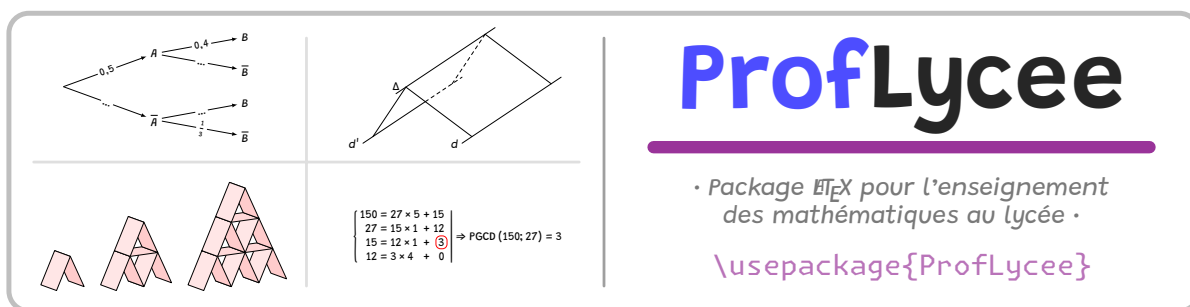




Cédric Pierquet

c pierquet -- at -- outlook . fr

<https://github.com/cpierquet/latex-packages/tree/main/proflycee>

Version 4.02c – 23 octobre 2026



Résumé : quelques commandes pour faciliter l'utilisation de $\LaTeX$ pour les mathématiques, au lycée – calculs formatés, tracés TikZ, présentation de code, et bien plus.

- ✦ résoudre, de manière approchée, des équations
- ✦ présenter du code python ou pseudocode, une console d'exécution Python
- ✦ tracer rapidement un pavé, un tétraèdre
- ✦ simplifier des calculs sous forme fractionnaire, simplifier des racines
- ✦ effectuer des calculs avec des suites récurrentes
- ✦ afficher et utiliser un cercle trigo
- ✦ afficher un petit schéma pour le signe d'une fonction affine ou d'un trinôme
- ✦ convertir entre bin/dec/hex avec détails
- ✦ présenter un calcul de PGCD
- ✦ effectuer des calculs de probas (lois binomiale, exponentielle, de Poisson, normale)
- ✦ créer des arbres de probas « classiques »
- ✦ déterminer la mesure principale d'un angle, calculer les lignes trigonométriques d'angles « classiques »
- ✦ résoudre une équation diophantienne « classique »
- ✦ ...

Merci à Anne et quark67 pour leurs retours et relectures!

Merci à Christophe, Denis et Franck-Olivier pour leurs retours et éclairages!

Merci aux membres du groupe du « Coin $\LaTeX$ » pour leur aide et leurs idées!

Suite aux allègements de la version [4.02a](#), une version *antérieure* et plus *complète* de la documentation est disponible ici :

<https://packages.cpierquet.fr/package.php?name=proflycee&category=maths>

$\LaTeX$

pdf $\LaTeX$

Lua $\LaTeX$

TikZ

TeXLive

MiKTeX

[v4.02c](#) : Arbres de probas flexibles (154)

[v4.02b](#) : Mise en conformité avec siunitx + correction de bugs mineurs

[v4.02a](#) : Factorisation et simplification de codes

À mon papa.

Table des matières

I	Introduction	11
1	Le package ProfLycee	11
1.1	« Philosophie » du package	11
1.2	Chargement du package	11
1.3	Librairies	12
1.4	Gestion des fontes	12
1.5	Gestion de siunitx	12
1.6	Gestion de babel	12
1.7	Gestion de fontawesome	13
1.8	Gestion de cancel	13
2	Compléments	13
2.1	Support multilingues	13
2.2	Le système de « clés/options »	14
2.3	Compilateur(s)	14
2.4	Problèmes éventuels...	14
2.5	Version allégée	14
2.6	Alias de commandes (en test)	15
II	Liste des commandes, par thème	18
2.7	Calculs formatés, conversions et intervalles	18
2.8	Analyse	18
2.9	Schémas de signe et compléments tkz-tab	19
2.10	Présentation de code	20
2.11	Géométrie (plan et espace)	20
2.12	Statistiques	21
2.13	Probabilités	22
2.14	Aléatoire	23
2.15	Arithmétique et dénombrement	24
2.16	Écritures, ensembles et trinômes	26
2.17	Trigonométrie	26
2.18	Compétences	26
2.19	Récréations mathématiques	26
2.20	Cliparts et logos	27
III	Écritures d'objets mathématiques	29
3	Commande spécifique de calcul/formatage	29
4	Commandes	30
4.1	Arrondi	30
4.2	Unités simplifiées	30
4.3	Ensembles et intervalles	30
4.4	Repères et coordonnées	31
4.5	Valeur absolue/module et norme	33
4.6	Divers	33
4.7	Probabilités	35
4.8	Intervalle de confiance, intervalle de fluctuation	36
5	Collection d'objets	39
5.1	Idée	39
5.2	Commande et options	39
IV	Outils pour l'analyse	41
6	Maximum, minimum d'une fonction	41

7	Résolution approchée d'une équation	42
7.1	Idée	42
7.2	Clés et options	42
8	Présentation d'une solution d'équation par balayage	44
8.1	Idée	44
8.2	Clés et arguments	44
8.3	Interaction avec la commande de résolution approchée	45
9	Suites récurrentes simples	46
9.1	Idées	46
9.2	Clés et arguments	46
9.3	Exemple d'utilisation	47
10	Valeur approchée d'une intégrale	48
10.1	Idée	48
10.2	Clés et arguments	48
10.3	Exemples	49
10.4	Commande simplifiée	49
11	Forme canonique, fonction homographique	51
11.1	Idée	51
11.2	Forme canonique	51
11.3	Fonction homographique	52
V	Outils graphiques	55
12	Préambule	55
13	Intervalles	55
13.1	Idée	55
13.2	Création de l'environnement	55
13.3	Représentation d'intervalles	56
14	Petits schémas pour le signe de fonctions classiques	59
14.1	Idée	59
14.2	Commandes	59
14.3	Intégration avec tkz-tab	61
15	Ligne de convexité en lien avec tkz-tab	62
15.1	Idée	62
15.2	Commande	62
16	Autres compléments pour tkz-tab	64
16.1	Marquer une valeur interdite ou particulière	64
16.2	Colorer une case ou une ligne	64
16.3	Ajouter un commentaire à droite d'une ligne	65
16.4	Variante de MiniSchemaSignesTkzTab	65
VI	Présentation de codes	68
17	Précautions	68
18	Code Python « simple » via le package listings	68
18.1	Introduction	68
18.2	Commande et options	68
18.3	Insertion via un fichier « externe »	69
18.4	Exemples	69
19	Code Python via le package piton	72
19.1	Introduction	72
19.2	Présentation de code Python	72
19.3	Console en partenariat avec Pyluatex	75

20 Pseudo-Code	76
20.1 Introduction	76
20.2 Présentation de Pseudo-Code	76
20.3 Compléments	77
21 PseudoCode via le package piton	78
21.1 Introduction	78
21.2 Présentation de PseudoCode	78
21.3 Exemples	79
22 Code et console Python, à la manière de Thonny	81
22.1 Introduction	81
22.2 Présentation de code, style éditeur	81
22.3 Présentation de code, style console	82
22.4 Exemple	83
23 Cartouche Capytale	84
23.1 Introduction	84
23.2 Commandes	84
VII Outils pour la géométrie	86
24 Pavé droit « simple »	86
24.1 Introduction	86
24.2 Commandes	86
24.3 Influence des paramètres	87
25 Tétraèdre « simple »	88
25.1 Introduction	88
25.2 Commandes	88
25.3 Influence des paramètres	89
26 Cercle trigo	90
26.1 Idée	90
26.2 Commandes	90
26.3 Équations trigos	91
27 Schémas type de géométrie dans l'espace	93
27.1 Idée	93
27.2 Commande	93
VIII Outils pour la géométrie analytique	101
28 Conseils d'utilisation	101
29 Affichage de coordonnées	101
29.1 Idée	101
29.2 Options et arguments	102
30 Équation cartésienne d'un plan de l'espace	103
30.1 Idée et commande	103
30.2 Clés et arguments	103
31 Équation paramétrique d'une droite de l'espace	105
31.1 Idée et commande	105
31.2 Clés et arguments	105
32 Équation cartésienne d'une droite du plan	107
32.1 Idée et commande	107
32.2 Clés et arguments	107

33 Norme d'un vecteur, distance entre deux points	109
33.1 Idée et commande	109
33.2 Clés et arguments	109
34 Distance d'un point à un plan	110
34.1 Idée et commande	110
34.2 Clés et arguments	110
35 Équation d'une sphère	112
35.1 Idée	112
35.2 Clés et arguments	112
35.3 Exemples	112
36 Équation réduite d'une droite du plan	114
36.1 Idée	114
36.2 Clés et arguments	114
36.3 Exemples	114
IX Outils pour les statistiques	118
37 Paramètres d'une série discrète	118
37.1 Idée	118
37.2 Arguments, exemples	118
38 Paramètres d'une régression linéaire par la méthode des moindres carrés	119
38.1 Idée	119
38.2 Commandes	119
38.3 Intégration dans un environnement TikZ	120
39 Statistiques à deux variables	123
39.1 Idées	123
39.2 Commandes, clés et options	124
39.3 Commandes annexes	127
39.4 Interactions avec CalculsRegLin	127
39.5 Exemple complémentaire, pour illustration	131
40 Boîtes à moustaches	133
40.1 Introduction	133
40.2 Clés et options	133
40.3 Commande pour placer un axe horizontal	134
41 Histogrammes	136
41.1 Introduction	136
41.2 Clés et options	137
41.3 Exemple avec des classes régulières	138
41.4 Exemple avec des classes non régulières	139
42 Courbe des ECC/FCC, paramètres	141
42.1 Introduction	141
42.2 Clés et options	143
42.3 Styles et exemples	143
X Outils pour les probabilités	146
43 Factorielle	146
43.1 Idée	146
43.2 Clés	146
44 Calculs de probabilités	148
44.1 Introduction	148
44.2 Calculs « simples »	148
44.3 Complément avec sortie « formatée »	150

45 Arbres de probabilités « classiques »	152
45.1 Introduction	152
45.2 Options et arguments	152
45.3 Exemples complémentaires	153
45.4 Arbres de probabilités flexibles	154
46 Petits schémas pour des probabilités continues	157
46.1 Idée	157
46.2 Commandes et options	157
46.3 Remarques et compléments	158
47 Nombres aléatoires	159
47.1 Idée	159
47.2 Clés et options	160
48 Combinatoire	161
48.1 Idée	161
48.2 Utilisation	161
49 Arbres de dénombrement	162
49.1 Idée	162
49.2 Styles et clés	163
49.3 Exemples	164
50 Fonction de répartition	168
50.1 Idée	168
50.2 Utilisation	168
51 Loi binomiale et graphiques	170
51.1 Idée	170
51.2 Utilisation de la commande autonome et de l'environnement	170
51.3 Le tracé de la loi normale	170
51.4 Exemples	171
XI Outils pour l'arithmétique	175
52 Réduction modulo	175
52.1 Idée	175
52.2 Clés et options	175
53 Conversions entre bases	176
53.1 Idée	176
53.2 Exemples	176
54 Division euclidienne	177
54.1 Idée	177
54.2 Clés et options	177
55 Conversions binaire/hexadécimal/décimal	179
55.1 Idée	179
55.2 Conversion décimal vers binaire	179
55.3 Conversion binaire vers hexadécimal	180
55.4 Conversion hexadécimal vers binaire	181
55.5 Conversion binaire ou hexadécimal en décimal	181
56 Conversion « présentée » d'un nombre en base décimale	183
56.1 Idée	183
56.2 Code et clés	183
57 Algorithme d'Euclide pour le PGCD	185
57.1 Idée	185
57.2 Options et clés	185
57.3 Compléments	186

58 Résolution d'une équation diophantienne	187
58.1 Idée	187
58.2 Options et clés	187
59 Nombres premiers	191
59.1 Idées	191
59.2 Test de primalité	191
59.3 Décomposition en facteurs premiers	191
60 Diviseurs	193
60.1 Idées	193
60.2 Options et clés	193
60.3 Compléments	196
61 Chiffrements	198
61.1 Idées	198
61.2 Chiffrement de César	198
61.3 Inverse modulo	199
61.4 Chiffrement affine	199
61.5 Chiffrement de Hill	200
62 Primorielle	201
62.1 Idée	201
62.2 Clés	201
63 Opérations posées	202
63.1 Idée	202
63.2 Clés et options	203
63.3 Exemples	203
XII Écritures, simplifications	207
64 Simplification sous forme d'une fractions	207
64.1 Idée	207
64.2 Commande et options	207
65 Écriture d'un trinôme, trinôme aléatoire	209
65.1 Idée	209
65.2 Clés et options	209
66 Simplification de racines	210
66.1 Idée	210
66.2 Exemples	210
67 Mesure principale d'un angle	211
67.1 Idée	211
67.2 Exemples	211
68 Lignes trigonométriques	212
68.1 Idée	212
68.2 Commande	212
68.3 Valeurs disponibles	214
69 Écriture sous forme de fraction irréductible d'un décimal périodique	215
69.1 Idées	215
69.2 Options et clés	215
XIII Jeux et récréations	218
70 SudoMaths, en TikZ	218
70.1 Introduction	218
70.2 Clés et options	219

71 Quelques fractales, en TikZ	221
71.1 Introduction	221
71.2 Flocon de Koch et triangle de Sierpinski	221
71.3 Affichage de plusieurs étapes pour les flocons de Koch	223
71.4 Affichage de plusieurs étapes pour les tapis de Sierpinski	224
72 Châteaux de cartes	225
72.1 Introduction	225
72.2 Clés et options	225
73 Allumettes	227
73.1 Introduction	227
73.2 Clés et options	227
74 Pyramide de sphères	229
74.1 Introduction	229
74.2 Clés et options	229
75 Commandes pour les dates	230
75.1 Introduction	230
75.2 Exemples	230
76 Machines à transformer, en TikZ	231
76.1 Introduction	231
76.2 Clés et options	232
 XIV Quelques cliparts	 236
77 Idées	236
78 Liste des cliparts, et aliases	237
79 Aperçu du logo "calculatrice interdite"	238
80 Cliparts 'Zone Bac/Brevet/BTS'	238
81 Cliparts flat 'Zone Bac/Brevet/BTS'	240
 XV Compétences au lycée	 242
82 Introduction	242
82.1 Fonctionnement global	242
82.2 Commandes	242
82.3 Liste des compétences lycée général et technologique	243
82.4 Liste des compétences lycée professionnel	244
82.5 Exemples	244
 XVI Outils pour les sujets d'examen	 247
83 Page de garde d'un sujet de baccalauréat	247
83.1 Idée	247
83.2 Clés	247
83.3 Exemples	249
 XVII En projet	 255
84 Préambule, avertissement(s)	255
85 Mini schémas de signe	255

86 Additions posées	256
86.1 Commande compatible avec tout compilateur	256
86.2 Commande compatible en LUA	257
87 Compléments sur la factorielle	258
88 Binôme de Newton	259
 XVIII Historique, depuis la version 4.00	 262

Thème

INTRODUCTION

Première partie

Introduction

1 Le package ProfLycee

1.1 « Philosophie » du package



Ce package, très *largement* inspiré (et beaucoup moins *abouti*) de l'excellent `\ProfCollege` de C. Poulain et des excellents `\tkz-*` d'A. Matthes, va définir quelques outils pour des situations particulières qui ne sont pas encore dans `\ProfCollege`.

On peut le voir comme un (maigre) complément à `\ProfCollege`, et je précise que la syntaxe est très proche (car pertinente de base) et donc pas de raison de changer une *équipe qui gagne*!

Il se charge de manière classique, dans le préambule, par `\usepackage{ProfLycee}`. Il charge des packages utiles, mais j'ai fait le choix de laisser l'utilisateur gérer ses autres packages, comme notamment `\amssymb` qui peut poser souci en fonction de la *position* de son chargement.

L'utilisateur est libre de charger ses autres packages utiles et habituels, ainsi que ses polices et encodages habituels!



2.7.2 Pour des soucis de compatibilités, `\xcolor` n'est plus chargé, par défaut, avec les options `[table,svgnames]`, les couleurs de base de `\xcolor` sont toutefois accessibles (une seule couleur, `CouleurVertForet` a été définie)!

Il est cependant possible, grâce à l'option `\xcolor` à passer au chargement du package, de charger `\xcolor` avec l'option `[table,svgnames]`.



Le package `\ProfLycee` charge et utilise des packages comme :

- `\mathtools`, `\amssymb`, `\xspace`, `\esvect`, `\interval`, `\mleftright`;
- `\tikz`, `\pgf`, `\pgffor`, `\nicefrac`, `\nicematrix`;
- `\tcolorbox` avec les librairies `breakable`, `fitting`, `skins`, `listings`, `listingsutf8`, `hooks`;
- `\xstring`, `\simplekv`, `\xinttools`;
- `\listofitems`, `\xintexpr`, `\xintbinhex`, `\xintgcd`;
- `\tabularray`, `\fontawesome5`, `\randomlist`, `\fancyvrb`.



J'ai utilisé les packages de C. Tellechea, je vous conseille d'aller jeter un œil sur ce qu'il est possible de faire en $\text{\LaTeX}$ (même si certains n'apprécient pas ce genre de travail d'empilage sans intérêt...) avec `\listofitems`, `\randomlist`, `\simplekv` ou encore `\xstring`!

1.2 Chargement du package

Code $\text{\LaTeX}$

```
%exemple de chargement pour une compilation en (pdf)latex
\documentclass{article}
\usepackage{ProfLycee}          % ou \usepackage[xcolor]{ProfLycee}
\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
...
```

Code $\text{\LaTeX}$

```
%exemple de chargement pour une compilation en (xe/lua)latex
\documentclass{article}
\usepackage{ProfLycee}          % ou \usepackage[xcolor]{ProfLycee}
\usepackage{fontspec}
...
```

1.3 Librairies



2.5.0 Le package fonctionne désormais avec un système de `librairies`, qui utilisent et chargent des packages spécifiques, avec des compilations particulières, donc l'utilisateur utilisera un système de chargement similaire à celui de `tcolorbox` ou `tikz`, dans le préambule, et une fois le package appelé.

Code $\text{\LaTeX}$

```
\usepackage{ProfLycee}
\useproflyclicb{...,...}
```



Depuis la version 4.02a, `ProfLycee` et `ProfLycee-Light` partagent le même moteur de chargement : la liste complète des librairies disponibles se trouve section 2.5 (« Version allégée »), qui s'applique aux deux points d'entrée.



2.5.8 Pour le package `piton`, la version minimale requise est la 1.5 pour bénéficier d'un rendu optimal (au niveau des marges) de la présentation du code Python.

1.4 Gestion des fontes



2.6.5 Sous $\text{\XeTeX}$ & $\text{\LuaTeX}$, `ProfLycee` utilisant le package `mathtools`, il est nécessaire de placer l'appel à `ProfLycee` avant l'appel des fontes.

Sous $\text{\XeTeX}$ & $\text{\LuaTeX}$, certaines fontes (par exemple `fourier-otf`) redéfinissent les fontes générées par le package `amssymb` et peuvent provoquer un « warning » au mieux, une erreur de compilation au pire.

Pour cela, on pourra appeler `ProfLycee` avec l'option `<nonamssymb>` (idée reprise de `ProfCollege`).

Pour éviter des *warnings* liés au chargement de `unicode-math`, il est possible d'utiliser l'option `<warning-soff>`.

Code $\text{\LaTeX}$

```
\documentclass{article}
\usepackage[nonamssymb,warningsoff]{ProfLycee} %ou \usepackage[compat]{ProfLycee}
\usepackage{fourier-otf}
```

1.5 Gestion de siunitx



3.11b Le package `siunitx` est chargé et utilisé par `ProfLycee`, avec les options *françaises*.

Pour éviter de charger les options *françaises* (et ainsi utiliser d'autres paramétrage(s)), il est possible d'utiliser l'option `<nonsiunitxfr>`.

Code $\text{\LaTeX}$

```
\documentclass{article}
\usepackage[nonsiunitxfr]{ProfLycee}
\sisetup{<options>}
```

1.6 Gestion de babel



3.10a Pour des raisons de compatibilité, il est possible de ne pas charger la librairie `TikZ` `babel`. Il est toutefois conseillé de la charger, et ce pour certaines commandes liées à `TikZ`.

Code $\text{\LaTeX}$

```
\usepackage[nonbabel]{ProfLycee}
```

1.7 Gestion de fontawesome



3.12d En prévision de la sortie de fontawesome6/7, il sera possible (c'est en pré-test) de spécifier la version de fontawesome à utiliser :

- fontawesome5 (par défaut);
- fontawesome6;
- fontawesome7.

Code $\LaTeX$

```
\usepackage{ProfLycee}           %fa5 par défaut
\usepackage[nonfa]{ProfLycee}    %sans fa
\usepackage[fa6]{ProfLycee}      %fa6
\usepackage[fa7]{ProfLycee}      %fa7
```

1.8 Gestion de cancel



3.11c Pour des raisons de compatibilité, il est possible de ne pas charger le package cancel (avec l'option [thicklines]).
Seule une commande liée à l'arithmétique (les équations diophantiennes) l'utilise.

Code $\LaTeX$

```
\usepackage[noncancel]{ProfLycee}
```

2 Compléments

2.1 Support multilingues



L'idée est de proposer des options (*experimental*) pour modifier la langue de ProfLycee pour les quelques commandes à rédaction.

À noter qu'une macro dédiée au *changement local* de langue existe, et elle peut être appelée à tout moment dans le document.

Code $\LaTeX$

```
%changement global de la langue
\usepackage[french]{ProfLycee}    %défaut
\usepackage[english]{ProfLycee}
\usepackage[german]{ProfLycee}
\usepackage[spanish]{ProfLycee}
```

Code $\LaTeX$

```
%changement local de la langue
\setproflyc{fr/en/de/es}
```



Il est également possible de modifier le style des symboles de comparaison, qui sont définis de manière globale dans ProfLycee.

Code $\LaTeX$

```
%comportement global/local geq(slant) + leq(slant) par défaut
\def\pflgeq{\geqslant}%
\def\pflleq{\leqslant}%

%comportement global/local geq(slant) + leq(slant) modifié
\def\pflgeq{\geq}%
\def\pflleq{\leq}%
```



Un travail d'*anglicisation* des clés, commandes et environnements est en cours.

À terme, il est donc prévu que :

- les commandes soient proposées avec des alias [fr] et [en] sous la forme `\pfl<macro>`;
- les clés soient proposées en version [fr] et [en];
- tout soit interchangeable.

C'est pour le moment en projet, mais déjà implémenté dans une majorité de commandes, et une documentation spécifique et plus concise sera proposée en marge de celle-ci.

2.2 Le système de « clés/options »



L'idée est de conserver – autant que faire se peut – l'idée de **Clés** qui sont :

- modifiables;
- définies (en majorité) par défaut pour chaque commande.

Pour certaines commandes, le système de **Clés** pose quelques soucis, de ce fait le fonctionnement est plus *basique* avec un système d'arguments *optionnels* (souvent entre [...]) ou *obligatoires* (souvent entre {...}).

À noter que les :

- les **Clés** peuvent être mises dans n'importe quel ordre, elles peuvent être omises lorsque la valeur par défaut est conservée;
- les arguments doivent, eux, être positionnés dans le *bon ordre*.

2.3 Compilateur(s)



Le package `ProfLycee` est compatible avec les compilateurs classiques : latex, pdflatex ou encore lualatex. En ce qui concerne les codes librairies, il faudra :

- `python` : compiler en Lua^{La}TeX (et shell-escape ou write18 avec pyluatex).

2.4 Problèmes éventuels...



Certaines commandes sont à intégrer dans un environnement TikZ, afin de pouvoir rajouter des éléments, elles ont été testées dans des environnement `tikzpicture`, à vérifier que la gestion des axes par l'environnement `axis` est compatible...

En dehors de cela, ce sont des tests multiples et variés qui permettront de détecter d'éventuels bugs!

2.5 Version allégée



En marge de la version *complète* du package, une version *allégée* est disponible!

Depuis la version `4.02a`, `ProfLycee` n'est plus qu'un point d'entrée qui charge `ProfLycee-Light` puis appelle `\useproflyclib{most}` : les deux partagent le même moteur, la seule différence est que la version *allégée* ne charge **aucun** module par défaut : c'est à l'utilisateur de choisir, via `\useproflyclib`, ceux dont il a besoin.



L'idée est de laisser l'utilisateur charger, de manière individuelle, les sous-packages. Les options du package `ProfLycee` sont disponibles également pour la version allégée!

Code $\text{\LaTeX}$

```
%seules les commandes de calculs formatés sont disponibles
\usepackage[options]{ProfLycee-Light}
```



Les modules à importer sont à donner grâce à la commande `\useproflyclib`.

Code $\text{\LaTeX}$

```
%bibliothèques individuelles, ou directement via \useproflyclib{most}
\useproflyclib{analyse}
\useproflyclib{graphiques}
\useproflyclib{listings}
\useproflyclib{trigo}
\useproflyclib{probas}
\useproflyclib{stats}
\useproflyclib{arithm}
\useproflyclib{aleatoire}
\useproflyclib{suites}
\useproflyclib{geom}
\useproflyclib{cliparts}
```

Code $\text{\LaTeX}$

```
%bibliothèques indépendantes
\useproflyclib{piton}
\useproflyclib{espace}
\useproflyclib{ecritures}
\useproflyclib{complexes}
\useproflyclib{recreat}
\useproflyclib{competences}
\useproflyclib{exams}
\useproflyclib{cas}
\useproflyclib{tikz2d}
\useproflyclib{tikz3d}
```

2.6 Alias de commandes (en test)



3.11a Certaines commandes de $\text{\LaTeX}$ ProfLycee disposent d'un alias sous la forme $\text{\LaTeX}$ \pf1..., pour rappeler rapidement qu'elles appartiennent à $\text{\LaTeX}$ ProfLycee (conseil fait par JFB). Ce n'est pour l'instant le cas que d'une petite partie des commandes du package : une généralisation progressive de ce préfixe est envisagée.

↔ Bonne(s) découverte(s) ↔

Thème

LISTE DES COMMANDES

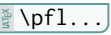
Deuxième partie

Liste des commandes, par thème



 Cette section contient un *résumé* des différentes commandes et environnements disponibles dans ProfLycee.

Elles sont présentées de manière *succincte*, mais elles sont présentées de manière *détaillée* dans la suite de la documentation.

Autant que peut se faire, la version *alias* en  est mentionnée, en sachant qu'elle reprend la syntaxe complète de la commande initiale.

 Cette section est désormais divisée par thème (voir les signets du PDF, ou la table des matières) pour une navigation plus rapide.

2.7 Calculs formatés, conversions et intervalles

Code $\text{\LaTeX}$

```
%calculs formatés
\pflnum(*)<type>[dec]{calcul} %version consolidée
\pflnumsqrt[formatage]{intérieur de la racine}
\pflnumfrac(*)[formatage]{calcul}
\pflnumrad(*)[formatage]{calcul}
```

Code $\text{\LaTeX}$

```
%conversion directe entre bases
\pflbasetobase{base initiale}{base finale}{nb}
```

Code $\text{\LaTeX}$

```
%formatage spéciaux avec sinuitx
\pflprix[arrondi]{px}
\pflangle[arrondi]{agl}
\pflpcent[arrondi]{pcent}
```

Code $\text{\LaTeX}$

```
%intervalles
\begin{RepIntervalles}[clés]<options tikz>...\end{RepIntervalles}
\tkzIntervalle[clés]{xmin}{xmax}
```

2.8 Analyse

Code $\text{\LaTeX}$

```
%Résolution approchée d'une équation f(x)=k
\ResolutionApprochee[clés]{équation}[macro]
\pflresolapproch
```

Code $\text{\LaTeX}$

```
%Présentation d'une solution par balayage (TVI)
\SolutionTVI[options]{fonction}{valeur}
\pflsoltvi
```

Code $\text{\LaTeX}$

```
%Calculer le terme d'une suite récurrente simple, toile pour une suite récurrente simple
\CalculTermeRecurrence[options]{fonction associée}
\pflcalcrecurr
\ToileRecurrence[clés][options du tracé][option supplémentaire des termes]
\pfltoilerecurr
```

Code $\LaTeX$

```
%Mise en forme de la conclusion d'un seuil
\SolutionSeuil[options]{fonction associée}{seuil}
\pflsolseuil
```

Code $\LaTeX$

```
%Valeur approchée d'une intégrale
\IntegraleApprochee[clés]{fonction}{a}{b}
\pflintegrapproch
\CalcIntegrale(*)[prec]<nb subdiv>{expr version xint}{binf}{bsup}
\pflvalintegr
```

Code $\LaTeX$

```
%Valeur moyenne d'une fonction
\ValeurMoyenneIntg(*)[prec]<nb subdiv>{expr version xint}{binf}{bsup}
\pflvalmoyintegr
```

Code $\LaTeX$

```
%forme canonique
\FormeCanonique(*)[option]{a}{b}{c}
\pflformcanoniq
```

Code $\LaTeX$

```
%décomposition d'une fonction homographique
\FonctionHomographique(*){a}{b}{c}{d}
\pflfcthomogr
```

Code $\LaTeX$

```
%max/min d'une fonction
\DetermineMax[précision]{expression xint}{a}{b}[\tmpmax][\tmpmaxvalx]
\pfldetmax
\DetermineMin[précision]{expression xint}{a}{b}[\tmpmin][\tmpminvalx]
\pfldetmin
```

2.9 Schémas de signe et compléments tkz-tab

Code $\LaTeX$

```
%schémas pour le signe affine/trinôme, dans un environnement tikz
\MiniSchemaSignes(*)[clés]<options tikz>
\pflschemasignes
\MiniSchemaSignesTkzTab[options]{numligne}[échelle][décalage horizontal]
\pflschemasignestkztab
```

2.10 Présentation de code

Code $\LaTeX$

```
%présentation de code Python
\begin{CodePythonLst}(*)[clés]{commandes tcbbox}...\end{CodePythonLst}
\begin{CodePythonLstAlt}(*)[clés]{commandes tcbbox}...\end{CodePythonLstAlt}
%:=bibliothèque python
\begin{CodePiton}[options piton]{commandes tcbbox}<option line-numbers>...\end{CodePiton}
\begin{PitonConsole}<clés>{commandes tcbbox}...\end{PitonConsole}
%:=bibliothèque pythontex
\begin{CodePythontex}[clés]{commandes tcbbox}...\end{CodePythontex}
\begin{CodePythontexAlt}[clés]{commandes tcbbox}...\end{CodePythontexAlt}
\begin{ConsolePythontex}[options]{...}\end{ConsolePythontex}
%présentation de pseudocode
\begin{PseudoCode}(*)[clés]{commandes tcbbox}...\end{PseudoCode}
\begin{PseudoCodeAlt}(*)[largeur][clés]{commandes tcbbox}...\end{PseudoCodeAlt}
\begin{PseudoCodePiton}[clés]{commandes tcbbox}<option line-numbers>...\end{PseudoCodePiton}

%style Thonny
\begin{PitonThonnyEditor}<clés>[options tcbbox]{largeur}...\end{PitonThonnyEditor}
\begin{PitonThonnyConsole}<clés>[options tcbbox]{largeur}...\end{PitonThonnyConsole}

\CartoucheCapytale(*)[options]{code capytale}
```

Code $\LaTeX$

```
%présentation de codes via piton (insertion de fichiers)
\CodePitonFichier[clés]{fichier}<option line-numbers>
\PseudoCodePitonFichier[clés]{fichier}<option line-numbers>
\PitonThonnyEditorFichier<clés>[largeur]{fichier}
```

2.11 Géométrie (plan et espace)

Code $\LaTeX$

```
%pavé et tétraèdre, dans un environnement tikz
\PaveTikz[options]
\TetraedreTikz[options]
```

Code $\LaTeX$

```
%cercle trigo, dans un environnement tikz
\CercleTrigo[clés]
\pflcercletrigo
```

Code $\LaTeX$

```
%Affichage des coordonnées d'un point (2 ou 3 coordonnées)
\AffPoint[options de formatage](liste des coordonnées)
%Affichage des coordonnées d'un vecteur (2 ou 3 coordonnées)
\AffVecteur[options de formatage]<options nicematrix>(liste des coordonnées)
```

Code $\LaTeX$

```
%Avec un vecteur normal et un point
\TrouveEqCartPlan[clés](vecteur normal)(point)
%Avec deux vecteurs directeurs et un point
\TrouveEqCartPlan[clés](vecteur dir1)(vecteur dir2)(point)
%Avec trois points
\TrouveEqCartPlan[clés](point1)(point2)(point3)
```

Code $\LaTeX$

```
%Avec un vecteur directeur et un point
\TrouveEqParamDroite[clés](vecteur directeur)(point)
%Avec deux points
\TrouveEqParamDroite[clés](point1)(point2)
```

Code $\LaTeX$

```
%Avec un vecteur normal (choix par défaut) et un point
\TrouveEqCartDroite[clés](vecteur normal)(point)
%Avec un vecteur directeur et un point
\TrouveEqCartDroite[clés,VectDirecteur](vecteur directeur)(point1)
%Avec deux points
\TrouveEqCartDroite[clés](point1)(point2)
```

Code $\LaTeX$

```
%Avec le point et le plan via vect normal + point
\TrouveDistancePtPlan(point)(vec normal du plan)(point du plan)
%Avec le point et le plan via vect normal + point
\TrouveDistancePtPlan(point)(équation cartésienne)
```

Code $\LaTeX$

```
\TrouveEqSphere[clés](centre ou point n°1)(rayon ou point ou éq plan)
```

Code $\LaTeX$

```
%Avec le vecteur
\TrouveNorme(vecteur)
%Avec deux points
\TrouveNorme(point 1)(point 2)
```

Code $\LaTeX$

```
%Équation réduite d'une droite
\EquationReduite[option]{A/xa/ya,B/xb/yb}
\pflqreduite
```

Code $\LaTeX$

```
%Schémas de cours de géométrie dans l'espace
\SchemaEspace(*)[clés]{type}
\pflschemesp
```

2.12 Statistiques

Code $\LaTeX$

```
%paramètres d'une régression linéaire, nuage de points
\CalculsRegLin[clés]{listeX}{listeY}
\PointsRegLin[clés]{listeX}{listeY}
\pflcalcreglin
```

Code $\LaTeX$

```
%stats à 2 variables, dans un environnement tikz
\GrilleTikz[options][options grille ppale][options grille second.]
\AxesTikz[options]
\AxexTikz[options]{valeurs} \AxyTikz[options]{valeurs}
\FenetreTikz \OrigineTikz
\FenetreSimpleTikz[options](opt axes)<opt axe Ox>{liste valx}<opt axe Oy>{liste valy}
\NuagePointsTikz[options]{listeX}{listeY}
\PointMoyenTikz[options]
\CourbeTikz[options]{formule}{domaine}
```

Code $\LaTeX$

```
%paramètres d'une série discrète
\DeterminerParamStats(*){liste}[\macromin]...[\macromax]
\pflparamstats
```

Code $\LaTeX$

```
%boîte à moustaches, dans un environnement tikz
\BoiteMoustaches[options]
\BoiteMoustachesAxe[options]
\pflboitemoustach
```

Code $\LaTeX$

```
%histogrammes
\Histogramme(*)[options]{données}
\pflhistogramme
```

Code $\LaTeX$

```
%courbe ECC/FCC
\CourbeECC[clés]{liste valeurs}{liste effectifs}
\pflcourbeecc

\begin{EnvCourbeECC}[clés]{liste valeurs}{liste effectifs}...\end{EnvCourbeECC}
\MedianeQuartilesECC{liste valeurs}{liste effectifs}
```

2.13 Probabilités

Code $\LaTeX$

```
%loi binomiale B(n,p)
\CalcBinomP{n}{p}{k}
\CalcBinomC{n}{p}{a}{b}
\BinomP(*)[prec]{n}{p}{k}
\BinomC(*)[prec]{n}{p}{a}{b}
\pflbinomp
\pflbinomc
```

Code $\LaTeX$

```
%loi de Poisson P(l)
\CalcPoissP{l}{k}
\CalcPoissC{l}{a}{b}
\PoissonP(*)[prec]{l}{k}
\PoissonC(*)[prec]{l}{a}{b}
\pflpoissonp
\pflpoissonc
```

Code $\LaTeX$

```
%loi géométrique G(p)
\CalcGeomP{p}{k}
\CalcGeomC{l}{a}{b}
\GeomP{p}{k}
\GeomC{l}{a}{b}
\pflgeomc
\pflgeomc
```

Code $\LaTeX$

```
%loi hypergéométrique H(N,n,m)
\CalcHypergeomP{N}{n}{m}{k}
\CalcHypergeomC{N}{n}{m}{a}{b}
\HypergeomP{N}{n}{m}{k}
\HypergeomC{N}{n}{m}{a}{b}
\pflhypergeomc
\pflhypergeomc
```

Code $\LaTeX$

```
%loi normale N(m,s)
\CalcNormC{m}{s}{a}{b}
\NormaleC(*)[prec]{m}{s}{a}{b}
\pflnormalec
```

Code $\LaTeX$

```
%loi exponentielle E(1)
\CalcExpoC{1}{a}{b}
\ExpoC(*)[prec]{1}{a}{b}
\pflexpoc
```

Code $\LaTeX$

```
%arbres de probas
\ArbreProbasTikz[options]<opts tikz>{donnees}
\pflarbreprobas

\begin{EnvArbreProbasTikz}[options]<opts tikz>{donnees}...\end{EnvArbreProbasTikz}
```

Code $\LaTeX$

```
%schémas lois continues
\LoiNormaleGraphe[options]<options tikz>{m}{s}{a}{b}
\LoiExpoGraphe[options]<options tikz>{l}{a}{b}
\pflnormgraph
\pflexpograph
```

Code $\LaTeX$

```
%fonction de répartition discrète, dans un environnement tikz
\FonctionRepartTikz[clés]{probas,borneinf,bornesup / probas,borneinf,bornesup / ...}
```

Code $\LaTeX$

```
%binomiale
\HistogrammeBinomiale[clés]{n}{p}
\pflhistobinom

\begin{HistoBinomiale}[clés]{n}{p}...\end{HistoBinomiale}

\TraceLoiNormale(*)[clés]<plage>{param1}{param2}
```

2.14 Aléatoire

Code $\LaTeX$

```
%entier aléatoire entre a et b
\NbAlea{a}{b}{macro}
\pflnbalea
```

Code $\LaTeX$

```
%nombre décimal (n chiffres après la virgule) aléatoire entre a et b+1 (exclus)
\NbAlea[n]{a}{b}{macro}
```

Code $\LaTeX$

```
%création d'un nombre aléatoire sous forme d'une macro
\VarNbAlea{macro}{calcul}
\pflvarnbalea
```

Code $\LaTeX$

```
%liste d'entiers aléatoires
\TirageAleatoireEntiers[options]{macro}
\pfltiragealeaent
```

2.15 Arithmétique et dénombrement

Code $\LaTeX$

```
%liste des diviseurs  
\ListeDiviseurs(*)[option]{nombre}  
\pflistediv
```

Code $\LaTeX$

```
%arbre des diviseurs  
\ArbreDiviseurs[options]{nombre}  
\pflarbrediv
```

Code $\LaTeX$

```
%test booléen nb premier  
\pflboolestpremier{nb}  
\pfltestpremier[code si vrai][code si faux]{nb}
```

Code $\LaTeX$

```
%décomposition avec exposants en nombres premiers  
\pfldecompn(*){nb}
```

Code $\LaTeX$

```
%arbre "avec remise"  
\ArbreChoix[clés]<options tikzpicture>{liste}  
\pflarbrechoix
```

Code $\LaTeX$

```
%arbre "sans remise"  
\ArbreChoixSansRemise[clés]<options tikzpicture>{liste}  
\pflarbrechoixssremise
```

Code $\LaTeX$

```
%factorielle  
\Factorielle(*)[clés]{n}  
\pflfacto  
  
%primorielle  
\Primorielle(*)[clés]{n}  
\pflprimor
```

Code $\LaTeX$

```
%arrangement Anp  
\Arrangement(*)[option]{p}{n}  
\pflarrang
```

Code $\LaTeX$

```
%arrangement Cnp (p parmi n)  
\Combinaison(*)[option]{p}{n}  
\pflcombin
```

Code $\LaTeX$

```
%conversions
\ConversionEntreBases(*)[espace blocs 4 pour le binaire]{dep->arrivée}{nb}
\pflconvbases

\ConversionDecBin(*)[clés]{nombre}
\pflconvdecbin

\ConversionBinHex[clés]{nombre}
\pflconvbinhex

\ConversionHexBin[clés]{nombre}
\pflconvhexbin

\ConversionVersDec[clés]{nombre}
\pflconvversdec

\ConversionBaseDix[clés]{nombre}{base de départ}
\pflconvdix

\ConversionDepuisBaseDix[options]{nombre en base 10}{base d'arrivée}
\pflconvdepuisdix
```

Code $\LaTeX$

```
%PGCD présenté
\PresentationPGCD[options]{a}{b}
\pflprespgcd
```

Code $\LaTeX$

```
%Opérations posées
\OperationPosee[clés]{opération}
\pfllopeposee
```

Code $\LaTeX$

```
%Équation diophantienne
\EquationDiophantienne[clés]{equation}
\pflquadioph
```

Code $\LaTeX$

```
%division euclidienne présentée
\DivEucl(*)[clés]{a}{b}
\pfldivideucl

%division euclidienne 'brute'
\DivisionEucl{a}{b}
\pflpresdivideucl
```

Code $\LaTeX$

```
%reste et quotient modulo de a par n
\QuoMod(*){a}{n}
\pflquomod

\ResteMod(*){a}{n}
\pflrestemod
```

Code $\LaTeX$

```
%inverse modulo
\InverseModulo(*){a}{modulo}
\pflinvmod
```

Code $\LaTeX$

```
%Chiffrement de César
\ChiffrementCesar[clés]{message}
\pflchiffrcesar
```

Code $\LaTeX$

```
%Chiffrement affine  
\ChiffrementAffine[clés]{message}  
\pflchiffraff
```

Code $\LaTeX$

```
%Chiffrement de Hill  
\ChiffrementHill[clés]{message}  
\pflchiffrhill
```

2.16 Écritures, ensembles et trinômes

Code $\LaTeX$

```
%conversion en fraction, simplification de racine  
\ConversionFraction(*)[option]{argument}  
\SimplificationRacine{expression}
```

Code $\LaTeX$

```
%ensemble d'éléments  
\EcritureEnsemble[clés]{liste}  
\pflecritens
```

Code $\LaTeX$

```
%trinôme, trinôme aléatoire  
\EcritureTrinome[options]{a}{b}{c}  
\pflecrittrinom
```

2.17 Trigonométrie

Code $\LaTeX$

```
%mesure principale  
\MesurePrincipale[options]{angle}  
\pflmesppale  
  
%lignes trigo  
\LigneTrigo(*)[booléens]{cos/sin/tan}(angle)  
\pflignetrig
```

2.18 Compétences

Code $\LaTeX$

```
%compétences LEGT  
\CompMathsLyc[clés]{choix}  
%compétences LPRO  
\CompMathsLycPro[clés]{choix}
```

2.19 Récréations mathématiques

Code $\LaTeX$

```
%sudomaths  
\SudoMaths[options]{liste}  
\pflsudomaths  
  
\begin{EnvSudoMaths}[options]{grille}...\end{EnvSudoMaths}
```

Code $\LaTeX$

```
%fractales
\FractaleTikz(*)[clés]<options tikz>
\pflfractaltikz

\EtapesFloconKoch[clés]{étapes}
\EtapesTapisSierpinski[clés]{étapes}
```

Code $\LaTeX$

```
%chateau de cartes
\ChateauCartes[clés]{niveau}<options tikz>
\pflchateaucartes
```

Code $\LaTeX$

```
%allumettes, dans un environnement tikz
\PFLAllumettes[clés]{liste départs>arrivées}
\pflallumettes
```

Code $\LaTeX$

```
%commandes sur les dates
\JourSelonDate(*){jj/mm/aaaa}
\pfljoursepondate

\DateComplete(*){jj/mm/aaaa}
\pfldatecomplet

\NbJoursEntreDates(*){jj/mm/aaaa}{jj/mm/aaaa}[\macro]
\pflnbjoursdates

\AjoutJoursDate(*){jj/mm/aaaa}{nb}
\pflajoutjoursdate
```

Code $\LaTeX$

```
%machine à transformer
\MachineTransformer[clés]{e/s}<options tikz>
\pflmachtransf
```

2.20 Cliparts et logos

Code $\LaTeX$

```
%insertion de clipart, en mode inline ou non
\InsererClipart*[options includegraphics]{clipart}
\pflclipart
```

Code $\LaTeX$

```
%logo sans calculatrice
\pflnocalc(*)[couleur]
```

Thème

ÉCRITURES MATHÉMATIQUES

Troisième partie

Écritures d'objets mathématiques

3 Commande spécifique de calcul/formatage



3.03b L'idée est de proposer une commande pour cumuler les possibilités des packages `xint` et `sinuitx` pour automatiser et formater des calculs.

