l'intervalle $]a ; b[$.

— Argument 2: borne supérieure b de l'intervalle $]a ; b[$.

`\interval0C {#1..#2}`

`\interval0C*{#1..#2}`

— Argument 1: borne inférieure a de l'intervalle $]a ; b]$.

— Argument 2: borne supérieure b de l'intervalle $]a ; b]$.

8. Intervalles

i. Intervalles réels – Notation américaine

Pour toutes les macros ci-dessous, la version non étoilée produit des délimiteurs qui s'étirent si besoin verticalement, tandis que la version étoilée ne le fait pas.

`\intervalCP {#1..#2}`
`\intervalCP*{#1..#2}`

- Argument 1: borne inférieure a de l'intervalle $[a; b)$.
 - Argument 2: borne supérieure b de l'intervalle $[a; b)$.
-

`\intervalP {#1..#2}`
`\intervalP*{#1..#2}`

- Argument 1: borne inférieure a de l'intervalle $(a; b)$.
 - Argument 2: borne supérieure b de l'intervalle $(a; b)$.
-

`\intervalPC {#1..#2}`
`\intervalPC*{#1..#2}`

- Argument 1: borne inférieure a de l'intervalle $(a; b]$.
 - Argument 2: borne supérieure b de l'intervalle $(a; b]$.
-

9. Intervalles

i. Intervalles discrets d'entiers

Pour toutes les macros ci-dessous, la version non étoilée produit des délimiteurs qui s'étirent si besoin verticalement, tandis que la version étoilée ne le fait pas.

`\ZintervalCO {#1..#2}`
`\ZintervalCO*{#1..#2}`

- Argument 1: borne inférieure a de l'intervalle $\llbracket a; b \llbracket$.
 - Argument 2: borne supérieure b de l'intervalle $\llbracket a; b \llbracket$.
-

`\ZintervalC {#1..#2}`
`\ZintervalC*{#1..#2}`

- Argument 1: borne inférieure a de l'intervalle $\llbracket a; b \rrbracket$.
 - Argument 2: borne supérieure b de l'intervalle $\llbracket a; b \rrbracket$.
-

`\ZintervalO {#1..#2}`
`\ZintervalO*{#1..#2}`

- Argument 1: borne inférieure a de l'intervalle $\llbracket a; b \rrbracket$.
 - Argument 2: borne supérieure b de l'intervalle $\llbracket a; b \rrbracket$.
-

`\ZintervalOC {#1..#2}`
`\ZintervalOC*{#1..#2}`

- Argument 1: borne inférieure a de l'intervalle $\llbracket a; b \rrbracket$.
- Argument 2: borne supérieure b de l'intervalle $\llbracket a; b \rrbracket$.

10. Unions et intersections en mode ligne

`\dcap`
`\dcup`
`\dsqcup`

11. Applications

i. Cardinal, image et compagnie

`\card`
`\card*`

`\dom`
`\codom`
`\im`

12. Applications

i. Application totale, partielle, injective, surjective et/ou bijective

<code>\to</code>	<code>\pto</code>
<code>\onetoone</code>	<code>\ponetoone</code>
<code>\onto</code>	<code>\ponto</code>
<code>\bijet</code>	<code>\pbijet</code>

13. Applications

i. Fonction identité

`\id`

14. Applications

i. Fonction caractéristique

`\caract`
`\caractone`

15. Applications

i. 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`.

16. Applications

i. Composition

`\compo` compo = compo-sition

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

- Option: la valeur par défaut est `r`. Voici les différentes valeurs possibles.

- | | |
|--|-----------------------------|
| 1. <code>r</code> : écriture utilisant des chevrons. | <code>r = r-after.</code> |
| 2. <code>exp</code> : écriture développée. | <code>exp = exp-and.</code> |
| 3. <code>dot</code> : écriture faussement développée utilisant des points de suspension. | |

- Argument 1: l'application.
- Argument 2: le nombre d'applications composées.