3.10a IL est également possible de spécifier des formats particuliers (fraction, racine, radians).

- la version étoilée force le calcul en flottant;
- l'argument optionnel, entre `[...]` permet de paramétrer l'arrondi en flottant;
- l'argument obligatoire, entre `{...}`, est quant à lui le calcul en langage `xint`.

Code $\text{\LaTeX}$

```
\pflnum{2^12+1}
\pflnum*{4096/(12+1)}
\pflnum*[4]{4096/(12+1)}
\pflnum*[5]{sqrt(1+exp(9.75))}
\pflnumfrac{74+1/5} \pflnumfrac[d]{74+1/5}
\pflnumrad{1250pi/4589} ou \pflnumrad[d]{-9pi/4} ou \pflnumrad*[d]{-9pi/4}
\pflnumsqrt[d]{0.45}
```

4 097
315,076 923 076 923 1
315,076 9
130,977 97
 $\frac{371}{5}$ $\frac{371}{5}$
 $\frac{1250\pi}{4589}$ ou $\frac{-9\pi}{4}$ ou $-\frac{9\pi}{4}$
 $\frac{3\sqrt{5}}{10}$



3.05a L'idée est de proposer une commande *interne* pour convertir d'une base à une autre, en accord avec les spécificités de conversions en $\text{\LaTeX}$ 3.

La commande ne s'occupe *que* de convertir d'une base à une autre, sans formatage ou autre.

Code $\text{\LaTeX}$

```
%\pflbasetobase{base initiale}{base finale}{nb}
\pflbasetobase{10}{2}{27}\\ %conversion en binaire
\pflbasetobase{10}{16}{456}\\ %conversion en hexa
\pflbasetobase{2}{16}{10011111}\\ %conversion en hexa
\pflbasetobase{5}{10}{2043}\\ %conversion b5-b10
\pflbasetobase{10}{5}{273}\\ %conversion b10-b5
\pflbasetobase{16}{7}{AB5F} %conversion b16-b7
```

11011
1C8
9F
273
2043
241622

4 Commandes



Les commandes de cette section sont disponibles en chargeant la librairie `ecritures`, car elles peuvent redéfinir des commandes personnelles déjà existantes!

4.1 Arrondi



Il est possible de calculer/arrondir/formater un calcul mathématique, grâce aux packages `siunitx` et `xint`.

- la version étoilée force l’affichage du « + » pour les nombres positifs;
- l’argument optionnel est la précision demandée (3 par défaut)
- l’argument obligatoire est le calcul, au langage `xint`.

Code $\text{\LaTeX}$

```
$1+\dfrac{7}{11} \approx \text{\Arrondi{1+7/11}} \approx \text{\Arrondi[5]{1+7/11}} \approx \text{\Arrondi*[2]{1+7/11}}$  
$\ln\big(1+\text{e}^4\big) \approx \text{\Arrondi[6]{log(1+exp(4))}}$
```

$1 + \frac{7}{11} \approx 1,636 \approx 1,636\,36 \approx +1,64$
 $\ln(1 + e^4) \approx 4,018\,150$

4.2 Unités simplifiées



Les commandes suivantes permettent de saisir des quantités avec unités classiques (€/°/%). Elles sont compatibles avec le mode `math` et les espacements sont gérés en interne par le package et par `siunitx`.
L’argument optionnel permet de préciser un arrondi pour l’affichage.

Code $\text{\LaTeX}$

```
On a un prix de  $\text{\pflprix{1199.99}}$  ou  $\text{\pflprix[1]{1199.99}}$ .\\  
L’angle mesure  $\text{\pflangle{164.46}}$  ou  $\text{\pflangle[1]{164.46}}$ .\\  
L’évolution est de  $\text{\pflpcent{47.567}}$  ou  $\text{\pflpcent[2]{47.567}}$  ou  $\text{\pflpcent*[2]{47.567}}$  (avec le signe  
↪ \log + \fg)
```

On a un prix de 1 199,99 € ou 1200,0 €.
L’angle mesure 164,46° ou 164,5°.
L’évolution est de 47,567 % ou 47,57 % ou +47,57 % (avec le signe « + »)

4.3 Ensembles et intervalles



Les commandes suivantes permettent de composer les ensembles traditionnels, en mode *tableau noir*. Elles sont dans un bloc `ensurmath`, donc les `$. . $` ne sont pas nécessaires.

À noter la macro particulière pour l’ensemble des quaternions, pour éviter des erreurs éventuelles de redéfinition.

Code $\text{\LaTeX}$

```
\N, \Z, \D, \Q, \R, \C, \ensH, \N*, \Z*, \D*, \Q*, \R*, \C*, \ensH*
```

$\text{N, Z, D, Q, R, C, H, N}^*, \text{Z}^*, \text{D}^*, \text{Q}^*, \text{R}^*, \text{C}^*, \text{H}^*$



Des intervalles peuvent être composés, grâce à la commande `\IntervalleXX`, dont la base est le package `interval` :

- la commande est insérée dans un bloc `ensuremath` ;
- le séparateur est le point-virgule ;
- l'espacement autour du point-virgule est laissé aux réglages du document (`\babel`, `frenchmath`, etc).

Code $\LaTeX$

```
\IntervalleF0{0}{+\infty} ou \Intervalle00{0}{+\infty} ou \Intervalle00{-\infty}{0} ou
↪ \Intervalle0F{-\infty}{0}
```

$[0; +\infty[$ ou $]0; +\infty[$ ou $]-\infty; 0[$ ou $]-\infty; 0]$

Code $\LaTeX$

```
\IntervalleFF{\frac{12}{\sqrt{26}}} ou \Intervalle00{\frac{12}{\sqrt{26}}} ou
\IntervalleF0{\dfrac{12}{\sqrt{26}}} ou \Intervalle0F{\dfrac{12}{\sqrt{26}}}
```

$\left[\frac{1}{2}; \sqrt{26}\right]$ ou $\left]\frac{1}{2}; \sqrt{26}\right[$ ou $\left[\frac{1}{2}; \sqrt{26}\right[$ ou $\left]\frac{1}{2}; \sqrt{26}\right]$

4.4 Repères et coordonnées



Des vecteurs/repères/coordonnées peuvent être composées :

- les vecteurs sont mis en forme par le package `esvect` (y compris en version étoilée pour les indices) ;
- des coordonnées de point/vecteur dans le plan ou l'espace ;
- des repères génériques, avec choix du séparateur entre le point et les vecteurs ;
- les versions étoilées des repères n'alignent pas les flèches.

Code $\LaTeX$

```
%vecteurs
\Vecteur{\imath} et \Vecteur{u} et \Vecteur{AB}

$\Vecteur{AB}+\Vecteur{BC} = \Vecteur{AC}$

$\Vecteur*{u}[1]+\Vecteur*{v}[2] = \Vecteur*{w}[3]$
```

$\vec{I}$ et $\vec{u}$ et $\vec{AB}$
 $\vec{AB} + \vec{BC} = \vec{AC}$
 $\vec{u}_1 + \vec{v}_2 = \vec{w}_3$

Code $\LaTeX$

```
%coordonnées points
\CoordPtP1{4}{-2} ou \CoordPtP1{\frac{12}{\sqrt{26}}}{\frac{47}{\sqrt{26}}}

\CoordPtEsp{4}{-2}{7} ou \CoordPtEsp{-2}{\frac{12}{\sqrt{26}}}{\frac{47}{\sqrt{26}}}
```

$(4; -2)$ ou $\left(\frac{1}{2}; \frac{4}{7}\right)$
 $(4; -2; 7)$ ou $\left(-2; \frac{1}{2}; \frac{4}{7}\right)$

Code $\LaTeX$

%coordonnées vecteurs

$\backslash\text{CoordVecPl}\{4\}{-2}$ ou $\backslash\text{CoordVecPl}\{\frac{12}{7}\}\{\frac{47}{7}\}$

$\backslash\text{CoordVecEsp}\{4\}{-2}\{7\}$ ou $\backslash\text{CoordVecEsp}\{-2\}\{\frac{12}{7}\}\{\frac{47}{7}\}$

%versions nicematrix, avec paramètre cell-space-limits

$\backslash\text{CoordVecPl}[1pt]\{4\}{-2}$ ou $\backslash\text{CoordVecPl}[1pt]\{\frac{12}{7}\}\{\frac{47}{7}\}$

$\backslash\text{CoordVecEsp}[1pt]\{4\}{-2}\{7\}$ ou $\backslash\text{CoordVecEsp}[1pt]\{-2\}\{\frac{12}{7}\}\{\frac{47}{7}\}$

%versions +pmatrix, compatible tblr

$\backslash\text{CoordVecPlTblr}\{4\}{-2}$ ou $\backslash\text{CoordVecPlTblr}\{\frac{12}{7}\}\{\frac{47}{7}\}$

$\backslash\text{CoordVecEspTblr}\{4\}{-2}\{7\}$ ou $\backslash\text{CoordVecEspTblr}\{-2\}\{\frac{12}{7}\}\{\frac{47}{7}\}$

$$\begin{pmatrix} 4 \\ -2 \end{pmatrix} \text{ ou } \begin{pmatrix} \frac{1}{2} \\ \frac{4}{7} \end{pmatrix}$$

$$\begin{pmatrix} 4 \\ -2 \\ 7 \end{pmatrix} \text{ ou } \begin{pmatrix} -2 \\ \frac{1}{2} \\ \frac{4}{7} \end{pmatrix}$$

$$\begin{pmatrix} 4 \\ -2 \end{pmatrix} \text{ ou } \begin{pmatrix} \frac{1}{2} \\ \frac{4}{7} \end{pmatrix}$$

$$\begin{pmatrix} 4 \\ -2 \\ 7 \end{pmatrix} \text{ ou } \begin{pmatrix} -2 \\ \frac{1}{2} \\ \frac{4}{7} \end{pmatrix}$$

$$\begin{pmatrix} 4 \\ -2 \end{pmatrix} \text{ ou } \begin{pmatrix} \frac{1}{2} \\ \frac{4}{7} \end{pmatrix}$$

$$\begin{pmatrix} 4 \\ -2 \\ 7 \end{pmatrix} \text{ ou } \begin{pmatrix} -2 \\ \frac{1}{2} \\ \frac{4}{7} \end{pmatrix}$$

Code $\LaTeX$

%matrices 2x2

$\backslash\text{MatDeux}\{4\}{-2}\{1\}{-5}$

$\backslash\text{MatDeux}\{\frac{47}{7}\}\{-1\}\{0\}\{-\frac{17}{7}\}$

%version nicematrix, avec paramètre cell-space-limits

$\backslash\text{MatDeux}[1pt]\{4\}{-2}\{1\}{-5}$

%versions +pmatrix, compatible tblr

$\backslash\text{MatDeuxTblr}\{4\}{-2}\{1\}{-5}$

$$\begin{pmatrix} 4 & -2 \\ 1 & -5 \end{pmatrix}$$

$$\begin{pmatrix} \frac{4}{7} & -1 \\ 0 & -\frac{1}{7} \end{pmatrix}$$

$$\begin{pmatrix} 4 & -2 \\ 1 & -5 \end{pmatrix}$$

$$\begin{pmatrix} 4 & -2 \\ 1 & -5 \end{pmatrix}$$

Code $\LaTeX$

%repères classiques

$\backslash\text{RepereOij}$ ou $\backslash\text{RepereOij*}$ ou $\backslash\text{RepereOij}[Sep=\{, \}]$ ou $\backslash\text{RepereOij*}[Sep=\{, \}]$

$\backslash\text{RepereOijk}$ ou $\backslash\text{RepereOijk*}$ ou $\backslash\text{RepereOijk}[Sep=\{, \}]$ ou $\backslash\text{RepereOijk*}[Sep=\{, \}]$

$\backslash\text{RepereOuv}$ ou $\backslash\text{RepereOuv*}$ ou $\backslash\text{RepereOuv}[Sep=\{, \}]$ ou $\backslash\text{RepereOuv*}[Sep=\{, \}]$

$(O; \vec{i}, \vec{j})$ ou $(O; \vec{i}, \vec{j})$ ou $(O, \vec{i}, \vec{j})$ ou $(O, \vec{i}, \vec{j})$
 $(O; \vec{i}, \vec{j}, \vec{k})$ ou $(O; \vec{i}, \vec{j}, \vec{k})$ ou $(O, \vec{i}, \vec{j}, \vec{k})$ ou $(O, \vec{i}, \vec{j}, \vec{k})$
 $(O; \vec{u}, \vec{v})$ ou $(O; \vec{u}, \vec{v})$ ou $(O, \vec{u}, \vec{v})$ ou $(O, \vec{u}, \vec{v})$

Code $\LaTeX$

%repères personnalisés

$\backslash\text{ReperePlan}\{A\}\{AB\}\{AC\}$ ou $\backslash\text{ReperePlan*}\{A\}\{AB\}\{AC\}$ ou $\backslash\text{ReperePlan*}\{O\}\{OI\}\{OJ\}$ ou
 $\backslash\text{ReperePlan}[Sep=\{, \}]\{O\}\{OI\}\{OK\}$

$\backslash\text{RepereEspace}\{A\}\{AB\}\{AC\}\{AD\}$ ou $\backslash\text{RepereEspace*}\{A\}\{AB\}\{AC\}\{k\}$ ou $\backslash\text{RepereEspace}\{D\}\{i\}\{j\}\{k\}$

$(A; \overrightarrow{AB}, \overrightarrow{AC})$ ou $(A; \overrightarrow{AB}, \overrightarrow{AC})$ ou $(O; \overrightarrow{OI}, \overrightarrow{OJ})$ ou $(O, \overrightarrow{OI}, \overrightarrow{OK})$
 $(A; \overrightarrow{AB}, \overrightarrow{AC}, \overrightarrow{AD})$ ou $(A; \overrightarrow{AB}, \overrightarrow{AC}, \vec{k})$ ou $(D; \vec{i}, \vec{j}, \vec{k})$

4.5 Valeur absolue/module et norme



Des valeurs absolues (ou modules) ou des normes peuvent être saisies :

- les délimiteurs s'adaptent à la hauteur de l'intérieur;
- les versions étoilées ne tiennent pas compte de la hauteur de l'intérieur.

Code $\LaTeX$

$\backslash\text{ValAbs}\{\backslash\text{dfrac}\{x+1\}\{x-1\}\}$ ou $\backslash\text{ValAbs*}\{\backslash\text{dfrac}\{x+1\}\{x-1\}\}$

$\backslash\text{Norme}\{\backslash\text{Vecteur}\{AB\}\}$ ou $\backslash\text{Norme*}\{\backslash\text{Vecteur}\{AB\}\}$

$\backslash\text{ModuleCplx}\{1+\backslash\text{dfrac}12\backslash\text{i}\}$

$\left| \frac{x+1}{x-1} \right|$ ou $\left| \frac{x+1}{x-1} \right|$
 $\| \overrightarrow{AB} \|$ ou $\| \overrightarrow{AB} \|$
 $\left| 1 + \frac{1}{2}i \right|$

4.6 Divers



La librairie `libecritures` permet également de définir des commandes pour :

- composer le nom d'une courbe;
- composer le « i » et le « e » en romain;
- composer le complexe j en mode algébrique ou exponentielle;
- composer un modulo avec choix de la congruence;
- composer une suite numérique;
- composer une intégrale (mode displaystyle);
- `\lim_{a \rightarrow b}` composer une limite.

À noter que les commandes sont (sauf mention contraire) dans un bloc ensuremath.

Code $\LaTeX$

```
%version normale := mathcal
%version étoilée := mathscr (si chargé)

\Courbe et \Courbe[f] et \Courbe[g^{-1}]
\Courbe* et \Courbe*[f] et \Courbe*[g^{-1}]
```

C et C_f et $C_{g^{-1}}$
 $\mathcal{C}$ et $\mathcal{C}_f$ et $\mathcal{C}_{g^{-1}}$

Code $\LaTeX$

```
 $\$e^{\{i \pi\}} = -1\$$ 
```

$e^{i\pi} = -1$

Code $\LaTeX$

```
On a  $\$j = \backslash jalg = \backslash jfexp\$$ 
```

On a $j = \frac{1}{2} + i\frac{\sqrt{3}}{2} = e^{i\frac{\pi}{3}}$

Code $\LaTeX$

```
%La version étoilée augmente l'espace, l'argument optionnel change la présentation
 $\$21 \backslash equiv 1 \backslash Modulo\{5\} \backslash equiv 1 \backslash Modulo[Par]\{5\} \backslash equiv 1 \backslash Modulo[Txt]\{5\} \$$ 
 $\$21 \backslash equiv 1 \backslash Modulo*\{5\} \backslash equiv 1 \backslash Modulo*[Par]\{5\} \backslash equiv 1 \backslash Modulo*[Txt]\{5\} \$$ 
```

$21 \equiv 1 \ [5] \equiv 1 \ (5) \equiv 1 \text{ modulo } 5$
 $21 \equiv 1 \ [5] \equiv 1 \ (5) \equiv 1 \text{ modulo } 5$

Code $\LaTeX$

```
Soient les suites \Suite{u} et \Suite[p]{v} et \Suite[q]{\Omega}.
```

Soient les suites (u_n) et (v_p) et (Ω_q) .

Code $\LaTeX$

```
 $\$I = \backslash Integrale f(x) \backslash dx = \backslash Integrale f(t) \backslash dx[t] \$$ 
```

$I = \int f(x) dx = \int f(t) dt$

Code $\LaTeX$

```
 $\$ \backslash Limite\{\dfrac{x^3-3x+1}{1+x^2}\}\{x \rightarrow +\infty\} = +\infty \$$ 
 $\$ \backslash Limite\{f(x)\}\{x \rightarrow 0\}[x > 0] = -\infty \$$ 
```

$\lim_{x \rightarrow +\infty} \frac{x^3 - 3x + 1}{1 + x^2} = +\infty$
 $\lim_{\substack{x \rightarrow 0 \\ x > 0}} f(x) = -\infty$

4.7 Probabilités



La librairie `us-ecritures` permet également de définir des commandes pour :

- composer une loi classique (binomiale, exponentielle, etc) avec `mathcal` ou `mathscr`;
- composer espérance, variance et écart-type;
- `us-3.02a` composer une probabilité conditionnelle, avec ou sans la formule et choix de la typographie de la proba.

À noter que les commandes sont dans un bloc `ensuremath`.

Code $\LaTeX$

```
%version normale := mathcal
%version étoilée := mathscr (si chargé)

\LoiNormale{150}{25} ou \LoiNormale*{150}{25}

\LoiBinomiale{150}{\num{0.45}} ou \LoiBinomiale*{150}{\num{0.45}}

\LoiPoisson{5} ou \LoiPoisson*{5}

\LoiExpo{\num{0.001}} ou \LoiExpo*{\num{0.001}}

\LoiUnif{\IntervalleFF{5}{60}} ou \LoiUnif*{\IntervalleFF{5}{60}}
```

$N(150; 25)$ ou $\mathcal{N}(150; 25)$
 $B(150; 0,45)$ ou $\mathcal{B}(150; 0,45)$
 P_5 ou $\mathcal{P}_5$
 $E_{0,001}$ ou $\mathcal{E}_{0,001}$
 $U_{[5;60]}$ ou $\mathcal{U}_{[5;60]}$

Code $\LaTeX$

```
%par défaut E et V en \mathbb
\Eesper{X} ou \Esper[E]{X} \\\
\Varianc{X^2} ou \Variance[V]{X^2} \\\
\EcType{X^2}
```

$\mathbb{E}(X)$ ou $E(X)$
 $\mathbb{V}(X^2)$ ou $V(X^2)$
 $\sigma(X^2)$

Code $\LaTeX$

```
%par défaut
$\ProbaCondit{A}{B}$ et $\ProbaCondit[Formule]{A}{B}$

$\ProbaCondit{\overline{A}}{\overline{B}}$ et $\ProbaCondit[Formule]{\overline{A}}{\overline{B}}$

%avec proba en minuscule
$\ProbaCondit[min]{A}{B}$ et $\ProbaCondit[min,Formule]{A}{B}$

%avec proba en \mathbb
$\ProbaCondit[BB]{A}{B}$ et $\ProbaCondit[BB,Formule]{A}{B}$
```

$$P_B(A) \text{ et } P_{\overline{B}}(A) = \frac{P(A \cap B)}{P(B)}$$

$$P_{\overline{B}}(\overline{A}) \text{ et } P_B(\overline{A}) = \frac{P(\overline{A} \cap \overline{B})}{P(\overline{B})}$$

$$p_B(A) \text{ et } p_{\overline{B}}(A) = \frac{p(A \cap B)}{p(B)}$$

$$\mathbb{P}_B(A) \text{ et } \mathbb{P}_{\overline{B}}(A) = \frac{\mathbb{P}(A \cap B)}{\mathbb{P}(B)}$$



Il est également possible de configurer *manuellement* sa *propre* notation pour la probabilité, grâce à `\renewcommand\notationproba{...}`.

4.8 Intervalle de confiance, intervalle de fluctuation



3.02a La librairie `ecritures` permet également de définir des commandes pour :

- composer un intervalle de fluctuation, avec ou sans détails et/ou calcul, et avec choix du *niveau* (2de ou terminale);
- composer un intervalle de confiance, avec ou sans détails et/ou calcul, et avec choix du *niveau* (2de ou terminale);
- composer une rédaction complète avec formule, détail et calcul.

Les niveaux de confiance possibles sont 90 %, 95 % et 99 %.

Code $\text{\LaTeX}$

%sorties par défaut

`\IntFluctu`

`\IntConf`

$$\left[p - 1,96 \frac{\sqrt{p(1-p)}}{\sqrt{n}} ; p + 1,96 \frac{\sqrt{p(1-p)}}{\sqrt{n}} \right]$$
$$\left[f - 1,96 \frac{\sqrt{f(1-f)}}{\sqrt{n}} ; f + 1,96 \frac{\sqrt{f(1-f)}}{\sqrt{n}} \right]$$

```
%visualisation des clés
$\IntFluctu[Details,p=0.85,n=1000]$

$\IntFluctu[Details,p=1/3,n=1000]$

$\IntFluctu[Calcul,p=0.85,n=1000]$

$\IntFluctu[Details,Calcul,p=0.3,n=50,Arrondi=3]$

$\IntFluctu[Seuil=99]$

$\IntFluctu[Details,Calcul,Classe=2de,p=0.3,n=10000,Symbole={},Arrondi=3]$

$\IntFluctu[Details,Calcul,Classe=2de,p=0.3,n=10001,Symbole=\subseteq,Arrondi=1]$

$\IntFluctu[Details,Calcul,Classe=2de,p=0.3,n=10001,Arrondi=2]$

$\IntFluctu[Details,Calcul,Classe=2de,p=0.3,n=123456,Arrondi=4]$

$\IntFluctu[Details,Calcul,Classe=2de,p=0.3,n=10,Arrondi=2]$

$\IntConf[Details,Calcul,Classe=Term,f=0.5,n=250,Arrondi=3]$
```

$$\left[p - 1,96 \frac{\sqrt{p(1-p)}}{\sqrt{n}} ; p + 1,96 \frac{\sqrt{p(1-p)}}{\sqrt{n}} \right] = \left[0,85 - 1,96 \frac{\sqrt{0,85(1-0,85)}}{\sqrt{1\,000}} ; 0,85 + 1,96 \frac{\sqrt{0,85(1-0,85)}}{\sqrt{1\,000}} \right]$$

$$\left[p - 1,96 \frac{\sqrt{p(1-p)}}{\sqrt{n}} ; p + 1,96 \frac{\sqrt{p(1-p)}}{\sqrt{n}} \right] = \left[\frac{1}{3} - 1,96 \frac{\sqrt{\frac{1}{3}\left(1-\frac{1}{3}\right)}}{\sqrt{1\,000}} ; \frac{1}{3} + 1,96 \frac{\sqrt{\frac{1}{3}\left(1-\frac{1}{3}\right)}}{\sqrt{1\,000}} \right]$$

$$\left[p - 1,96 \frac{\sqrt{p(1-p)}}{\sqrt{n}} ; p + 1,96 \frac{\sqrt{p(1-p)}}{\sqrt{n}} \right] \approx [0,82 ; 0,88]$$

$$\left[p - 1,96 \frac{\sqrt{p(1-p)}}{\sqrt{n}} ; p + 1,96 \frac{\sqrt{p(1-p)}}{\sqrt{n}} \right] = \left[0,3 - 1,96 \frac{\sqrt{0,3(1-0,3)}}{\sqrt{50}} ; 0,3 + 1,96 \frac{\sqrt{0,3(1-0,3)}}{\sqrt{50}} \right] \approx [0,172 ; 0,428]$$

$$\left[p - 2,58 \frac{\sqrt{p(1-p)}}{\sqrt{n}} ; p + 2,58 \frac{\sqrt{p(1-p)}}{\sqrt{n}} \right]$$

$$\left[p - \frac{1}{\sqrt{n}} ; p + \frac{1}{\sqrt{n}} \right] = \left[0,3 - \frac{1}{\sqrt{10\,000}} ; 0,3 + \frac{1}{\sqrt{10\,000}} \right] = [0,29 ; 0,31]$$

$$\left[p - \frac{1}{\sqrt{n}} ; p + \frac{1}{\sqrt{n}} \right] = \left[0,3 - \frac{1}{\sqrt{10\,001}} ; 0,3 + \frac{1}{\sqrt{10\,001}} \right] \subseteq [0,2 ; 0,4]$$

$$\left[p - \frac{1}{\sqrt{n}} ; p + \frac{1}{\sqrt{n}} \right] = \left[0,3 - \frac{1}{\sqrt{10\,001}} ; 0,3 + \frac{1}{\sqrt{10\,001}} \right] \approx [0,29 ; 0,31]$$

$$\left[p - \frac{1}{\sqrt{n}} ; p + \frac{1}{\sqrt{n}} \right] = \left[0,3 - \frac{1}{\sqrt{123\,456}} ; 0,3 + \frac{1}{\sqrt{123\,456}} \right] \approx [0,297\,1 ; 0,302\,9]$$

$$\left[p - \frac{1}{\sqrt{n}} ; p + \frac{1}{\sqrt{n}} \right] = \left[0,3 - \frac{1}{\sqrt{10}} ; 0,3 + \frac{1}{\sqrt{10}} \right] \approx [0 ; 0,62]$$

$$\left[f - 1,96 \frac{\sqrt{f(1-f)}}{\sqrt{n}} ; f + 1,96 \frac{\sqrt{f(1-f)}}{\sqrt{n}} \right] = \left[0,5 - 1,96 \frac{\sqrt{0,5(1-0,5)}}{\sqrt{250}} ; 0,5 + 1,96 \frac{\sqrt{0,5(1-0,5)}}{\sqrt{250}} \right] \approx [0,438 ; 0,562]$$

%rédaction complète, via amsmath

`\RedactionIntFluct[Arrondi=3,p=0.55,n=189]{IF}` %1=clés,2=nom intervalle

`\RedactionIntConf[Arrondi=3,f=0.55,n=189]{IC}` %1=clés,2=nom intervalle

$$\begin{aligned}
 IF &= \left[p - 1,96 \frac{\sqrt{p(1-p)}}{\sqrt{n}} ; p + 1,96 \frac{\sqrt{p(1-p)}}{\sqrt{n}} \right] \\
 &= \left[0,55 - 1,96 \frac{\sqrt{0,55(1-0,55)}}{\sqrt{189}} ; 0,55 + 1,96 \frac{\sqrt{0,55(1-0,55)}}{\sqrt{189}} \right] \\
 &\approx [0,479 ; 0,621]
 \end{aligned}$$

$$\begin{aligned}
 IC &= \left[f - 1,96 \frac{\sqrt{f(1-f)}}{\sqrt{n}} ; f + 1,96 \frac{\sqrt{f(1-f)}}{\sqrt{n}} \right] \\
 &= \left[0,55 - 1,96 \frac{\sqrt{0,55(1-0,55)}}{\sqrt{189}} ; 0,55 + 1,96 \frac{\sqrt{0,55(1-0,55)}}{\sqrt{189}} \right] \\
 &\approx [0,479 ; 0,621]
 \end{aligned}$$

5 Collection d'objets

5.1 Idée



L'idée est d'obtenir une commande pour simplifier l'écriture d'un ensemble d'éléments, en laissant gérer les espaces.
Les délimiteurs de l'ensemble créé sont toujours $\{ \}$.

Code $\LaTeX$

```
\EcritureEnsemble[clés]{liste}
```

5.2 Commande et options



Peu d'options pour ces commandes :

- le premier argument, *optionnel*, permet de spécifier les **Clés** :
 - clé **Sep** qui correspond au délimiteur des éléments de l'ensemble; défaut **(;)**
 - clé **Option** qui est un code (par exemple `strut...`) inséré avant les éléments; défaut **(vide)**
 - un booléen **Mathpunct** qui permet de préciser si on utilise l'espacement mathématique `mathpunct`. défaut **(true)**
- le second, *obligatoire*, est la liste des éléments, séparés par `/`.

Code $\LaTeX$

```
\EcritureEnsemble{a/b/c/d/e}$  
\EcritureEnsemble[Mathpunct=false]{a/b/c/d/e}$  
\EcritureEnsemble[Sep=,]{a/b/c/d/e}$  
\EcritureEnsemble[Option={\strut}]{a/b/c/d/e}$ % \strut pour "augmenter" un peu la  
↪ hauteur des {}  
\EcritureEnsemble{\frac{1}{1+\frac{1}{3}} / b / c / d / \frac{1}{2}}$
```

Code $\LaTeX$

```
{a;b;c;d;e}  
{a;b;c;d;e}  
{a,b,c,d,e}  
{a;b;c;d;e}  
 $\left\{ \frac{1}{1+\frac{1}{3}}; b; c; d; \frac{1}{2} \right\}$ 
```



Attention cependant au comportement de la commande avec des éléments en mode mathématique, ceux-ci peuvent générer une erreur si `displaystyle` n'est pas utilisé...

Thème

OUTILS POUR L'ANALYSE

Quatrième partie

Outils pour l'analyse

6 Maximum, minimum d'une fonction



 L'idée est de proposer une commande pour déterminer, par balayage, le maximum ou le minimum d'une fonction *classique*, sur un intervalle donné.

Le package  `xint` se charge des calculs, et les résultats sont stockés dans des macros personnalisables, pour exploitations ultérieures.

⚠ Attention cependant à ne pas travailler sur un intervalle trop grand et/ou de demander une précision trop petite, car les calculs risquent d'être longs...

Code $\LaTeX$

```
%arguments optionnels = macros de stockage  
\DetermineMax[précision]{expression xint}{a}{b}[\tmpmax][\tmpmaxvalx]  
\DetermineMin[précision]{expression xint}{a}{b}[\tmpmin][\tmpminvalx]
```

Code $\LaTeX$

```
\DetermineMax{80*x*exp(-0.2*x)}{3}{10}\tmpmax en \tmpmaxvalx
```

147.1517764685769 en 5

Code $\LaTeX$

```
\DetermineMax[0.1]  
{0.6*cos(4.5*(x-4)+2.1)-1.2*sin(x-4)+0.1*x+0.2}{1}{4}  
[\MonMax]  
[\MonMaxX]  
\MonMax en \MonMaxX
```

2.164719329044054 en 2.2

Code $\LaTeX$

```
\DetermineMin[0.001]  
{log(exp(-x)+1)+0.25*x}{1}{1.2}  
[\MonMin]  
[\MonMinX]  
\MonMin en \MonMinX
```

0.5623351587104050 en 1.099

7 Résolution approchée d'une équation

7.1 Idée



2.1.4 L'idée est de proposer une commande pour résoudre, de manière approchée, une équation du type $f(x) = k$ sur un intervalle (fermé) donné.

La méthode utilisée est la **dichotomie**, pour plus de rapidité que la méthode *simple* par balayage.

Code $\LaTeX$

```
\ResolutionApprochee[clés]{équation}[macro]
```

Code $\LaTeX$

```
\ResolutionApprochee[Intervalle=0:10]{x**3-2*x**2-x-1=2}%  
$x_0 \approx \num[minimum-decimal-digits=2]{\masolutiond}$ par défaut ;\\  
$x_0 \approx \num[minimum-decimal-digits=2]{\masolutione}$ par excès ;\\  
$x_0 \approx \num[minimum-decimal-digits=2]{\masolutiona}$ arrondi à  $10^{-2}$ .\\  
  
\hfill\includegraphics[scale=0.45]{./graphics/pl-solve_a}\hfill~
```

$x_0 \approx 2,75$ par défaut;
 $x_0 \approx 2,76$ par excès;
 $x_0 \approx 2,76$ arrondi à 10^{-2} .

rad SOLVEUR	
Equations	
Solution	
x1	2.757278921
Δ	-439

7.2 Clés et options



Quelques explications sur les **clés** et sur les arguments :

- la clé **Precision** pour le nombre de chiffres après la virgule de la solution; défaut **2**
- la clé (obligatoire!) **Intervalle** qui permet de préciser l'intervalle initial de recherche;
- la clé **Variable** qui permet de spécifier la variable de l'équation; défaut **x**
- l'argument *obligatoire* est l'équation, sous la forme $f(\dots) = k$ (ou $f(\dots)$ pour $f(\dots) = 0$);
- l'argument *optionnel* est la base de la *macro* qui sert à stocker les valeurs : défaut **masolution**
 - `\<macro>d` pour la valeur approchée par défaut;
 - `\<macro>e` pour la valeur approchée par excès;
 - `\<macro>a` pour la valeur approchée.

```

\ResolutionApprochee[Precision=4,Intervalle=0:2]{exp(0.5*x)+x**2-4=0}%
Une valeur approchée, à  $10^{-4}$  près, d'une solution de  $\text{e}^{0,5x}+x^2-4=0$  sur
↪  $\left[0;2\right]$  est  $\beta$  avec :
\begin{itemize}
\item  $\beta \approx \text{num}[\text{minimum-decimal-digits}=4]{\text{masolutiond}}$  par défaut ;
\item  $\beta \approx \text{num}[\text{minimum-decimal-digits}=4]{\text{masolutione}}$  par excès ;
\item  $\beta \approx \text{num}[\text{minimum-decimal-digits}=4]{\text{masolutiona}}$ .
\end{itemize}
\ResolutionApprochee[Variable=t,Intervalle=-1:2]{3*t*exp(-0.5*t+1)=4}[SolA]%
Une valeur approchée, à  $10^{-2}$  près d'une solution de  $3t\text{e}^{-0,5t+1}=4$  est  $t_1$  avec :
\begin{itemize}
\item  $t_1 \approx \text{num}[\text{minimum-decimal-digits}=2]{\text{SolAd}}$  par défaut ;
\item  $t_1 \approx \text{num}[\text{minimum-decimal-digits}=2]{\text{SolAe}}$  par excès ;
\item  $t_1 \approx \text{num}[\text{minimum-decimal-digits}=2]{\text{SolAa}}$ .
\end{itemize}
\ResolutionApprochee[Precision=3,Variable=t,Intervalle=2:10]{3*t*exp(-0.5*t+1)=4}[SolB]
Une valeur approchée, à  $10^{-2}$  près d'une solution de  $3t\text{e}^{-0,5t+1}=4$  est  $t_2$  avec :
\begin{itemize}
\item  $t_2 \approx \text{num}[\text{minimum-decimal-digits}=2]{\text{SolBd}}$  par défaut ;
\item  $t_2 \approx \text{num}[\text{minimum-decimal-digits}=2]{\text{SolBe}}$  par excès ;
\item  $t_2 \approx \text{num}[\text{minimum-decimal-digits}=2]{\text{SolBa}}$ .
\end{itemize}
\end{itemize}

\medskip

\hfill\includegraphics[scale=0.45]{./graphics/pl-solve_b}~~
\includegraphics[scale=0.45]{./graphics/pl-solve_c}~~
\includegraphics[scale=0.45]{./graphics/pl-solve_d}\hfill~

```

Une valeur approchée, à 10^{-4} près, d'une solution de $\text{e}^{0,5x} + x^2 - 4 = 0$ sur $[0; 2]$ est β avec :

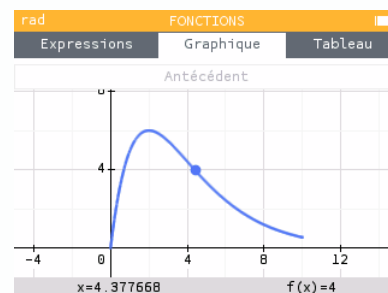
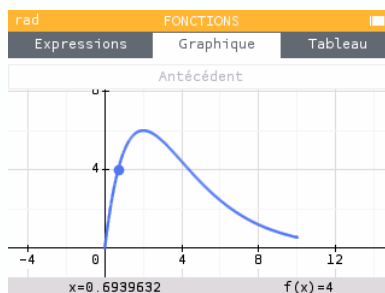
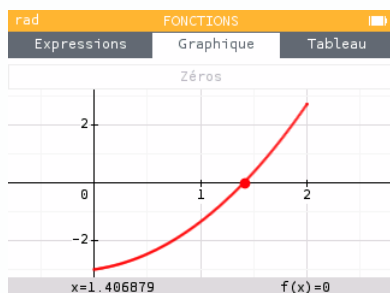
- $\beta \approx 1,4068$ par défaut;
- $\beta \approx 1,4069$ par excès;
- $\beta \approx 1,4069$.

Une valeur approchée, à 10^{-2} près d'une solution de $3t\text{e}^{-0,5t+1} = 4$ est t_1 avec :

- $t_1 \approx 0,69$ par défaut;
- $t_1 \approx 0,70$ par excès;
- $t_1 \approx 0,69$.

Une valeur approchée, à 10^{-2} près d'une solution de $3t\text{e}^{-0,5t+1} = 4$ est t_2 avec :

- $t_2 \approx 4,377$ par défaut;
- $t_2 \approx 4,378$ par excès;
- $t_2 \approx 4,378$.



8 Présentation d'une solution d'équation par balayage

8.1 Idée



2.0.4 L'idée est de présenter l'obtention d'une solution approchée d'équation par balayage, dans le cadre du TVI par exemple. Les calculs et tests sont effectués grâce au package `xinttools`, et le formatage par `tabularray` et `sinuitx`.



Le code ne trouve pas la solution, il met *juste* en forme mais effectue quand même les calculs d'images et les tests.

Code $\LaTeX$

```
\SolutionTVI[options]{fonction}{valeur}
```

8.2 Clés et arguments



Plusieurs **(Clés)** sont disponibles pour cette commande, relative à une équation du type $f(x) = k$:

- la clé **(NomFct)** qui permet de spécifier le nom de la fonction; défaut **(f)**
- la clé **(NomSol)** qui permet de spécifier le nom de la fonction; défaut **(\alpha)**
- les clés **(va)** et **(vb)** qui sont les bornes inférieure et supérieure de l'encadrement;
- la clé **(Precision)** qui est la précision des calculs pour les images; défaut **(2)**
- la clé **(Stretch)** qui permet d'espacer les lignes; défaut **(1.15)**
- les booléens **(Balayage)** ou **(Calculatrice)** pour afficher un texte en amont; défaut **(false)**
- le booléen **(Majuscule)** qui affiche le texte avant, avec une majuscule au début. défaut **(true)**

Le premier argument *obligatoire* est la fonction, en syntaxe `xint` et avec comme variable x , et le second la valeur de k .

Code $\LaTeX$

Pour $f(x)=0$ avec $f(x)=x^2-2$. On obtient `\SolutionTVI[va=1.414,vb=1.415,Precision=3]{x**2-2}{0}`.

Pour $f(x) = 0$ avec $f(x) = x^2 - 2$. On obtient
$$\left\{ \begin{array}{l} f(1,414) \approx -0,001 < 0 \\ f(1,415) \approx 0,002 > 0 \end{array} \right\} \Rightarrow 1,414 < \alpha < 1,415.$$

Code $\LaTeX$

Avec $\varphi(t)=3t\,e^{-0,5t+1}=5$,
`\SolutionTVI[Majuscule=false,Calculatrice,va=1.02,vb=1.03,NomFct=\varphi]{3*x*exp(-0.5*x+1)}{5}`

Avec $\varphi(t) = 3t e^{-0,5t+1} = 5$, par calculatrice, on obtient
$$\left\{ \begin{array}{l} \varphi(1,02) \approx 4,99 < 5 \\ \varphi(1,03) \approx 5,02 > 5 \end{array} \right\} \Rightarrow 1,02 < \alpha < 1,03$$

Code $\LaTeX$

On s'intéresse à $g(x)=\ln(x)$ avec $g(x)=\ln(x)$.
`\SolutionTVI[Balayage,Stretch=1.5,va=4.48,vb=4.49,NomFct=g,Precision=4,NomSol={x_0}]{log(x)}{1.5}`.

On s'intéresse à $g(x) = 1,5$ avec $g(x) = \ln(x)$.

Par balayage, on obtient
$$\left\{ \begin{array}{l} g(4,48) \approx 1,4996 < 1,5 \\ g(4,49) \approx 1,5019 > 1,5 \end{array} \right\} \Rightarrow 4,48 < x_0 < 4,49.$$

8.3 Interaction avec la commande de résolution approchée



2.1.4 L'idée est de récupérer les valeurs par défaut et par excès pour le TVI grâce à la commande `\ResolutionApprochee`.

Code $\text{\LaTeX}$

On s'intéresse à $g(x) = \text{num}\{1,5\}$ avec $g(x) = \ln(x)$ sur l'intervalle $\left[3;5\right]$.

```
\ResolutionApprochee[Intervalle=3:5]{log(x)=1.5}[SolLn]
\SolutionTVI%
[Balayage,Stretch=1.5,va={\SolLnd},vb={\SolLne},
NomFct=g,Precision=4,NomSol={x_0}]{log(x)}{1.5}.
```

On s'intéresse à $g(x) = 1,5$ avec $g(x) = \ln(x)$ sur l'intervalle $[3 ; 5]$.

Par balayage, on obtient $\left\{ \begin{array}{l} g(4,48) \approx 1,499\,6 < 1,5 \\ g(4,49) \approx 1,501\,9 > 1,5 \end{array} \right. \Rightarrow 4,48 < x_0 < 4,49.$



À terme, peut-être que la commande `\ResolutionApprochee` sera intégrée dans la commande `\SolutionTVI` afin d'automatiser encore plus le procédé.

9 Suites récurrentes simples

9.1 Idées



2.0.3 L'idée est de proposer des commandes pour effectuer des calculs avec des suites récurrentes du type $u_{n+1} = f(u_n)$:

- calcul de termes avec possibilité d'arrondir;
- présentation de la conclusion de la recherche d'un seuil du type $u_n > S$ ou $u_n < S$.



2.1.0 Le code pour le seuil **trouve** également le rang cherché, il met en forme et effectue les calculs d'images.

2.0.5 Le choix a été fait de faire les calculs en mode `float` pour éviter les dépassements de capacité de `xint` liés aux boucles...

Code $\LaTeX$

```
%commande pour calculer et formater  
\CalculTermeRecurrence[options]{fonction associée}  
  
%mise en forme de la conclusion d'un seuil  
\SolutionSeuil[options]{fonction associée}{seuil}
```

9.2 Clés et arguments



Plusieurs **Clés** sont disponibles pour la commande du calcul d'un terme :

- la clé **<No>** qui est le rang initial de la suite;
- la clé **<UNo>** qui est le terme initial de la suite;
- la clé **<Precision>** qui précise l'arrondi éventuel; défaut **<3>**
- la clé **<N>** qui est l'indice du terme à calculer.

L'argument *obligatoire* est la fonction associée à la suite, en syntaxe `xint` et avec comme variable x .

Code $\LaTeX$

```
Avec  $\begin{cases} u_0 = 50 \\ u_{n+1} = \frac{1}{u_n + 2} \end{cases}$ .  
On obtient  $u_{10} \approx \text{\textbackslash CalculTermeRecurrence[No=0,UNo=50,N=10]\{1/(x+2)\}}$ .  
On obtient  $u_{15} \approx \text{\textbackslash CalculTermeRecurrence[Precision=4,No=0,UNo=50,N=15]\{1/(x+2)\}}$ .  
On obtient  $u_{20} \approx \text{\textbackslash CalculTermeRecurrence[Precision=6,No=0,UNo=50,N=20]\{1/(x+2)\}}$ .
```

Code $\LaTeX$

Avec $u_0 = 50$ et $u_{n+1} = \frac{1}{u_n + 2}$.

On obtient $u_{10} \approx 0,414$

On obtient $u_{15} \approx 0,414\,2$

On obtient $u_{20} \approx 0,414\,214$

sortie par défaut.

avec choix de la précision à 10^{-4} .

avec choix de la précision à 10^{-6} .



Plusieurs **(Clés)** sont disponibles pour la commande du seuil :

- la clé **<NomSuite>** qui est le nom de la suite; défaut **<u>**
- la clé **<No>** qui est le rang initial de la suite;
- la clé **<UNo>** qui est le terme initial de la suite;
- la clé **<Precision>** qui précise l'arrondi éventuel; défaut **<2>**
- la clé **<Stretch>** qui permet d'espacer les lignes; défaut **<1.15>**
- les booléens **<Balayage>** ou **<Calculatrice>** pour afficher un texte en amont; défaut **<false>**
- le booléen **<Simple>** pour une présentation plus neutre; défaut **<false>**
- le booléen **<Majuscule>** qui affiche le texte avant, avec une majuscule au début; défaut **<true>**
- les booléens **<Exact>** et **<ExactB>** qui affichent $\approx$ au lieu de $\backslash approx$; défaut **<false>**
- le booléen **<Conclusion>** pour afficher la conclusion ou non; défaut **<true>**
- la clé **<Sens>** parmi **<< / > / <= / >= >** pour indiquer le type de seuil. défaut **<>>**

Le premier argument *obligatoire* est la fonction associée à la suite, en syntaxe $\backslash xint$ et avec comme variable x , et le second est le seuil à dépasser.

Code $\LaTeX$

```
Avec  $\backslash begin{dcases}$   $u_1 = 2$   $\backslash \backslash u_{n+1} = 1 + \frac{1+u_n^2}{1+u_n}$   $\backslash end{dcases}$  $,
on cherche $n$ tel que $u_n > 5$.  $\backslash \backslash$ 
 $\backslash SolutionSeuil[Balayage,No=1,UNo=2]\{1+(1+x**2)/(1+x)\}\{5\}$ .
 $\hookrightarrow \backslash SolutionSeuil[Calculatrice,Precision=4,No=1,UNo=2,Conclusion=false]\%$ 
 $\{1+(1+x**2)/(1+x)\}\{5\}$ .
```

Avec $\begin{cases} u_1 = 2 \\ u_{n+1} = 1 + \frac{1+u_n^2}{1+u_n} \end{cases}$, on cherche n tel que $u_n > 5$.

Par balayage, on obtient $\begin{cases} u_7 \approx 4,868 \leq 5 \\ u_8 \approx 5,209 > 5 \end{cases} \Rightarrow n \geq 8$. Par calculatrice, on obtient $\begin{cases} u_7 \approx 4,8681 \leq 5 \\ u_8 \approx 5,2089 > 5 \end{cases}$.

9.3 Exemple d'utilisation

Code $\LaTeX$

```
Avec  $\backslash begin{dcases}$   $u_1 = 2$   $\backslash \backslash u_{n+1} = 1 + \frac{1+u_n^2}{1+u_n}$   $\backslash end{dcases}$  $,
on obtient le tableau de valeurs suivant :
 $\backslash begin{tabular}{c|c}$ 
 $\backslash n$  &  $\backslash u_n$   $\backslash \backslash \hline$ 
1 & 2  $\backslash \backslash$ 
 $\backslash xintFor* \#1 in \{ \backslash xintSeq{2}\{7\} \}$   $\backslash do \{ \#1 \}$  &  $\backslash CalculTermeRecurrence[No=1,UNo=2,N=\#1]\{1+(1+x**2)/(1+x)\}$   $\backslash \backslash$ 
 $\backslash end{tabular}$   $\backslash \backslash$ 

 $\backslash SolutionSeuil[Precision=4,No=1,UNo=2,Simple]\{1+(1+x**2)/(1+x)\}\{10\}$  (Ainsi $u_n > 10$ à partir de
 $\hookrightarrow \backslash n = \backslash the \backslash CompteurSeuil$ )
```

Avec $\begin{cases} u_1 = 2 \\ u_{n+1} = 1 + \frac{1+u_n^2}{1+u_n} \end{cases}$, on obtient le tableau de valeurs suivant :	n	u_n
	1	2
	2	2,667
	3	3,212
	4	3,687
	5	4,114
	6	4,505
	7	4,868
$u_{28} \approx 9,9408 \leq 10$ et $u_{29} \approx 10,1236 > 10$ (Ainsi $u_n > 10$ à partir de $n = 29$)		

10 Valeur approchée d'une intégrale

10.1 Idée



2.6.1 L'idée est de proposer plusieurs approximations pour le calcul d'une intégrale, en utilisant :

- une méthode des rectangles (Gauche, Droite ou Milieu);
- la méthode des trapèzes;
- la méthode de Simpson.



Il s'agit de valeurs approchées, mais la méthode de Simpson donne des valeurs satisfaisantes! Les méthodes *Rectangles* ou *Trapezes* seront plutôt utiles pour des résultats obtenus par algorithme par exemple.

Code $\LaTeX$

```
\IntegraleApprochee[clés]{fonction}{a}{b}
```

10.2 Clés et arguments



Plusieurs **Clés** sont disponibles pour la commande de calcul :

- le booléen **ResultatBrut** qui donne le résultat obtenu grâce à `\xint`; défaut : **false**
- la clé **Methode**, parmi **RectanglesGauche** / **RectanglesDroite** / **RectanglesMilieu** / **Trapezes** / **Simpson** pour spécifier la méthode utilisée; défaut : **Simpson**
- la clé **NbSubDiv** précise le nombre de subdivisions pour le calcul; défaut : **10**
- le booléen **AffFormule** qui affiche au préalable l'intégrale; défaut **false**
- la clé **Expr** qui indique ce qui doit être affiché dans l'intégrale; défaut **f(x)**
- la clé **Signe** qui indique le signe à afficher entre l'intégrale et le résultat; défaut **\approx**
- la clé **Variables** qui indique la variable à afficher dans le dx. défaut **x**

Concernant les arguments obligatoires :

- le premier est la fonction à intégrer, en langage `\xint`, avec comme variable x;
- les deux autres arguments sont les bornes de l'intégrale.

À noter que la commande, hormis dans sa version **ResultatBrut**, est à insérer de préférence dans un mode mathématique.

Code $\LaTeX$

```
On s'intéresse à  $\displaystyle \int_4^{10} f(x) \, dx$  avec  $f(x) = \sqrt{x}$  :  
\begin{itemize}[itemsep=6pt, leftmargin=4cm]  
  \item[sortie par défaut :] \IntegraleApprochee{sqrt(x)}{4}{10}  
  \item[résultat brut :] \IntegraleApprochee[ResultatBrut]{sqrt(x)}{4}{10}  
  \item[résultat formaté :]  
    \displaystyle \IntegraleApprochee[NbSubDiv=100, AffFormule, Precision=5, Expr={\sqrt{x}}]{  
      sqrt(x)}{4}{10}$  
\end{itemize}
```

On s'intéresse à $\int_4^{10} f(x) dx$ avec $f(x) = \sqrt{x}$:

sortie par défaut : 15,749

résultat brut : 15.74851726347158

résultat formaté : $\int_4^{10} \sqrt{x} dx \approx 15,748\,52$

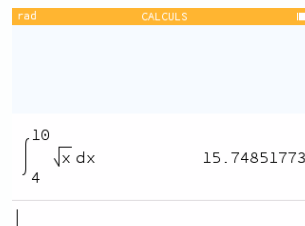
10.3 Exemples

Code $\LaTeX$

```
%tableau
\IntegraleApprochee[NbSubDiv=10,ResultatBrut]{sqrt(x)}{4}{10}
\IntegraleApprochee[NbSubDiv=10,Methode=RectanglesGauche,ResultatBrut]{sqrt(x)}{4}{10}
\IntegraleApprochee[NbSubDiv=10,Methode=RectanglesDroite,ResultatBrut]{sqrt(x)}{4}{10}
\IntegraleApprochee[NbSubDiv=10,Methode=RectanglesMilieu,ResultatBrut]{sqrt(x)}{4}{10}
\IntegraleApprochee[NbSubDiv=10,Methode=Trapezes,ResultatBrut]{sqrt(x)}{4}{10}
$\displaystyle\IntegraleApprochee[NbSubDiv=10,AffFormule,Expr={\sqrt{x}}]{\sqrt{x)}{4}{10}$
```

Code $\LaTeX$

Méthode utilisée	Valeur brute obtenue
$f(x) = \sqrt{x}$ et $n = 10$; $\int_4^{10} f(x) dx$	
Simpson	15.74851726347158
rectangles Gauche	15.39707973922291
rectangles Droite	16.09444633532393
rectangles Milieu	15.74989437657067
trapèzes	15.74576303727343
$\int_4^{10} \sqrt{x} dx \approx 15,749$	

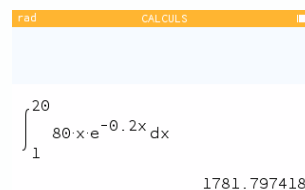


Code $\LaTeX$

```
%tableau
\IntegraleApprochee[NbSubDiv=100,Methode=Simpson,ResultatBrut]{80*x*exp(-0.2*x)}{1}{20}
\IntegraleApprochee[NbSubDiv=100,Methode=RectanglesGauche,ResultatBrut]{80*x*exp(-0.2*x)}{1}{20}
\IntegraleApprochee[NbSubDiv=100,Methode=RectanglesDroite,ResultatBrut]{80*x*exp(-0.2*x)}{1}{20}
\IntegraleApprochee[NbSubDiv=100,Methode=RectanglesMilieu,ResultatBrut]{80*x*exp(-0.2*x)}{1}{20}
\IntegraleApprochee[NbSubDiv=100,Methode=Trapezes,ResultatBrut]{80*x*exp(-0.2*x)}{1}{20}
$\displaystyle\IntegraleApprochee[NbSubDiv=100,AffFormule,Expr={80x\,\text{e}^{-0,2x}}]{80x*exp(-0.2*x)}{1}{20}$
```

Code $\LaTeX$

Méthode utilisée	Valeur brute obtenue
$f(x) = 80x e^{-0,2x}$ et $n = 100$; $\int_1^{20} f(x) dx$	
Simpson	1781.797415154050
rectangles Gauche	1785.064951643106
rectangles Droite	1778.188198418496
rectangles Milieu	1781.882835215674
trapèzes	1781.626575030801
$\int_1^{20} 80x e^{-0,2x} dx \approx 1781,797$	



10.4 Commande simplifiée



3.04a À noter qu'il existe une commande, simplifiée, pour le calcul de la valeur approchée d'une intégrale :

- la méthode utilisée est la méthode de Simpson;
- il n'existe pas de **(clés)** pour cette commande.

Code $\LaTeX$

```
\CalcIntegrale(*)[prec]<nb subdiv>\{expr version xint\}\{binf\}\{bsup\}
```



La version étoilée force l’affichage avec $\texttt{\sinuitx}$.

L’argument optionnel, et entre $\texttt{[...]}$, est la précision éventuelle souhaitée (résultat brut sinon).

L’argument optionnel, et entre $\texttt{<...>}$, est le nombre de subdivisions, qui permet de gagner en précision.

Les trois arguments obligatoires sont quant à eux :

- l’expression de la fonction, en langage $\texttt{xint}$;
- les bornes de l’intégrale.

Code $\LaTeX$

```
\displaystyle\int_1^{20} 80x\mathrm{e}^{-0,2x} \, \mathrm{d}x \approx  
\CalcIntegrale\{80*x*\exp(-0.2*x)\}\{1\}\{20\}
```

$$\int_1^{20} 80x\mathrm{e}^{-0,2x} \mathrm{d}x \approx 1781.797364979251$$

Code $\LaTeX$

```
\displaystyle\int_1^{20} 80x\mathrm{e}^{-0,2x} \, \mathrm{d}x \approx  
\CalcIntegrale*\{80*x*\exp(-0.2*x)\}\{1\}\{20\}
```

$$\int_1^{20} 80x\mathrm{e}^{-0,2x} \mathrm{d}x \approx 1\,781,797\,364\,979\,251$$

Code $\LaTeX$

```
\displaystyle\int_1^{20} 80x\mathrm{e}^{-0,2x} \, \mathrm{d}x \approx  
\CalcIntegrale*\{3\}\{80*x*\exp(-0.2*x)\}\{1\}\{20\}
```

$$\int_1^{20} 80x\mathrm{e}^{-0,2x} \mathrm{d}x \approx 1\,781,797$$



$\texttt{3.04a}$ On peut également utiliser la commande, avec exactement le même fonctionnement, pour déterminer une estimation d’une valeur moyenne par intégrale.

Code $\LaTeX$

```
\ValeurMoyenneIntg(*)[prec]<nb subdiv>\{expr version xint\}\{binf\}\{bsup\}
```

Code $\LaTeX$

```
\displaystyle\dfrac{1}{20-1}\int_1^{20} 80x\mathrm{e}^{-0,2x} \, \mathrm{d}x \approx  
\ValeurMoyenneIntg*\{3\}<100>\{80*x*\exp(-0.2*x)\}\{1\}\{20\}
```

$$\frac{1}{20-1} \int_1^{20} 80x\mathrm{e}^{-0,2x} \mathrm{d}x \approx 93,779$$

11 Forme canonique, fonction homographique

11.1 Idée



 3.03a L'idée est de proposer des commandes pour :

- mettre un trinôme sous forme canonique $ax^2 + bx + c = a(x - \alpha)^2 + \beta$ ($a \neq 0$);
- décomposer une fonction homographique $\frac{ax+b}{cx+d} = A + \frac{C}{x-\alpha}$ ($c \neq 0$).

Code $\LaTeX$

```
\FormeCanonique(*)[option]{a}{b}{c}
```

Code $\LaTeX$

```
\FonctionHomographique(*)[option]{a}{b}{c}{d}
```

11.2 Forme canonique



La version étoilée force l'affichage (si besoin) du signe « - », sous la forme $a(x - \alpha)^2 + \beta$; et le code se charge de gérer les cas particuliers ($|a| = 1$, $\alpha = 0$, $\beta = 0$).

Le code devrait fonctionner correctement avec des entiers et/ou des décimaux, mais les résultats affichés le seront toujours sous forme fractionnaire (par défaut, l'argument optionnel vaut `<[d]>` pour formater en

 `\displaystyle`, mais on, peut utiliser `<[t/n/dec]>`).

Code $\LaTeX$

```
$2x^2+4x-6 = \FormeCanonique{2}{4}{-6} = \FormeCanonique*{2}{4}{-6}$
```

$$2x^2 + 4x - 6 = 2(x + 1)^2 - 8 = 2(x - (-1))^2 - 8$$

Code $\LaTeX$

```
$-x^2+4x-6 = \FormeCanonique{-1}{4}{-6}$
```

$$-x^2 + 4x - 6 = -(x - 2)^2 - 2$$

Code $\LaTeX$

```
$x^2+3x+7 = \FormeCanonique{1}{3}{7} = \FormeCanonique*{1}{3}{7}$
```

$$x^2 + 3x + 7 = \left(x + \frac{3}{2}\right)^2 + \frac{19}{4} = \left(x - \left(-\frac{3}{2}\right)\right)^2 + \frac{19}{4}$$

Code $\LaTeX$

```
$3x^2-7x = \FormeCanonique{3}{-7}{0} = \FormeCanonique[t]{3}{-7}{0}$
```

$$3x^2 - 7x = 3\left(x - \frac{7}{6}\right)^2 - \frac{49}{12} = 3\left(x - \frac{7}{6}\right)^2 - \frac{49}{12}$$

Code $\LaTeX$

```
$3x^2-7 = \FormeCanonique{3}{0}{-7}$
```

$$3x^2 - 7 = 3x^2 - 7$$

Code $\LaTeX$

```
 $\$x^2+2x+1 = \backslashFormeCanonique\{1\}\{2\}\{1\}\$$ 
```

$$x^2 + 2x + 1 = (x + 1)^2$$

Code $\LaTeX$

```
 $\$1,5x^2-7,25x-14 = \backslashFormeCanonique\{1.5\}\{-7.25\}\{-14\}\$$ 
```

$$1,5x^2 - 7,25x - 14 = \frac{3}{2}\left(x - \frac{29}{12}\right)^2 - \frac{2185}{96}$$

11.3 Fonction homographique



La version étoilée force l'affichage (si besoin) du signe « - » au dénominateur, sous la forme $x - \alpha$; et le code se charge de gérer les cas particuliers ($|A| = 1$, $\alpha = 0$, $B = 0$).

Le code devrait fonctionner correctement avec des entiers et/ou des décimaux, mais les résultats affichés le seront toujours sous forme fractionnaire :

- le coefficient A est toujours en mode `\dfrac`, tout comme la fraction *principale*;
- les coefficients B et α sont par défaut en `\tfrac`, mais l'argument optionnel peut valoir `\{d/t/n/dec\}`.

Code $\LaTeX$

```
 $\$\dfrac{2x-11}{3x-6} = \backslashFonctionHomographique\{2\}\{-11\}\{3\}\{-6\}\$$ 
```

$$\frac{2x - 11}{3x - 6} = \frac{2}{3} + \frac{-\frac{7}{3}}{x - 2}$$

Code $\LaTeX$

```
 $\$\dfrac{11}{3x+6} = \backslashFonctionHomographique\{0\}\{11\}\{3\}\{6\} = \backslashFonctionHomographique*\{0\}\{11\}\{3\}\{6\}\$$ 
```

$$\frac{11}{3x + 6} = \frac{\frac{11}{3}}{x + 2} = \frac{\frac{11}{3}}{x - (-2)}$$

Code $\LaTeX$

```
 $\$\dfrac{2x+5}{7x-19} = \backslashFonctionHomographique\{2\}\{5\}\{7\}\{-19\}\$$ 
```

$$\frac{2x + 5}{7x - 19} = \frac{2}{7} + \frac{\frac{73}{49}}{x - \frac{19}{7}}$$

Code $\LaTeX$

```
 $\$\dfrac{2x+5}{4x} = \backslashFonctionHomographique\{2\}\{5\}\{4\}\{0\}\$$ 
```

$$\frac{2x + 5}{4x} = \frac{1}{2} + \frac{\frac{5}{4}}{x}$$

Code $\LaTeX$

```
 $\dfrac{-x+7}{7x-5} = \text{\FonctionHomographique{-1}{7}{7}{-5}} = \text{\FonctionHomographique*{-1}{7}{7}{-5}}$ 
```

$$\frac{-x+7}{7x-5} = -\frac{1}{7} + \frac{\frac{44}{49}}{x-\frac{5}{7}} = -\frac{1}{7} + \frac{\frac{44}{49}}{x-\frac{5}{7}}$$

Thème

OUTILS GRAPHIQUES

Cinquième partie

Outils graphiques

12 Préambule



Les commandes *graphiques* fournies par `ProfLycee` ont servi de base au package `tkz-grapheur` ([lien]) qui en reprend les éléments constructeurs.

Cette partie (hormis par exemple la section *intervalles*) n'est plus officiellement *maintenue*, mais elle reste présente dans le package et la documentation pour des raisons de rétro-compatibilité.

13 Intervalles

13.1 Idée



`3.00a` L'idée est de proposer un environnement pour travailler sur des intervalles :

- création de l'environnement (basé sur TikZ) avec paramètres globaux (l'unité est le cm);
- commande pour créer un intervalle.

Code $\text{\LaTeX}$

```
\begin{RepIntervalles}[clés]<options tikz>
  \tkzIntervalle[clés]{xmin}[labelxmin]{xmax}[labelxmax]
\end{RepIntervalles}
```

Code $\text{\LaTeX}$

```
\begin{RepIntervalles}
  \tkzIntervalle{-4}{5}
\end{RepIntervalles}
```



13.2 Création de l'environnement

Code $\text{\LaTeX}$

```
\begin{RepIntervalles}[clés]<options tikz>
  %corps
\end{RepIntervalles}
```



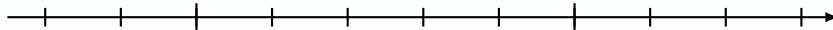
Les **<clés>** (globales pour l'ensemble de l'environnement, y compris pour la création des intervalles) disponibles pour l'environnement `RepIntervalles` sont :

- **<xmin>** : valeur minimale pour l'axe; défaut **<-8>**
- **<xmax>** : valeur maximale pour l'axe; défaut **<8>**
- **<Elargir>** : coefficient d'élargissement de l'axe, en pourcentage de largeur; défaut **<0.05>**
- **<Largeur>** : largeur (en cm) de l'axe (hors élargissement); défaut **<12>**
- **<Unite>** : permet au code de calculer lui-même l'unité, sinon à donner en cm; défaut **<auto>**
- **<EpTrait>** : épaisseur de base des tracés; défaut **<0.8pt>**
- **<Graduations>** : liste des graduations (traits) à afficher;
- **<GraduationsAlt>** : liste des graduations (traits un peu plus grands) à afficher;
- **<HautGrad>** : hauteur de base des graduations; défaut **<7pt>**
- **<Hauteur>** : hauteur des crochets des intervalles; défaut **<16pt>**
- **<Valeurs>** : valeurs (formatées par `\siunitx`!) à afficher sous l'axe;
- **<Police>** : police globale des labels. défaut **<\normalsize\normalfont>**

L'argument optionnel, et entre `<...>`, permet de passer des options spécifiques à l'environnement TikZ.

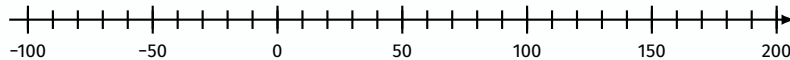
Code $\text{\LaTeX}$

```
\begin{RepIntervalles}[Unite=1,xmin=-2,xmax=8,Graduations={-2,-1,...,8},GraduationsAlt={0,5}]
%corps
\end{RepIntervalles}
```



Code $\text{\LaTeX}$

```
\begin{RepIntervalles}[Elargir=0.025,xmin=-100,xmax=200,Unite=auto,Largeur=10,
↪ Graduations={-100,-90,...,200},GraduationsAlt={-100,-50,...,200},
↪ Valeurs={-100,-50,...,200},Police=\scriptsize]
%corps
\end{RepIntervalles}
```



13.3 Représentation d'intervalles



L'idée est de pouvoir représenter des intervalles dans l'environnement créé précédemment :

- en prenant en compte le type d'intervalles et les bornes infinies;
- en personnalisant la décoration éventuelle.

Code $\text{\LaTeX}$

```
\begin{RepIntervalles}[clés]<options tikz>
\tkzIntervalle[clés]{xmin}[labelxmin]{xmax}[labelxmax]
\end{RepIntervalles}
```



Les **clés** (certains paramètres sont hérités des clés de l'environnement) disponibles pour la commande `\tkzIntervalle` sont :

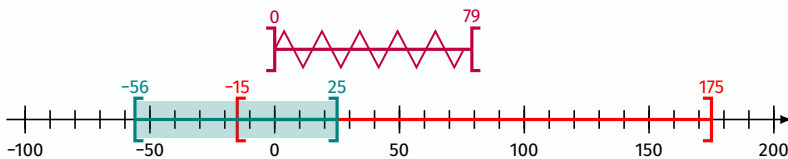
- **<Couleur>** : couleur de base des tracés; défaut **<red>**
- **<Type>** (parmi `FF / FO / OO / OF`) : choix du type d'intervalle; défaut **<FF>**
- **<NiveauV>** : pour positionner l'intervalle décalé verticalement (en pourcentage de la hauteur); défaut **<0>**
- **<Offset>** : pour décaler légèrement la décoration (si superposition par exemple); défaut **<Opt>**
- **<Decor>** parmi `fond / zigzag / hach` : pour décorer l'intervalle (`hach/angle` pour choisir un angle des hachures); défaut **<{}>**
- **<AffValeurs>** : booléen pour afficher les valeurs de bornes; défaut **<false>**
- **<NumInf>** : booléen pour afficher la borne inf grâce à `\siunitx`; défaut **<true>**
- **<NumSup>** : booléen pour afficher la borne sup grâce à `\siunitx`; défaut **<true>**
- **<PosValeurs>** : choix de la position des valeurs des bornes. défaut **<above>**

Les deux arguments obligatoires, et entre `{...}`, permettent de préciser les bornes de l'intervalle (en langage TikZ, et avec `*` pour l'infini.)

`3.00b` Les deux arguments optionnels correspondants, et entre `[...]`, permettent de spécifier les labels des bornes (dans le cas de valeurs rationnelles, ou bien dans le cas de valeurs littérales), et dans ce cas il sera opportun de mettre les booléens **<NumInf/NumSup>** à **<false>**.

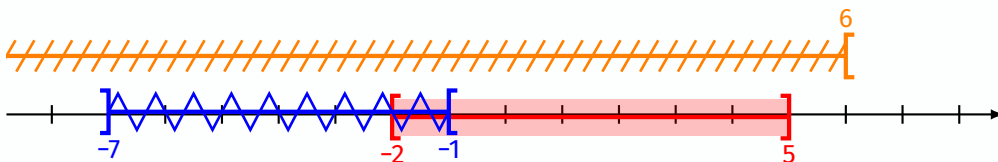
Code $\text{\LaTeX}$

```
\begin{RepIntervalles}[Elargir=0.025,xmin=-100,xmax=200,Unite=auto,Largeur=10,
  ↳ Graduations={-100,-90,...,200},GraduationsAlt={-100,-50,...,200},
  ↳ Valeurs={-100,-50,...,200},Police=\scriptsize,EpTrait=0.6pt]
  \tkzIntervalle[AffValeurs]{-15}{175}
  \tkzIntervalle[Decor=fond,Couleur=teal,AffValeurs]{-56}{25}
  \tkzIntervalle[Decor=zigzag,NiveauV=1.5,Couleur=purple,AffValeurs,Type=OO]{0}{79}
\end{RepIntervalles}
```



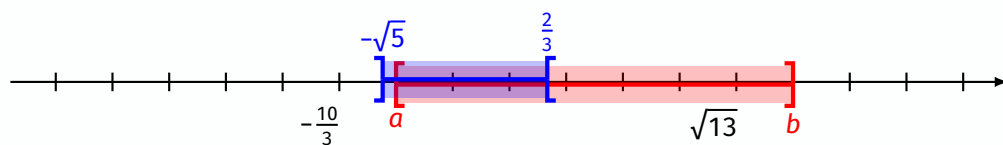
Code $\text{\LaTeX}$

```
\begin{RepIntervalles}[Graduations={-8,-7,...,8}]
  \tkzIntervalle[Decor=fond,Offset=-1pt,AffValeurs,PosValeurs=below]{-2}{5}
  \tkzIntervalle[Type=OO,Couleur=blue,Decor=zigzag,Offset=1pt, AffValeurs,PosValeurs=below]{-7}{-1}
  \tkzIntervalle[Type=FO,Couleur=orange,NiveauV=1.25,Decor=hach,AffValeurs]{*}{6}
\end{RepIntervalles}
```



%la commande \PlaceValeursAxe permet de placer des valeurs quelconques

```
\begin{RepIntervalles}[Graduations={-8,-7,...,8}]
\PlaceValeursAxe{-10/3$-\frac{10}{3}$,sqrt(13)$\sqrt{13}$}
\tkzIntervalle[Decor=fond,Offset=-1pt,AffValeurs,PosValeurs=below,
  ↳ NumInf=false,NumSup=false]{-2}[$a$][$b$]
\tkzIntervalle[Couleur=blue,Type=00,Decor=fond,Offset=1pt,AffValeurs,
  ↳ NumInf=false,NumSup=false]{-sqrt(5)}[$-\sqrt{5}$]{2/3}[$\tfrac{2}{3}$]
\end{RepIntervalles}
```



14 Petits schémas pour le signe de fonctions classiques

14.1 Idée



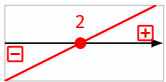
L'idée est d'obtenir une commande pour tracer (en TikZ) un petit schéma pour *visualiser* le signe d'une fonction affine ou d'un trinôme.

Le code est largement inspiré de celui du package `tnsana` même si la philosophie est un peu différente. Comme pour les autres commandes TikZ, l'idée est de laisser la possibilité à l'utilisateur de définir et créer son environnement TikZ, et d'insérer la commande `\MiniSchemaSignes` pour afficher le schéma.

2.1.9 Il est à noter que la version *étoilée* rend la commande autonome, sans besoin de créer l'environnement TikZ.

Code $\LaTeX$

```
\MiniSchemaSignes*
```



14.2 Commandes

Code $\LaTeX$

```
\begin{tikzpicture}[<options>]  
  \MiniSchemaSignes[clés]  
\end{tikzpicture}
```

Code $\LaTeX$

```
{\tikz[options] \MiniSchemaSignes[clés]}  
%ou  
\MiniSchemaSignes*[clés]<options tikzpicture>
```



2.1.9 La version *étoilée* de la commande permet de basculer en mode *autonome*, c'est-à-dire sans avoir besoin de créer son environnement TikZ.

Le premier argument, *optionnel* et entre [...], contient les **(Clés)** suivantes :

- la clé **<Code>** qui permet de définir le type d'expression (voir en-dessous); défaut **<da+>**
- la clé **<Couleur>** qui donne la couleur de la représentation; défaut **<red>**
- la clé **<Racines>** qui définit la ou les racines; défaut **<2>**
- la clé **<Largeur>** qui est la largeur du schéma; défaut **<2>**
- la clé **<Hauteur>** qui est la hauteur du schéma; défaut **<1>**
- un booléen **<Cadre>** qui affiche un cadre autour du schéma. défaut **<true>**

Le second argument, *optionnel* et entre <...>, permet de spécifier (pour la commande *étoilée*), des options à passer à l'environnement `\tikzpicture`.



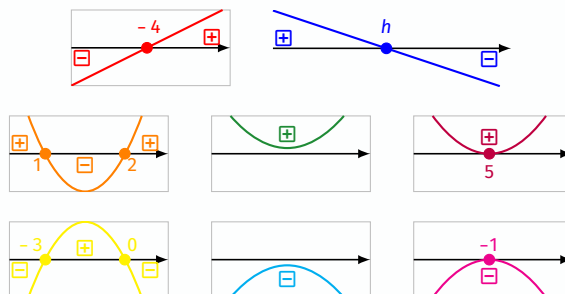
Pour la clé **<code>**, il est construit par le type (a pour affine ou p comme parabole) puis les éléments caractéristiques (a+ pour $a > 0$, d0 pour $\Delta = 0$, etc) :

- **<Code=da+>** := une droite croissante;
- **<Code=da->** := une droite décroissante;
- **<Code=pa+d+>** := une parabole *souriante* avec deux racines;
- etc

Code $\LaTeX$

```
\begin{center}
  \MiniSchemaSignes*[Code=da+,Racines=-4]
  ~~~~
  \MiniSchemaSignes*[Code=da-,Racines={h},Couleur=blue,Largeur=3,Cadre=false]
\end{center}
%
\begin{center}
  \MiniSchemaSignes*[Code=pa+d+,Racines={1/2},Couleur=orange]
  ~~~~
  \MiniSchemaSignes*[Code=pa+d-,Couleur=CouleurVertForet]
  ~~~~
  \MiniSchemaSignes*[Code=pa+d0,Racines={5},Couleur=purple]
\end{center}
%
\begin{center}
  \MiniSchemaSignes*[Code=pa-d+,Racines={-3/0},Couleur=yellow]
  ~~~~
  \MiniSchemaSignes*[Code=pa-d-,Couleur=cyan]
  ~~~~
  \MiniSchemaSignes*[Code=pa-d0,Racines={-1},Couleur=magenta]
\end{center}
```

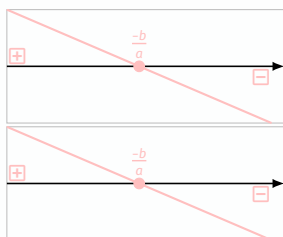
Code $\LaTeX$



Code $\LaTeX$

```
\begin{tikzpicture}
  \MiniSchemaSignes[Largeur=3.5,Hauteur=1.5,Code=da-,Racines=\tfrac{-b}{a},Couleur=pink]
\end{tikzpicture}

\MiniSchemaSignes*[Code=da-,Racines=\tfrac{-b}{a},Couleur=pink]<x=1.75cm,y=1.5cm>
```



Les *pictogrammes* pour les signes sont définis par défaut grâce à `fontawesome`, mais peuvent être redéfinis via `\def\pflfacplus` et `\def\pflfacminus`.

14.3 Intégration avec tkz-tab



Ces schémas peuvent être de plus utilisés, via la commande `\MiniSchemaSignesTkzTab` pour illustrer les signes obtenus dans un tableau de signes présentés grâce au package `tkz-tab`.

Pour des raisons internes, le fonctionnement de la commande `\MiniSchemaSignesTkzTab` est légèrement différent et, pour des raisons que j'ignore, le code est légèrement différent en interne (avec une *déconnexion* des caractères : et \) pour que la librairie TikZ `calc` puisse fonctionner (mystère pour le moment...)

Code $\text{\LaTeX}$

```
\begin{tikzpicture}
%commandes tkztab
\MiniSchemaSignesTkzTab[options]{numligne}[echelle][décalage horizontal]
\end{tikzpicture}
```



Les **<Clés>** pour le premier argument *optionnel* sont les mêmes que pour la version *initiale* de la commande précédente.

En ce qui concerne les autres arguments :

- le deuxième argument, *obligatoire*, est le numéro de la ligne à côté de laquelle placer le schéma ;
- le troisième argument, *optionnel* et valant **<0.85>** par défaut, est l'échelle à appliquer sur l'ensemble du schéma (à ajuster en fonction de la hauteur de la ligne) ;
- le quatrième argument, *optionnel* et valant **<1.5>** par défaut, est lié à l'écart horizontal entre le bord de la ligne du tableau et le schéma.

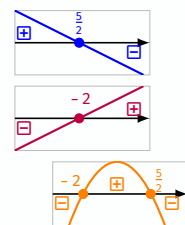
À noter que si l'un des arguments optionnels (le n°3 et/ou le n°4) sont utilisés, il vaut mieux préciser les 2 !

Code $\text{\LaTeX}$

```
\begin{center}
\begin{tikzpicture}
\tkzTabInit[[]]{$x$/1,$-2x+5$/1,$2x+4$/1,$p(x)$/1}{$-\infty$,$-2$,$2,5$,$+\infty$}
\tkzTabLine{,+ ,t ,+ ,z ,-,}
\tkzTabLine{- ,z ,+ ,t ,+ ,}
\tkzTabLine{- ,z ,+ ,z ,-,}
\MiniSchemaSignesTkzTab[Code=da-,Racines={\tfrac{5}{2}},Couleur=blue]{1}
\MiniSchemaSignesTkzTab[Code=da+,Racines={-2},Couleur=purple]{2}
\MiniSchemaSignesTkzTab[Code=pa-d+,Racines={-2/{\tfrac{5}{2}}},Couleur=orange]{%
{3}[0.85][2]
\end{tikzpicture}
\end{center}
```

Code $\text{\LaTeX}$

x	$-\infty$	-2	$2,5$	$+\infty$	
$-2x + 5$	$+$	$+$	0	$-$	
$2x + 4$	$-$	0	$+$	$+$	
$p(x)$	$-$	0	$+$	0	$-$



15 Ligne de convexité en lien avec tkz-tab

15.1 Idée



L'idée est d'obtenir une commande pour rajouter une ligne de *convexité* dans un tableau créé par `tkz-tab`. Cette commande est donc à utiliser dans un environnement correctement défini par les macros de `tkz-tab` car elle utilise les *nœuds* créés.

Code $\text{\LaTeX}$

```
\begin{tikzpicture}
%commandes tkztab
\tkzTabLineConvex(*){num ligne}{liste 'cases'}
\end{tikzpicture}
```

15.2 Commande



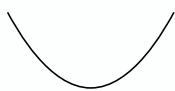
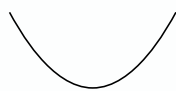
La version *étoilée* permet de forcer l'affichage des attributs en mode *texte*. Le premier argument, *obligatoire* et entre $\langle \{ \dots \} \rangle$, est le numéro de la ligne sur laquelle on souhaite afficher les attributs de convexité.

Le second argument, *obligatoire* et entre $\langle \{ \dots \} \rangle$, donne – en langage `tkz-tab`, le contenu de la ligne de convexité, avec les mêmes *règles* pour les commandes de `tkz-tab`, avec en plus :

- $\langle i^* \rangle$:= point d'inflexion (avec *texte*);
- $\langle i \rangle$:= point d'inflexion (sans *texte*);
- $\langle ccv \rangle$:= concave (schéma, ou *texte* en mode *étoilé*);
- $\langle cvx \rangle$:= convexe (schéma, ou *texte* en mode *étoilé*).

Code $\text{\LaTeX}$