II. Méthodes formelles en logique

1. Espace après la négation logique

`\neg`
`\stdneg` (pour retrouver le symbole par défaut)

2. Différents types de comparaisons « standard »

i. Opérateurs décorés – Les textes

<code>\textopappli{}</code>	<code>\textopcons{}</code>	<code>\textopid{}</code>
<code>\textopchoice{}</code>	<code>\textopdef{}</code>	<code>\textopplot{}</code>
<code>\textopcond{}</code>	<code>\textophyp{}</code>	<code>\textoptest{}</code>

ii. Opérateurs de comparaison supplémentaires

<code>\eqdef</code>	<code>\eqid</code>	<code>\eqplot</code>	<code>\eqchoice</code>	<code>\eqcons</code>	<code>\eqtest</code>
<code>\eqdef*</code>	<code>\eqid*</code>	<code>\eqappli</code>	<code>\eqcond</code>	<code>\eqhyp</code>	

<code>\neqid</code>	<code>\neqappli</code>	<code>\neqcond</code>	<code>\neqhyp</code>
<code>\neqplot</code>	<code>\neqchoice</code>	<code>\neqcons</code>	<code>\neqtest</code>

<code>\lessplot</code>	<code>\lesschoice</code>	<code>\lesscons</code>	<code>\lesstest</code>
<code>\lessappli</code>	<code>\lesscond</code>	<code>\lesshyp</code>	

<code>\nlessplot</code>	<code>\nlesschoice</code>	<code>\nlesscons</code>	<code>\nlesstest</code>
<code>\nlessappli</code>	<code>\nlesscond</code>	<code>\nlesshyp</code>	

<code>\leqplot</code>	<code>\leqchoice</code>	<code>\leqcons</code>	<code>\leqtest</code>
<code>\leqappli</code>	<code>\leqcond</code>	<code>\leqhyp</code>	

<code>\nleqplot</code>	<code>\nleqchoice</code>	<code>\nleqcons</code>	<code>\nleqtest</code>
<code>\nleqappli</code>	<code>\nleqcond</code>	<code>\nleqhyp</code>	

<code>\gtrplot</code>	<code>\gtrchoice</code>	<code>\gtrcons</code>	<code>\gtrtest</code>
<code>\gtrappli</code>	<code>\gtrcond</code>	<code>\gtrhyp</code>	

3. Équivalences et implications

i. Des symboles logiques supplémentaires

<code>\iff</code>	<code>\iffchoice</code>	<code>\iffcons</code>	<code>\ifftest</code>
<code>\iffappli</code>	<code>\iffcond</code>	<code>\iffhyp</code>	

<code>\niff</code>	<code>\niffchoice</code>	<code>\niffcons</code>	<code>\nifftest</code>
<code>\niffappli</code>	<code>\niffcond</code>	<code>\niffhyp</code>	

<code>\implies</code>	<code>\implieschoice</code>	<code>\impliescons</code>	<code>\impliestest</code>
<code>\impliesappli</code>	<code>\impliescond</code>	<code>\implieshyp</code>	

<code>\nimplies</code>	<code>\nimplieschoice</code>	<code>\nimpliescons</code>	<code>\nimpliestest</code>
<code>\nimpliesappli</code>	<code>\nimpliescond</code>	<code>\nimplieshyp</code>	

<code>\liesimp</code>	<code>\liesimpchoice</code>	<code>\liesimpcons</code>	<code>\liesimptest</code>
<code>\liesimpappli</code>	<code>\liesimpcond</code>	<code>\liesimphyp</code>	

<code>\nliesimp</code>	<code>\nliesimpchoice</code>	<code>\nliesimpcons</code>	<code>\nliesimptest</code>
<code>\nliesimpappli</code>	<code>\nliesimpcond</code>	<code>\nliesimphyp</code>	

4. Équivalences et implications

i. Équivalences et implications verticales

<code>\viff</code>	<code>\vimplies</code>	<code>\vliesimp</code>
<code>\nviff</code>	<code>\nvimplies</code>	<code>\nvliesimp</code>

5. Des versions alternatives du quantificateur existentiel

`\existmulti {#1}`
`\nexistmulti{#1}`

— Argument 1: une écriture mathématique servant à préciser la portée du quantificateur.

`\existsTsone`
`\nexistsTsone`

6. Détailler un raisonnement simple

i. Détailler un raisonnement simple

`\begin{explain} [#opt]`
`...`
`\end{explain}`

— Option: la valeur utilise une syntaxe de type clé-valeur. Voici les différentes clés disponibles.

1. `ope` sert à définir l'opérateur utilisé dans tout l'environnement qui sera rédigé en mode mathématique. La valeur par défaut est `{=}` (et non *juste* `=`).
2. `style` sert à définir le style de mise en forme. Voici les différentes valeurs possibles.
 - (a) `u`, la valeur par défaut, est pour u-niversity.

- (b) `ar` est pour `ar-row`.
- (c) `sar` est pour `s-hort ar-row`.

3. `com` permet de demander l’alignement ou non des commentaires non étoilés entre eux.