```
\begin{tikzpicture}
\tkzTabInit[lgt=3,espcl=4]{\$x\$/1,\$f''(x)\$/1,Convexité\\de \$f\$/2}{\$0\$, \$1\$, \$3\$, \$5\$}
\tkzTabLine{,+,z,-,z,+}
\tkzTabLineConvex{2}{,cvx,i*,ccv,i*,cvx,}
\end{tikzpicture}
```

x	0	1	3	5	
f''(x)	+	0	-	0	+
Convexité de f					
		point d'inflexion	point d'inflexion		

Code $\LaTeX$

```
\begin{tikzpicture}
\tkzTabInit[lgt=3,espcl=4]{$x$/1,$f''(x)$/1,Convexit  de $f$/2}{$0$, $1$, $3$, $5$}
\tkzTabLine{+,z,-,z,+}
\tkzTabLineConvex*{2}{,cvx,i,ccv,i,cvx,}
\end{tikzpicture}
```

x	0	1	3	5	
$f''(x)$	+	0	-	0	+
Convexit�� de f	concave		concave		concave

Code $\LaTeX$

```
\begin{tikzpicture}
\tkzTabInit[lgt=3,espcl=4]{$x$/1,$f''(x)$/1,Convexit  de $f$/2}{$0$, $1$, $3$, $5$}
\tkzTabLine{+,z,-,d,+}
\tkzTabLineConvex*{2}{,cvx,t,ccv,d,cvx,}
\end{tikzpicture}
```

x	0	1	3	5
f''(x)	+	0	-	+
Convexit�� de f	concave		concave	

16 Autres compléments pour tkz-tab



Quelques commandes supplémentaires, développées par-dessus `tkz-tab`, pour annoter ou mettre en valeur des éléments d'un tableau de variations/signes. Comme pour `tkzTabLineConvex` (section précédente), elles s'utilisent à l'intérieur d'un environnement `tkz-tab` déjà en place, car elles s'appuient sur les *nœuds* qu'il crée.

16.1 Marquer une valeur interdite ou particulière

Code $\LaTeX$

```
\tkzTabMarqueVI[couleur]{liste de colonnes}
```



Place un petit **(D)** au-dessus de chacune des colonnes indiquées (par exemple pour repérer une valeur interdite dans un tableau de signes).

- l'argument *optionnel*, valant **(red)** par défaut, est la couleur du marqueur;
- l'argument *obligatoire* est la liste des numéros de colonnes concernées, séparés par des virgules (ex : **{2,4}**).

Code $\LaTeX$

```
\begin{tikzpicture}
\tkzTabInit[{$x$/1,$f(x)$/2}]{$-\infty$,$-2$,$2$,$+\infty$}
\tkzTabLine{+,d,-,d,+}
\tkzTabMarqueVI[red]{2,3}
\end{tikzpicture}
```

x	$-\infty$	-2	2	$+\infty$
		D	D	
$f(x)$	+	-	+	

16.2 Colorer une case ou une ligne

Code $\LaTeX$

```
\tkzTabColoreCases[couleur]{liste "ligne/colonne"}
\tkzTabColoreLigne[couleur]{liste de lignes}
```



`tkzTabColoreCases` colorie, en arrière-plan, une ou plusieurs cases *intérieures* précises du tableau : l'argument *obligatoire* est une liste de couples **(ligne/colonne)**, séparés par des virgules. Couleur par défaut : **(red !25)**.

`tkzTabColoreLigne` colorie une ou plusieurs lignes *entières* : l'argument *obligatoire* est une liste de numéros de lignes (en commençant à 0), séparés par des virgules. Couleur par défaut : **(gray !30)**.

Dans les deux cas, la couleur se règle avec l'argument *optionnel*.

Code $\LaTeX$

```
\begin{tikzpicture}
\tkzTabInit[]{$x$/1,$f'(x)$/1,$f(x)$/2}{$-\infty$,$-1$,$3$,$+\infty$} \tkzTabLine{,-,z,+,z,-,}
\tkzTabColoreLigne[yellow!20]{1}
\tkzTabColoreCases[green!20]{3/2}
\end{tikzpicture}
```

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

16.3 Ajouter un commentaire à droite d'une ligne

Code $\LaTeX$

```
\tkzTabComLigne[couleur]{numligne}{texte}[décalage]
```



Place un **<texte>** libre à droite d'une ligne du tableau, verticalement centré sur celle-ci.

- l'argument *optionnel*, valant **<blue>** par défaut, est la couleur du texte;
- le premier argument *obligatoire* est le numéro de la ligne concernée;
- le second argument *obligatoire* est le texte à afficher;
- le dernier argument *optionnel*, valant **<0.5>** par défaut, est le décalage horizontal par rapport au bord du tableau.

Code $\LaTeX$

```
\begin{tikzpicture}
\tkzTabInit[]{$x$/1,$f'(x)$/2}{$-\infty$,$1$,$+\infty$}
\tkzTabLine{,-,z,+,}
\tkzTabComLigne[teal]{1}{minimum en $1$}
\end{tikzpicture}
```

x	$-\infty$	1	$+\infty$
$f'(x)$	-	0	+

minimum en 1

16.4 Variante de MiniSchemaSignesTkzTab



`\tkzTabMiniSchemaSignes` rend le même service que `\tkzTabMiniSchemaSignesTkzTab` (voir plus haut, 14) – placer un mini-schéma de signes à côté d'une ligne du tableau – mais avec une implémentation *légèrement* différente (positionnement par un **<node>** plutôt qu'un **<pic>** directement sur le chemin) et un décalage par défaut différent (**<0.5>** au lieu de **<1.5>**). `\tkzTabMiniSignes`, de son côté, est un simple alias de `\tkzTabMiniSchemaSignesTkzTab`.

Code $\LaTeX$

```
\tkzTabMiniSchemaSignes[codeaidetikz]{numligne}[échelle][décalage]
```



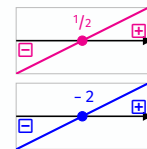
Mêmes arguments que `\MiniSchemaSignesTkzTab` : le premier (optionnel) est le `<code>` du mini-schéma (voir 14), le second (obligatoire) le numéro de ligne, le troisième (optionnel, `<0.85>` par défaut) l'échelle, et le dernier (optionnel, `<0.5>` par défaut) le décalage horizontal.

Code $\LaTeX$

```
\begin{tikzpicture}[double distance=0.7mm]
\tkzTabInit[]{$x$/0.75,$6x-3$/1,$x+2$/1,expr/1}{$-\infty$,$-2$,$\frac{1}{2}$,$+\infty$}
\tkzTabLine{,-,t,-,z,+}
\tkzTabLine{,-,z,+,t,+}
\tkzTabLine{+,d,-,z,+}
\tkzTabMiniSchemaSignes[Code=da+,Racines={\nicefrac{1}{2}},Couleur=magenta]{1}[0.875]
\tkzTabMiniSchemaSignes[Code=da+,Racines={-2},Couleur=blue]{2}[0.875]
\tkzTabMarqueVI{2}
\tkzTabColoreCases[red!25]{4/1,4/3} %attention aux numéros
\tkzTabColoreLigne[orange!5]{0} %attention au numéro
\tkzTabColoreLigne[magenta!5]{1} %attention au numéro
\tkzTabColoreLigne[blue!5]{2} %attention au numéro
\tkzTabComLigne[orange]{0}{\textleftarrow\enspace valeurs importantes de $x$}
\tkzTabComLigne[red]{3}{\textleftarrow\enspace mise en évidence des solutions}
\end{tikzpicture}
```

x	$-\infty$	-2	$\frac{1}{2}$	$+\infty$
$6x - 3$	-	-	0	+
$x + 2$	-	0	+	+
expr	+	-	0	+

← valeurs importantes de x



← mise en évidence des solutions

Thème

PRÉSENTATION DE CODES

Sixième partie

Présentation de codes

17 Précautions



L'idée est de proposer des environnements pour présenter du code :

- Python;
- PseudoCode.

Dans la mesure du possible (mis à part pour certains points avec l'utilisation des packages `python`), les environnements seront composés :

- dans une boîte `tcolorbox`;
- de deux styles : `CodeXXXX` ou `CodeXXXXAlt`;
- de clés pour paramétrer la **Largeur** et le début de la numérotation (**PremLigne**);
- d'une version étoilée pour ne pas numérotée les lignes;
- d'options éventuelles à donner en langage `tcolorbox`.



Avec la mise à jour `2.7.5` et la possibilité de modifier la numérotation des lignes, certains environnements ont vu leur fonctionnement légèrement modifié, donc il est conseillé d'être prudent avec les nouvelles spécificités.

Il est prévu, à plus ou moyen terme, d'uniformiser le fonctionnement de tous les environnements, mais cela demande de reprendre une bonne partie du code.

18 Code Python « simple » via le package listings

18.1 Introduction



Le package `listings` permet d'insérer et de formater du code, notamment du code Python. En *partenariat* avec `tcolorbox`, on peut donc présenter *joliment* du code Python!



Le package `listings` ne nécessite pas de compilation particulière, au contraire d'autres (comme `pythontex` ou `python`) qui seront présentés ultérieurement.



Les styles utilisés pour formater le code Python ne sont pas modifiables. Ils donnent un rendu proche de celui des packages comme `pythontex` ou `python`.

Donc, si plusieurs *méthodes* sont utilisées pour insérer du code Python (via les *méthodes* suivantes), le rendu pourra être légèrement différent.

18.2 Commande et options



L'environnement `CodePythonLst` permet de présenter du code Python, dans une `tcolorbox` avec deux styles particuliers (`2.5.8`).

Code $\LaTeX$

```
\begin{CodePythonLst}(*)[clés]{commandes tcolorbox}
...
\end{CodePythonLst}
```

Code $\LaTeX$

```
\begin{CodePythonLstAlt}(*)[clés]{commandes tcolorbox}
...
\end{CodePythonLstAlt}
```



Plusieurs **(arguments)** sont disponibles :

- la version *étoilée* qui permet de ne pas afficher les numéros de lignes;
- le premier argument (*optionnel*), comprend la clé **<Largeur>** de la `\tcbox` (**<\linewidth>** par défaut) et la clé **<PremLigne>** (**<1>** par défaut) et la clé **<EspaceNum>** (**<14pt>** par défaut);
- le second argument (*obligatoire*), concerne des **<options>** de la `\tcbox` en *langage* `tcolorbox`, comme l'alignement.



Les environnements créés par `\tcolorbox` et `\listings` ne sont pas compatibles avec les options **<gobble>** (pour supprimer les tabulations d'environnement), donc il faut bien penser à « aligner » le code à gauche, pour éviter des tabulations non esthétiques!

18.3 Insertion via un fichier « externe »



Pour des raisons pratiques, il est parfois intéressant d'avoir le code Python dans un fichier externe au fichier `\tex`, ou bien créé directement par le fichier `\tex` (via `\scontents`, notamment, mais non chargé par `\ProfLycee`).

Dans ce cas, il n'est pas nécessaire d'aligner le code « à gauche », en utilisant une commande alternative. Si cette méthode est utilisée, il ne faut oublier de charger le package `\scontents`, et être attentif à la syntaxe.

Code $\LaTeX$

```
\usepackage{scontents} %si script déclaré dans le fichier tex
...
\CodePythonLstFichier(*)[largeur]{commandes tcbox}{script}
```

18.4 Exemples

Code $\LaTeX$

```
\begin{CodePythonLst}{} %les {}, même vides, peuvent être nécessaires (bug avec # sinon !)
#environnement par défaut
nb = int(input("Saisir un entier positif"))
if (nb % 7 == 0) :
    print(f"{nb} est bien divisible par 7")
#endif

def f(x) :
    return x**2
\end{CodePythonLst}
```

Code $\LaTeX$

```
1 #environnement par défaut
2 nb = int(input("Saisir un entier positif"))
3 if (nb % 7 == 0) :
4     print(f"{nb} est bien divisible par 7")
5 #endif
6
7 def f(x) :
8     return x**2
```

Code Python

Code $\LaTeX$

```
\begin{CodePythonLst}[PremLigne=10]{}
nb = int(input("Saisir un entier positif"))
if (nb % 7 == 0) :
    print(f"{nb} est bien divisible par 7")
#endif
\end{CodePythonLst}
```

Code $\LaTeX$

 Code Python

```
10 nb = int(input("Saisir un entier positif"))
11 if (nb % 7 == 0) :
12     print(f"{nb} est bien divisible par 7")
13 #endif
```



Code $\LaTeX$

```
\begin{CodePythonLstAlt}*[Largeur=0.75\linewidth]{flush right}
#largeur de 75%, sans numéro, et aligné à droite
nb = int(input("Saisir un entier Python positif"))
if (nb % 7 == 0) :
    print(f"{nb} est bien divisible par 7")
#endif

def f(x) :
    return x**2
\end{CodePythonLstAlt}
```

Code $\LaTeX$

 Code Python

```
#largeur de 50%, sans numéro, et aligné à droite
nb = int(input("Saisir un entier Python positif"))
if (nb % 7 == 0) :
    print(f"{nb} est bien divisible par 7")
#endif

def f(x) :
    return x**2
```



Code $\LaTeX$

```
\begin{scontents}[overwrite,write-out=testscript.py]
# Calcul de la factorielle en langage Python
def factorielle(x):
    if x < 2:
        return 1
    else:
        return x * factorielle(x-1)

# rapidité de tracé
import matplotlib.pyplot as plt
import time
def trace_parabole_tableaux():
    depart=time.clock()
    X = [] # Initialisation des listes
    Y = []
    a = -2
    h = 0.001
    while a<2:
        X.append(a) # Ajout des valeurs
        Y.append(a*a) # au "bout" de X et Y
        a = a+h
    # Tracé de l'ensemble du tableau de valeurs
    plt.plot(X,Y,".b")
    fin=time.clock()
    return "Temps : " + str(fin-depart) + " s."
\end{scontents}

%environnement centré, avec numéros, largeur 9cm
\CodePythonLstFichier[9cm]{center}{testscript.py}
```

```
1 # Calcul de la factorielle en langage Python
2 def factorielle(x):
3     if x < 2:
4         return 1
5     else:
6         return x * factorielle(x-1)
7
8 # rapidité de tracé
9 import matplotlib.pyplot as plt
10 import time
11 def trace_parabole_tableaux():
12     depart=time.clock()
13     X = [] # Initialisation des listes
14     Y = []
15     a = -2
16     h = 0.001
17     while a<2:
18         X.append(a) # Ajout des valeurs
19         Y.append(a*a) # au "bout" de X et Y
20         a = a+h
21     # Tracé de l'ensemble du tableau de
    valeurs
22     plt.plot(X,Y,".b")
23     fin=time.clock()
24     return "Temps : " + str(fin-depart) + "
    s."
```

19 Code Python via le package `piton`

19.1 Introduction



`2.5.0` Cette section nécessite de charger la librairie `piton` dans le préambule.
`2.5.7` Une console Python est disponible, elle nécessite le package `pyluatex`, qui n'est pas chargé par ProfLycee, du fait de l'obligation de spécifier le *chemin* pour l'exécutable Python!

Code $\text{\LaTeX}$

```
\usepackage[executable=...]{pyluatex} %si utilisation de la console REPL
\useproflyclic{piton}
```



La librairie `piton` (qui charge `piton`, est compatible uniquement avec $\text{\LaTeX}$!) permet d'insérer du code Python avec une coloration syntaxique en utilisant la bibliothèque Lua LPEG.

En partenariat avec `tcolorbox`, on peut avoir une présentation de code Python!

`3.03c` Les options pour `(gobble)` sont disponibles et **à adapter en fonction des situations** (voir la doc de `piton`).

`3.04d` Afin de faciliter l'écriture de codes, et de s'affranchir de l'aspect *tabulation*, il est possible d'utiliser des **commandes** (avec les mêmes clés que les environnements) qui vont insérer le code inclus dans un fichier existant (ou créé avec `filecontents/scontents`).



Le package `piton` nécessite donc obligatoirement l'emploi de $\text{\LaTeX}$!
Ce package n'est chargé que si la compilation détectée est en $\text{\LaTeX}$!

`2.5.7` L'utilisation de la console **REPL** nécessite une compilation en `--shell-escape` ou `-write18`!

`2.5.7` Les packages `pyluatex` et `pythontex` utilisent des commandes de même nom, donc la présente documentation n'utilisera pas le package `pyluatex`. Une documentation annexe spécifique est disponible.

19.2 Présentation de code Python

Code $\text{\LaTeX}$

```
\begin{CodePiton}[clés]{options tbox}<option line-numbers>
...
\end{CodePiton}

%ou

\CodePitonFichier[clés]{fichier}<option line-numbers>[clés piton]
```



Plusieurs **clés** sont disponibles :

- la clé booléenne **<Lignes>** pour afficher ou non les numéros de lignes; défaut **<true>**
- 3.03c la clé **<Gobble>** pour spécifier des options liées au gobble, parmi **<nb/auto/>**, et à adapter en fonction des situations; défaut **<vide>**
- la clé **<Largeur>** qui correspond à la largeur de la tcbbox; défaut **<\linewidth>**
- la clé **<TaillePolice>** pour la taille des caractères; défaut **<\footnotesize>**
- la clé **<Alignement>** qui paramètre l'alignement de la tcbbox; défaut **<center>**
- 3.11e la clé **<Style>** (parmi **<Moderne / Classique / Onglet / Carbon / CarbonClair>**) pour changer le style; défaut **<Classique>**
- 2.5.7 le booléen **<Filigrane>** pour afficher, le logo  en filigrane; défaut **<false>**
- 2.5.7 le booléen **<BarreTitre>** (si **<Style=Moderne>**) pour afficher le titre; défaut **<true>**
- 2.5.7 le booléen **<Cadre>** (si **<Style=Moderne>**) pour afficher le cadre; défaut **<true>**
- 3.04f la clé **<CouleurNombres>** pour la couleur des nombres; défaut **<orange !75 !black>**
- 3.11e la clé **<Couleur>** pour la couleur du style **<Onglet>**; défaut **<red>**
- 3.11e la clé **<IconesCarbon>** pour les icônes des styles **<Carbon*>**;
- 4.01c les clés **<PremLigne/DerLigne>**.



Du fait du paramétrage des boîtes tcolorbox, il se peut que le rendu soit non conforme si elle doit être insérée dans une autre tcolorbox... (normalement corrigé en 2.6.9)!

La visualisation des différents styles est davantage présentée dans le document ProfLycee-exemples-pyluatex.pdf.



Même si les options Gobble sont disponibles, la version *par défaut* donne des résultats satisfaisants, à condition d'aligner le code au même niveau que le `\begin{...}\end{...}`.



Pour éviter des problèmes avec le code interprété par piton, les {} de l'argument obligatoire sont nécessaires au bon fonctionnement du code.

Code $\LaTeX$

```
\begin{CodePiton}{} %pour éviter un bug avec le caractère #
#environnement piton avec numéros de ligne, pleine largeur, style moderne
def arctan(x,n=10):
    if x < 0:
        return -arctan(-x) #> (appel récursif)
    elif x > 1:
        return pi/2 - arctan(1/x) #> (autre appel récursif)
    else:
        return sum( (-1)**k/(2*k+1)*x**(2*k+1) for k in range(n) )
\end{CodePiton}
```

 Code Python

```
1 #environnement piton avec numéros de ligne, pleine largeur, style classique
2 def arctan(x,n=10):
3     if x < 0:
4         return -arctan(-x) (appel récursif)
5     elif x > 1:
6         return pi/2 - arctan(1/x) (autre appel récursif)
7     else:
8         return sum( (-1)**k/(2*k+1)*x**(2*k+1) for k in range(n) )
```

Code L^AT_EX

```
\begin{CodePiton}[Style=Moderne,Filigrane]{}<start=10>
#environnement piton avec numéros (début=10), style moderne, filigrane
def arctan(x,n=10):
    if x < 0:
        return -arctan(-x) #> (appel récursif)
    elif x > 1:
        return pi/2 - arctan(1/x) #> (autre appel récursif)
    else:
        return sum( (-1)**k/(2*k+1)*x**(2*k+1) for k in range(n) )
\end{CodePiton}
```

Code Python

```
10 #environnement piton avec numéros, style moderne, filigrane
11 def arctan(x,n=10):
12     if x < 0:
13         return -arctan(-x) (appel récursif)
14     elif x > 1:
15         return pi/2 - arctan(1/x) (autre appel récursif)
16     else:
17         return sum( (-1)**k/(2*k+1)*x**(2*k+1) for k in range(n) )
```

Code L^AT_EX

```
\begin{CodePiton}[Alignement=flush right,Largeur=13cm,CouleurFond=cyan!5]{}
def f(x) :
    return x**2
\end{CodePiton}

\begin{CodePiton}[Alignement=flush left,Largeur=11cm]{}
def f(x) :
    return x**2
\end{CodePiton}

\begin{itemize} %Avec des indentations d'environnement :
    \item On essaye avec un \texttt{itemize} :
    %
    \begin{CodePiton}[Gobble=tabs,Largeur=12cm,Style=Classique,Cadre=false]{}
    def f(x) :
        return x**2
    \end{CodePiton}
    \item Et avec un autre \texttt{itemize} :
    %
    \begin{CodePiton}[Gobble=tabs,Largeur=12cm,Style=Moderne,Cadre=false,BarreTitre=false]{}
    #avec numéros, de largeur 12cm, centré, moderne, sans cadre/titre
    def f(x) :
        return x**2
    \end{CodePiton}
\end{itemize}
\vspace*{-\baselineskip}\leavevmode
```

Code Python

```
1 #avec numéros, de largeur 13cm, aligné à droite
2 def f(x) :
3     return x**2
```

Code Python

```
1 #avec numéros, de largeur 11cm, aligné à gauche
2 def f(x) :
3     return x**2
```

— On essaye avec un itemize :

Code Python

```
1 #avec numéros, de largeur 12cm, centré, classique, sans cadre
2 def f(x) :
3     return x**2
```

— Et avec un autre itemize :

```
1 #avec numéros, de largeur 12cm, centré, moderne, sans \  
  ↪ cadre/titre  
2 def f(x) :  
3     return x**2
```

19.3 Console en partenariat avec Pyluatex



2.5.7 Une console d'exécution (type REPL) est disponible, et la documentation associée est en marge de la présente documentation.

20 Pseudo-Code

20.1 Introduction



Le package `listings` permet d'insérer et de présenter du code, et avec `tcolorbox` on peut obtenir une présentation similaire à celle du code Python. Pour le moment la *philosophie* de la commande est un peu différente de celle du code Python, avec son système de **Clés**.

20.2 Présentation de Pseudo-Code



Les environnements `PseudoCode` ou `PseudoCodeAlt` permet de présenter du (pseudo-code) dans une `tcolorbox`, avec deux styles à disposition (`2.5.8`).



De plus, le package `listings` avec `tcolorbox` ne permet pas de gérer le paramètre *autogobble*, donc il faudra être vigilant quant à la position du code (pas de tabulation en fait...)

Code $\LaTeX$

```
\begin{PseudoCode}(*)[clés]{options tcolorbox}
%attention à l'indentation, gobble ne fonctionne pas...
...
\end{PseudoCode}
```

Code $\LaTeX$

```
\begin{PseudoCodeAlt}(*)[clés]{options tcolorbox}
%attention à l'indentation, gobble ne fonctionne pas...
...
\end{PseudoCodeAlt}
```



Plusieurs **arguments** (optionnels) sont disponibles :

- la version *étoilée* qui permet de ne pas afficher les numéros de lignes;
- le 1^{er} argument (*optionnel*), comprend la clé **Largeur** de la `tcolorbox` (`\linewidth` par défaut) et la clé **PremLigne** (`1` par défaut) et la clé **EspaceNum** (`14pt` par défaut);
- `2.7.5` une clé booléenne **Couleur** est également disponible pour mettre en évidence trois niveaux (elles peuvent être redéfinies) de mots clés en pseudo-code (`false` par défaut);
- `2.5.8` l'argument obligatoire entre `{...}` concerne les **options** de la `tcolorbox`.

Code $\LaTeX$

```
%en pas oublier les {}, même vides !
\begin{PseudoCode}{} %non centré, de largeur par défaut (12cm) avec lignes
List = [...]          # à déclarer au préalable
n = longueur(List)
Pour i allant de 0 à n-1 Faire
  Afficher(List[i])
FinPour
\end{PseudoCode}
```

Code $\LaTeX$

```
1 List ← [...]          # à déclarer au préalable
2 n ← longueur(List)
3 Pour i allant de 0 à n-1 Faire
4   Afficher(List[i])
5 FinPour
```

Pseudo-Code

Code $\LaTeX$

```

\begin{PseudoCodeAlt}[Largeur=15cm,PremLigne=7,Couleur]{center} %centré, de largeur 15cm
List = [...]          # à déclarer au préalable
n = longueur(List)
Pour i allant de 0 à n-1 Faire
    Afficher(List[i])
FinPour
\end{PseudoCodeAlt}

```

Code $\LaTeX$

```

</> PseudoCode
7 List ← [...]          # à déclarer au préalable
8 n ← longueur(List)
9 Pour i allant de 0 à n-1 Faire
10     Afficher(List[i])
11 FinPour

```

20.3 Compléments



À l'instar de packages existants, la *philosophie* ici est de laisser l'utilisateur gérer son langage pseudo-code.

J'ai fait le choix de ne pas forcément définir des mots clés à mettre en valeur car cela reviendrait à *imposer* des choix! Donc ici, pas de coloration syntaxique (uniquement via la clé $\langle$ **Couleur** $\rangle$) ou de mise en évidence de mots clés, uniquement un formatage basique!

Code $\LaTeX$

```

%couleurs par défaut des mots clés, modifiables si besoin
\colorlet{MotsClesPseudoCodeA}{blue!75}
\colorlet{MotsClesPseudoCodeB}{green!50!black}
\colorlet{MotsClesPseudoCodeChaine}{red!75}

```



Le style `listings` utilisé par la commande a l'option $\langle$ **mathescape** $\rangle$ activée, et accessible grâce aux délimiteurs $\langle$ **(...*)** $\rangle$.

Cela permet d'insérer du code $\LaTeX$ dans l'environnement `PseudoCode` (attention au fontes par contre!).

Code $\LaTeX$

```

\begin{PseudoCode}*[Largeur=12cm]{} % pour éviter un bug avec #
#Utilisation du mode mathescape
Afficher (*\og*) .....(*\fg*)
m = (*$\tfrac{\texttt{1}}{\texttt{2}}$*)
\end{PseudoCode}

```

Code $\LaTeX$

```

#Utilisation du mode mathescape
Afficher « ..... »
m ←  $\frac{1}{2}$ 

```

Pseudo-Code

21 PseudoCode via le package piton

21.1 Introduction



3.01f Avec une version supérieure ou égale à 2.4 du package `piton`, il est possible d'utiliser un langage minimal, ce qui permet de pouvoir *formater* du *pseudocode*.

3.04d Afin de faciliter l'écriture de codes, et de s'affranchir de l'aspect *tabulation*, il est possible d'utiliser des **commandes** (avec les mêmes clés que les environnements) qui vont insérer le code inclus dans un fichier existant (ou créé avec `filecontents/scontents`).

Code $\LaTeX$

```
\begin{PseudoCodePiton}[options][options tcbox]<option line-numbers>
...
\end{PseudoCodePiton}

%ou

\PseudoCodePitonFichier[clés]{fichier}<option line-numbers>[options piton]
```



Le package `piton` nécessite donc obligatoirement l'emploi de $\text{Lua}\LaTeX$! Ce package n'est chargé que si la compilation détectée est en $\text{Lua}\LaTeX$!

21.2 Présentation de PseudoCode



Plusieurs **clés** sont disponibles :

- la clé booléenne **<Lignes>** pour afficher ou non les numéros de lignes; défaut **<true>**
- **3.03c** la clé **<Gobble>** pour spécifier des options liées au gobble, parmi **<nb/auto/>**, et à adapter **en fonction des situations**; défaut **<vide>**
- la clé **<Largeur>** qui correspond à la largeur de la `tcbox`; défaut **<\linewidth>**
- la clé **<TaillePolice>** pour la taille des caractères; défaut **<\footnotesize>**
- la clé **<Alignement>** qui paramètre l'alignement de la `tcbox`; défaut **<center>**
- le booléen **<Filigrane>** pour afficher, le logo  en filigrane; défaut **<false>**
- le booléen **<BarreTitre>** pour afficher le titre; défaut **<true>**
- le booléen **<Cadre>** pour afficher le cadre; défaut **<true>**
- le booléen **<Couleurs>** pour colorer les *mots clés* éventuels. défaut **<true>**
- **3.11e** la clé **<Style>** (parmi **<Classique / Onglet / Carbon / CarbonClair>**) pour changer le style; défaut **<Classique>**
- **3.11e** la clé **<Couleur>** pour la couleur du style **<Onglet>**; défaut **<red>**
- **3.11e** la clé **<IconesCarbon>** pour les icônes des styles **<Carbon*>**;
- **4.01c** les clés **<PremLigne/DerLigne>**.



À l'instar de packages existants, la *philosophie* ici est de pouvoir laisser l'utilisateur gérer *son* langage pseudo-code.

3.01f Il est possible, avec `piton` (version supérieure à 2.4), de spécifier une liste de mots clés.

Code $\LaTeX$

```
%couleurs par défaut des mots clés, modifiables si besoin
\colorlet{MotsClesPseudoCodeA}{blue!75}
\colorlet{MotsClesPseudoCodeB}{green!50!black}
\colorlet{MotsClesPseudoCodeChaine}{red!75}

%liste de mots clés en mode [Couleurs=true]
\SetPitonIdentifiant[minimal]%
{
  %
  Algorithme,Fonction,Début,Paramètre,Paramètres,Faire,
  Fin,Si,Pour,TantQue,que,alors,Alors,Sinon,SinonSi,
  FinSi,FinPour,FinTantQue,TantQue,Variable,Variables,Procédure
}
{\color{MotsClesPseudoCodeA}}

\SetPitonIdentifiant[minimal]%
{
  %
  Afficher,Retourner,Saisir
}
{\color{MotsClesPseudoCodeB}}
```

21.3 Exemples



Même si les options Gobble sont disponibles, la version *par défaut* donne des résultats satisfaisants, à condition d'aligner le code au même niveau que le `\begin{...}\end{...}`.

Code $\LaTeX$

```
\begin{PseudoCodePiton}[Filigrane][<start=10>
Algorithme : Périmètre de rectangles
Variables : Long, Larg, Perim (réels)
           Choix (chaîne) # pour refaire ou non l'exécution

Début
  # initialisation de l'indicateur pour entrer dans la boucle
  Choix ← "o"
  # boucle TantQue permettant de refaire le traitement selon le choix
  TantQue Choix = "o" Faire
    # traitement
    Afficher("Donner les dimensions du rectangle") et Saisir(Long,Larg)
    Perim ← 2 × (Long + Larg)
    Afficher("Le périmètre du rectangle est", Perim)
    #saisie du choix de recommencer ou non
    Afficher("Voulez-vous continuer (o/n) ?") et Saisir(Choix)
  FinTantQue
Fin
\end{PseudoCodePiton}
```

Pseudo-Code

```
10 Algorithme : Périmètre de rectangles
11 Variables : Long, Larg, Perim (réels)
12           Choix (chaîne) # pour refaire ou non l'exécution
13
14 Début
15   # initialisation de l'indicateur pour entrer dans la boucle
16   Choix ← "o"
17   # boucle TantQue permettant de refaire le traitement selon le choix
18   TantQue Choix = "o" Faire
19     # traitement
20     Afficher("Donner les dimensions du rectangle") et Saisir(Long,Larg)
21     Perim ← 2 × (Long + Larg)
22     Afficher("Le périmètre du rectangle est", Perim)
23     #saisie du choix de recommencer ou non
24     Afficher("Voulez-vous continuer (o/n) ?") et Saisir(Choix)
25   FinTantQue
26 Fin
```

```

\begin{PseudoCodePiton}[Lignes=false,Filigrane=false,Couleurs=false]{}
Algorithme : Périmètre de rectangles
Variables : Long, Larg, Perim (réels)
            Choix (chaîne) # pour refaire ou non l'exécution

Début
  # initialisation de l'indicateur pour entrer dans la boucle
  Choix ← "o"
  # boucle TantQue permettant de refaire le traitement selon le choix
  TantQue Choix = "o" Faire
    # traitement
    Afficher("Donner les dimensions du rectangle") et Saisir(Long,Larg)
    Perim ← 2 × (Long + Larg)
    Afficher("Le périmètre du rectangle est", Perim)
    #saisie du choix de recommencer ou non
    Afficher("Voulez-vous continuer (o/n) ?") et Saisir(Choix)
  FinTantQue
Fin
\end{PseudoCodePiton}

```

Pseudo-Code

```

Algorithme : Périmètre de rectangles
Variables : Long, Larg, Perim (réels)
            Choix (chaîne) # pour refaire ou non l'exécution

Début
  # initialisation de l'indicateur pour entrer dans la boucle
  Choix ← "o"
  # boucle TantQue permettant de refaire le traitement selon le choix
  TantQue Choix = "o" Faire
    # traitement
    Afficher("Donner les dimensions du rectangle") et Saisir(Long,Larg)
    Perim ← 2 × (Long + Larg)
    Afficher("Le périmètre du rectangle est", Perim)
    #saisie du choix de recommencer ou non
    Afficher("Voulez-vous continuer (o/n) ?") et Saisir(Choix)
  FinTantQue
Fin

```

22 Code et console Python, à la manière de Thonny

22.1 Introduction



3.02e L'idée est de proposer des environnements (l'un pour le code et l'autre pour la console) pour présenter du code Python à la manière d'un éditeur, du style Thonny.

3.04d Afin de faciliter l'écriture de codes, et de s'affranchir de l'aspect *tabulation*, il est possible d'utiliser des **commandes** (avec les mêmes clés que les environnements) qui vont insérer le code inclus dans un fichier existant (ou créé avec `filecontents/scontents`).

Code $\LaTeX$

```
\begin{PitonThonnyEditor}<clés>[options tcolorbox]{largeur}
...
\end{PitonThonnyEditor}

%ou

\PitonThonnyEditorFichier<clés>[largeur]{fichier}
```

Code $\LaTeX$

```
\begin{PitonThonnyConsole}<clés>[options tcolorbox]{largeur}
...
\end{PitonThonnyConsole}
```



Le package `piton` nécessite donc obligatoirement l'emploi de $\text{Lua}\LaTeX$! Ce package n'est chargé que si la compilation détectée est en $\text{Lua}\LaTeX$! L'utilisation de ces environnements nécessitent l'utilisation de `pyluatex`, et donc une compilation adaptée est nécessaire (`--shell-escape`).



Les exemples de cette section sont disponibles en marge de la présente documentation. La présente documentation ne présente que les codes d'entrées, sans les sorties (ceci étant dû à la spécificité d'utilisation de `pyluatex`.)

22.2 Présentation de code, style éditeur

Code $\LaTeX$

```
\begin{PitonThonnyEditor}<clés>[options tcolorbox]{largeur}
...
\end{PitonThonnyEditor}
```



Le premier argument, optionnel et entre `<...>`, propose les **clés** :

- la clé `<NomFichier>` pour afficher le nom du fichier dans le cartouche *éditeur*; défaut `<script.py>`
- **3.04f** la clé `<CouleurNombres>` pour la couleur des nombres. défaut `<orange !75 !black>`
- **3.03c** la clé `<Gobble>` pour spécifier des options liées au gobble, parmi `<nb/auto/...>`, et à adapter en fonction des situations. défaut `<vide>`

Le deuxième argument, optionnel et entre `<...>`, correspond à des options à passer à la `tcolorbox`.
Le dernier argument, obligatoire et entre `{...}`, correspond à la largeur de la `tcolorbox`.



Même si les options Gobble sont disponibles, la version *par défaut* donne des résultats satisfaisants, à condition d'aligner le code au même niveau que le `\begin{...}\end{...}`.

Code $\LaTeX$

```
\begin{PitonThonnyEditor}<NomFichier=tpcapytale.py>{15cm}
#PROJET CAPYTALE
from math import gcd

def est_duffy(n) :
    nb_div, somme_div = 0, 0
    for i in range(1, n+1) :
        if n % i == 0 :
            nb_div += 1
            somme_div += i
    if gcd(somme_div, n) == 1 :
        return True
    else :
        return False
\end{PitonThonnyEditor}
```

22.3 Présentation de code, style console



 Une console d'exécution (type REPL et style Thonny) est disponible, et la documentation associée est en marge de la présente documentation.

Code $\LaTeX$

```
\begin{PitonThonnyConsole}<clés>[options tcolorbox]{largeur}
...
\end{PitonThonnyConsole}
```



Le premier argument, optionnel et entre $\langle \dots \rangle$, propose les (clés) :

- la clé $\langle \text{NomConsole} \rangle$ pour afficher le nom de la console; défaut $\langle \text{console} \rangle$
-  la clé $\langle \text{CouleurNombres} \rangle$ pour la couleur des nombres. défaut $\langle \text{orange !75 !black} \rangle$
- la clé $\langle \text{IntroConsole} \rangle$ pour afficher le message d'accueil de la console.

Le deuxième argument, optionnel et entre $\langle \dots \rangle$, correspond à des options à passer à la tcolorbox .
Le dernier argument, obligatoire et entre $\langle \dots \rangle$, correspond à la largeur de la tcolorbox .

Code $\LaTeX$

```
%code python
\begin{python}
from math import gcd
def est_duffy(n) :
    nb_div, somme_div = 0, 0
    for i in range(1, n+1) :
        if n % i == 0 :
            nb_div += 1
            somme_div += i
    if gcd(somme_div, n) == 1 :
        return True
    else :
        return False
\end{python}

%console style Thonny
\begin{PitonThonnyConsole}<NomFichier=tpcapytale.py>{15cm}
#Run tpcapytale.py
est_duffy(6)
est_duffy(13)
\end{PitonThonnyConsole}
```

22.4 Exemple

tpcapytale.py ×

```
1 #PROJET CAPYTALE
2 from math import gcd
3
4 def est_duffy(n) :
5     nb_div = 0
6     somme_div = 0
7     for i in range(1, n+1) :
8         if n % i == 0 :
9             nb_div += 1
10            somme_div += i
11 if gcd(somme_div, n) == 1 :
12     return True
13 else :
14     return False
```

console ×

```
Python 3.11.6 /usr/bin/python
>>> #Run tpcapytale.py
>>> est_duffy(6)
False
>>> est_duffy(13)
True
>>> est_duffy(265)
True
>>>
>>> from random import randint
>>> nb = randint(1,100000)
>>> nb, est_duffy(nb)
(42563, True)
```

23 Cartouche Capytale

23.1 Introduction



L'idée est d'obtenir des cartouches tels que Capytale les présente, pour partager un code afin d'accéder à une activité Python.

23.2 Commandes

Code $\LaTeX$

```
\CartoucheCapytale(*)[options]{code capytale}
```



Peu d'options pour ces commandes :

- la version *étoilée* qui permet de passer de la police $\langle\text{sffamily}\rangle$ à la police $\langle\text{ttfamily}\rangle$, et donc dépendante des fontes du document;
- le deuxième, *optionnel*, permet de rajouter des caractères après le code (comme un espace);
défaut $\langle\text{vide}\rangle$
- le troisième, *obligatoire*, est le code capytale à afficher.

Code $\LaTeX$

```
\CartoucheCapytale{abcd-12345}           %lien simple, en sf
\CartoucheCapytale[~]{abcd-12345}         %lien avec ~ à la fin, en sf
\CartoucheCapytale*{abcd-12345}           %lien simple, en tt
\CartoucheCapytale*[~]{abcd-12345}        %lien avec ~ à la fin, en tt
```

Code $\LaTeX$

```
abcd-12345 
abcd-12345 
abcd-12345 
abcd-12345 
```



Le cartouche peut être « cliquable » grâce à $\LaTeX$ `\href`.

Code $\LaTeX$

```
\usepackage{hyperref}
\urlstyle{same}
...
\href{https://capytale2.ac-paris.fr/web/c/abcd-12345}{\CartoucheCapytale{abcd-12345}}
```

Code $\LaTeX$

```
abcd-12345 
```

Thème

OUTILS POUR LA GÉOMÉTRIE

Septième partie

Outils pour la géométrie

24 Pavé droit « simple »

24.1 Introduction



L'idée est d'obtenir un pavé droit, dans un environnement TikZ, avec les nœuds créés et nommés directement pour utilisation ultérieure.

24.2 Commandes

Code $\text{\LaTeX}$

```
\begin{tikzpicture}[options tikz]
  \PaveTikz[options]
  ...
\end{tikzpicture}
```

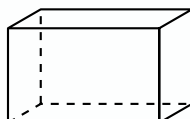


Quelques **clés** sont disponibles pour cette commande :

- **⟨Largeur⟩** : largeur du pavé; défaut **⟨2⟩**
- **⟨Profondeur⟩** : profondeur du pavé; défaut **⟨1⟩**
- **⟨Hauteur⟩** : hauteur du pavé; défaut **⟨1.25⟩**
- **⟨Angle⟩** : angle de fuite de la perspective; défaut **⟨30⟩**
- **⟨Fuite⟩** : coefficient de fuite de la perspective; défaut **⟨0.5⟩**
- **⟨Sommets⟩** : liste des sommets (avec délimiteur §!); défaut **⟨A§B§C§D§E§F§G§H⟩**
- **⟨Math⟩** : booléen pour forcer le mode math des sommets; défaut **⟨false⟩**
- **⟨Epaisseur⟩** : épaisseur des arêtes (en *langage simplifié TikZ*); défaut **⟨thick⟩**
- **⟨Aff⟩** : booléen pour afficher les noms des sommets; défaut **⟨false⟩**
- **⟨Plein⟩** : booléen pour ne pas afficher les arêtes *invisibles*; défaut **⟨false⟩**
- **⟨Cube⟩** : booléen pour préciser qu'il s'agit d'un cube (seule la valeur **⟨Largeur⟩** est util(isé)e). défaut **⟨false⟩**

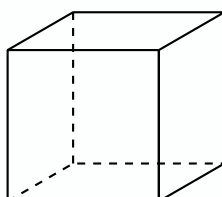
Code $\text{\LaTeX}$

```
%code tikz
\PaveTikz
```



Code $\text{\LaTeX}$

```
%code tikz
\PaveTikz[Cube,Largeur=2]
```





La ligne est de ce fait à insérer dans un environnement `TikZ`, avec les options au choix pour cet environnement.

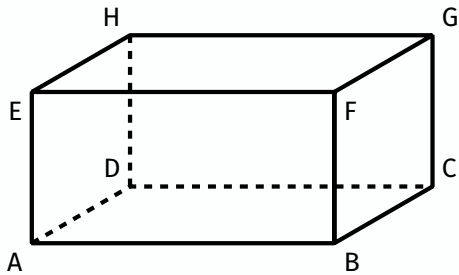
Le code crée les nœuds relatifs aux sommets, et les nomme comme les sommets, ce qui permet de les réutiliser pour éventuellement compléter la figure!

24.3 Influence des paramètres

Code $\text{\LaTeX}$

```
\begin{tikzpicture}[line join=bevel]
  \PaveTikz[Aff,Largeur=4,Profondeur=3,Hauteur=2,Epaisseur={ultra thick}]
\end{tikzpicture}
```

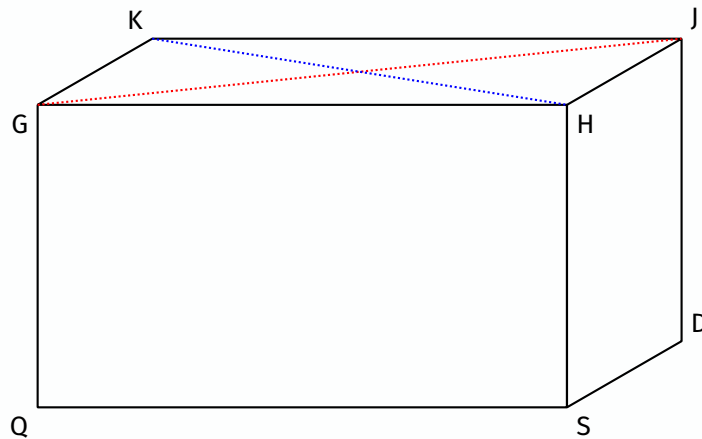
Code $\text{\LaTeX}$



Code $\text{\LaTeX}$

```
\begin{center}
\begin{tikzpicture}[line join=bevel]
  \PaveTikz[Plein,Aff,Largeur=7,Profondeur=3.5,Hauteur=4,Sommets=Q$S$D$F$G$H$J$K]
  \draw[thick,red,densely dotted] (G)--(J) ;
  \draw[thick,blue,densely dotted] (K)--(H) ;
\end{tikzpicture}
\end{center}
```

Code $\text{\LaTeX}$



25 Tétraèdre « simple »

25.1 Introduction



L'idée est d'obtenir un tétraèdre, dans un environnement *TikZ*, avec les nœuds créés et nommés directement pour utilisation ultérieure.

25.2 Commandes

Code $\text{\LaTeX}$

```
\begin{tikzpicture}[options tikz]
  \TetraedreTikz[options]
  ...
\end{tikzpicture}
```

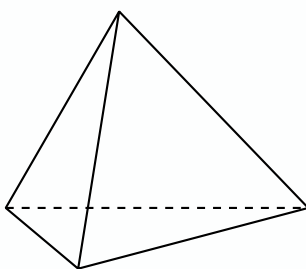


Quelques **clés** sont disponibles pour cette commande :

- **⟨Largeur⟩** : largeur du tétraèdre; défaut **⟨4⟩**
- **⟨Profondeur⟩** : profondeur du tétraèdre; défaut **⟨1.25⟩**
- **⟨Hauteur⟩** : hauteur du tétraèdre; défaut **⟨3⟩**
- **⟨Alpha⟩** : angle du sommet de devant; défaut **⟨40⟩**
- **⟨Beta⟩** : angle du sommet du haut; défaut **⟨60⟩**
- **⟨Sommets⟩** : liste des sommets (avec délimiteur §!); défaut **⟨A§B§C§D⟩**
- **⟨Math⟩** : booléen pour forcer le mode math des sommets; défaut **⟨false⟩**
- **⟨Epaisseur⟩** : épaisseur des arêtes (en *langage simplifié TikZ*); défaut **⟨thick⟩**
- **⟨Aff⟩** : booléen pour afficher les noms des sommets; défaut **⟨false⟩**
- **⟨Plein⟩** : booléen pour ne pas afficher l'arête invisible. défaut **⟨false⟩**

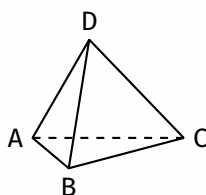
Code $\text{\LaTeX}$

```
%code tikz
\TetraedreTikz
```



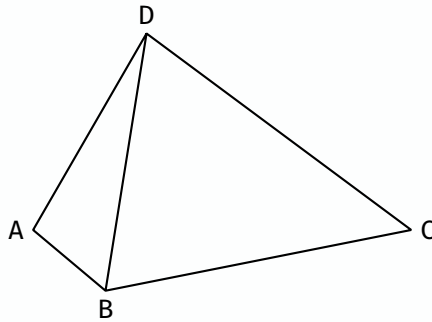
Code $\text{\LaTeX}$

```
%code tikz
\TetraedreTikz[Aff,Largeur=2,Profondeur=0.625,Hauteur=1.5]
```



Code $\text{\LaTeX}$

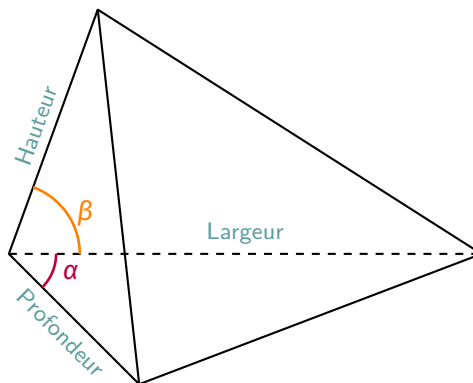
```
%code tikz
\TetraedreTikz[Plein,Aff,Largeur=5,Beta=60]
```



25.3 Influence des paramètres



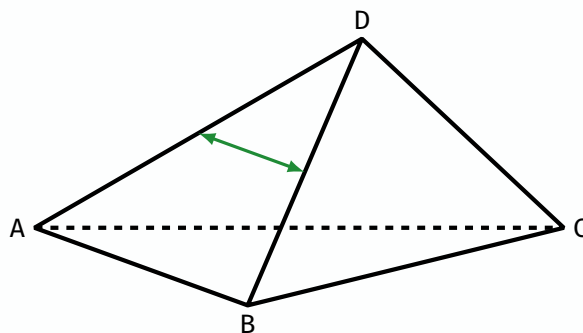
Pour *illustrer* un peu les **clés**, un petit schéma, avec les différents paramètres utiles.



Code $\text{\LaTeX}$

```
\begin{center}
\begin{tikzpicture}[line join=bevel]
\TetraedreTikz[Aff,Largeur=7,Profondeur=3,Hauteur=5,Epaisseur={ultra thick},Alpha=20,Beta=30]
\draw[very thick,CouleurVertForet,<->,>=latex] ($(A)!0.5!(D)$)--($(B)!0.5!(D)$) ;
\end{tikzpicture}
\end{center}
```

Code $\text{\LaTeX}$



26 Cercle trigo

26.1 Idée

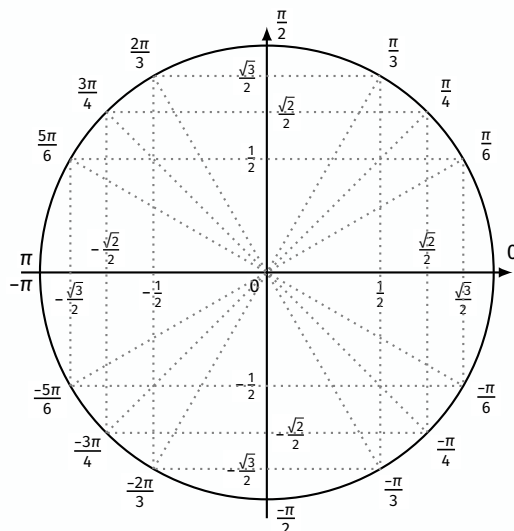


L'idée est d'obtenir une commande pour tracer (en TikZ) un cercle trigonométrique, avec personnalisation des affichages.

Comme pour les autres commandes TikZ, l'idée est de laisser l'utilisateur définir et créer son environnement TikZ, et d'insérer la commande `\CercleTrigo` pour afficher le cercle.

Code $\text{\LaTeX}$

```
%code tikz
\CercleTrigo
```



26.2 Commandes

Code $\text{\LaTeX}$

```
...
\begin{tikzpicture}[options tikz]
...
\CercleTrigo[clés]
...
\end{tikzpicture}
```



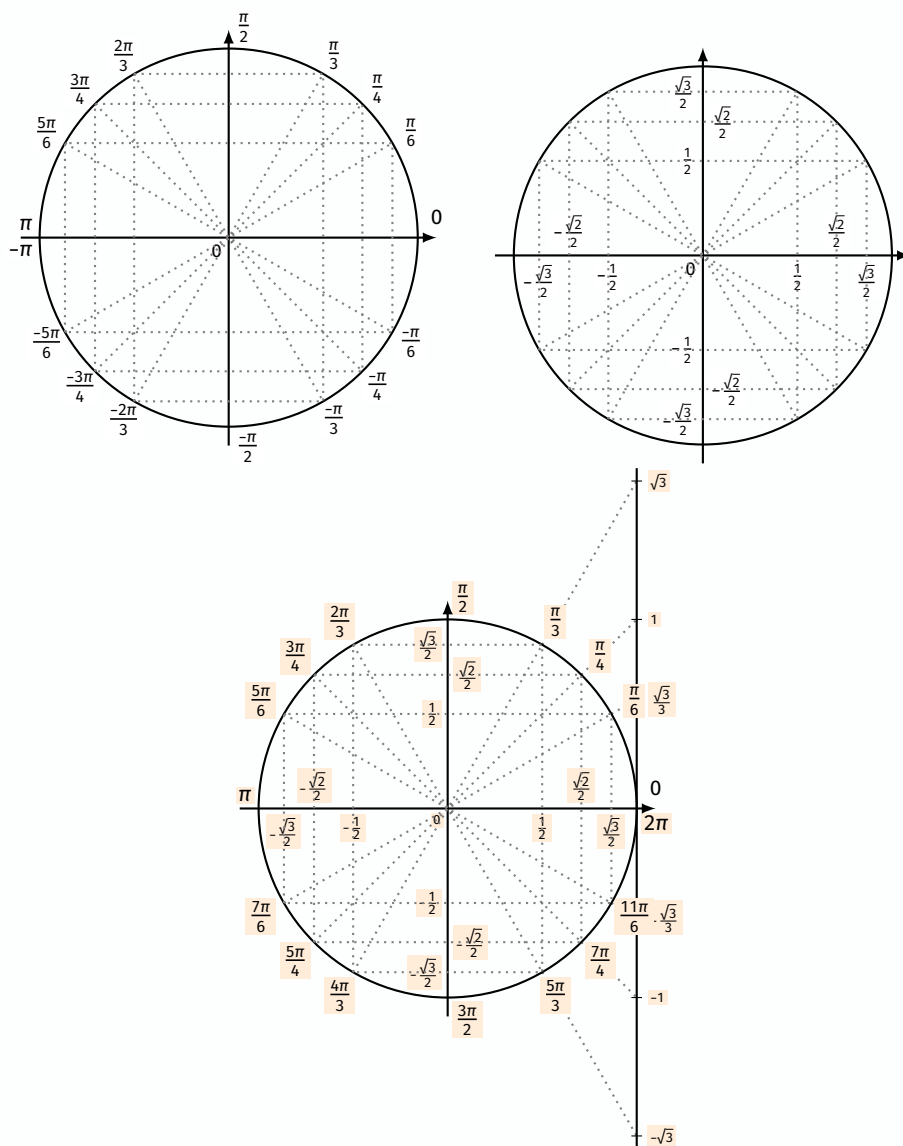
Plusieurs **(Clés)** sont disponibles pour cette commande :

- la clé **(Rayon)** qui définit le rayon du cercle; défaut **(3)**
- la clé **(Epaisseur)** qui donne l'épaisseur des traits de base; défaut **(thick)**
- la clé **(Marge)** qui est l'écartement de axes; défaut **(0.25)**
- la clé **(TailleValeurs)** qui est la taille des valeurs remarquables; défaut **(scriptsize)**
- la clé **(TailleAngles)** qui est la taille des angles; défaut **(footnotesize)**
- la clé **(CouleurFond)** qui correspond à la couleur de fond des labels; défaut **(white)**
- la clé **(Decal)** qui correspond au décalage des labels par rapport au cercle; défaut **(10pt)**
- un booléen **(MoinsPi)** qui bascule les angles « -pi » à « zero deux pi »; défaut **(true)**
- un booléen **(AffAngles)** qui permet d'afficher les angles; défaut **(true)**
- un booléen **(AffTraits)** qui permet d'afficher les *traits de construction*; défaut **(true)**
- un booléen **(ValeursTan)** qui permet de rajouter les éléments de la tangente; défaut **(false)**
- un booléen **(AffValeurs)** qui permet d'afficher les valeurs remarquables. défaut **(true)**

Code $\text{\LaTeX}$

```
\begin{center}
\begin{tikzpicture}[line join=bevel]
  \CercleTrigo[Rayon=2.5,AffValeurs=false,Decal=8pt]
\end{tikzpicture}
~~~~~
\begin{tikzpicture}[line join=bevel]
  \CercleTrigo[Rayon=2.5,AffAngles=false]
\end{tikzpicture}
~~~~~
\begin{tikzpicture}[line join=bevel]
  \CercleTrigo[Rayon=2.5,MoinsPi=false,CouleurFond=orange!15]
\end{tikzpicture}
\end{center}
```

Code $\text{\LaTeX}$



26.3 Équations trigos



En plus des **Clés** précédentes, il existe un complément pour *visualiser* des solutions d'équations simples du type $\cos(x) = \dots$ ou $\sin(x) = \dots$.



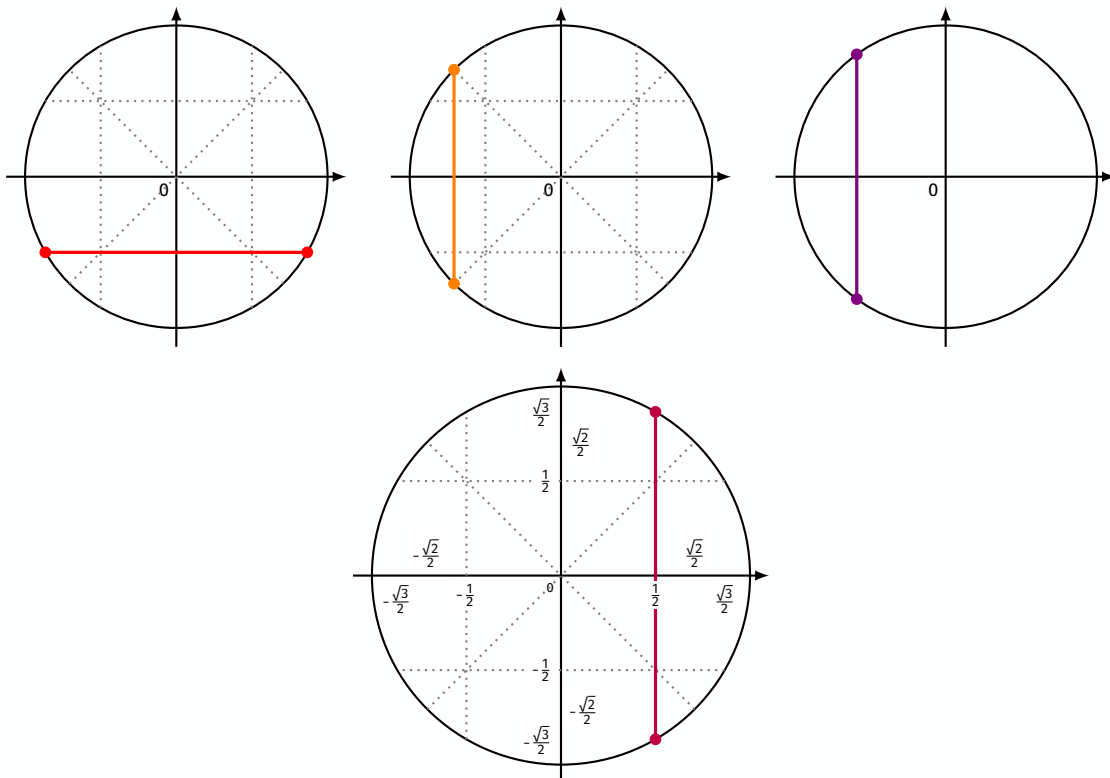
Les **Clés** pour cette possibilité sont :

- un booléen **Equationcos** pour activer « cos = »; défaut **false**
- un booléen **Equationsin** pour activer « sin = »; défaut **false**
- la clé **sin** qui est la valeur de l'angle (en degrés) du sin; défaut **30**
- la clé **cos** qui est la valeur de l'angle (en degrés) cos; défaut **45**
- 2.6.2 un booléen **AffTraitsEq** qui permet d'afficher les *traits de construction secondaires* pour les équations; défaut **true**
- la clé **CouleurSol** qui est la couleur des *solutions*. défaut **blue**

Code $\text{\LaTeX}$

```
\begin{center}
\begin{tikzpicture}
  \CercleTrigo[%
    AffAngles=false,AffValeurs=false,Rayon=2,Equationsin,sin=-30, CouleurSol=red]
\end{tikzpicture}
\begin{tikzpicture}
  \CercleTrigo[%
    AffAngles=false,AffValeurs=false,AffTraits=false,Rayon=2,Equationcos,cos=135, CouleurSol=orange]
\end{tikzpicture}
\begin{tikzpicture}
  \CercleTrigo[%
    AffAngles=false,AffValeurs=false,AffTraits=false,AffTraitsEq=false,Rayon=2,
    Equationcos,cos=126,CouleurSol=violet]
\end{tikzpicture}
\begin{tikzpicture}
  \CercleTrigo[%
    AffTraits=false,AffAngles=false,Rayon=2.5,Equationcos,cos=60,CouleurSol=purple, TailleValeurs=\tiny]
\end{tikzpicture}
\end{center}
```

Code $\text{\LaTeX}$



27 Schémas type de géométrie dans l'espace

27.1 Idée



3.02f L'idée est d'obtenir une commande pour obtenir *facilement* des petits schémas pour illustrer des propriétés de géométrie dans l'espace.

Les commandes sont accessibles en chargeant la librairie **espace**.

Les schémas sont réalisés en METAPOST, donc une compilation adaptée est nécessaire :

- en Lua $\text{\TeX}$!



3.03c Le code METAPOST est largement inspiré du travail de David Nivaud sur <https://melusine.eu.org/syracuse/exemples/> et une comptabilité avec ProfCollege a été déployée.

Code $\text{\LaTeX}$

```
\useproflyclib{espace}  
...  
\SchemaEspace(*)[clé]{type}
```

27.2 Commande



La version étoilée force l'affichage des labels des plans en mode **mathscr** (si chargé), sinon il se fait en mode **mathcal**.

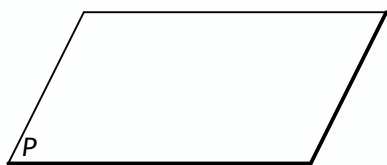
La **<Clé>** disponible est la clé **<Echelle>** (en cm) pour tracer le schéma.

L'argument obligatoire, et entre **{...}**, est le type de schéma souhaité, parmi :

- plan;
- interplans;
- plan3points;
- plandroitessecantes;
- plandroitepoint;
- plandroitespara;
- droitesnoncopla;
- incidence;
- droiteparaplans;
- toit;
- planspara;
- droiteplanpara;
- droitesortho;
- droiteorthoplan;
- plansparadroiteortho;
- plansparadroitesortho;
- plansperp;
- plansperpplan.

Code $\text{\LaTeX}$

```
\SchemaEspace{plan}
```



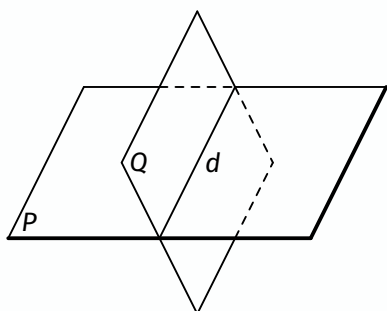
Code $\text{\LaTeX}$

```
\SchemaEspace[Echelle=1.5]{plan}
```



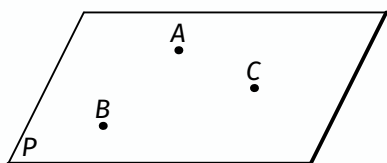
Code $\text{\LaTeX}$

```
\SchemaEspace{interplans}
```



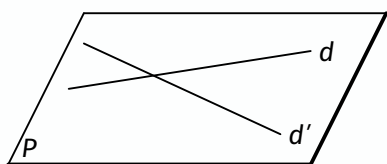
Code $\text{\LaTeX}$

```
\SchemaEspace{plan3points}
```



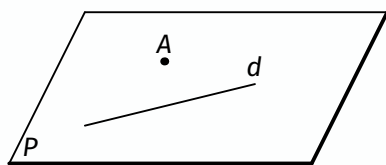
Code $\text{\LaTeX}$

```
\SchemaEspace{plandroitessecantes}
```



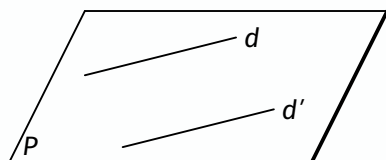
Code $\text{\LaTeX}$

`\SchemaEspace{plandroitepoint}`



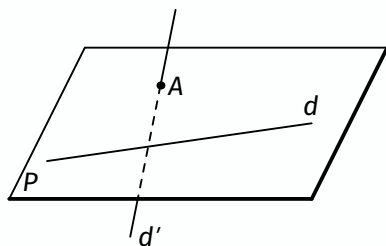
Code $\text{\LaTeX}$

`\SchemaEspace{plandroitespara}`



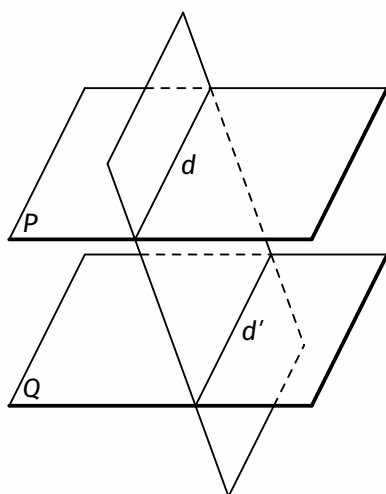
Code $\text{\LaTeX}$

`\SchemaEspace{droitesnoncopla}`



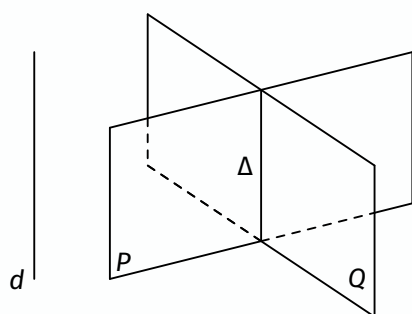
Code $\text{\LaTeX}$

`\SchemaEspace{incidence}`



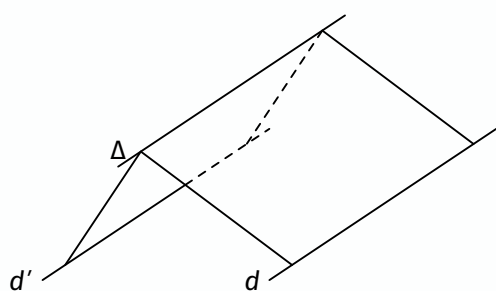
Code $\LaTeX$

```
\SchemaEspace{droiteparaplans}
```



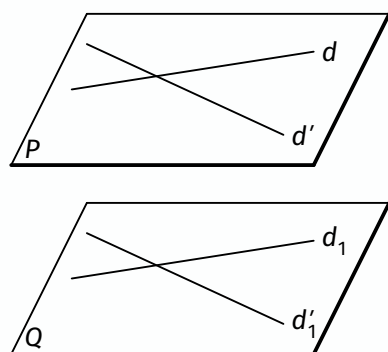
Code $\LaTeX$

```
\SchemaEspace{toit}
```



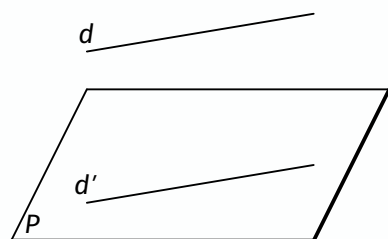
Code $\LaTeX$

```
\SchemaEspace{planspara}
```



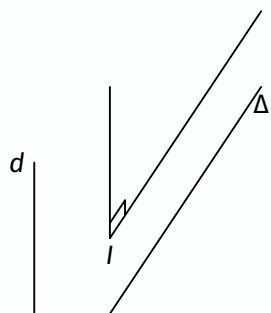
Code $\LaTeX$

```
\SchemaEspace{droiteplanpara}
```



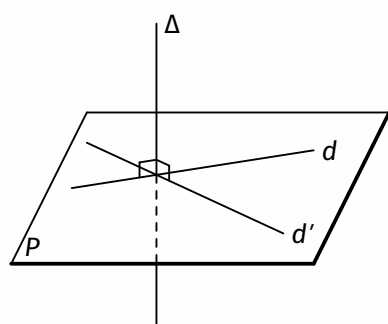
Code $\LaTeX$

```
\SchemaEspace{droitesortho}
```



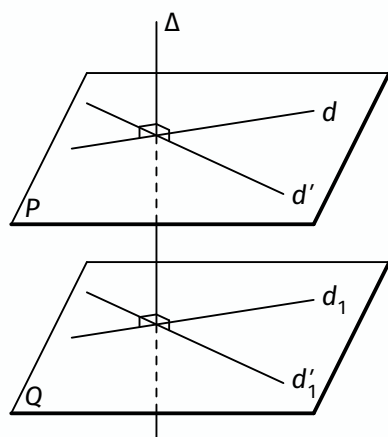
Code $\LaTeX$

```
\SchemaEspace{droiteorthoplan}
```



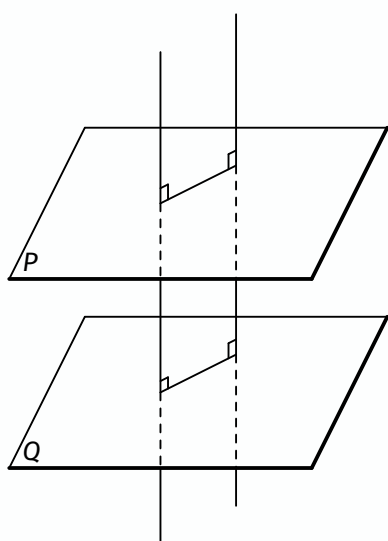
Code $\LaTeX$

```
\SchemaEspace{plansparadroiteortho}
```



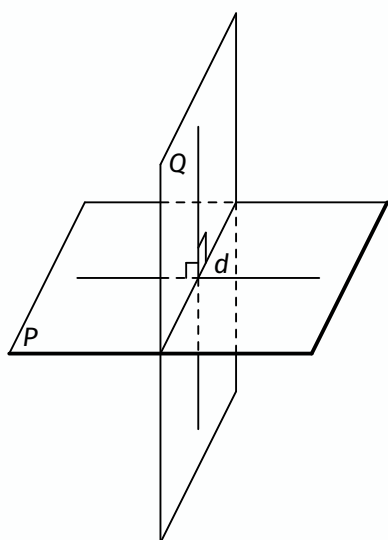
Code $\LaTeX$

```
\SchemaEspace{plansparadroitesortho}
```



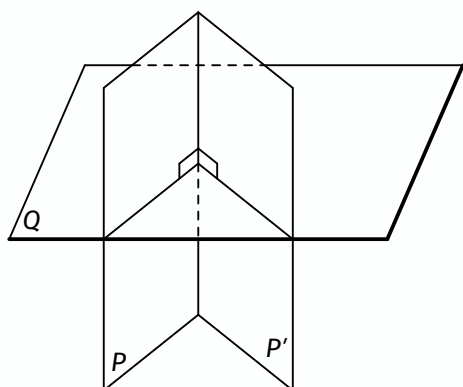
Code $\LaTeX$

```
\SchemaEspace{plansperp}
```

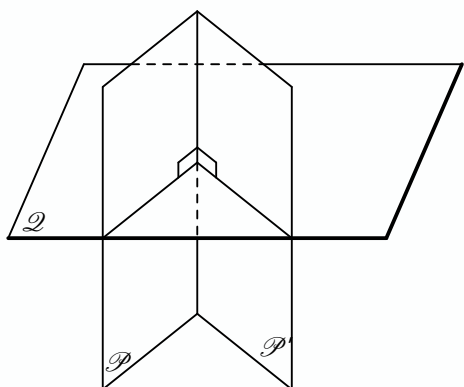


Code $\LaTeX$

```
\SchemaEspace{plansperpplan}
```



```
\SchemaEspace*{plansperpplan}
```



Thème

OUTILS POUR LA GÉOMÉTRIE ANALYTIQUE

Huitième partie

Outils pour la géométrie analytique

28 Conseils d'utilisation



2.6.5 Il est conseillé d'utiliser Lua $\text{\LaTeX}$ pour les commandes (vectorielles) de géométrie analytique, même s'il est toutefois possible d'utiliser pdf $\text{\LaTeX}$.

Il est possible que les simplifications demandées (coefficients entiers, ou premiers entre eux) ne donnent pas entière satisfaction, donc prudence sur l'utilisation de celles-ci (ce sont des tests et retours de *bugs* qui montreront les limites des commandes).

29 Affichage de coordonnées

29.1 Idée



2.6.4 L'idée est de proposer des commandes pour simplifier la saisie de coordonnées de vecteurs ou de points (plan ou espace), en saisissant les coordonnées *en ligne*.

À noter que les calculs et résultats sont traités par la commande de *conversion de fraction* de [ProfLycee](#).



Logiquement les commandes (à insérer dans un environnement mathématique) doivent donner des résultats satisfaisants pour tout ce qui est *rationnel*, mais cela ne sera pas pertinent dans le cas de coordonnées irrationnelles...

Code $\text{\LaTeX}$

```
%Affichage des coordonnées d'un point (2 ou 3 coordonnées)
\AffPoint[options de formatage](liste des coordonnées)

%Affichage des coordonnées d'un vecteur (2 ou 3 coordonnées)
\AffVecteur[options de formatage]<options nicematrix>(liste des coordonnées)
```



Dans cette partie liée à la géométrie analytique, j'ai choisi de saisir les arguments (coordonnées) via les délimiteurs $\text{\LaTeX}(\dots)$:

- avec le séparateur $\text{\LaTeX},$ pour les points;
- avec le séparateur $\text{\LaTeX};$.

De ce fait, le code *sait* s'il est face à un point ou à un vecteur, et adapte sa méthode de calcul en conséquence!

29.2 Options et arguments



Concernant les arguments des commandes :

- le premier argument, optionnel et entre `[...]` permet de spécifier la ou les caractéristiques de formatage des coordonnées, de manière globale ou individuelle, et de manière cohérente avec les options disponibles pour la commande de *conversion en fraction* de `\ProfLycee` :
 - `<d>` : pour un formatage en `\dfrac` si nécessaire;
 - `<t>` : pour un formatage en `\tfrac` si nécessaire;
 - `<n>` : pour un formatage en `\nicefrac` si nécessaire;
 - `<dec>` : pour la forme décimale (brute);
 - `<dec=k>` : pour la forme décimale à 10^{-k} .

Il est possible de spécifier des formatages différents en utilisant une *liste* sous la forme :

- `<f1,f2>` ou `<f1,f2,f3>` pour les points;
- `<f1 ;f2>` ou `<f1 ;f2 ;f3>`;
- l'argument *optionnel* et entre `<...>` (uniquement pour les vecteurs!) permet de spécifier des options de type *nicematrix*;
- l'argument obligatoire, et entre `{...}` est quant à lui la liste des coordonnées, en ligne et au format *naturel xint*.



Il est donc possible de mettre des *calculs* dans l'argument des coordonnées.

Il suffit *juste* d'utiliser une syntaxe compréhensible par les commandes du package `\xint`.

Code $\LaTeX$

```
%Point, avec affichage classique en dfrac
$\AffPoint(1,2/3)$ \\\
%Point, avec affichage en décimal + dfrac + dfrac
$\AffPoint[dec,d,d](-0.5,1,2/3)$ \\\
%Vecteurs, avec affichages classiques
$\AffVecteur(1;2)$ et $\AffVecteur(1;2;3)$ \\\
%Vecteurs, avec option nicematrix et affichage en décimal + tfrac
$\AffVecteur[dec;t]<cell-space-limits=2pt>(0.5;2/3)$ \\\
%Vecteurs, avec option nicematrix et affichage en décimal
$\AffVecteur[dec]<cell-space-limits=2pt>(0.5;0.6;0.75)$ \\\
%Vecteurs, avec cacluls et affichage classique
$\AffVecteur((2-(-3));(5-6);(1-1))$
```

```
(1; 2/3)
(-1/2; 1; 2/3)
(1) et (1)
(2) (2)
(1)
(2)
(3)
(3)
(5)
(-1)
0
```

30 Équation cartésienne d'un plan de l'espace

30.1 Idée et commande



2.6.4 L'idée est de proposer une commande pour déterminer une équation cartésienne d'un plan dans l'un des cas suivants :

- en donnant un vecteur normal et un point;
- en donnant deux vecteurs directeurs et un point;
- en donnant trois points.

Code $\text{\LaTeX}$

```
%Avec un vecteur normal et un point
\TrouveEqCartPlan[clés](vecteur normal)(point)
%Avec deux vecteurs directeurs et un point
\TrouveEqCartPlan[clés](vecteur dir1)(vecteur dir2)(point)
%Avec trois points
\TrouveEqCartPlan[clés](point1)(point2)(point3)
```

30.2 Clés et arguments



Concernant les arguments des commandes :

- le premier argument, optionnel et entre $\text{\LaTeX}[\dots]$ contient les clés :
 - **<OptionCoeffs>** pour spécifier un formatage *global* des coefficients; défaut : **<d>**
 - **<SimplifCoeffs>** pour forcer des coefficients simples (entiers et premiers entre eux); défaut : **<false>**
 - **<Facteur>** pour spécifier un facteur personnalisé aux simplifications. défaut : **<1>**
- les arguments suivants, entre $\text{\LaTeX}(\dots)$ correspondent aux données utilisées (entre 2 et 3).

À noter que les séparateurs $\text{\LaTeX},$ ou $\text{\LaTeX};$ permettent de spécifier point ou vecteur.

Code $\text{\LaTeX}$

```
Une équation cartésienne du plan  $\mathcal{P}$  de vecteur normal  $\mathbf{v}$   $\text{\LaTeX}\text{\AffVecteur}(1;2;3)$  et passant
↳ par le point A de coordonnées  $\text{\LaTeX}\text{\AffPoint}(4,5,6)$  est  $\mathcal{P}$  :
↳  $\text{\LaTeX}\text{\TrouveEqCartPlan}(1;2;3)(4,5,6)$ 
```

Une équation cartésienne du plan P de vecteur normal $\vec{n}\begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$ et passant par le point A de coordonnées $(4; 5; 6)$ est $P : x + 2y + 3z - 32 = 0$

Code $\text{\LaTeX}$

```
Une équation cartésienne du plan  $\mathcal{P}$  de vecteur normal  $\mathbf{v}$   $\text{\LaTeX}\text{\AffVecteur}[n](1/2;2/3;3/5)$  et
↳ passant par le point A de coordonnées  $\text{\LaTeX}\text{\AffPoint}(4,5,6)$  est  $\mathcal{P}$  :
↳  $\text{\LaTeX}\text{\TrouveEqCartPlan}(1/2;2/3;3/5)(4,5,6) \text{\LaTeX}\text{\Leftrightarrow}$ 
↳  $\text{\LaTeX}\text{\TrouveEqCartPlan}[\text{SimplifCoeffs}](1/2;2/3;3/5)(4,5,6)$ 
```

Une équation cartésienne du plan P de vecteur normal $\vec{n}\begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$ et passant par le point A de coordonnées $(4; 5; 6)$ est $P : \frac{1}{2}x + \frac{2}{3}y + \frac{3}{5}z - \frac{134}{15} = 0 \Leftrightarrow 15x + 20y + 18z - 268 = 0$

Code $\LaTeX$

Une équation cartésienne du plan $\mathcal{P}$ de vecteur normal $\vec{v} \begin{pmatrix} 1 \\ 2/3 \\ 0 \end{pmatrix}$ et passant par le point A de coordonnées $\begin{pmatrix} 0.75 \\ 0.56 \\ 1/3 \end{pmatrix}$ est $\mathcal{P}$:

```
\TrouveEqCartPlan(1;2/3;0)(0.75,0.56,1/3) \Leftrightarrow
```

```
\TrouveEqCartPlan[SimplifCoeffs](1;2/3;0)(0.75,0.56,1/3)
```

Une équation cartésienne du plan P de vecteur normal $\vec{n} \begin{pmatrix} 1 \\ 2 \\ 3 \\ 0 \end{pmatrix}$ et passant par le point A de coordonnées $\begin{pmatrix} 3 \\ 4 \\ 14 \\ 25 \\ 1 \\ 3 \end{pmatrix}$ est $P : x + \frac{2}{3}y - \frac{337}{300} = 0 \Leftrightarrow 300x + 200y - 337 = 0$

Code $\LaTeX$

Une équation cartésienne du plan $\mathcal{P}_3$ passant par les points $A \begin{pmatrix} 2 \\ 0 \\ 1 \end{pmatrix}$, $B \begin{pmatrix} 3 \\ 1 \\ 1 \end{pmatrix}$ et $C \begin{pmatrix} 1 \\ -2 \\ 0 \end{pmatrix}$ est

```
\[ \mathcal{P}_3 \text{ : } \] \TrouveEqCartPlan(2,0,1)(3,1,1)(1,-2,0)\]
```

Une équation cartésienne du plan P_3 passant par les points A(2 ; 0 ; 1), B(3 ; 1 ; 1) et C(1 ; -2 ; 0) est

$$P_3 : -x + y - z + 3 = 0$$

Code $\LaTeX$

Une équation cartésienne du plan $\mathcal{R}$ passant par les points $A \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$, $B \begin{pmatrix} 4 \\ 2 \\ 3 \end{pmatrix}$ et $C \begin{pmatrix} -3 \\ 1 \\ 1 \end{pmatrix}$ est

```
\[ \mathcal{R} \text{ : } \] \TrouveEqCartPlan[SimplifCoeffs](0,0,1)(4,2,3)(-3,1,1)\]
```

```
\[ \mathcal{R} \text{ : } \] \TrouveEqCartPlan[SimplifCoeffs,Facteur=-1](0,0,1)(4,2,3)(-3,1,1)\]
```

Une équation cartésienne du plan R passant par les points A(0 ; 0 ; 1), B(4 ; 2 ; 3) et C(-3 ; 1 ; 1) est

$$R : -x - 3y + 5z - 5 = 0$$

$$R : x + 3y - 5z + 5 = 0$$

Code $\LaTeX$

Une équation cartésienne du plan $\mathcal{P}_0$ dirigé par les vecteurs $\vec{v} \begin{pmatrix} 9 \\ 7 \\ -8 \end{pmatrix}$ et $\vec{v} \begin{pmatrix} -2 \\ 2 \\ -1 \end{pmatrix}$ et passant par le point $A \begin{pmatrix} 5 \\ 1 \\ -1 \end{pmatrix}$ est :

```
\[ \mathcal{P}_0 \text{ : } \] \TrouveEqCartPlan[SimplifCoeffs](9;7;-8)(-2;2;-1)(5,1,-1)\]
```

Une équation cartésienne du plan P_0 dirigé par les vecteurs $\begin{pmatrix} 9 \\ 7 \\ -8 \end{pmatrix}$ et $\begin{pmatrix} -2 \\ 2 \\ -1 \end{pmatrix}$ et passant par le point A(5 ; 1 ; -1) est :

$$P_0 : 9x + 25y + 32z - 38 = 0$$

31 Équation paramétrique d'une droite de l'espace

31.1 Idée et commande



2.6.4 L'idée est de proposer une commande pour déterminer un système d'équations paramétriques d'une droite de l'espace dans l'un des cas suivants :

- en donnant un vecteur directeur et un point;
- en donnant deux points.

Code $\LaTeX$

```
%Avec un vecteur directeur et un point
\TrouveEqParamDroite[clés](vecteur directeur)(point)
%Avec deux points
\TrouveEqParamDroite[clés](point1)(point2)
```

31.2 Clés et arguments



Concernant les arguments des commandes :

- le premier argument, optionnel et entre $\boxed{\text{...}}$ contient les clés :
 - **<OptionCoeffs>** pour spécifier un formatage *global* des coefficients; défaut : **<d>**
 - **<Reel>** pour coder le paramètre réel; défaut : **<k>**
 - le booléen **<Oppose>** pour utiliser plutôt l'opposé du vecteur directeur; défaut : **<false>**
 - le booléen **<Rgras>** pour utiliser le symbole **R** ou lieu de $\mathbb{R}$ (si $\boxed{\text{amsfonts}}$ est chargé!). défaut : **<false>**
- les arguments suivants, entre $\boxed{\text{...}}$ correspondent aux données utilisées.

À noter que les séparateurs $\boxed{\text{,}}$ ou $\boxed{\text{;}}$ permettent de spécifier point ou vecteur.

Code $\LaTeX$

```
Une équation paramétrique de la droite  $\$(d)\$$  dirigée par le vecteur  $\$\vv{u}\$ \backslash\text{AffVecteur}(2;5;-4)\$$  et
passant par  $\$A\backslash\text{AffPoint}(-1,-1,-1)\$$  est
 $\boxed{\backslash\text{TrouveEqParamDroite}(2;5;-4)(-1,-1,-1) \backslash}$ 
```

Une équation paramétrique de la droite (d) dirigée par le vecteur $\vec{u} \begin{pmatrix} 2 \\ 5 \\ -4 \end{pmatrix}$ et passant par $A(-1; -1; -1)$ est

$$\begin{cases} x = -1 + 2k \\ y = -1 + 5k, k \in \mathbb{R} \\ z = -1 - 4k \end{cases}$$

Code $\LaTeX$