- (a) `nal`, la valeur par défaut, est pour `n-ot al-igned`.
- (b) `al` est pour `al-igned`.

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

`expl = expl-ain`

— **Option**: le symbole à utiliser pour une explication, la valeur par défaut étant celle du symbole de l’environnement `explain` où `\explnext` est utilisé.

— **Argument**: le texte de l’explication qui peut être vide si aucune explication n’est à afficher.

ATTENTION! La macro `\explnext` est à utiliser sans argument ni option au tout début du contenu de l’environnement `explain` en cas d’utilisation du style `sar`.

`\explnext* [#opt] {#1..#2}`

— **Option**: le symbole à utiliser pour une explication, la valeur par défaut étant celle du symbole de l’environnement `explain` où `\explnext` est utilisé.

— **Argument 1**: le texte de l’explication pour la 1^{re} ligne. Ce texte peut être vide (*voir l’environnement `aexplain` pour la raison de ceci*).

— **Argument 2**: le texte de l’explication pour la 2^e ligne. Ce texte peut être vide (*voir l’environnement `aexplain` pour la raison de ceci*).

`\comthis {#1}`

`com = com-ment`

`\comthis*{#1}`

— **Argument**: le texte d’un court commentaire.

ii. Détailler un raisonnement simple – Mise en forme du texte

Les macros suivantes sont juste utilisées par l’environnement `explain`.

`\expltxtspacein{}`

`\expltxt{#1}`

— **Argument**: le texte de l’explication que l’on veut mettre en forme.

`\expltxttdown{#1}`

— **Argument**: le texte de l’explication du haut vers le bas que l’on veut mettre en forme.

`\expltxtup{#1}`

— **Argument**: le texte de l’explication du bas vers le haut que l’on veut mettre en forme.

```
\expltxtupdown{#1..#2}
```

— Argument 1: le texte de l'explication du haut vers le bas que l'on veut mettre en forme.

— Argument 1: le texte de l'explication du bas vers le haut que l'on veut mettre en forme.

```
\explcom{#1}
```

— Argument: le texte d'un court commentaire.

7. Détailler un « vrai » raisonnement

i. Détailler un « vrai » raisonnement via un tableau

```
\begin{demoexplain} [#opts]
```

```
...
```

```
\end{demoexplain}
```

— Clé "start": le début de la numérotation des identifiants des justifications. La valeur par défaut est 1 et la valeur spéciale `last` permet de reprendre la numérotation là où elle s'était arrêtée le dernier environnement `demoexplain` ou `demoexplain*` utilisé.

— Clé "hyps ": les hypothèses, au format texte, vérifiées au départ. Cet argument peut être vide et ne doit pas rentrer en conflit avec l'option `hyp`.

— Clé "hyp ": une unique hypothèse, au format texte, vérifiée au départ. Cet argument peut être vide et ne doit pas rentrer en conflit avec l'option `hyps`.

— Clé "ccl ": la conclusion, au format texte, du raisonnement détaillé. Cet argument peut être vide.

```
\begin{demoexplain*} [#opt]
```

```
...
```

```
\end{demoexplain*}
```

— Clé "start": le début de la numérotation des identifiants des justifications. La valeur par défaut est 1 et la valeur spéciale `last` permet de reprendre la numérotation là où elle s'était arrêtée le dernier environnement `demoexplain` ou `demoexplain*` utilisé.

```
\demostep [#opt]
```

— Option: un texte qui sera utilisé comme label global référençant le numéro d'une justification.

```
\explref{#1}
```

ref = ref-erence

— Argument: un numéro de 1 ou 2 chiffres qui sera encadré comme le sont les numérotations des indications.

```
\explref*{#1}
```

— Argument: un texte correspondant à un label global référençant le numéro d'une justification.

ii. Détailler un « vrai » raisonnement via un tableau - Textes utilisés

<code>\textdemoID{}</code>	ID = ID-entifier
<code>\textdemoKNOWN{}</code>	
<code>\textdemoPROP{}</code>	PROP = PROP-osition
<code>\textdemoCONS{}</code>	CONS = CONS-equence
<code>\textdemoHYP{}</code>	HYP = HYP-othesis (S-everal ones)
<code>\textdemoHYP{}</code>	HYP = HYP-othesis (just one)
<code>\textdemoCCL{}</code>	CCL = C-on-CL-usion
<code>\textdemoNEXTPAGE{}</code>	

III. Géométrie

1. Points et lignes

i. Points

`\pt{#1}`

— Argument : un texte donnant le nom d'un point.

`\pt*{#1..#2}`

— Argument 1 : un texte indiquant UP dans le nom UP_{down} d'un point.

— Argument 2 : un texte indiquant *down* dans le nom UP_{down} d'un point.

2. Points et lignes

i. Lignes

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

g = g-eometry

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

p = p-oint

— Option : pour indiquer les parenthèses ou crochets à utiliser, les valeurs possibles étant 0, valeur par défaut, C, CO et OC.

— Argument 1 : le 1^{er} point géométrique.

— Argument 2 : le 2^e point géométrique.

`\hgline {#1..#2}`

h = h-alf

`\phgline {#1..#2}`

phg = p + h + g

`\segment {#1..#2}`

`\psegment {#1..#2}`

— Argument 1 : le 1^{er} point géométrique.

— Argument 2 : le 2^e point géométrique.

3. Points et lignes

i. Droites parallèles ou non

`\parallel`

`\nparallel`

`\stdparallel` (pour utiliser le symbole par défaut)

`\stdnparallel` (pour utiliser le symbole proposé par `amssymb`)

4. Vecteurs

i. Les écrire

`\vect{#1}`

— **Argument**: un texte donnant le nom d'un vecteur.

`\vect*{#1..#2}`

— **Argument 1**: un texte indiquant *up* dans le nom $\overrightarrow{up}_{down}$ d'un vecteur.

— **Argument 2**: un texte indiquant *down* dans le nom $\overrightarrow{up}_{down}$ d'un vecteur.

5. Vecteurs

i. Norme

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

— **Option**: la valeur par défaut est `b`. Deux options disponibles.

1. `b` : des doubles barres extensibles sont utilisées.
2. `s` : des doubles barres non extensibles sont utilisées.

— **Argument**: le vecteur sur lequel appliquer la norme.

`\vnorm{#1}`

`v = v-ector`

— **Argument**: le nom du vecteur sur lequel appliquer la norme.

6. Vecteurs

i. Produit scalaire

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

— **Option**: la valeur par défaut est `u` pour `u`-sual soit « *habituel* » en anglais. Voici les différentes valeurs possibles.

1. `u` : écriture habituelle avec un point.
2. `b` : écriture habituelle mais avec une puce.
3. `p` : écriture « universitaire » avec des parenthèses extensibles.
4. `sp` : écriture « universitaire » avec des parenthèses non extensibles.
5. `r` : écriture « à la physicienne » avec des chevrons extensibles.
6. `sr` : écriture « à la physicienne » avec des chevrons non extensibles.

— Argument 1: le 1^{er} vecteur qu'il faut taper via la macro `\vect`.

— Argument 2: le 2^e vecteur qu'il faut taper via la macro `\vect`.

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

`v = v-ector`

— Option: voir les explications précédentes données pour `\dotprod`.

— Argument 1: le nom du 1^{er} vecteur sans utiliser la macro `\vect`.

— Argument 2: le nom du 2^e vecteur sans utiliser la macro `\vect`.

7. Vecteurs

i. 3D – Produit vectoriel

Écriture symbolique

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

— Option: la valeur par défaut est `w` pour `w-edge` soit « *coin* » en anglais. Voici les valeurs possibles.

1. `w` : écriture classique en France.
2. `t` : écriture alternative avec le symbole `\times`.

— Argument 1: le 1^{er} vecteur qu'il faut taper via la macro `\vect`.

— Argument 2: le 2^e vecteur qu'il faut taper via la macro `\vect`.

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

`v = v-ector`

— Option: voir les explications précédentes données pour `\crossprod`.

— Argument 1: le nom du 1^{er} vecteur sans utiliser la macro `\vect`.

— Argument 2: le nom du 2^e vecteur sans utiliser la macro `\vect`.

8. Vecteurs

i. 3D – Produit vectoriel

Explication du mode de calcul

`\calccrossprod` {`#1`..`#9`}

`calc = calc-ulate`

`\calccrossprod*` {`#1`..`#9`}

`\calccrossprod**` {`#1`..`#9`}

— Argument 1: `vec` ou `novec` suivant que l'on veut afficher ou non les vecteurs.

— Argument 2: le 1^{er} vecteur qu'il faut taper via la macro `\vect`.

— Arguments 3..5: les coordonnées du 1^{er} vecteur.

— Argument 6: le 2^e vecteur qu'il faut taper via la macro `\vect`.

— Arguments 7..9: les coordonnées du 2^e vecteur.