```
Une équation paramétrique de la droite  $\$(d)\$$  passant par  $\$A\backslash\text{AffPoint}(2,5,-4)\$$  et  $\$A\backslash\text{AffPoint}(-1,-1,-1)\$$  est
 $\boxed{\backslash\text{TrouveEqParamDroite}[\text{Oppose}](2,5,-4)(-1,-1,-1) \backslash\text{ou} \backslash\text{TrouveEqParamDroite}(2,5,-4)(-1,-1,-1) \backslash}$ 
```

Une équation paramétrique de la droite (d) passant par $(2; 5; -4)$ et $(-1; -1; -1)$ est

$$\begin{cases} x = 2 + 3k \\ y = 5 + 6k, k \in \mathbb{R} \\ z = 5 + 6k \end{cases} \text{ ou } \begin{cases} x = 2 - 3k \\ y = 5 - 6k, k \in \mathbb{R} \\ z = 5 - 6k \end{cases}$$

Code $\LaTeX$

Une équation paramétrique de la droite (d) dirigée par le vecteur $\vec{u} \begin{pmatrix} 0 \\ -1 \\ 3 \end{pmatrix}$ et passant par $O(0;0;0)$ est

```
\[ \TrouveEqParamDroite(0;-1;3)(0,0,0) \]
```

Une équation paramétrique de la droite (d) dirigée par le vecteur $\vec{u} \begin{pmatrix} 0 \\ -1 \\ 3 \end{pmatrix}$ et passant par $O(0;0;0)$ est

$$\begin{cases} x = 0 \\ y = -k, k \in \mathbb{R} \\ z = 3k \end{cases}$$

Code $\LaTeX$

Une équation paramétrique de la droite (d) dirigée par le vecteur $\vec{u} \begin{pmatrix} -1 \\ 2 \\ 3 \end{pmatrix}$ et passant par $A(2;0;-3)$ est

```
\[ \TrouveEqParamDroite[Reel=\ell,Rgras](-1;2;3)(2,0,-3) \]
```

Une équation paramétrique de la droite (d) dirigée par le vecteur $\vec{u} \begin{pmatrix} -1 \\ 2 \\ 3 \end{pmatrix}$ et passant par $A(2;0;-3)$ est

$$\begin{cases} x = 2 - \ell \\ y = 2\ell \\ z = -3 + 3\ell \end{cases}, \ell \in \mathbb{R}$$

32 Équation cartésienne d'une droite du plan

32.1 Idée et commande



2.6.4 L'idée est de proposer une commande pour déterminer une équation cartésienne d'une droite du plan dans l'un des cas suivants :

- en donnant un vecteur directeur et un point;
- en donnant un vecteur normal et un point;
- en donnant deux points.

Code $\LaTeX$

```
%Avec un vecteur normal (choix par défaut) et un point
\TrouveEqCartDroite[clés](vecteur normal)(point)
%Avec un vecteur directeur et un point
\TrouveEqCartDroite[clés,VectDirecteur](vecteur directeur)(point1)
%Avec deux points
\TrouveEqCartDroite[clés](point1)(point2)
```

32.2 Clés et arguments



Concernant les arguments des commandes :

- le premier argument, optionnel et entre `[...]` contient les clés :
 - `<OptionCoeffs>` pour spécifier un formatage *global* des coefficients; défaut : `<d>`
 - le booléen `<SimplifCoeffs>` pour forcer des coeffs simples (entiers et premiers entre eux); défaut : `<false>`
 - `<Facteur>` pour spécifier un facteur personnalisé aux simplifications; défaut : `<1>`
 - le booléen `<VectDirecteur>` pour préciser que le vecteur utilisé est directeur. défaut : `<false>`
- les arguments suivants, entre `(...)` correspondent aux données utilisées.

À noter que les séparateurs `[ , ]` ou `[ Vect ]` permettent de spécifier point ou vecteur.

Code $\LaTeX$

```
Une équation cartésienne de la droite  $\mathcal{D}$  de vecteur normal  $\vec{v}_n$  \AffVecteur(1;2) et
passant par le point A de coordonnées \AffPoint(4,5) est  $\mathcal{D}$  :
\TrouveEqCartDroite[VectNormal](1;2)(4,5)
```

Une équation cartésienne de la droite D de vecteur normal $\vec{n}\left(\begin{smallmatrix} 1 \\ 2 \end{smallmatrix}\right)$ et passant par le point A de coordonnées $(4; 5)$ est $D : x + 2y - 14 = 0$

Code $\LaTeX$

```
Une équation cartésienne de la droite  $\mathcal{D}$  de vecteur directeur  $\vec{v}_u$  \AffVecteur[n](1/2;2/3)
et passant par le point A de coordonnées \AffPoint(5,6) est  $\mathcal{D}$  :
\TrouveEqCartDroite[VectDirecteur](1/2;2/3)(5,6) \Leftrightarrow
\TrouveEqCartDroite[SimplifCoeffs,VectDirecteur](1/2;2/3)(5,6) \Leftrightarrow
\TrouveEqCartDroite[SimplifCoeffs,VectDirecteur,Facteur=-1](1/2;2/3)(5,6)
```

Une équation cartésienne de la droite D de vecteur directeur $\vec{u}\left(\begin{smallmatrix} 1 \\ 2 \\ 3 \end{smallmatrix}\right)$ et passant par le point A de coordonnées $(5; 6)$ est $D : -\frac{2}{3}x + \frac{1}{2}y + \frac{1}{3} = 0 \Leftrightarrow -4x + 3y + 2 = 0 \Leftrightarrow 4x - 3y - 2 = 0$

Code $\LaTeX$

Une équation cartésienne de la droite $\mathcal{D}$ passant par les points $A(2,4)$ et $B(-4,2)$ est $\left[\mathcal{D} \quad \text{ : } \right] \rightarrow \text{TrouveEqCartDroite}(2,4)(-4,2) \rightarrow \text{TrouveEqCartDroite}[\text{SimplifCoeffs}](2,4)(-4,2)$

Une équation cartésienne de la droite D passant par les points $(2;4)$ et $(-4;2)$ est

$$D : 2x - 6y + 20 = 0 \Leftrightarrow x - 3y + 10 = 0$$

33 Norme d'un vecteur, distance entre deux points

33.1 Idée et commande



2.6.5 L'idée est de proposer une commande pour déterminer la distance entre deux points, ou la norme d'un vecteur :

- en donnant le vecteur;
- en donnant deux points.

Code $\LaTeX$

```
%Avec le vecteur
\TrouveNorme(vecteur)
%Avec deux points
\TrouveNorme(point 1)(point 2)
```



Le résultat étant souvent écrit à l'aide d'une racine carrée, le code se charge de simplifier le résultat sous la forme $\frac{a\sqrt{n}}{b}$.

Dans le cas où les coordonnées ne seraient pas rationnelles, le résultat risque de ne pas être conforme à celui attendu.

33.2 Clés et arguments



Concernant les arguments de cette commande :

- les séparateurs $\LaTeX$, ou $\LaTeX$; permettent de spécifier point ou vecteur pour les arguments 1 et 2.

Code $\LaTeX$

La distance AB avec $A(-5,2)$ et $B(4,-3)$ vaut
 $\$d=\displaystyle\TrouveNorme(-5,2)(4,-3)\$$

La distance AB avec $A(-5;2)$ et $B(4;-3)$ vaut $d = \sqrt{106}$

Code $\LaTeX$

La distance AB avec $A(2,1,2)$ et $B(-4,1,1)$ vaut
 $\$d=\displaystyle\TrouveNorme(2,1,2)(-4,1,1)\$$

La distance AB avec $A(2;1;2)$ et $B(-4;1;1)$ vaut $d = \sqrt{37}$

Code $\LaTeX$

La norme de $\text{\texttt{\textit{AffVecteur}}}(2;4)$ vaut
 $\$d=\displaystyle\TrouveNorme(2;4)\$$

La norme de $\begin{pmatrix} 2 \\ 4 \end{pmatrix}$ vaut $d = 2\sqrt{5}$

Code $\LaTeX$

La norme de $\text{\texttt{\textit{AffVecteur}}}[d;d;n](2;4;0.5)$ vaut
 $\$d=\displaystyle\TrouveNorme(2;4;0.5)\$$

La norme de $\begin{pmatrix} 2 \\ 4 \\ 1 \\ 2 \end{pmatrix}$ vaut $d = \frac{9}{2}$

34 Distance d'un point à un plan

34.1 Idée et commande



2.6.4 L'idée est de proposer une commande pour déterminer la distance d'un point à un plan :

- en donnant le point puis le plan défini par vecteur normal & point;
- en donnant le point puis le plan défini par une équation cartésienne.

Code $\LaTeX$

```
%Avec le point et le plan via vect normal + point
\TrouveDistancePtPlan(point)(vec normal du plan)(point du plan)
%Avec le point et le plan via vect normal + point
\TrouveDistancePtPlan(point)(équation cartésienne)
```



Le résultat étant souvent écrit à l'aide d'une racine carrée, le code se charge de simplifier le résultat sous la forme $\frac{a\sqrt{n}}{b}$.

Dans le cas où les coordonnées ne seraient pas rationnelles, le résultat risque de ne pas être conforme à celui attendu.

34.2 Clés et arguments



Concernant les arguments de cette commande :

- si on travaille avec une équation cartésienne, elle est à donner sous la forme $ax+by+cz=0$ ou $ax+by+cz$
- les séparateurs pt , ou vec permettent de spécifier point ou vecteur pour les arguments 1 et 3.

Code $\LaTeX$

La distance entre le point $\text{\texttt{\$}\AffPoint(1,2,3)\$}$ et le plan de vecteur normal $\text{\texttt{\$}\AffVecteur(-1;-2;3)\$}$ et passant par $\text{\texttt{\$}\AffPoint(5,0,2)\$}$ vaut
 $\text{\texttt{\[d = \displaystyle\TrouveDistancePtPlan(1,2,3)(-1;-2;3)(5,0,2) \]}}$

La distance entre le point $(1 ; 2 ; 3)$ et le plan de vecteur normal $\begin{pmatrix} -1 \\ -2 \\ 3 \end{pmatrix}$ et passant par $(5 ; 0 ; 2)$ vaut

$$d = \frac{3\sqrt{14}}{14}$$

Code $\LaTeX$

La distance entre le point $\text{\texttt{\$}\AffPoint(1,2,3)\$}$ et le plan d'équation $\text{\texttt{\$}x+2y+2z-7=0\$}$ vaut
 $\text{\texttt{\[d = \displaystyle\TrouveDistancePtPlan(1,2,3)(x+2y-2z+7) \]}}$

La distance entre le point $(1 ; 2 ; 3)$ et le plan d'équation $x + 2y + 2z - 7 = 0$ vaut

$$d = 2$$

Code $\LaTeX$

La distance entre le point $\text{\texttt{\$}\AffPoint(-7,0,4)\$}$ et le plan d'équation $\text{\texttt{\$}0,5x+2y-z-1=0\$}$ vaut
 $\text{\texttt{\[d = \displaystyle\TrouveDistancePtPlan(-7,0,4)(0.5x+2y-z-1=0) \]}}$

La distance entre le point $(-7 ; 0 ; 4)$ et le plan d'équation $0,5x + 2y - z - 1 = 0$ vaut

$$d = \frac{17\sqrt{21}}{21}$$

Code $\LaTeX$

La distance entre le point $H(0,4,8)$ et le plan d'équation $-x+y+z-4=0$ vaut
`\[ d = \displaystyle\TrouveDistancePtPlan(0,4,8)(-x+y+z-4=0) \]`

La distance entre le point $H(0 ; 4 ; 8)$ et le plan d'équation $-x + y + z - 4 = 0$ vaut

$$d = \frac{8\sqrt{3}}{3}$$

Code $\LaTeX$

La distance entre le point $H(0,0,5)$ et le plan d'équation $z-1=0$ vaut
`\[ d = \displaystyle\TrouveDistancePtPlan(0,0,5)(z-1=0) \]`

La distance entre le point $H(0 ; 0 ; 5)$ et le plan d'équation $z - 1 = 0$ vaut

$$d = 4$$

35 Équation d'une sphère

35.1 Idée



4.00b L'idée est de proposer une commande pour travailler sur les équations de sphères :

- en spécifiant centre et rayon;
- en spécifiant centre et point;
- en spécifiant un diamètre;
- en spécifiant centre et plan tangent.

À noter que les calculs et résultats sont traités par la commande de *conversion de fraction* de [ProfLycee](#).

Code $\LaTeX$

```
\TrouveEqSphere[clés](centre ou point n°1)(rayon ou point ou éq plan)
```

35.2 Clés et arguments



Concernant les clés de cette commande :

- | | |
|---|-------------------------------|
| — <OptionCoeffs> pour spécifier un formatage <i>global</i> des coefficients; | défaut : <d> |
| — <Diam> (booléen) pour préciser que les 2 points forment un diamètre; | défaut : <false> |
| — <Develop> (booléen) pour développer l'équation; | défaut : <true> |
| — <Tri> (booléen) pour trier les termes. | défaut : <true> |

35.3 Exemples

Code $\LaTeX$

```
%centre + rayon
$\TrouveEqSphere(1,1,1)(2)$\\
$\TrouveEqSphere[Develop=false](1,1,1)(2)$\\
$\TrouveEqSphere[Tri=false](1,1,1)(2)$
```

$$\begin{aligned}x^2 - 2x + y^2 - 2y + z^2 - 2z - 1 &= 0 \\(x - 1)^2 + (y - 1)^2 + (z - 1)^2 &= 4 \\x^2 + y^2 + z^2 - 2x - 2y - 2z - 1 &= 0\end{aligned}$$

Code $\LaTeX$

```
%centre + point
$\TrouveEqSphere(3,1,4)(0,0,0)$\\
$\TrouveEqSphere[Develop=false](3,1,4)(0,0,0)$\\
$\TrouveEqSphere[Tri=false](3,1,4)(0,0,0)$
```

$$\begin{aligned}x^2 - 6x + y^2 - 2y + z^2 - 8z &= 0 \\(x - 3)^2 + (y - 1)^2 + (z - 4)^2 &= 26 \\x^2 + y^2 + z^2 - 6x - 2y - 8z &= 0\end{aligned}$$

Code $\LaTeX$

```
%diamètre
$\TrouveEqSphere[Diam](3,1,4)(0,0,0)$\\
$\TrouveEqSphere[Diam,Develop=false](3,1,4)(0,0,0)$\\
$\TrouveEqSphere[Diam,Tri=false](3,1,4)(0,0,0)$
```

$$\begin{aligned}x^2 - 3x + y^2 - y + z^2 - 4z &= 0 \\ \left(x - \frac{3}{2}\right)^2 + \left(y - \frac{1}{2}\right)^2 + (z - 2)^2 &= \frac{13}{2} \\ x^2 + y^2 + z^2 - 3x - y - 4z &= 0\end{aligned}$$

```
%centre + plan tangent
 $\backslash$ TrouveEqSphere[](1,2,4)(2x+3y-3z+1=0) $\$\\$ 
 $\backslash$ TrouveEqSphere[Develop=false](1,2,4)(2x+3y-3z+1=0) $\$\\$ 
 $\backslash$ TrouveEqSphere[Tri=false](1,2,4)(2x+3y-3z+1=0) $\$$ 
```

$$x^2 - 2x + y^2 - 4y + z^2 - 8z + \frac{453}{22} = 0$$

$$(x - 1)^2 + (y - 2)^2 + (z - 4)^2 = \frac{9}{22}$$

$$x^2 + y^2 + z^2 - 2x - 4y - 8z + \frac{453}{22} = 0$$

36 Équation réduite d'une droite du plan

36.1 Idée



2.6.3 L'idée est de proposer une commande pour déterminer l'équation réduite d'une droite passant par deux points :

- en traitant les cas particuliers *horizontale*, *verticale*;
- en affichant une méthode de résolution;
- en travaillant sous forme exacte fractionnaire (les racines carrées ou autres ne seront pas gérés).

À noter que les calculs et résultats sont traités par la commande de *conversion de fraction* de [ProfLycee](#).



La commande se charge de formater (normalement!) correctement les différentes étapes de calculs (il se peut quand même que cela puisse ne pas donner le résultat réellement escompté...) :

- en travaillant en fraction;
- en mettant les parenthèses nécessaires devant les éventuels nombres négatifs;
- en traitant les cas particuliers $m = \pm 1$ et $b = 0$.

Code $\text{\LaTeX}$

```
\EquationReduite[option]{A/xa/ya,B/xb/yb}
```

36.2 Clés et arguments



Concernant le fonctionnement de la commande :

- le premier argument, optionnel et entre $\text{\LaTeX}$ [...] et valant $\langle [d] \rangle$ par défaut, permet de formater les fractions éventuelles en mode $\text{\LaTeX}$ \displaystyle;
- le second argument, obligatoire et entre $\text{\LaTeX}$ {...}, permet de donner les coordonnées des points concernés.

Code $\text{\LaTeX}$

```
\EquationReduite{C/2/0,D/-2/-8}
```

Afin de déterminer l'équation réduite d'une droite passant par les points C et D , on doit d'abord déterminer le coefficient directeur m :

$$m = \frac{y_D - y_C}{x_D - x_C} = \frac{-8 - 0}{-2 - 2} = \frac{-8}{-4} = 2$$

L'équation réduite de la droite est donc de la forme $(CD) : y = 2x + p$.

Il faut enfin déterminer l'ordonnée à l'origine p .

On sait que la droite passe par le point C , donc les coordonnées $C(2; 0)$ vérifient l'équation. On a alors :

$$y_C = 2 \times x_C + p \implies 0 = 2 \times 2 + p \implies p = 0 - (2 \times 2) \implies p = -4$$

Donc l'équation réduite de (CD) est $y = 2x - 4$.

36.3 Exemples

Code $\text{\LaTeX}$

```
\EquationReduite{I/-4/5,J/-4/12}
```

Étant donné que $x_I = x_J$, la droite (IJ) est verticale, dont une équation est $x = -4$.

Code $\LaTeX$

$\backslash\text{EquationReduite}\{U/-4/5,V/-4/5\}$

Les deux points donnés sont identiques, donc pas de droite...

Code $\LaTeX$

$\backslash\text{EquationReduite}\{L/10/7,M/-2/7\}$

Étant donné que $y_L = y_M$, la droite (LM) est horizontale, dont une équation est $y = 7$.

Code $\LaTeX$

$\backslash\text{EquationReduite}\{L/\{1/3\}/2.5,M/\{-5/7\}/\{3/5\}\}$

Afin de déterminer l'équation réduite d'une droite passant par les points L et M , on doit d'abord déterminer le coefficient directeur m :

$$m = \frac{y_M - y_L}{x_M - x_L} = \frac{\frac{3}{5} - 2,5}{-\frac{5}{7} - \frac{1}{3}} = \frac{-\frac{19}{10}}{-\frac{22}{21}} = \frac{399}{220}$$

L'équation réduite de la droite est donc de la forme (LM) : $y = \frac{399}{220}x + p$.

Il faut enfin déterminer l'ordonnée à l'origine p .

On sait que la droite passe par le point L , donc les coordonnées $L\left(\frac{1}{3}; 2,5\right)$ vérifient l'équation. On a alors :

$$y_L = \frac{399}{220} \times x_L + p \Rightarrow 2,5 = \frac{399}{220} \times \frac{1}{3} + p \Rightarrow p = 2,5 - \left(\frac{399}{220} \times \frac{1}{3}\right) \Rightarrow p = \frac{417}{220}$$

Donc l'équation réduite de (LM) est $y = \frac{399}{220}x + \frac{417}{220}$.

Code $\LaTeX$

$\backslash\text{EquationReduite}\{P/4/-4,Q/-2/2\}$

Afin de déterminer l'équation réduite d'une droite passant par les points P et Q , on doit d'abord déterminer le coefficient directeur m :

$$m = \frac{y_Q - y_P}{x_Q - x_P} = \frac{2 - (-4)}{-2 - 4} = \frac{6}{-6} = -1$$

L'équation réduite de la droite est donc de la forme (PQ) : $y = -x + p$.

Il faut enfin déterminer l'ordonnée à l'origine p .

On sait que la droite passe par le point P , donc les coordonnées $P(4; -4)$ vérifient l'équation. On a alors :

$$y_P = -1 \times x_P + p \Rightarrow -4 = -1 \times 4 + p \Rightarrow p = -4 - (-1 \times 4) \Rightarrow p = 0$$

Donc l'équation réduite de (PQ) est $y = -x$.

$\text{\EquationReduite}\{G/-4/5,H/10/4\}$

Afin de déterminer l'équation réduite d'une droite passant par les points G et H , on doit d'abord déterminer le coefficient directeur m :

$$m = \frac{y_H - y_G}{x_H - x_G} = \frac{4 - 5}{10 - (-4)} = \frac{-1}{14} = -\frac{1}{14}$$

L'équation réduite de la droite est donc de la forme $(GH) : y = -\frac{1}{14}x + p$.

Il faut enfin déterminer l'ordonnée à l'origine p .

On sait que la droite passe par le point G , donc les coordonnées $G(-4 ; 5)$ vérifient l'équation. On a alors :

$$y_G = -\frac{1}{14} \times x_G + p \implies 5 = -\frac{1}{14} \times (-4) + p \implies p = 5 - \left(-\frac{1}{14} \times (-4)\right) \implies p = \frac{33}{7}$$

Donc l'équation réduite de (GH) est $y = -\frac{1}{14}x + \frac{33}{7}$.

Thème

OUTILS POUR LES STATISTIQUES

Neuvième partie

Outils pour les statistiques

37 Paramètres d'une série discrète

37.1 Idée



 3.10c L'idée est de proposer une commande pour déterminer les paramètres d'une série statistique discrète.

Le code se charge de calculer les paramètres (min, q1, méd, q3, max) et de les stocker dans des macros réutilisables.

Code $\text{\LaTeX}$

```
\DeterminerParamStats(*){liste}[\macromin]...[\macromax]
```

37.2 Arguments, exemples



La version étoilée permet de spécifier que la liste est donnée sous la forme $\text{\LaTeX}$ val/eff1,val2/eff2,..., sinon la liste est à donner sous forme brute $\text{\LaTeX}$ val1,val2,....

Les (5) macros par défaut sont

- la macro $\langle \backslash\text{monmin} \rangle$ pour stocker min;
- la macro $\langle \backslash\text{monquartileun} \rangle$ pour stocker q1;
- la macro $\langle \backslash\text{mamediane} \rangle$ pour stocker méd;
- la macro $\langle \backslash\text{monquartiletrois} \rangle$ pour stocker q3;
- la macro $\langle \backslash\text{monmax} \rangle$ pour stocker max.

Code $\text{\LaTeX}$

```
%avec des données brutes
\def\listedonnees{0.7426,0.7429,0.7435,0.7448,0.7456,0.7478,0.7478,0.748,0.7487,0.7494,0.7494,0.7496}

\DeterminerParamStats{\listedonnees}
\monmin/\monquartileun/\mamediane/\monquartiletrois/\monmax
```

0.7426/0.7435/0.7478/0.7487/0.7496

Code $\text{\LaTeX}$

```
%avec des données/effectifs
\def\listedonneesregroup{2/25,3/18,4/17,5/10,6/5,7/20,8/2}

\DeterminerParamStats*{\listedonneesregroup}
La valeur minimale est \monmin.\\
La médiane est \mamediane.\\
Le premier quartile est \monquartileun.\\
Le troisième quartile est \monquartiletrois.\\
La valeur maximale est \monmax.
```

La valeur minimale est 2.
La médiane est 4.
Le premier quartile est 2.
Le troisième quartile est 6.
La valeur maximale est 8.

38 Paramètres d'une régression linéaire par la méthode des moindres carrés

38.1 Idée



L'idée est d'utiliser une commande qui va permettre de calculer les paramètres principaux d'une régression linéaire par la méthode des moindres carrés.

Le package `pgfpots` permet de le faire nativement, mais le moteur de calculs de `pgf` peut poser souci avec de grandes valeurs, donc ici cela passe par `xfp` qui permet de *gagner* en précision!

L'idée est que cette macro calcule et stocke les paramètres dans des variables (le nom peut être personnalisé!) pour exploitation ultérieure :

- en calculs *purs*;
- dans un environnement *TikZ* via `pgfplots` ou bien en *natif*;
- dans un environnement `PSTricks`;
- dans un environnement `METAPOST` (à vérifier quand même);
- ...

Code $\text{\LaTeX}$

```
...  
\CalculsRegLin[clés]{listeX}{listeY} %listes avec éléments séparés par des ,  
...
```



La commande `\CalculsRegLin` va définir également des macros pour chaque coefficient, qui de ce fait seront réutilisables après!

38.2 Commandes



Quelques **(Clés)** sont disponibles pour cette commande, essentiellement pour *renommer* les paramètres :

- | | |
|--|------------------------------|
| — la clé <code>\NomCoeffa</code> qui permet de définir la variable qui contiendra a ; | défaut <code>\COEFFa</code> |
| — la clé <code>\NomCoeffb</code> qui permet de définir la variable qui contiendra b ; | défaut <code>\COEFFb</code> |
| — la clé <code>\NomCoeffr</code> qui permet de définir la variable qui contiendra r ; | défaut <code>\COEFFr</code> |
| — la clé <code>\NomCoeffrd</code> qui permet de définir la variable qui contiendra r^2 ; | défaut <code>\COEFFrd</code> |
| — la clé <code>\NomXmin</code> qui permet de définir la variable qui contiendra $x_{\min}$; | défaut <code>\LXmin</code> |
| — la clé <code>\NomXmax</code> qui permet de définir la variable qui contiendra $x_{\max}$. | défaut <code>\LXmax</code> |

Code $\text{\LaTeX}$

```
%les espaces verticaux n'ont pas été écrits ici  
\def\LLX{1994,1995,1996,1997,1998,1999,2000,2001,2002,2004,2005,2006,2007,2008, 2009,2010}  
\def\LLY{1718,1710,1708,1700,1698,1697,1691,1688,1683,1679,1671,1670,1663,1661, 1656,1649}  
\CalculsRegLin{\LLX}{\LLY}
```

Code $\text{\LaTeX}$

```
%vérif des calculs (noms non modifiables...)  
Liste des X := \showitems\LX.  
Liste des Y := \showitems\LY.  
Somme des X := \LXSomme{} et somme des Y := \LYSomme.  
Moyenne des X := \LXmoy{} et moyenne des Y := \LYmoy.  
Variance des X := \LXvar{} et variance des Y := \LYvar{}  
Covariance des X/Y := \LXYvar.  
%les coefficients, avec des noms modifiables !  
Min des X := \LXmin{} et Max des X := \LXmax.  
Coefficient $a$=\COEFFa$.  
Coefficient $b$=\COEFFb$.  
Coefficient $r$=\COEFFr$.  
Coefficient $r^2$=\COEFFrd$.
```

Code $\LaTeX$

Liste des X := 1994 1995 1996 1997 1998 1999 2000 2001 2002 2004 2005 2006 2007 2008 2009 2010 .

Liste des Y := 1718 1710 1708 1700 1698 1697 1691 1688 1683 1679 1671 1670 1663 1661 1656 1649 .

Somme des X := 32031 et somme des Y := 26942.

Moyenne des X := 2001.9375 et moyenne des Y := 1683.875.

Variance des X := 25.43359375 et variance des Y := 403.984375

Covariance des X/Y := -100.9453125.

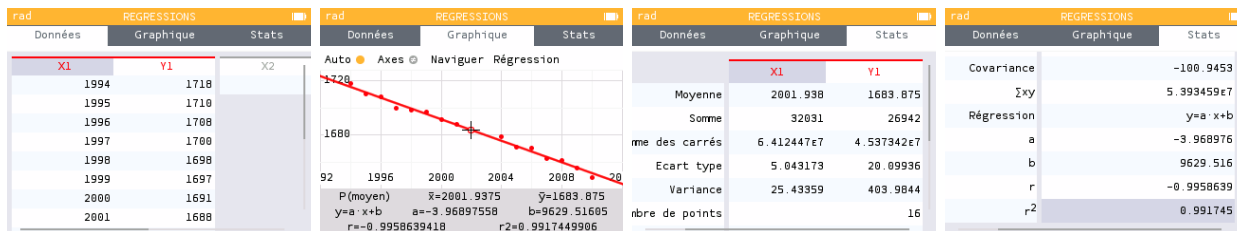
Min des X := 1994 et Max des X := 2010.

Coefficient $a = -3.968975579788051$.

Coefficient $b = 9629.516049761941$.

Coefficient $r = -0.9958639418357528$.

Coefficient $r^2 = 0.9917449906486436$.



Les macros qui contiennent les paramètres de la régression sont donc réutilisables, en tant que nombres réels, donc exploitables par `\siunitx` et `\xfp` pour affichage *fin* ! Ci-dessous un exemple permettant de visualiser tout cela.

Code $\LaTeX$

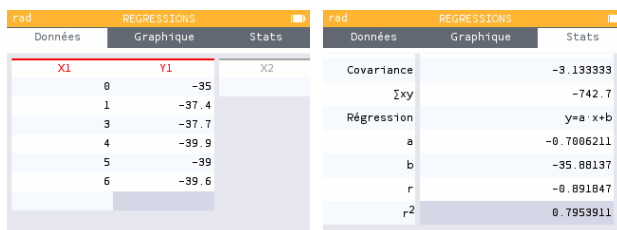
```
%les espaces verticaux n'ont pas été écrits ici
\def\LstX{0,1,3,4,5,6}
\def\LstY{-35,-37.4,-37.7,-39.9,-39,-39.6}
%on lance les calculs et on change le nom des "macros-résultats"
\CalculsRegLin[NomCoeffa=TESTa,NomCoeffb=TESTb,NomCoeffr=TESTr,NomCoeffrd=TESTrd,%
  NomXmin=TESTmin,NomXmax=TESTmax]{\LstX}{\LstY}
%commandes complémentaires
\DeclareDocumentCommand\arrond{ s O{3} m }{% * signe / précision / nb
  \IfBooleanTF{#1}{\num[print-implicit-plus]{\fpeval{round(#3,#2)}}}{\num{\fpeval{round(#3,#2)}}}
}
%paramètres
Les valeurs extr. de X sont \TESTmin{} et \TESTmax. Une équ. est $y=\arrond[3]{\TESTa}x
↪ \arrond*[3]{\TESTb}$.
Le coeff. de corrélation est $r=\arrond[4]{\TESTr}$, et son carré est $r^2=\arrond[4]{\TESTrd}$.
```

Code $\LaTeX$

Les valeurs extrêmes de x sont 0 et 6. Une équation de la droite de régression de y en x est

$$y = -0,701x - 35,881.$$

Le coefficient de corrélation linéaire est $r = -0,8918$, et son carré est $r^2 = 0,7954$.



38.3 Intégration dans un environnement TikZ



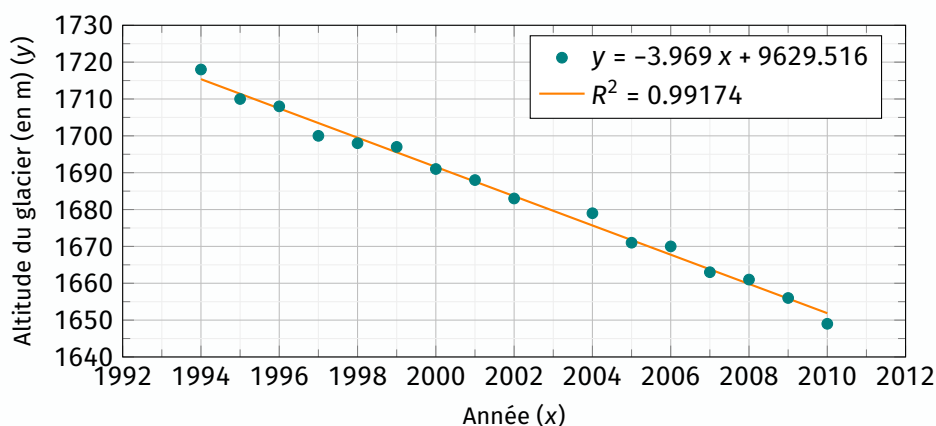
La commande étant « autonome », elle va pouvoir être intégrée dans des environnements graphiques pour permettre un tracé *facile* de la droite de régression.

Code $\LaTeX$

```

\begin{tikzpicture}
\begin{axis}[options des axes, non présentées ici...]
\addplot[teal, only marks] table{
  X Y
  1994 1718 1995 1710 1996 1708 1997 1700 1998 1698 1999 1697 2000 1691 2001 1688
  2002 1683 2004 1679 2005 1671 2006 1670 2007 1663 2008 1661 2009 1656 2010 1649
};
\def\LLX{1994,1995,1996,1997,1998,1999,2000,2001,2002,2004,2005,2006,2007,2008, 2009,2010}
\def\LLY{1718,1710,1708,1700,1698,1697,1691,1688,1683,1679,1671,1670,1663,1661, 1656,1649}
\CalculsRegLin{\LLX}{\LLY}
\addplot [thick,orange,domain=\LXmin:\LXmax,samples=2]{\COEFFa*x+\COEFFb};
\addlegendentry{$y = \fpeval{round(\COEFFa,3)}\,x + \fpeval{round(\COEFFb,3)}$};
\addlegendentry{$R^2=\fpeval{round(\COEFFrd,5)}$};
\end{axis}
\end{tikzpicture}

```

Code $\LaTeX$ 

Il existe également une commande auxiliaire, `\PointsRegLin` pour afficher le nuage de points avec quelques options, dans un environnement TikZ classique (sans pgfplot)...

Code $\LaTeX$

```

...
\begin{tikzpicture}[<options>]
...
\PointsRegLin[clés]{listeX}{listeY}
...
\end{tikzpicture}

```



Quelques **Clés** sont disponibles pour cette commande, essentiellement pour la mise en forme du nuage :

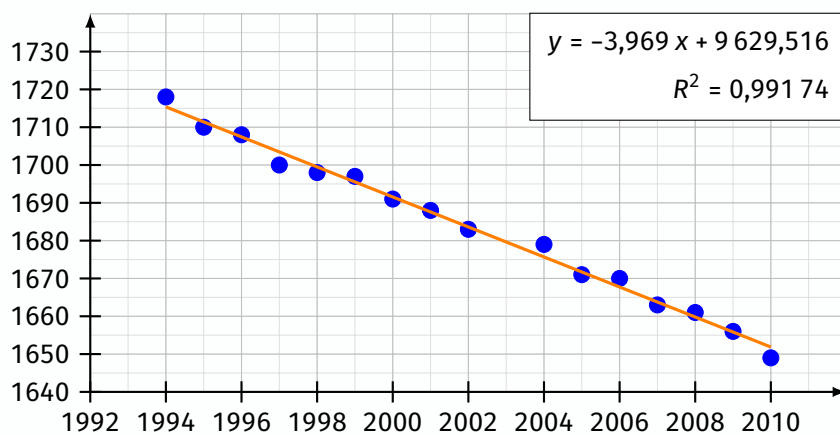
- la clé **<Couleur>** pour la couleur des points du nuage; défaut **<teal>**
- la clé **<Taille>** pour la taille des points (type *cercle*); défaut **<2pt>**
- la clé **<Ox>** pour spécifier la valeur initiale Ox (si changement d'origine); défaut **<0>**
- la clé **<Oy>** pour spécifier la valeur initiale Oy (si changement d'origine). défaut **<0>**

Code $\LaTeX$

```

\begin{tikzpicture}[x=0.5cm,y=0.05cm]
\draw[xstep=1,ystep=5,lightgray!50,very thin] (0,0) grid (20,100);
\draw[xstep=2,ystep=10,lightgray,thin] (0,0) grid (20,100);
\draw[thick,->,>=latex] (0,0)--(20,0) ;
\draw[thick,->,>=latex] (0,0)--(0,100) ;
\foreach \x in {1992,1994,...,2010} \draw[thick] ({\x-1992},4pt)--({\x-1992},-4pt) node[below] {$\x$} ;
\foreach \y in {1640,1650,...,1730} \draw[thick] (4pt,{\y-1640})--(-4pt,{\y-1640}) node[left] {$\y$} ;
\def\LLX{1994,1995,1996,1997,1998,1999,2000,2001,2002,2004,2005,2006,2007,2008, 2009,2010}
\def\LLY{1718,1710,1708,1700,1698,1697,1691,1688,1683,1679,1671,1670,1663,1661, 1656,1649}
\def\Ox{1992}\def\Oy{1640}
\CalcuLSRegLin{\LLX}{\LLY}
\PointsRegLin[Ox=1992,Oy=1640,Couleur=blue,Taille=3pt]{\LLX}{\LLY}
\draw[orange,very thick,samples=2,domain=\LXmin:\LXmax] plot ({\x-\Ox},{\COEFFa*({\x}-\Ox)+\COEFFb-\Oy}) ;
\matrix [draw,fill=white,below left] at (current bounding box.north east) {
  \node {$y=\num{\fpeval{round(\COEFFa,3)}}\x+\num{\fpeval{round(\COEFFb,3)}}$} ; \\
  \node {$R^2=\num{\fpeval{round(\COEFFrd,5)}}$} ; \\
};
\end{tikzpicture}

```

Code $\LaTeX$ 

39 Statistiques à deux variables

39.1 Idées



L'idée est de *prolonger* le paragraphe précédent pour proposer un environnement TikZ adapté à des situations venant de statistiques à deux variables.

Un des soucis pour ces situations est le fait que le repère dans lequel on travaille n'a pas forcément pour origine (0 ; 0).

De ce fait – pour éviter des erreurs de `dimension too large` liées à TikZ – il faut *décaler les axes* pour se ramener à une origine en 0.

Le code, intimement lié à un environnement `tikzpicture`, va donc :

- préciser les informations utiles comme `xmin`, `xmax`, `Ox`, `xgrille`, etc
- proposer des commandes (sans se soucier des *translations*!) pour :
 - tracer une grille (principale et/ou secondaire);
 - tracer les axes (avec légendes éventuelles) et éventuellement les graduer;

En utilisant les commandes de régression linéaire du paragraphe précédent, il sera de plus possible (sans calculs!) de :

- représenter le nuage de points;
- placer le point moyen;
- tracer la droite d'ajustement (obtenue par `ProfLycee`) ou une autre courbe.



Le package `pgfplots` peut être utilisé pour traiter ce genre de situation, mais ne l'utilisant pas, j'ai préféré préparer des macros permettant de s'affranchir de ce package (est-ce pertinent, ça c'est une autre question...).

Code $\text{\LaTeX}$

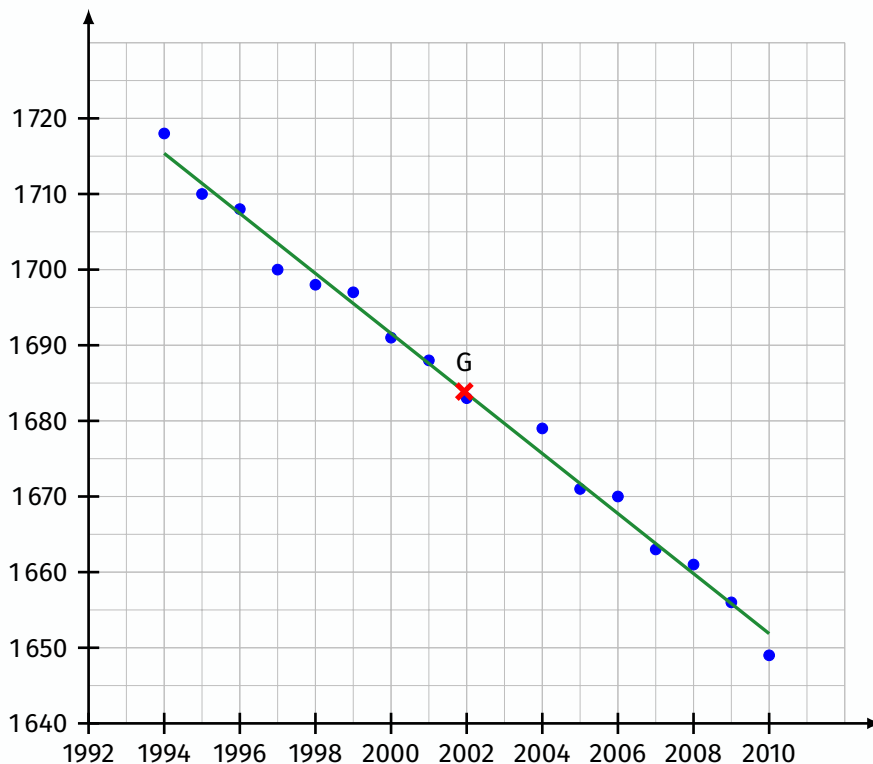
```
%Listes et calculs
\def\LLX{1994,1995,1996,1997,1998,1999,2000,2001,2002,2004,2005,2006,2007,2008, 2009,2010}
\def\LLY{1718,1710,1708,1700,1698,1697,1691,1688,1683,1679,1671,1670,1663,1661, 1656,1649}
\CalculsRegLin{\LLX}{\LLY}
```

Code $\text{\LaTeX}$

```
%tracé (simple), les options seront présentées juste après
\begin{tikzpicture}%
[x=0.5cm,y=0.1cm,                                %unités
Ox=1992,xmin=1992,xmax=2012,xgrille=2,xgrilles=1,  %axe Ox
Oy=1640,ymin=1640,ymax=1730,ygrille=10,ygrilles=5] %axe Oy
\GrilleTikz \AxesTikz                             %grilles et axes
\AxexTikz[Annee]{1992,1994,...,2010}               %axeOx
\AxyTikz{1640,1650,...,1720}                       %axeOy
\NuagePointsTikz{\LLX}{\LLY}                      %nuage
\CourbeTikz[line width=1.25pt,CouleurVertForet,samples=2]%
{\COEFFa*x+\COEFFb}{\LXmin:\LXmax}                %droite de régression
\PointMoyenTikz                                    %point moyen
\end{tikzpicture}
```

Code $\text{\LaTeX}$

```
%tracé avec options fenêtre par défaut
\begin{tikzpicture}%
[....]
\FenetreSimpleTikz<Annee>{1992,1994,...,2010}{1640,1650,...,1720} %paramètres
\NuagePointsTikz{\LLX}{\LLY} %fenêtre "simple"
\CourbeTikz[line width=1.25pt,CouleurVertForet,samples=2]% %nuage
{\COEFFa*x+\COEFFb}{\LXmin:\LXmax} %droite de régression
\PLnuageptmoy %point moyen
\end{tikzpicture}
```



39.2 Commandes, clés et options



Les **paramètres** nécessaires à la bonne utilisation des commandes suivantes sont à déclarer directement dans l'environnement `tikzpicture`, seules les versions « x » sont présentées ici :

- **xmin**, stockée dans `\xmin`; défaut **-3**
- **xmax**, stockée dans `\xmax`; défaut **3**
- **Ox**, stockée dans `\axexOx`, origine de l'axe (Ox); défaut **0**
- **xgrille**, stockée dans `\xgrille`, graduation principale; défaut **1**
- **xgrilles**, stockée dans `\xgrilles`, graduation secondaire. défaut **0.5**

La fenêtre d'affichage (de sortie) sera donc *portée* par le rectangle de coins (xmin;ymin) et (xmax;ymax); ce qui correspond en fait à la fenêtre TikZ *portée* par le rectangle de coins (xmin-Ox;ymin-Oy) et (xmax-Ox;ymax-Oy).

Les commandes ont – pour certaines – pas mal de **clés** pour des réglages fins, mais dans la majorité des cas elles ne sont pas forcément *utiles*.



Pour illustrer les commandes et options de ce paragraphe, la base sera le graphique présenté précédemment.