`\vcalccrossprod` {`#1`..`#8`}

`calc = calc-ulate` et `v = v-ector`

`\vcalccrossprod*` {`#1`..`#8`}

`\vcalccrossprod**{#1..#8}`

- Argument 1: `vec` ou `novec` suivant que l'on veut afficher ou non les vecteurs.
- Argument 2: le 1^{er} vecteur sans utiliser la macro `\vect`.
- Arguments 3..5: les coordonnées du 1^{er} vecteur.
- Argument 6: le 2^e vecteur sans utiliser la macro `\vect`.
- Arguments 7..9: les coordonnées du 2^e vecteur.

9. Vecteurs

i. 3D – Produit vectoriel

Les coordonnées

`\coordcrossprod[#opt]{#1..#6}`

`coord = coord-inate`

— Option: la valeur par défaut est `p,s`. Voici les différentes valeurs possibles pour la mise en forme des coordonnées uniquement (*voir la section i. page 121*).

1. `p` : écriture horizontale avec des parenthèses extensibles.
2. `sp` : écriture horizontale avec des parenthèses non extensibles.
3. `vp` : écriture verticale avec des parenthèses.
4. `b` : écriture horizontale avec des crochets extensibles.
5. `sb` : écriture horizontale avec des crochets non extensibles.
6. `vb` : écriture verticale avec des crochets.

Pour les produits, voici ce qui est proposé.

1. `s` : un espace pour séparer les facteurs de chaque produit.
2. `t` : `\times` comme opérateur de multiplication.
3. `c` : `\cdot` comme opérateur de multiplication.

On peut combiner deux types de choix en les séparant par une virgule comme dans `p,s` la valeur par défaut de l'option.

- Arguments 1..3: les coordonnées du 1^{er} vecteur.
- Arguments 4..6: les coordonnées du 2^e vecteur.

10. Vecteurs

i. 2D – Critère de colinéarité de deux vecteurs

`\colicriteria [#opt]{#1..#6}`

`coli = coli-nearity`

— Option: la valeur par défaut est `vec`. Voici les deux valeurs possibles.

1. `vec` : les vecteurs sont affichés si besoin.
2. `novec` : les vecteurs ne sont jamais affichés.

- Argument 1: le 1^{er} vecteur qu'il faut taper via la macro `\vect`.
 - Arguments 2..3: les coordonnées du 1^{er} vecteur.
 - Argument 4: le 2^e vecteur qu'il faut taper via la macro `\vect`.
 - Arguments 5..6: les coordonnées du 2^e vecteur.
-

`\colicriteria` [*#opt*] {*#1..#6*}

v = v-ector

— Option: voir les indications données pour la macro `\colicriteria`.

— Arguments 1..6: voir les indications données pour la macro `\colicriteria`.

11. Vecteurs

i. 2D – Déterminant de deux vecteurs

`\calcdetplane` [*#opt*] {*#1..#6*}

calc = calc-ulate

`\calcdetplane*` [*#opt*] {*#1..#6*}

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

1. *vec* : les vecteurs sont affichés si besoin.
2. *novec* : les vecteurs ne sont jamais affichés.
3. *exp* : ceci demande d'afficher une formule développée en utilisant un espace pour séparer les facteurs de chaque produit.
4. *cexp* : comme *exp* mais avec le symbole \cdot obtenu via `\cdot`.
5. *texp* : comme *exp* mais avec le symbole \times .

— Argument 1: le 1^{er} vecteur qu'il faut taper via la macro `\vect`.

— Arguments 2..3: les coordonnées du 1^{er} vecteur.

— Argument 4: le 2^e vecteur qu'il faut taper via la macro `\vect`.

— Arguments 5..6: les coordonnées du 2^e vecteur.

`\vcalcdetplane` [*#opt*] {*#1..#6*}

calc = calc-ulate et *v* = v-ector

`\vcalcdetplane*` [*#opt*] {*#1..#6*}

— Option: voir les indications données pour les macros `\calcdetplane` et `\calcdetplane*`.

— Arguments 1..6: voir les indications données pour les macros `\calcdetplane` et `\calcdetplane*`.

12. Géométrie cartésienne

i. Coordonnées

`\coord` [*#opt*] {*#1*}

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

1. *p* : écriture horizontale avec des parenthèses extensibles.
2. *sp* : écriture horizontale avec des parenthèses non extensibles.
3. *vp* : écriture verticale avec des parenthèses.
4. *b* : écriture horizontale avec des crochets extensibles.
5. *sb* : écriture horizontale avec des crochets non extensibles.
6. *vb* : écriture verticale avec des crochets.