```
%...code tikz
\GrilleTikz[options][options grille ppale][options grille second.]
```



Cette commande permet de tracer une grille principale et/ou une grille secondaire :

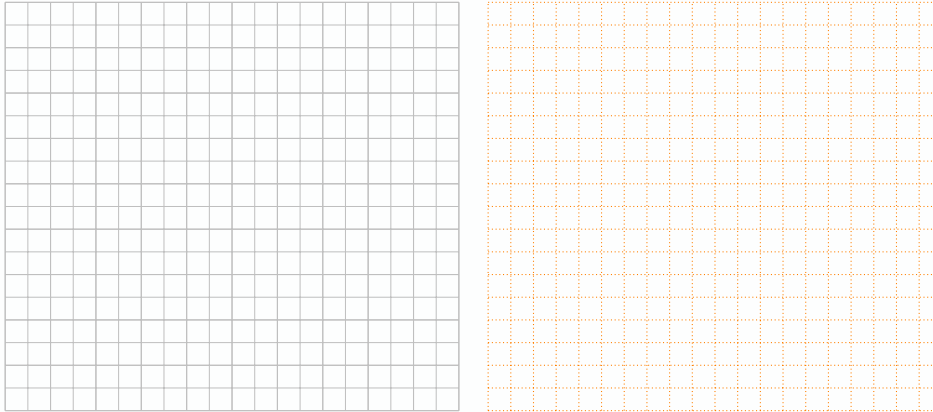
- les premières **clés** sont les booléens **Affp** et **Affs** qui affichent ou non les grilles; défaut **true**
- les options des grilles sont en TikZ. défaut **thin,lightgray** et **very thin,lightgray**

Code $\LaTeX$

```

\begin{tikzpicture}%
[x=0.3cm,y=0.06cm,%
Ox=1992,xmin=1992,xmax=2012,xgrille=2,xgrilles=1,%
Oy=1640,ymin=1640,ymax=1730,ygrille=10,ygrilles=5]
\GrilleTikz
\end{tikzpicture}
~~
\begin{tikzpicture}%
[x=0.3cm,y=0.06cm,%
Ox=1992,xmin=1992,xmax=2012,xgrille=2,xgrilles=1,%
Oy=1640,ymin=1640,ymax=1730,ygrille=10,ygrilles=5]
\GrilleTikz[Affp=false][][orange,densely dotted]
\end{tikzpicture}

```

Code $\LaTeX$ Code $\LaTeX$

```

%...code tikz
\AxesTikz[options]

```



Cette commande permet de tracer les axes, avec des **clés** :

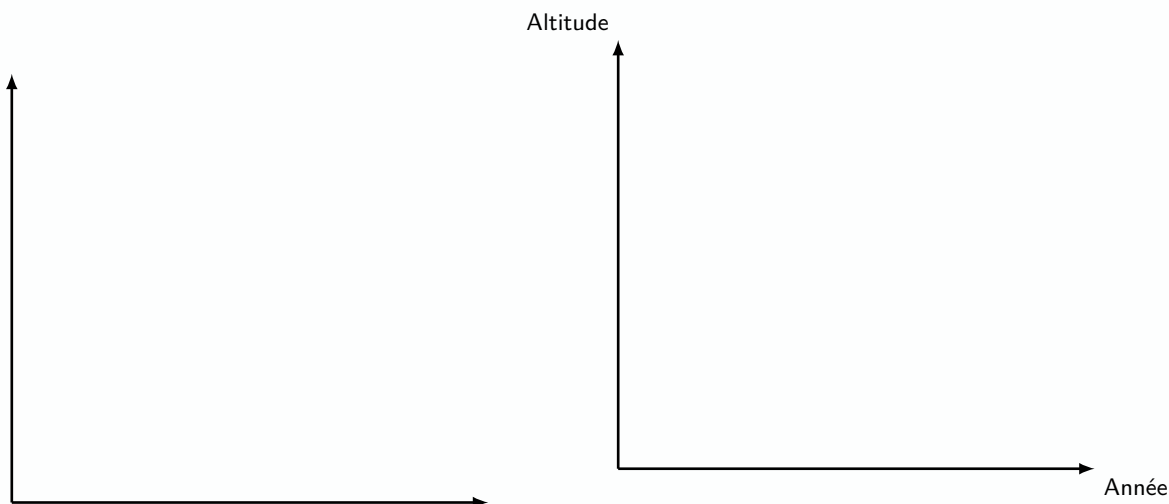
- **<Épaisseur>** qui est l'épaisseur des traits; défaut **<1.25pt>**
- **<Police>** qui est le style des labels des axes; défaut **<\normalsize\normalfont>**
- **<2.1.2> <ElargirOx>** qui est le % l'élargissement **<global>** ou **<G/D>** de l'axe (Ox); défaut **<0/0.05>**
- **<2.1.2> <ElargirOy>** qui est le % l'élargissement **<global>** ou **<B/H>** de l'axe (Oy); défaut **<0/0.05>**
- **<Labelx>** qui est le label de l'axe (Ox); défaut **<\$x\$>**
- **<Labely>** qui est le label de l'axe (Oy); défaut **<\$y\$>**
- **<AffLabel>** qui est le code pour préciser quels labels afficher, entre **<x>**, **<y>** ou **<xy>**; défaut **<vide>**
- **<PosLabelx>** pour la position du label de (Ox) en bout d'axe; défaut **<right>**
- **<PosLabely>** pour la position du label de (Oy) en bout d'axe; défaut **<above>**
- **<EchelleFleche>** qui est l'échelle de la flèche des axes; défaut **<1>**
- **<TypeFleche>** qui est le type de la flèche des axes. défaut **<latex>**

Code $\LaTeX$

```
%code tikz
\AxesTikz

%code tikz
\AxesTikz%
[AffLabel=xy,Labelx={Année},Labely={Altitude},%
PosLabelx={below right},PosLabely={above left},%
Police=\small\sffamily]
```

Code $\LaTeX$



Code $\LaTeX$

```
%...code tikz
\AxexTikz[options]{valeurs}
\AxeYtikz[options]{valeurs}
```



Ces commande permet de tracer les graduations des axes, avec des **clés** identiques pour les deux directions :

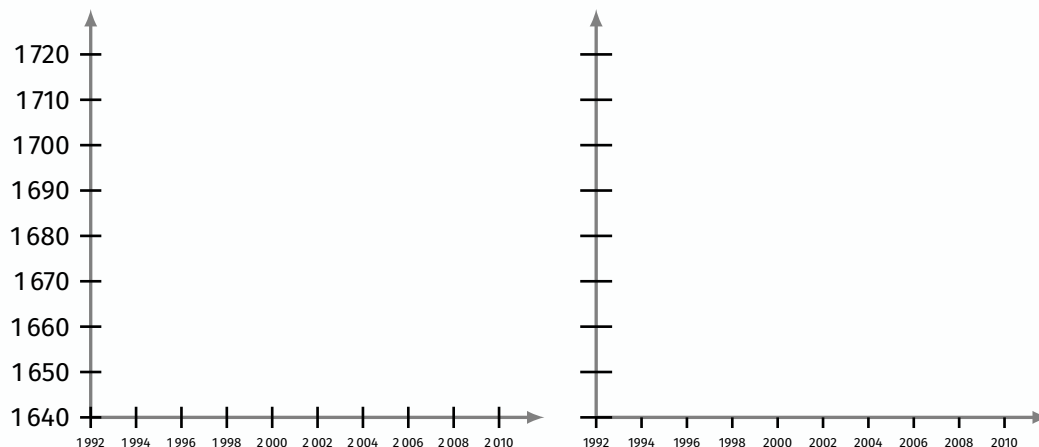
- **Épaisseur** qui est l'épaisseur des graduations; défaut **1pt**
- **Police** qui est le style des labels des graduations; défaut $\langle \backslash \text{normalsize} \backslash \text{normalfont} \rangle$
- **PosGrad** qui est la position des graduations par rapport à l'axe; défaut **below** et **left**
- **HautGrad** qui est la position des graduations (sous la forme **lgt** ou **lgt a/lgt b**); défaut **4pt**
- le booléen **AffGrad** pour afficher les valeurs (formatés avec `\num` donc dépendant de `\sisetup`) des graduations; défaut **true**
- le booléen **AffOrigine** pour afficher la graduation de l'origine; défaut **true**
- le booléen **Annee** qui permet de ne pas formater les valeurs des graduations (type année). défaut **false**

Code $\LaTeX$

```
%code tikz
\AxexTikz[Police=\small]{1992,1994,...,2010}
\AxexTikz{1640,1650,...,1720}

%code tikz
\AxeYtikz[Police=\small,Annee,HautGrad=0pt/4pt]{1992,1994,...,2010}
\AxeYtikz[AffGrad=false,HautGrad=6pt]{1640,1650,...,1720}

%des axes fictifs (en gris) sont rajoutés pour la lisibilité du code de sortie
```



39.3 Commandes annexes



Il existe, de manière marginale, quelques commandes complémentaires qui ne seront pas trop détaillées mais qui sont présentes dans l'introduction :

- `FenetreTikz` qui restreint les tracés à la fenêtre (utile pour des courbes qui *débordent*);
- `FenetreSimpleTikz` qui permet d'automatiser le tracé des grilles/axes/graduations dans leurs versions par défaut, avec peu de paramétrages;
- `OrigineTikz` pour rajouter le libellé de l'origine si non affiché par les axes.

```
%code tikz
\FenetreTikz %on restreint les tracés
\FenetreSimpleTikz[opt](opt axes)<opt axe Ox>{liste valx}<opt axe Oy>{liste valy}
```

39.4 Interactions avec CalculsRegLin

```
%...code tikz
\NuagePointsTikz[options]{listeX}{listeY}
```



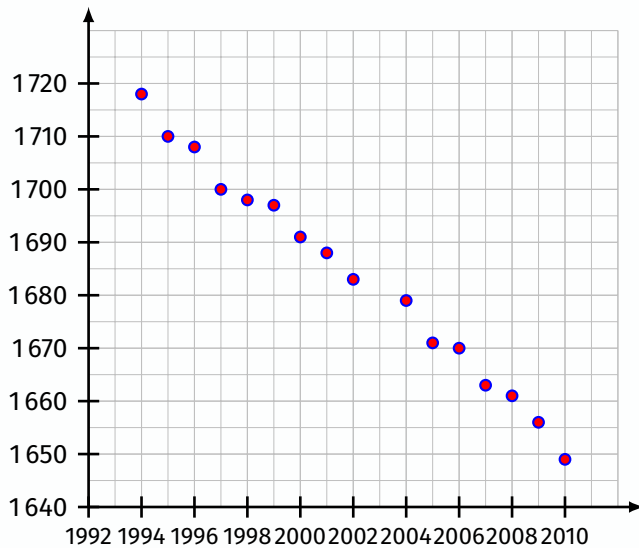
Cette commande, liée à la commande `CalculsRegLin` permet de représenter le nuage de points associé aux deux listes, avec les **clés** suivantes :

- **⟨Taille⟩** qui est la taille des points du nuage; défaut **⟨2pt⟩**
- **⟨Style⟩** parmi **⟨o⟩** (rond) ou **⟨x⟩** (croix) ou **⟨+⟩** (plus); défaut **⟨o⟩**
- **⟨Couleur⟩** qui est la couleur (éventuellement **⟨couleurA/couleurB⟩** pour les ronds). défaut **⟨blue⟩**

```
\def\LLX{1994,1995,1996,1997,1998,1999,2000,2001,2002,2004,2005,2006,2007,2008, 2009,2010}
\def\LLY{1718,1710,1708,1700,1698,1697,1691,1688,1683,1679,1671,1670,1663,1661, 1656,1649}
```

```
\begin{tikzpicture}[...]
\NuagePointsTikz[Couleur=blue/red]{\LLX}{\LLY}
\end{tikzpicture}
```

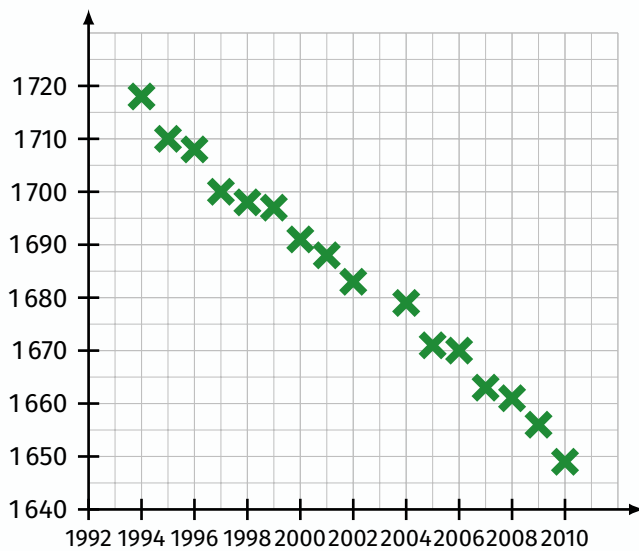
Code $\text{\LaTeX}$



Code $\text{\LaTeX}$

```
\begin{tikzpicture}[...]
\NuagePointsTikz[Couleur=CouleurVertForet,Style=x,Taille=6pt]{\LLX}{\LLY}
\end{tikzpicture}
```

Code $\text{\LaTeX}$



Code $\text{\LaTeX}$

```
%...code tikz
\PointMoyenTikz[options]
```



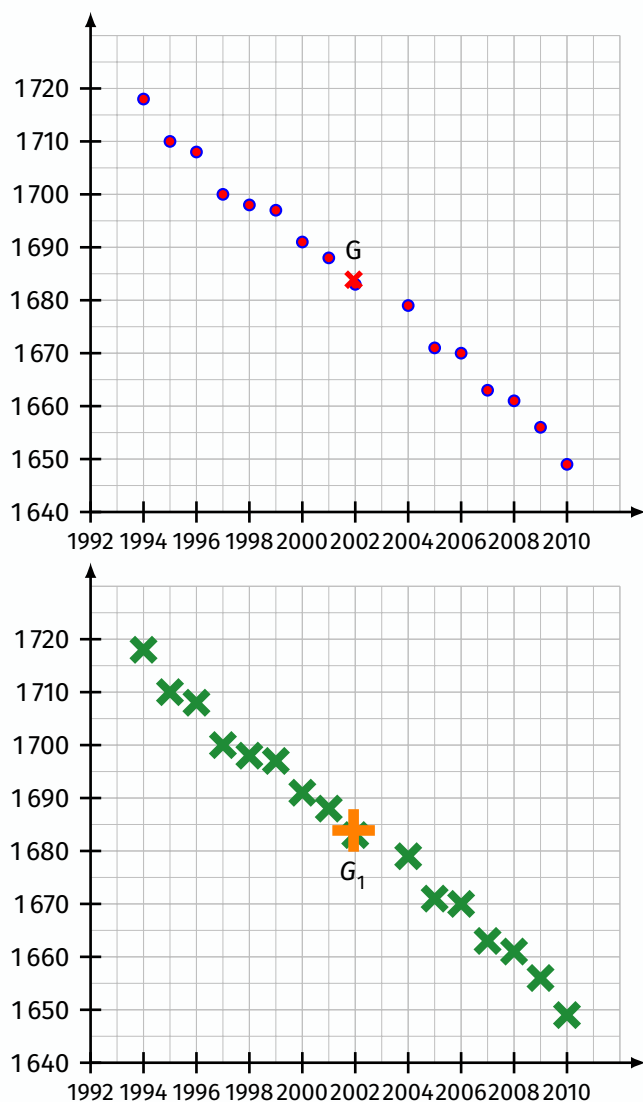
Cette commande permet de rajouter le point moyen du nuage, calculé par la commande `\CalculsRegLin`, avec les **clés** :

- | | |
|--|--|
| – Police , comme précédemment; | défaut <code>\normalsize\normalfont</code> ; |
| – Taille , taille du point moyen; | défaut <code>4pt</code> |
| – Couleur , couleur du point moyen; | défaut <code>red</code> |
| – Style parmi <code>o</code> (rond) ou <code>x</code> (croix) ou <code>+</code> (plus); | défaut <code>o</code> |
| – xg , abscisse du point moyen, récupérable via <code>\CalculsRegLin</code> ; | défaut <code>\LXmoy</code> |
| – yg , ordonnée du point moyen, récupérable via <code>\CalculsRegLin</code> ; | défaut <code>\LYmoy</code> |
| – Nom , label du point moyen; | défaut <code>G</code> |
| – Pos qui est la position du label par rapport au point; | défaut <code>above</code> |
| – Decal qui est l'éloignement de la position du label par rapport au point; | défaut <code>0pt</code> |
| – la booléen AffNom qui affiche ou non le libellé. | défaut <code>true</code> |

Code $\text{\LaTeX}$

```
\def\LLX{1994,1995,1996,1997,1998,1999,2000,2001,2002,2004,2005,2006,2007,2008, 2009,2010}
\def\LLY{1718,1710,1708,1700,1698,1697,1691,1688,1683,1679,1671,1670,1663,1661, 1656,1649}
\CalculsRegLin{\LLX}{\LLY}

\begin{tikzpicture}[...]
\NuagePointsTikz[Couleur=blue/red]{\LLX}{\LLY}
\PointMoyenTikz
\end{tikzpicture}
~~
\begin{tikzpicture}[...]
\NuagePointsTikz[Couleur=CouleurVertForet,Style=x,Taille=6pt]{\LLX}{\LLY}
\PointMoyenTikz[Couleur=orange,Taille=8pt,Style=+,Nom={G_1$},Pos=below]
\end{tikzpicture}
```



```
%...code tikz
\CourbeTikz[options]{formule}{domaine}
```



Cette commande permet de rajouter une courbe sur le graphique (sans se soucier de la transformation de son expression) avec les arguments :

- **optionnels** qui sont – en TikZ – les paramètres du tracé;
- le premier *obligatoire*, est – en langage TikZ – l'expression de la fonction à tracer, donc avec $\LaTeX$ comme variable;
- le second *obligatoire* est le domaine du tracé, sous la forme $\LaTeX$ valxmin:valxmax.



L'idée principale est de récupérer les variables de la régression linéaire pour tracer la droite d'ajustement à *moindres frais*!



Toute courbe peut être tracée sur ce principe, par contre il faudra saisir la fonction à *la main*.

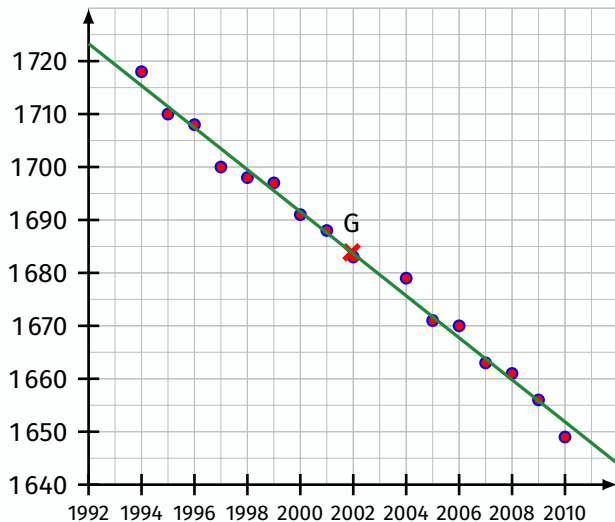
Code $\LaTeX$

```

\def\LLX{1994,1995,1996,1997,1998,1999,2000,2001,2002,2004,2005,2006,2007,2008, 2009,2010}
\def\LLY{1718,1710,1708,1700,1698,1697,1691,1688,1683,1679,1671,1670,1663,1661, 1656,1649}
\CalculsRegLin{\LLX}{\LLY}

\begin{tikzpicture}[...]
\NuagePointsTikz[Couleur=blue/red]{\LLX}{\LLY} \PointMoyenTikz
\CourbeTikz[line width=1.25pt,CouleurVertForet,samples=2]{\COEFFa*\x+\COEFFb}{\xmin:\xmax}
\end{tikzpicture}

```

Code $\LaTeX$ 

39.5 Exemple complémentaire, pour illustration

Code $\LaTeX$

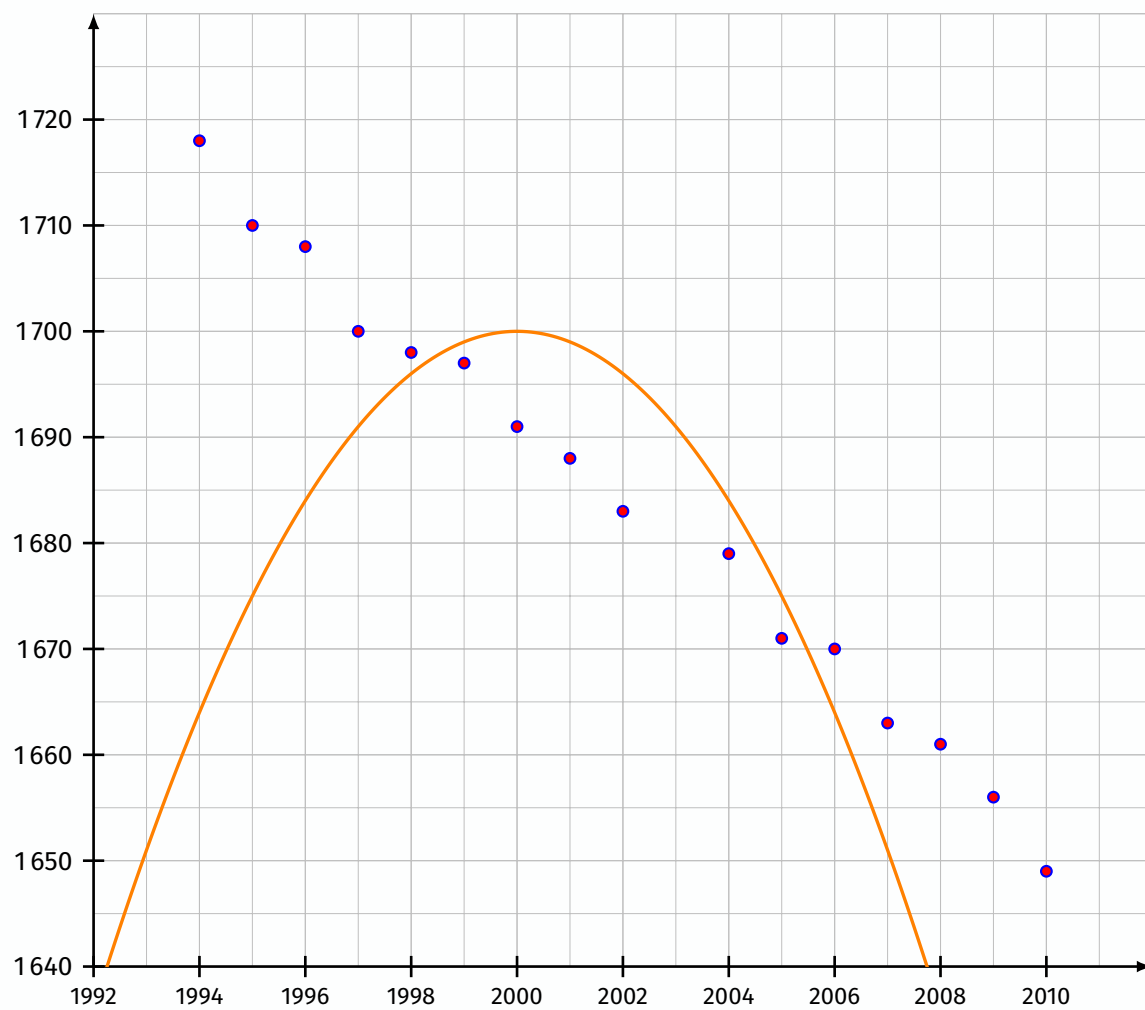
```

\def\LLX{1994,1995,1996,1997,1998,1999,2000,2001,2002,2004,2005,2006,2007,2008,2009,2010}
\def\LLY{1718,1710,1708,1700,1698,1697,1691,1688,1683,1679,1671,1670,1663,1661,1656,1649}

%la courbe n'a pas de lien avec le nuage
%elle illustre l'interaction des commandes "nuage" avec les autres commandes

\begin{tikzpicture}[...]
\NuagePointsTikz[Couleur=blue/red]{\LLX}{\LLY} \FenetreTikz %on fixe la fenêtre
\CourbeTikz[line width=1.25pt,orange,samples=250]{-(\x-2000)*(\x-2000)+1700}{\xmin:\xmax}
\end{tikzpicture}

```



40 Boîtes à moustaches

40.1 Introduction

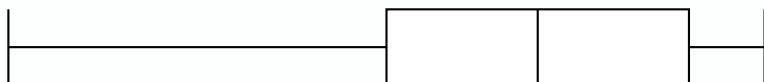


L'idée est de proposer une commande, à intégrer dans un environnement TikZ, pour tracer une boîte à moustaches grâce aux paramètres, saisis par l'utilisateur.

Le code ne calcule pas les paramètres, il ne fait *que* tracer la boîte à moustaches!

Code $\text{\LaTeX}$

```
\begin{tikzpicture}
\BoiteMoustaches{10/15/17/19/20}
\end{tikzpicture}
```



Étant donnée que la commande est intégrée dans un environnement TikZ, les unités peuvent/doivent donc être précisées, *comme d'habitude*, si besoin.

40.2 Clés et options

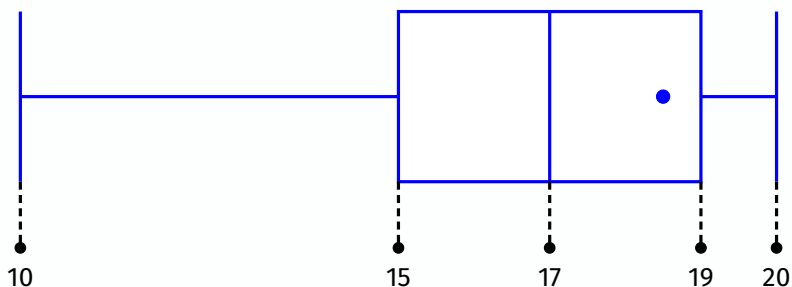


Quelques **clés** sont disponibles pour cette commande :

- la clé **Couleur** qui est la couleur de la boîte; défaut **black**
- la clé **Elevation** qui est la position verticale (ordonnée des moustaches) de la boîte; défaut **1.5**
- la clé **Hauteur** qui est la hauteur de la boîte; défaut **1**
- la clé **Moyenne** qui est la moyenne (optionnelle) de la série;
- la clé **Epaisseur** qui est l'épaisseur des traits de la boîte; défaut **thick**
- la clé **Remplir** qui est la couleur de remplissage de la boîte; défaut **white**
- le booléen **AffMoyenne** qui permet d'afficher ou non la moyenne (sous forme d'un point); défaut **false**
- le booléen **Pointilles** qui permet d'afficher des pointillés au niveau des paramètres; défaut **false**
- le booléen **Valeurs** qui permet d'afficher les valeurs des paramètres au niveau des abscisses. défaut **false**

Code $\text{\LaTeX}$

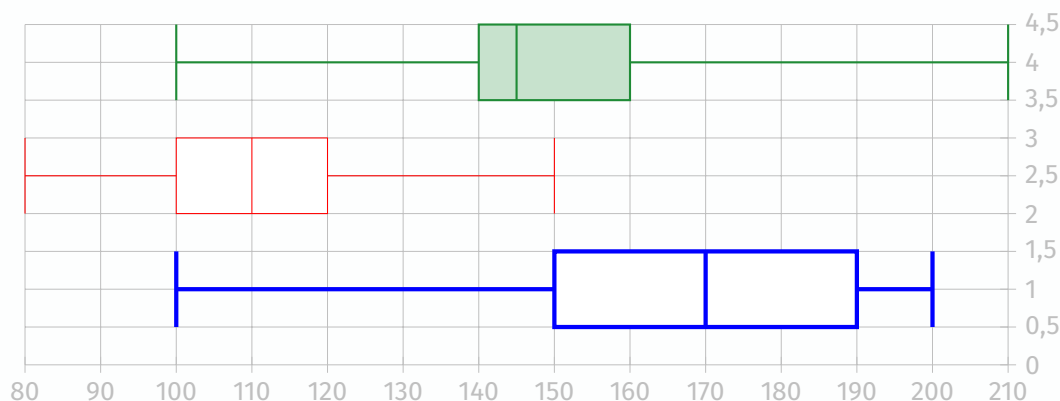
```
\begin{tikzpicture}
\BoiteMoustaches[Epaisseur=very thick,Moyenne=18.5,Couleur=blue,AffMoyenne,%
Pointilles,Valeurs,Hauteur=2.25,Elevation=2]{10/15/17/19/20}
\end{tikzpicture}
```



Code $\LaTeX$

```
%une grille a été rajoutée pour visualiser la "position verticale"
\begin{center}
\begin{tikzpicture}[x=0.1cm]
  \BoiteMoustaches[Epaisseur=ultra thick,Couleur=blue]{100/150/170/190/200}
  \BoiteMoustaches[Epaisseur=thin,Elevation=2.5,Couleur=red]{80/100/110/120/150}
  \BoiteMoustaches%
    [Elevation=4,Couleur=CouleurVertForet,Remplir=CouleurVertForet!25]{100/140/145/160/210}
\end{tikzpicture}
\end{center}
```

Code $\LaTeX$



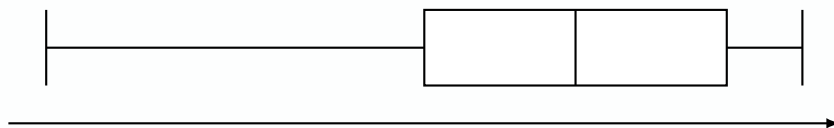
40.3 Commande pour placer un axe horizontal



L'idée est de proposer, en parallèle de la commande précédente, une commande pour tracer un axe horizontal « sous » les éventuelles boîtes à moustaches.

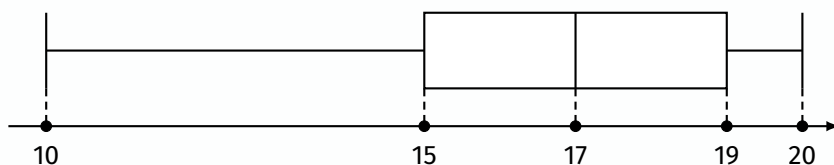
Code $\LaTeX$

```
\begin{tikzpicture}
  \BoiteMoustachesAxe[Min=10,Max=20]
  \BoiteMoustaches{10/15/17/19/20}
\end{tikzpicture}
```



Code $\LaTeX$

```
\begin{tikzpicture}
  \BoiteMoustachesAxe[Min=10,Max=20]
  \BoiteMoustaches[Valeurs,Pointilles]{10/15/17/19/20}
\end{tikzpicture}
```





Quelques **clés** sont disponibles pour cette commande :

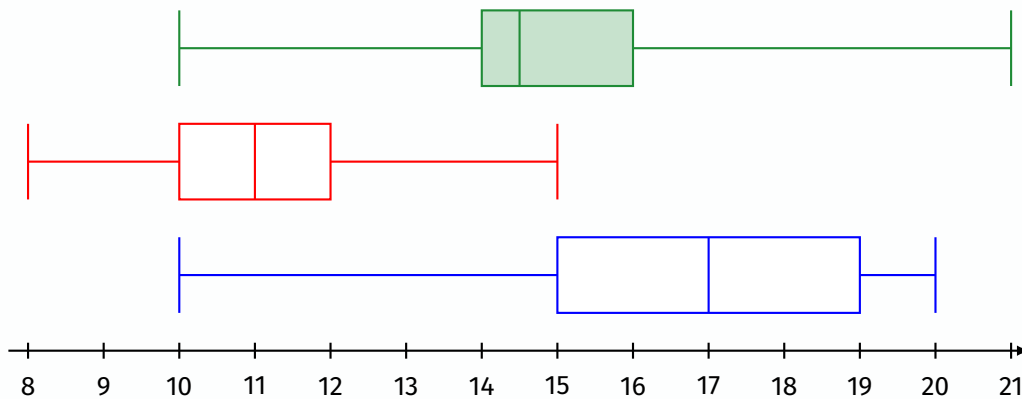
- la clé **Min** qui est la valeur minimale de l'axe horizontal;
- la clé **Max** qui est la valeur maximale de l'axe horizontal;
- la clé **Elargir** qui est le pourcentage l'élargissement de l'axe;
- la clé **Epaisseur** qui est l'épaisseur des traits de la boîte;
- la clé **Valeurs** qui est la liste (compréhensible en TikZ) des valeurs à afficher.

défaut **0.1**

défaut **thick**

Code $\text{\LaTeX}$

```
\begin{tikzpicture}
\BoiteMoustachesAxe[Min=8,Max=21,AffValeurs,Valeurs={8,9,...,21},Elargir=0.02]
\BoiteMoustaches[Moyenne=18.5,Couleur=blue]{10/15/17/19/20}
\BoiteMoustaches[Elevation=2.5,Couleur=red]{8/10/11/12/15}
\BoiteMoustaches[Elevation=4,Couleur=CouleurVertForet,Remplir=CouleurVertForet!25]{10/14/14.5/16/21}
\end{tikzpicture}
```



Le placement des différentes boîtes n'est pas automatique, donc il faut penser à cela avant de se lancer dans le code.

Sachant que la hauteur par défaut est de 1, il est – a priori – intéressant de placer les boîtes à des **élevations** de 1 puis 2,5 puis 4 etc

41 Histogrammes

41.1 Introduction



2.6.7 L'idée est de proposer une commande pour tracer un histogramme à classes régulières ou non. La commande, qui utilise `TikZ`, est autonome (ceci étant dû à la gestion en interne des unités!), et ne permet pas de rajouter une fois le graphique affiché.



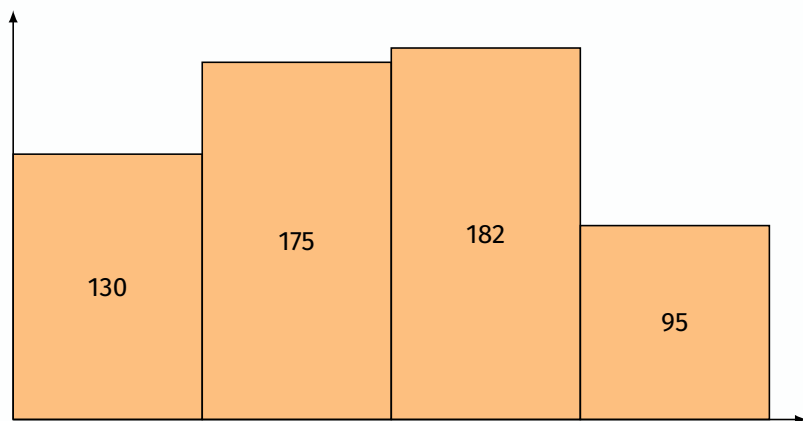
La commande fonctionne avec des données classe/effectif, qui seront à traduire sous la forme `BorneInf/BorneSup/Effectif`.

Code $\LaTeX$

```
\Histogramme(*)[options]{données}
```

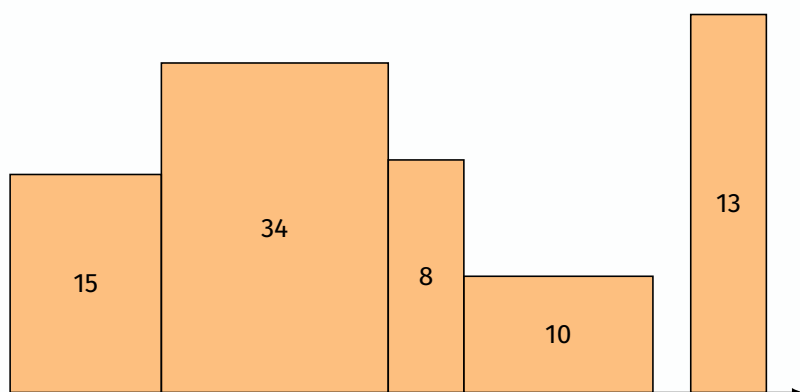
Code $\LaTeX$

```
%classes régulières  
\Histogramme{7/9/130 9/11/175 11/13/182 13/15/95}
```



Code $\LaTeX$

```
%classes non régulières  
\Histogramme*{0/20/15 20/50/34 50/60/8 60/85/10 90/100/13}
```



Contrairement aux autres commandes graphiques, qui sont souvent à intégrer dans un environnement `TikZ`, la commande `\Histogramme` aura besoin de connaître les dimensions finales du graphique pour fonctionner!

Les dimensions correspondent à celles des rectangles avec les éventuelles modifications horizontales et/ou verticales spécifiées.

41.2 Clés et options



La version *étoilée* permet de préciser que les classes ne sont pas d'amplitudes régulières.

Le premier argument, optionnel et entre `[...]` propose les **clés** principales suivantes :

- **⟨DebutOx⟩** : permet de préciser le début de l'axe horizontal (sinon c'est par défaut la borne inférieure de la première classe);
défaut : **⟨vide⟩**
- **⟨FinOx⟩** : permet de préciser la fin de l'axe horizontal (sinon c'est par défaut la borne supérieure de la dernière classe);
défaut : **⟨vide⟩**
- **⟨Largeur⟩** : largeur en cm du graphique créé (entre **⟨DebutOx⟩** et **⟨FinOx⟩**);
défaut : **⟨10⟩**
- **⟨Hauteur⟩** : hauteur en cm du graphique créé (par rapport à l'effectif maximal ou la grille éventuelle);
défaut : **⟨5⟩**
- **⟨ListeCouleurs⟩** : liste des couleurs des rectangles (unique ou sous la forme `{Cou1A,Cou1B,...}`);
défaut : **⟨orange⟩**
- **⟨ElargirX⟩** et **⟨ElargirY⟩** : pour rajouter une petite longueur au bout des axes;
défaut : **⟨5mm⟩**
- **⟨LabelX⟩** et **⟨LabelY⟩** : pour les labels des axes;
défaut : **⟨vide⟩**
- **⟨GradX⟩** et **⟨GradY⟩** : pour les graduations et valeurs des axes (langage `tikz`);
défaut : **⟨vide⟩**
- **⟨AffEffectifs⟩** : booléen pour afficher les effectifs;
défaut : **⟨true⟩**
- **⟨PosEffectifs⟩** : choix de la position des effectifs parmi **⟨bas,milieu,haut,dessus⟩**;
défaut : **⟨milieu⟩**
- **⟨Remplir⟩** : booléen pour remplir les rectangles;
défaut : **⟨true⟩**
- **⟨Opacite⟩** : choix de l'opacité du remplissage;
défaut : **⟨0.5⟩**
- **⟨AffBornes⟩** : booléen pour afficher les bornes des classes;
défaut : **⟨false⟩**
- **⟨GrilleV⟩** : booléen pour afficher une grille verticale (pour les classes régulières, à la manière d'un tableau);
défaut : **⟨true⟩**
- **⟨PoliceAxes⟩** : police pour les axes;
défaut : **⟨\normalsize\normalfont⟩**
- **⟨PoliceEffectifs⟩** : police pour les effectifs;
défaut : **⟨\normalsize\normalfont⟩**
- **⟨EpaisseurTraits⟩** : épaisseur des traits (langage `tikz`).
défaut : **⟨semithick⟩**

2.6.8 Quelques clés sont spécifiques à la grille (éventuelle) des histogrammes non réguliers (avec ajustement vertical et légende) :

- **⟨Grille⟩** : création de la grille, sous la forme **⟨GradX/UniteAire⟩**;
défaut : **⟨vide⟩**
- **⟨ExtraGrilleY⟩** : pour rajouter une ligne à la grille en vertical;
défaut : **⟨0⟩**
- **⟨PosLegende⟩** : pour préciser le *carreau* de la légende éventuelle.
défaut : **⟨vide⟩**

Le second argument, obligatoire et entre `{...}` permet de préciser les données utilisées sous la forme

`BorneInf/BorneSup/Effectif BorneInf/BorneSup/Effectif ...`.

Code $\text{\LaTeX}$