— Argument: l'argument est une suite de "morceaux" séparés par des barres | , chaque morceau étant une coordonnée. Il peut n'y avoir qu'un seul morceau.

`\pcoord` [*#opt*] {*#1..#2*}

p = p-oint

— Option: voir les indications données pour la macro `\coord`.

- **Argument 1**: le point auquel sera appliqué automatiquement la macro `\pt`.
- **Argument 2**: voir les indications données pour l'unique argument obligatoire de la macro `\coord`.

`\pcoord* [#opt] {#1..#2}`

- **Option**: voir les indications données pour la macro `\coord`.
- **Argument 1**: le point auquel ne sera pas appliqué automatiquement la macro `\pt`.
- **Argument 2**: voir les indications données pour l'unique argument obligatoire de la macro `\coord`.

`\vcoord [#opt] {#1..#2}` `v = v-ertical`

- **Option**: voir les indications données pour la macro `\coord`.
- **Argument 1**: le vecteur sans utiliser la macro `\vect`.
- **Argument 2**: voir les indications données pour l'unique argument obligatoire de la macro `\coord`.

`\vcoord* [#opt] {#1..#2}`

- **Option**: voir les indications données pour la macro `\coord`.
- **Argument 1**: le vecteur qu'il faut taper via la macro `\vect`.
- **Argument 2**: voir les indications données pour l'unique argument obligatoire de la macro `\coord`.

13. Géométrie cartésienne

i. Nommer un repère

`\axes {#1}`

`\axes* {#1}`

- **Argument**: l'argument est une suite de "morceaux" séparés par des barres |.
 - Le premier morceau est l'origine du repère.
 - Les morceaux suivants sont des points ou des vecteurs qui "définissent" chaque axe.

`\paxes [#opt] {#1..# }` `p = p-oint`

- **Argument**: l'argument est une suite de "morceaux" séparés par des barres |.
 - Le premier morceau est le nom de l'origine du repère sur laquelle la macro-commande `\pt` sera automatiquement appliquée.
 - Viennent ensuite les noms des points "définissant" chaque axe. Pour chacun de ces points la macro-commande `\pt` sera automatiquement appliquée.

`\vaxes [#opt] {#1..# }` `v = v-ector`

- **Argument**: l'argument est une suite de "morceaux" séparés par des barres |.
 - Le premier morceau est l'origine du repère.
 - Viennent ensuite les noms des vecteurs "définissant" chaque axe. Pour chacun de ces vecteurs la macro-commande `\vect` sera automatiquement appliquée.

 $\backslash\text{pvaxes}[\text{\#opt}]{\text{\#1}..\text{\#}}\}$
 $\text{pv} = \text{p} + \text{v}$

— **Argument**: l'argument est une suite de "morceaux" séparés par des barres |.

- Le premier morceau est le nom de l'origine du repère sur laquelle la macro-commande $\backslash\text{pt}$ sera automatiquement appliquée.
- Viennent ensuite les noms des vecteurs "définissant" chaque axe. Pour chacun de ces vecteurs la macro-commande $\backslash\text{vect}$ sera automatiquement appliquée.

14. Arcs circulaires

 $\backslash\text{circarc}\{\text{\#1}\}$
 $\text{circ} = \text{circ-ular}$

— **Argument**: un texte donnant le nom d'un arc circulaire.

 $\backslash\text{circarc}^*\{\text{\#1}..\text{\#2}\}$

— **Argument 1**: un texte indiquant *up* dans le nom \widehat{up}_{down} d'un arc circulaire.

— **Argument 2**: un texte indiquant *down* dans le nom \widehat{up}_{down} d'un arc circulaire.

15. Angles

i. Angles géométriques « intérieurs »

 $\backslash\text{anglein}\{\text{\#1}\}$
 $\text{in} = \text{in-terior}$

— **Argument**: un texte donnant le nom d'un angle intérieur.

 $\backslash\text{anglein}^*\{\text{\#1}..\text{\#2}\}$

— **Argument 1**: un texte indiquant *up* dans le nom \widehat{up}_{down} d'un angle intérieur.

— **Argument 2**: un texte indiquant *down* dans le nom \widehat{up}_{down} d'un angle intérieur.

16. Angles

i. Angles orientés de vecteurs

 $\backslash\text{angleorient}[\text{\#opt}]{\text{\#1}..\text{\#2}\}$

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

1. *p* : écriture habituelle avec des parenthèses extensibles.
2. *sp* : écriture habituelle avec des parenthèses non extensibles.
3. *h* : écriture avec un chapeau et des parenthèses extensibles.
4. *sh* : écriture avec un chapeau et des parenthèses non extensibles.

— **Argument 1**: le premier vecteur qu'il faut taper via la macro $\backslash\text{vect}$.

— **Argument 2**: le second vecteur qu'il faut taper via la macro $\backslash\text{vect}$.

 $\backslash\text{vangleorient}[\text{\#opt}]{\text{\#1}..\text{\#2}\}$
 $\text{v} = \text{v-ector}$

— **Option**: voir les explications précédentes données pour $\backslash\text{angleorient}$.

- Argument 1 : le nom du premier vecteur sans utiliser la macro `\vect`.
- Argument 2 : le nom du second vecteur sans utiliser la macro `\vect`.

IV. Analyse

1. Constantes et paramètres

i. Constantes classiques ajoutées

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

2. Constantes et paramètres

i. Constantes latines personnelles

`\param{#1}`

- Argument : un texte utilisant l'alphabet latin.

3. Une variable « symbolique »

`\symvar[#opt]`

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

- 1 donne \bullet .
- 2 donne \circ .
- 3 donne \blacksquare .

4. La fonction valeur absolue

`\abs {#1}`

`\abs*{#1}`

- Argument : l'expression sur laquelle appliquer la fonction valeur absolue.

5. Fonctions nommées spéciales

i. Sans paramètre

`\acos`

`\asin`

`\atan`

`\arccosh`

`\arcsinh`

`\arctanh`

`\acosh`

`\asinh`
`\atanh`

<code>\fch</code>	<code>f = f-rench</code>
<code>\fsh</code>	<code>f = f-rench</code>
<code>\fth</code>	<code>f = f-rench</code>

`\afch`
`\afsh`
`\afth`

6. Fonctions nommées spéciales

i. Avec un paramètre

`\exp[#opt]`

— Option: la base de l'exponentielle

`\log[#opt]`

— Option: la base du logarithme

7. 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.

8. 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.

9. 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. `srqt` 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 grande¹.

10. Calcul intégral

i. Le symbole standard revisité

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

11. Calcul intégral

i. 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} .

12. Calcul intégral

i. L'opérateur « crochet »

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

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

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

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.

1. 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.

- 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.

V. Suites

1. Des notations complémentaires pour des suites spéciales

`\seqplus{#1..#2}`

- Argument 1: l'exposant à droite.
- Argument 2: l'indice à droite.

`\seqhypergeo{#1..#2}`

- Argument 1: l'indice à gauche.
- Argument 2: l'indice à droite.

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

- Argument 1: l'indice à gauche.
- Argument 2: l'indice à droite.
- Argument 3: l'exposant à droite.
- Argument 4: l'exposant à gauche.

2. Sommes et produits en mode ligne

Les opérateurs suivants ont un comportement spécifique vis à vis des mises en index et en exposant.

`\dprod`

`\dsum`

3. Comparaison asymptotique de suites et de fonctions

i. Les notations \mathcal{O} et \mathcal{o}

`\bigO {#1}`

`\smallO{#1}`

- Argument: un argument vide est ignoré, sinon il est mis entre des parenthèses après \mathcal{O} ou \mathcal{o} .

ii. La notation Ω

`\bigomega{#1}`

- Argument: un argument vide est ignoré, sinon il est mis entre des parenthèses après Ω .

iii. La notation Θ

`\bigtheta{#1}`

— **Argument**: un argument vide est ignoré, sinon il est mis entre des parenthèses après Θ .

VI. Probabilité

1. Généralités

i. Probabilité « simple »

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

— **Option**: le nom de la probabilité. La valeur par défaut est p .

— **Argument**: l'ensemble dont on veut calculer la probabilité.

2. Généralités

i. Probabilité conditionnelle

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

`\probacond* [#opt]{#1..#2}`

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

`\eprobacond* [#opt]{#1..#2}`

— **Option**: le nom de la probabilité. La valeur par défaut est p .

— **Argument 1**: l'ensemble qui donne la condition.

— **Argument 2**: l'ensemble dont on veut calculer la probabilité.

3. Généralités

i. Évènement contraire

`\nevent{#1}`

— **Argument**: l'ensemble dont on veut indiquer le contraire.

4. Généralités

i. Espérance, variance et écart-type

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

— **Option**: le nom de la fonction espérance. La valeur par défaut est E obtenue via `\mathrm{E}`.

— **Argument**: la variable aléatoire dont on veut calculer l'espérance.

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

— **Option**: le nom de la fonction variance. La valeur par défaut est V obtenue via `\mathrm{V}`.

— **Argument**: la variable aléatoire dont on veut calculer la variance.

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

— **Option**: le nom de la fonction écart-type. La valeur par défaut est σ obtenue via `\sigma`.

— **Argument**: la variable aléatoire dont on veut calculer l'écart-type.

5. Arbres pondérés

`\begin{probatree}`

...

`\end{probatree}`

`\begin{probatree*}`

...

`\end{probatree*}`

— **Contenu**: un arbre codé en utilisant la syntaxe supportée par le package `forest`.

— Clé `"pweight"`: pour écrire un poids sur le milieu d'une branche.

— Clé `"apweight"`: pour écrire un poids au-dessus le milieu d'une branche.

— Clé `"bpweight"`: pour écrire un poids en-dessous du milieu d'une branche.

— Clé `"pframe"`: pour encadrer un sous-arbre depuis un noeud vers toutes les feuilles de celui-ci.

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

`p = p-robabilty`

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

— **Arguments 1..3**: noms de la sous-racine (à gauche), du noeud final en haut (à droite) et du noeud final en bas (à droite). En fait l'ordre n'est pas important ici.

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

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

— **Argument 1**: le nom de la feuille.

— **Argument 2**: le texte du commentaire.

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

`\ptreeFocus* [#opt] {#1}`

`\ptreeFocus** [#opt] {#1}`

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

— **Argument**: les noms des noeuds dans le bon ordre et séparés par des barres verticales `|`.

VII. Arithmétique

1. Opérateurs de base

`\divides`

`\ndivides`

`\nequiv`

`\modulo`

2. Fonctions nommées spéciales

`\pgcd`

`\ppcm`

`\lcm`

3. Fractions continuées

i. Fractions continuées standard

`\contfrac {#1}`

`\contfrac*{#1}`

— Argument : tous les éléments de la fraction continuée séparés par des |.

4. Fractions continuées

i. Fractions continuées généralisées

`\contfracgene {#1}`

`\contfracgene*{#1}`

— Argument : tous les éléments de la fraction continuée généralisée séparés par des |.

5. Fractions continuées

i. Comme une fraction continuée isolée

`\singlecontfrac{#1..#2}`

— Argument 1 : le pseudo numérateur.

— Argument 2 : le pseudo dénominateur.

6. Fractions continuées

i. L'opérateur \mathcal{K}

La macro suivante sans argument a un comportement spécifique vis à vis des mises en index et en exposant.

`\contfracope`

VIII. Algèbre linéaire

1. Matrices via nicematrix

i. Calculs expliqués des déterminants 2×2

`\calcdettwo` `[#opt]` `{#1..#4}` `c = c-alculate`
`\calcdettwo*` `[#opt]` `{#1..#4}`

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

1. `std` : on utilise l'écriture matricielle.
2. `exp` : ceci demande d'afficher une formule développée en utilisant \times pour les produits.
3. `cexp` : comme `exp` mais avec le symbole \cdot obtenu via `\cdot`.
4. `sexp` : comme `exp` mais avec un espace pour séparer les facteurs de chaque produit.

— **Argument 1**: l'entrée à la position (1, 1)

— **Argument 2**: l'entrée à la position (1, 2)

— **Argument 3**: l'entrée à la position (2, 1)

— **Argument 4**: l'entrée à la position (2, 2)

IX. Polynômes et séries formelles

1. Polynômes

`\setpoly` `{#1..#2}`
`\setpolyfrac` `{#1..#2}`

— **Argument 1**: l'ensemble auquel les coefficients appartiennent.

— **Argument 2**: cet argument est une suite de "morceaux" séparés par des barres `|`, chaque morceau étant une variable formelle.

2. Séries formelles classiques

`\setserie` `{#1..#2}`
`\setseriefrac` `{#1..#2}`

— **Argument 1**: l'ensemble auquel les coefficients appartiennent.

— **Argument 2**: cet argument est une suite de "morceaux" séparés par des barres `|`, chaque morceau étant une variable formelle.

3. Polynômes et séries formelles de Laurent

`\setpolylaurent` `{#1..#2}`
`\setserielaurent` `{#1..#2}`

— **Argument 1**: l'ensemble auquel les coefficients appartiennent.

— **Argument 2**: cet argument est une suite de "morceaux" séparés par des barres `|`, chaque morceau étant une variable formelle.