```
%style pour la grille
\tikzset{stylegrillehisto/.style={line width=0.4pt,lightgray}}
```

41.3 Exemple avec des classes régulières

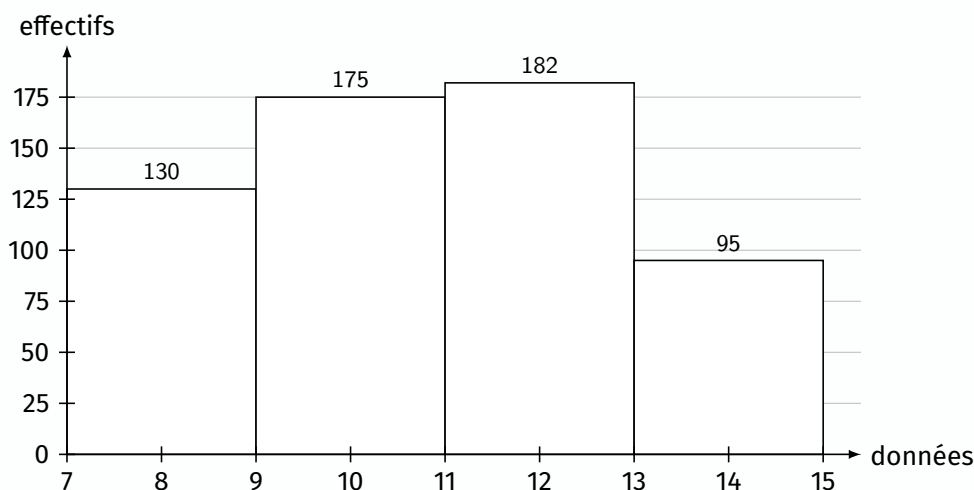


Avec la série suivante :

Classes	[7;9[	[9;11[	[11;13[	[13;15]
Effectifs	130	175	182	95

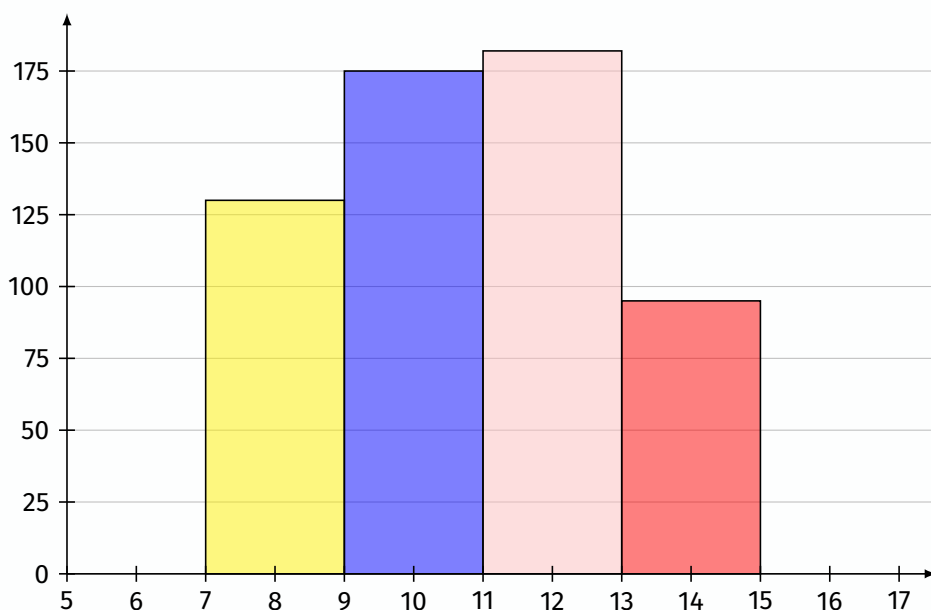
Code $\text{\LaTeX}$

```
\Histogramme[ListeCouleurs={white},Opacite=1,%  
  GradX={7,8,...,15},LabelX={données},GradY={0,25,...,175},LabelY={effectifs},%  
  PoliceEffectifs=\small\sffamily,PosEffectifs=dessus]%  
{7/9/130 9/11/175 11/13/182 13/15/95}
```



Code $\text{\LaTeX}$

```
\Histogramme[Largeur=11,Hauteur=7,%  
  ListeCouleurs={yellow,blue,pink,red},%  
  DebutOx=5,FinOx=17,GradX={5,6,...,17},GradY={0,25,...,175},%  
  AffEffectifs=false]%  
{7/9/130 9/11/175 11/13/182 13/15/95}
```



41.4 Exemple avec des classes non régulières

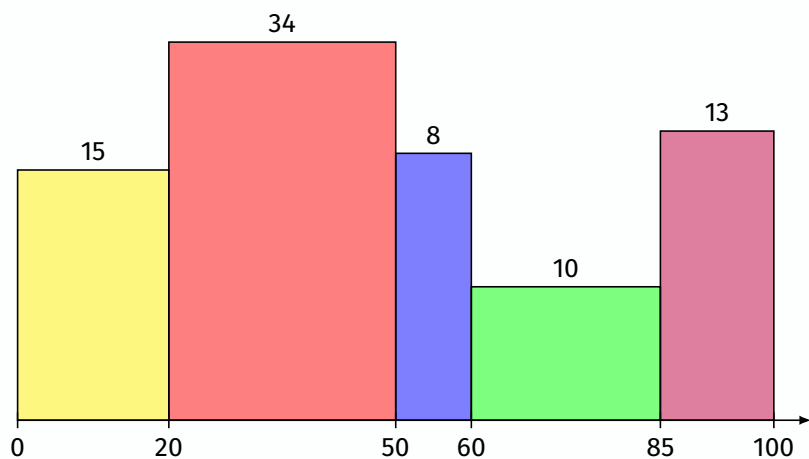


Avec la série suivante :

Classes	[0;20[	[20;50[	[50;60[	[60;85[	[85;100]
Effectifs	15	34	8	10	13

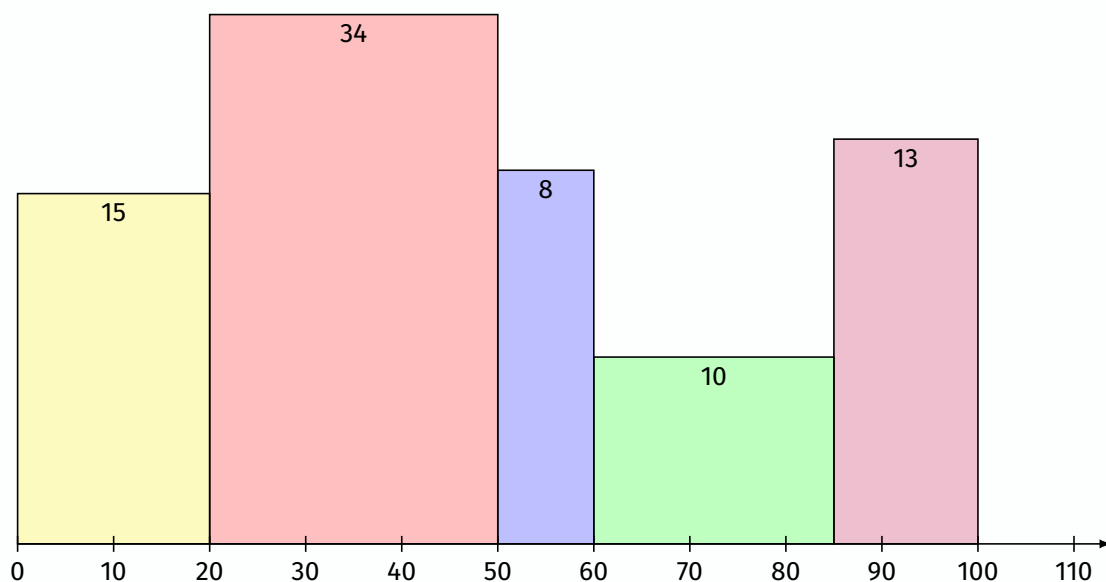
Code $\LaTeX$

```
\Histogramme*[%  
  ListeCouleurs={yellow,red,blue,green,purple},%  
  PosEffectifs=dessus,AffBornes]  
{0/20/15 20/50/34 50/60/8 60/85/10 85/100/13}
```



Code $\LaTeX$

```
\Histogramme*[%  
  Largeur=14,Hauteur=7,FinOx=110,%  
  ListeCouleurs={yellow,red,blue,green,purple},Opacite=0.25,%  
  GradX={0,10,...,110},%  
  PosEffectifs=haut]  
{0/20/15 20/50/34 50/60/8 60/85/10 85/100/13}
```





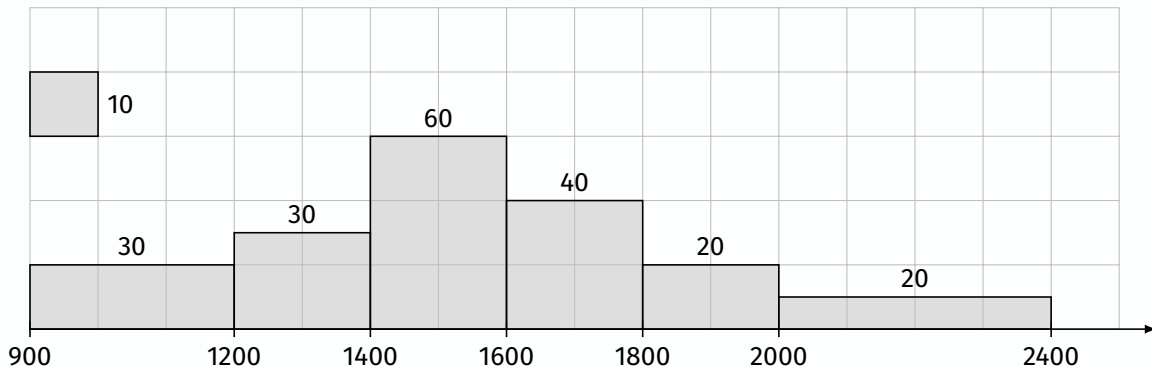
Avec la série suivante :

Classes	[900;1 200[	[1 200;1 400[	[1 400;1 600[	[1 600;1 800[	[1 800;2 000[	[2 000;2 400]
Effectifs	30	30	60	40	20	20

Code $\text{\LaTeX}$

%choix des unités 0.85cm par petit carreau avec 17H et 5V

```
\Histogramme*[%  
  Largeur=13.6,Hauteur=4.25,FinOx=2500,%  
  PosLegende=0/3,Grille=100/10,ExtraGrilleY=1,%  
  ListeCouleurs=lightgray,%  
  AffBornes,PosEffectifs=dessus]  
{900/1200/30 1200/1400/30 1400/1600/60 1600/1800/40 1800/2000/20 2000/2400/20}
```



42 Courbe des ECC/FCC, paramètres

42.1 Introduction



3.01a L'idée est de proposer des commandes (et un environnement) pour tracer automatiquement une courbe des ECC (ou des FCC), et/ou déterminer (par interpolation) une valeur approchée des paramètres d'une série continue.

La commande, qui utilise *TikZ*, est autonome (ceci étant dû à la gestion en interne des unités!), et ne permet pas de rajouter une fois le graphique affiché.

L'environnement permet de rajouter des éléments complémentaires sur le graphique.

Il existe également une commande indépendante qui ne fait *que* déterminer les paramètres par interpolation.



Contrairement aux autres commandes graphiques, qui sont souvent à intégrer dans un environnement *TikZ*, le code aura besoin de connaître les dimensions finales du graphique pour fonctionner!

Les dimensions correspondent à celles des rectangles avec les éventuelles modifications horizontales et/ou verticales spécifiées.

Le code se charge de déterminer une valeur des paramètres, pour utilisation ultérieure (avec arrondis éventuels car ils sont obtenus par *conversions*) :

- le premier quartile, Q_1 , est stocké dans la macro `\ValPremQuartile`;
- la médiane, méd, est stocké dans la macro `\ValMed`;
- le troisième quartile, Q_3 , est stocké dans la macro `\ValTroisQuartile`.

Code $\LaTeX$

```
%commande pour déterminer les paramètres
\MedianeQuartilesECC{liste valeurs}{liste effectifs}
```

Code $\LaTeX$

```
\CourbeECC[clés]{liste valeurs}{liste effectifs}
```

Code $\LaTeX$

```
\begin{EnvCourbeECC}[clés]{liste valeurs}{liste effectifs}
%commandes tikz
\end{EnvCourbeECC}
```

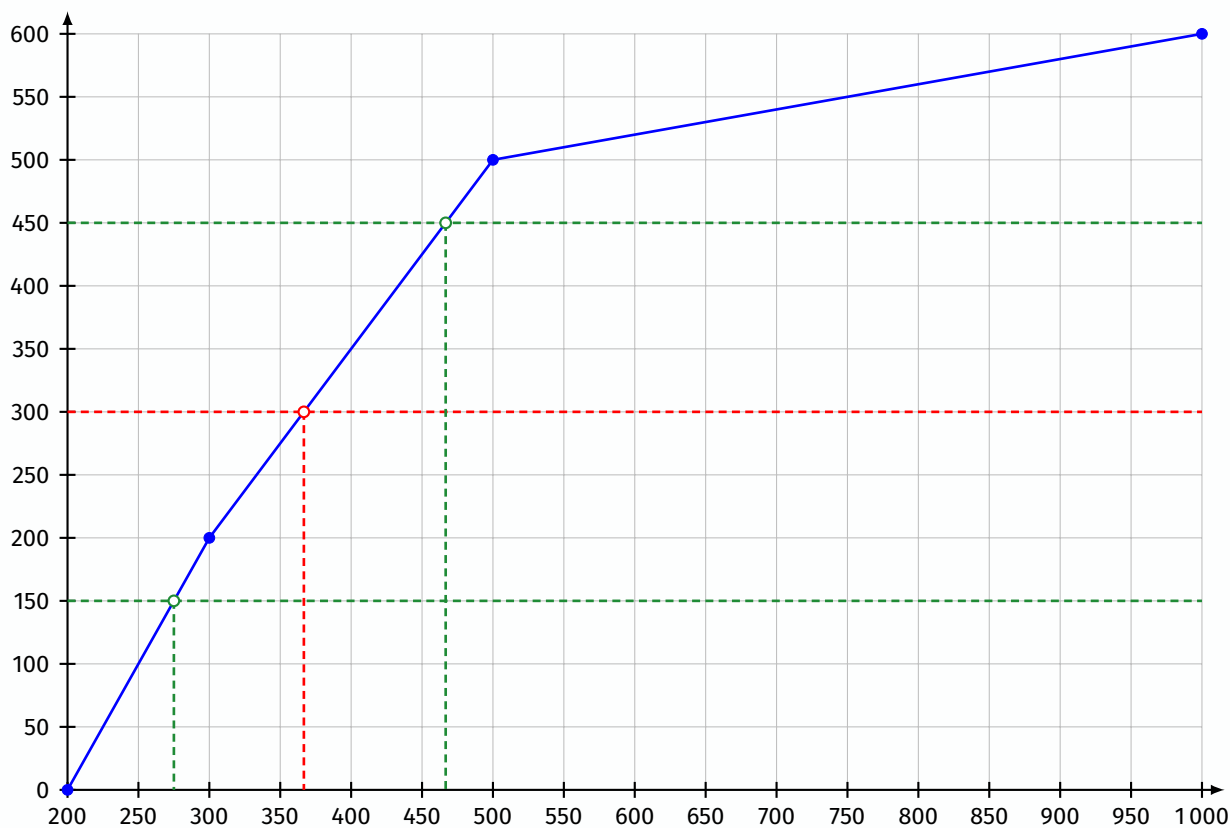
Code $\LaTeX$

```
%valeurs brutes, non arrondies, stockées dans les macros ci-dessous
\MedianeQuartilesECC{200,300,500,1000}{200,300,100}
$Q_1 = \ValPremQuartile$\par
$\text{Méd} = \ValMed$\par
$Q_3 = \ValTroisQuartile$\par
```

```
Q1 = 275
Méd = 366.6666666666667
Q3 = 466.6666666666667
```

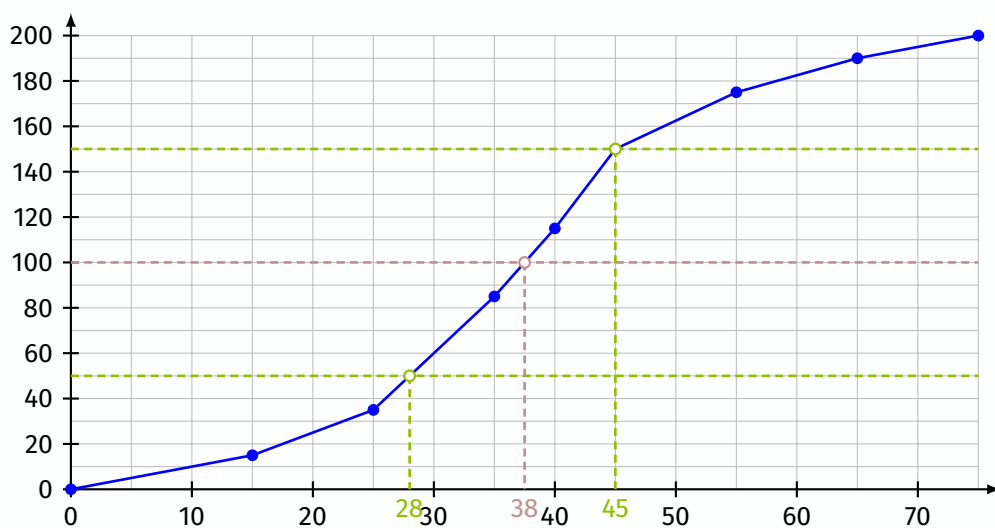
Code $\LaTeX$

```
\CourbeECC%
[GraduationsX={200,250,...,1000},GraduationsY={0,50,...,600},PoliceAxes=\small]%
{200,300,500,1000}%borne des classes
{200,300,100}%effectifs
```



Code $\LaTeX$

```
\begin{EnvCourbeECC}%
[Hauteur=6,Largeur=12,PasX=5,PasY=10,GraduationsX={0,10,...,75},%
GraduationsY={0,20,...,200},CouleursParams={lime!75!black/pink!75!black}]%
{0,15,25,35,40,45,55,65,75}%bornes des classes
{15,20,50,30,35,25,15,10}%effectifs
\draw[lime!75!black] (\ValPremQuartile,0) node[below] {\Arrondi[0]{\ValPremQuartile}} ;
\draw[pink!75!black] (\ValMed,0) node[below] {\Arrondi[0]{\ValMed}} ;
\draw[lime!75!black] (\ValTroisQuartile,0) node[below] {\Arrondi[0]{\ValTroisQuartile}} ;
\end{EnvCourbeECC}
```



42.2 Clés et options



Les clés et arguments ont un fonctionnement identique pour la commande ou pour l'environnement. Le premier argument, optionnel et entre `[...]` propose les **clés** principales suivantes :

- **⟨Largeur⟩** : largeur du graphique (en cm, et sans les graduations); défaut : **⟨15⟩**
- **⟨Hauteur⟩** : hauteur du graphique (en cm, et sans les graduations); défaut : **⟨10⟩**
- **⟨PasX⟩** : pas pour la grille horizontale; défaut : **⟨50⟩**
- **⟨PasY⟩** : pas pour la grille verticale; défaut : **⟨50⟩**
- **⟨Couleur⟩** : couleur de la ligne brisée; défaut : **⟨blue⟩**
- **⟨AffParams⟩** : booléen pour les traits de construction; défaut : **⟨true⟩**
- **⟨CouleursParams⟩** : couleur(s) des paramètres; défaut : **⟨CouleurVertForet/red⟩**
- **⟨GraduationsX⟩** : liste des graduations pour l'axe des abscisses;
- **⟨GraduationsY⟩** : liste des graduations pour l'axe des ordonnées;
- **⟨PoliceAxes⟩** : police des valeurs sur les axes; défaut : **⟨\normalsize\normalfont⟩**
- **⟨ElargirOx⟩** et **⟨ElargirOy⟩** : booléens pour agrandir un peu les axes; défaut : **⟨true⟩**
- **⟨ElargirAxes⟩** : longueur d'agrandissement des axes. défaut : **⟨3mm⟩**

Le deuxième argument, obligatoire et entre `{...}` permet de préciser les bornes des classes (sans répétition...)

Le dernier argument, obligatoire et entre `{...}` permet de préciser les effectifs des classes.

Le dernier argument comprend donc un élément de moins que le deuxième!

42.3 Styles et exemples



Le style de certains éléments sont fixés, mais ils peuvent être modifiés grâce à :

- `\traitsparamecc` pour les traits de construction;
- `\courbeecc` pour la ligne brisée;
- `\gradsecc` pour l'épaisseur des graduations;
- `\axececc` pour le tracé des axes.

La macro `\CourbeECCStylesDefault` permet de réinitialiser les valeurs par défaut si modification.

Code $\text{\LaTeX}$

```
\tikzset{traitsparamecc/.style={line width=1pt,densely dashed}}
\tikzset{courbeecc/.style={line width=1pt}}
\tikzset{gradsecc/.style={thick}}
\tikzset{axececc/.style={gradsecc,->,>=latex}}
```



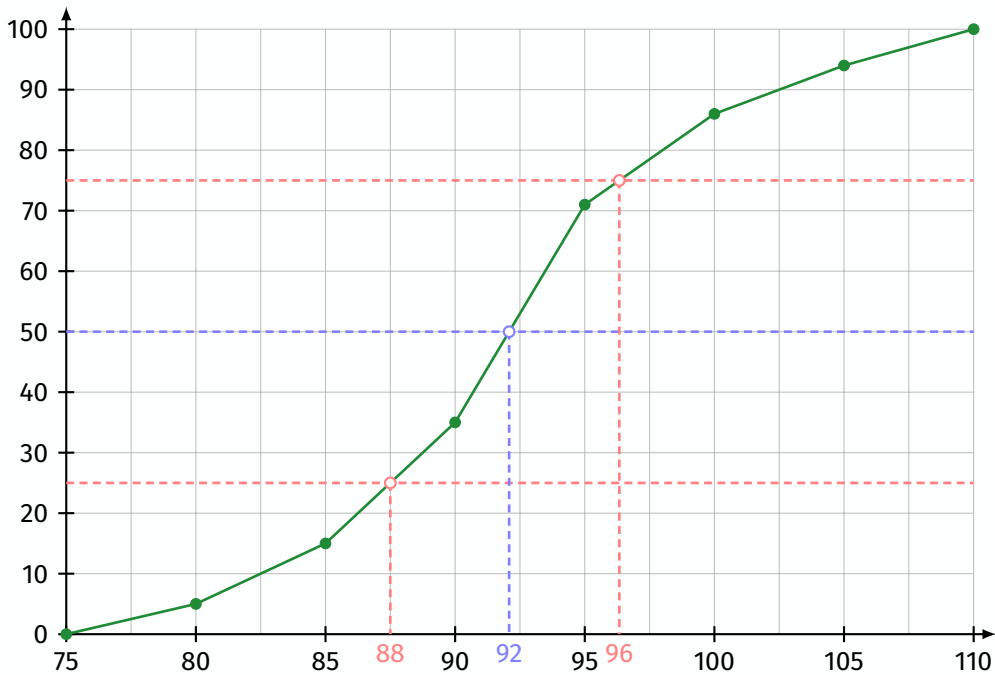
On va utiliser les données suivantes :

Classe	[75 ; 80[	[80 ; 85[	[85 ; 90[	[90 ; 95[	[95 ; 100[	[100 ; 105[	[105 ; 110[
Fréquence	0,05	0,10	0,20	0,36	0,15	0,08	0,06

Code $\text{\LaTeX}$

```
\begin{EnvCourbeECC}%
[Hauteur=8,Largeur=12,PasX=2.5,PasY=10,GraduationsX={75,80,...,110},%
GraduationsY={0,10,...,100},CouleursParams={red!50/blue!50},%
Couleur=CouleurVertForêt]%
{75,80,85,90,95,100,105,110}%bornes des classes
{5,10,20,36,15,8,6}%fréquences, en %
\draw[red!50] (\ValPremQuartile,0) node[below] {\Arrondi[0]{\ValPremQuartile}} ;
\draw[blue!50] (\ValMed,0) node[below] {\Arrondi[0]{\ValMed}} ;
\draw[red!50] (\ValTroisQuartile,0) node[below] {\Arrondi[0]{\ValTroisQuartile}} ;
\draw ($ (75,100)+(0,3mm)$) node[above] {FCC, en \%} ;
\end{EnvCourbeECC}
```

FCC, en %



Thème

OUTILS POUR LES PROBABILITÉS

Dixième partie

Outils pour les probabilités

43 Factorielle

43.1 Idée



3.04a L'idée est de proposer de quoi travailler avec les factorielles.
Les calculs sont effectués par le package `xint`, et le formatage peut être effectué par le package `siunitx`.

Code $\text{\LaTeX}$

```
\Factorielle*[clés]{nb}
```

Code $\text{\LaTeX}$

```
\Factorielle{9}\  
\Factorielle{17}
```

362 880

355 687 428 096 000

43.2 Clés



La version *étoilée* ne formate pas les résultats par `siunitx`.

Les **clés** disponibles pour cette commande sont :

- **Enonce** : booléen pour afficher l'énoncé; défaut **false**
- **Partiel** : booléen pour afficher la formule partielle; défaut **false**
- **Complet** : booléen pour afficher la formule complète; défaut **false**
- **Grand** : booléen pour les grands résultats; défaut **false**
- **Sens** : préciser l'ordre pour les détails **d/m**; défaut **m**
- **ChSignif** : nombre de chiffres significatifs pour les grands résultats. défaut **9**
- **Espace** : espace avant le « ! »; défaut `\mkern1.5mu\relax1.5`

Code $\text{\LaTeX}$

```
\Factorielle[Enonce,Complet]{9}
```

$9! = 1 \times 2 \times 3 \times 4 \times 5 \times 6 \times 7 \times 8 \times 9 = 362\,880$

Code $\text{\LaTeX}$

```
\Factorielle[Enonce,Complet,Sens=d]{9}
```

$9! = 9 \times 8 \times 7 \times 6 \times 5 \times 4 \times 3 \times 2 \times 1 = 362\,880$

Code $\text{\LaTeX}$

```
\Factorielle[Enonce,Partiel]{9}
```

$9! = 1 \times 2 \times \dots \times 8 \times 9 = 362\,880$

Code $\LaTeX$

```
\Factorielle[Enonce,Partiel,Sens=d]{9}
```

$9! = 9 \times 8 \times \dots \times 2 \times 1 = 362\,880$

Code $\LaTeX$

```
\Factorielle[Grand,Enonce,Partiel]{147}
```

$147! = 1 \times 2 \times \dots \times 146 \times 147 \approx 1,727\,245\,89 \times 10^{256}$

44 Calculs de probabilités

44.1 Introduction



L'idée est de proposer des commandes permettant de calculer des probabilités avec des lois classiques :

- binomiale;
- normale;
- exponentielle;
- de Poisson;
- géométrique;
- hypergéométrique.



Les commandes sont de deux natures :

- des commandes pour calculer, grâce au package `xintexpr`;
- des commandes pour formater le résultat de `xintexpr`, grâce à `siunitx`.

De ce fait, les options de `siunitx` de l'utilisateur affecteront les formatages du résultat, la commande va « forcer » les arrondis et l'écriture scientifique.

44.2 Calculs « simples »

Code $\LaTeX$

```
%loi binomiale B(n,p)
\CalcBinomP{n}{p}{k}          %P(X=k)
\CalcBinomC{n}{p}{a}{b}      %P(a<=X<=b)

%loi de Poisson P(l)
\CalcPoissP{l}{k}            %P(X=k)
\CalcPoissC{l}{a}{b}         %P(a<=X<=b)

%loi géométrique G(p)
\CalcGeomP{p}{k}             %P(X=k)
\CalcGeomC{l}{a}{b}          %P(a<=X<=b)

%loi hypergéométrique H(N,n,m)
\CalcHypergeomP{N}{n}{m}{k}  %P(X=k)
\CalcHypergeomC{N}{n}{m}{a}{b} %P(a<=X<=b)

%loi normale N(m,s)
\CalcNormC{m}{s}{a}{b}       %P(a<=X<=b)

%loi exponentielle E(l)
\CalcExpoC{l}{a}{b}          %P(a<=X<=b)
```



Les probabilités calculables sont donc – comme pour beaucoup de modèles de calculatrices – les probabilités **P**onctuelles ($P(X = k)$) et **C**umulées ($P(a \leq X \leq b)$).

Pour les probabilités cumulées, on peut utiliser le caractère `*` comme borne (a ou b), pour les probabilités du type $P(X \leq b)$ et $P(X \geq a)$.

Code $\LaTeX$

```
% X -> B(5,0.4)
$P(X=3) \approx \backslash\text{CalcBinomP}\{5\}\{0.4\}\{3\}$.
$P(X\leqslant 1) \approx \backslash\text{CalcBinomC}\{5\}\{0.4\}\{*\}\{1\}$.

% X -> B(100,0.02)
$P(X=10) \approx \backslash\text{CalcBinomP}\{100\}\{0.02\}\{10\}$.
$P(15\leqslant X\leqslant 25) \approx \backslash\text{CalcBinomC}\{100\}\{0.02\}\{15\}\{25\}$.

% Y -> P(5)
$P(Y=3) \approx \backslash\text{CalcPoissP}\{5\}\{3\}$.
$P(Y\geqslant 2) \approx \backslash\text{CalcPoissC}\{5\}\{2\}\{*\}$.

% T -> G(0.5)
$P(T=100) \approx \backslash\text{CalcPoissP}\{0.5\}\{3\}$.
$P(T\leqslant 5) \approx \backslash\text{CalcPoissC}\{0.5\}\{*\}\{5\}$.

% W -> H(50,10,5)
$P(W=4) \approx \backslash\text{CalcHypergeomP}\{50\}\{10\}\{5\}\{4\}$.
$P(1\leqslant W\leqslant 3) \approx \backslash\text{CalcHypergeomC}\{50\}\{10\}\{5\}\{1\}\{3\}$.
```

Code $\LaTeX$

- $X \hookrightarrow B(5 ; 0, 4)$:
 $P(X = 3) \approx 0.2304$.
 $P(X \leqslant 1) \approx 0.33696$.
- $X \hookrightarrow B(100 ; 0, 02)$:
 $P(X = 10) \approx 0.00002877077765846743$.
 $P(15 \leqslant X \leqslant 25) \approx 0.000000001670210428685021$.
- $Y \hookrightarrow P_5$:
 $P(Y = 3) \approx 0.1403738958142806$.
 $P(Y \geqslant 2) \approx 0.9595723180054873$.
- $T \hookrightarrow G_{0,5}$:
 $P(T = 3) \approx 0.125$.
 $P(T \leqslant 5) \approx 0.96875$.
- $W \hookrightarrow H(50 ; 10 ; 5)$:
 $P(W = 4) \approx 0.003964583058015065$.
 $P(1 \leqslant W \leqslant 3) \approx 0.6853536974456758$.

Code $\LaTeX$

```
% X -> N(0,1)
$P(X\leqslant 1) \approx \backslash\text{CalcNormC}\{0\}\{1\}\{*\}\{1\}$.
$P(-1,96\leqslant Z\leqslant 1,96) \approx \backslash\text{CalcNormC}\{0\}\{1\}\{-1.96\}\{1.96\}$.

% X -> N(550,30)
$P(Y\geqslant 600) \approx \backslash\text{CalcNormC}\{550\}\{30\}\{600\}\{*\}$.
$P(500\leqslant Y\leqslant 600) \approx \backslash\text{CalcNormC}\{550\}\{30\}\{500\}\{600\}$.

% Z -> E(0.001)
$P(Z\geqslant 400) \approx \backslash\text{CalcExpoC}\{0.001\}\{400\}\{*\}$.
$P(300\leqslant Z\leqslant 750) \approx \backslash\text{CalcExpoC}\{0.001\}\{300\}\{750\}$.
```

Code $\LaTeX$

- $X \hookrightarrow N(0 ; 1)$:
 $P(X \leqslant 1) \approx 0.841344680841397$.
 $P(-1,96 \leqslant Z \leqslant 1,96) \approx 0.9500039553976748$.
- $Y \hookrightarrow N(550 ; 30)$:
 $P(Y \geqslant 600) \approx 0.0477903462453939$.
 $P(500 \leqslant Y \leqslant 600) \approx 0.9044193075092122$.
- $Z \hookrightarrow E_{0,001}$:
 $P(Z \geqslant 400) \approx 0.6703200460356393$.
 $P(300 \leqslant Z \leqslant 750) \approx 0.2684516679407032$.

44.3 Complément avec sortie « formatée »



L'idée est ensuite de formater le résultat obtenu par `\xintexpr`, pour un affichage homogène. L'utilisateur peut donc utiliser « sa » méthode pour formater les résultats obtenus par `\xintexpr` !

Code $\text{\LaTeX}$

```
%avec un formatage manuel
\num[exponent-mode=scientific]{\CalcBinomP{100}{0.02}{10}}
```

Code $\text{\LaTeX}$

```
$\bullet\leadsto X \hookrightarrow \mathcal{B}(100\backslash, ; 0,02)$ :
$P(X=10) \approx \num[exponent-mode=scientific]{\CalcBinomP{100}{0.02}{10}}$.
```

- $X \hookrightarrow B(100 ; 0,02)$:
 $P(X = 10) \approx 2,877\,077\,765\,846\,743 \times 10^{-5}$.



Le package `\ProfLycee` propose – en complément – des commandes pour formater, grâce à `\siunitx`, le résultat.

Les commandes ne sont donc, dans ce cas, pas préfixées par `\calc` :

- formatage sous forme décimale *pure* : 0,00 ... ;
- formatage sous forme scientifique : $n, \dots \times 10^{\dots}$.

Code $\text{\LaTeX}$

```
%loi binomiale B(n,p)
\BinomP(*)[prec]{n}{p}{k}          %P(X=k)
\BinomC(*)[prec]{n}{p}{a}{b}       %P(a<=X<=b)

%loi de Poisson P (l)
\PoissonP(*)[prec]{l}{k}           %P(X=k)
\PoissonC(*)[prec]{l}{a}{b}        %P(a<=X<=b)

%loi géométrique G (p)
\GeomP{p}{k}                       %P(X=k)
\GeomC{l}{a}{b}                    %P(a<=X<=b)

%loi hypergéométrique H (N,n,m)
\HypergeomP{N}{n}{m}{k}            %P(X=k)
\HypergeomC{N}{n}{m}{a}{b}         %P(a<=X<=b)

%loi normale N(m,s)
\NormaleC(*)[prec]{m}{s}{a}{b}      %P(a<=X<=b)

%loi exponentielle E(l)
\ExpoC(*)[prec]{l}{a}{b}           %P(a<=X<=b)
```



Quelques précisions sur les commandes précédentes :

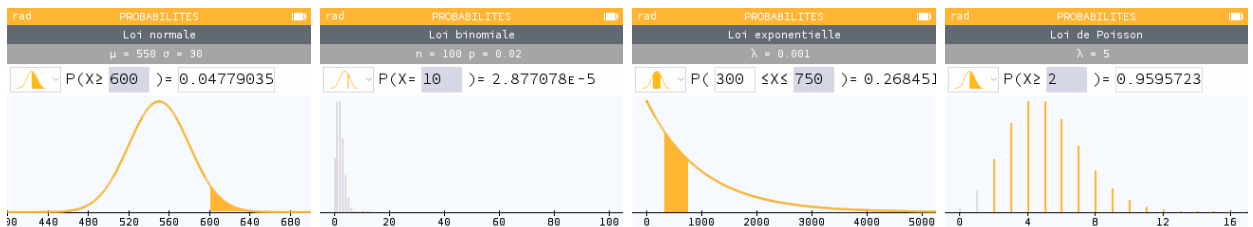
- la version étoilée `*` des commandes formate le résultat en mode scientifique ;
- l'argument optionnel (par défaut `3`) correspond à quant à lui à l'arrondi.

Code $\LaTeX$

```
% X -> N(550,30)
$P(Y\geqslant 600) \approx \backslash\mathrm{NormaleC}[4]{550}{30}{600}{*}$.
$P(500\leqslant Y\leqslant 600) \approx \backslash\mathrm{NormaleC}[4]{550}{30}{500}{600}$.
% X -> B(100,0.02)
$P(X=10) \approx \backslash\mathrm{BinomP}[7]{100}{0.02}{10} \approx \backslash\mathrm{BinomP*}[7]{100}{0.02}{10}$.
$P(15\leqslant X\leqslant 25) \approx \backslash\mathrm{BinomC}[10]{100}{0.02}{15}{25} \approx \backslash\mathrm{BinomC*}[10]{100}{0.02}{15}{25}$.
% H -> H(50,10,5)
$P(W=4) \approx \backslash\mathrm{HypergeomP}[5]{50}{10}{5}{4}$.
$P(1\leqslant W\leqslant 3) \approx \backslash\mathrm{HypergeomC}[4]{50}{10}{5}{1}{3}$.
% Z -> E(0,001)$ :
$P(Z\geqslant 400) \approx \backslash\mathrm{ExpoC}[0.001]{400}{*}$.
$P(300\leqslant Z\leqslant 750) \approx \backslash\mathrm{ExpoC}[0.001]{300}{750}$.
% T -> P(5)
$P(T=3) \approx \backslash\mathrm{PoissonP}[5]{3}$.
$P(T\geqslant 2) \approx \backslash\mathrm{PoissonC}[4]{5}{2}{*}$.
```

Code $\LaTeX$

- $Y \hookrightarrow N(550 ; 30)$:
 $P(Y \geq 600) \approx 0,0478$.
 $P(500 \leq Y \leq 600) \approx 0,9044$.
- $X \hookrightarrow B(100 ; 0,02)$:
 $P(X = 10) \approx 0,0000288 \approx 2,88 \times 10^{-5}$.
 $P(15 \leq X \leq 25) \approx 0,000000017 \approx 1,7 \times 10^{-9}$.
- $W \hookrightarrow H(50 ; 10 ; 5)$:
 $P(W = 4) \approx 0,00396$.
 $P(1 \leq W \leq 3) \approx 0,6854$.
- $Z \hookrightarrow E_{0,001}$:
 $P(Z \geq 400) \approx 0,670$.
 $P(300 \leq Z \leq 750) \approx 0,268$.
- $T \hookrightarrow P_5$:
 $P(T = 3) \approx 0,140$.
 $P(T \geq 2) \approx 0,9596$.



45 Arbres de probabilités « classiques »

45.1 Introduction



L'idée est de proposer des commandes pour créer des arbres de probabilités classiques (et homogènes), en TikZ, de format :

- 2×2 ou 2×3 ;
- 3×2 ou 3×3 .

Les (deux) commandes sont donc liées à un environnement `\tikzpicture`, et elles créent les nœuds de l'arbre, pour exploitation ultérieure éventuelle.

Code L^AT_EX

```
%commande simple pour tracé de l'arbre
\ArbreProbasTikz[options]{donnees}

%environnement pour tracé et exploitation éventuelle
\begin{EnvArbreProbasTikz}[options]{donnees}
  code tikz supplémentaire
\end{EnvArbreProbasTikz}
```

45.2 Options et arguments



Les **⟨donnees⟩** seront à préciser sous forme

`<sommet1>/<proba1>/<position1>,<sommet2>/<proba2>/<position2>,...`

avec comme « sens de lecture » de la gauche vers la droite puis du haut vers le bas (on balaye les sous-arbres), avec comme possibilités :

- `2.5.3` une donnée **⟨proba⟩** peut être laissée vide ou spécifiée avec des macros;
- une donnée **⟨position⟩** peut valoir **⟨above⟩** (au-dessus), **⟨below⟩** (en-dessous) ou être laissée **⟨vide⟩** (sur);
- `3.02b` la clé **⟨PositionProbas⟩** (vide par défaut) peut permettre de positionner toutes les probas suivant :
 - **⟨auto⟩** := position automatique des probas;
 - **⟨dessus⟩** := position au-dessus des branches;
 - **⟨dessous⟩** := position au-dessous des branches;
 - **⟨sur⟩** := position sur les branches.



Quelques **⟨Clés⟩** (communes) pour les deux commandes :

- la clé **⟨Unite⟩** pour préciser l'unité de l'environnement TikZ; défaut **⟨1cm⟩**
- la clé **⟨EspaceNiveau⟩** pour l'espace (H) entre les étages; défaut **⟨3.25⟩**
- la clé **⟨EspaceFeuille⟩** pour l'espace (V) entre les feuilles; défaut **⟨1⟩**
- la clé **⟨Type⟩** pour le format, parmi **⟨2x2⟩** ou **⟨2x3⟩** ou **⟨3x2⟩** ou **⟨3x3⟩**; défaut **⟨2x2⟩**
- la clé **⟨Police⟩** pour la police des nœuds; défaut **⟨\normalfont\normalsize⟩**
- la clé **⟨PoliceProbas⟩** pour la police des probas; défaut **⟨\normalfont\small⟩**
- le booléen **⟨InclineProbas⟩** pour incliner les probas; défaut **⟨true⟩**
- la clé **⟨Racine⟩** pour afficher une racine; défaut **⟨vide⟩**
- le booléen **⟨Fleche⟩** pour afficher une flèche sur les branches; défaut **⟨false⟩**
- la clé **⟨StyleTrait⟩** pour les branches, en langage TikZ; défaut **⟨vide⟩**
- la clé **⟨EpaisseurTrait⟩** pour l'épaisseur des branches, en langage TikZ; défaut **⟨semithick⟩**

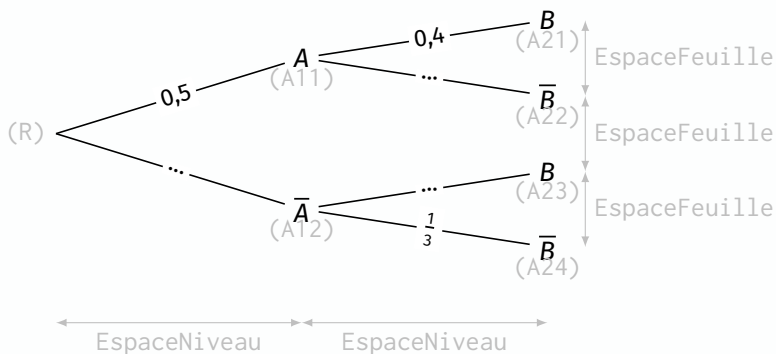
Code $\text{\LaTeX}$

```
\def\ArbreDeuxDeux{
  $A$/\num{0.5}/,
  $B$/\num{0.4}/,
  $\overline{B}$/\dots/,
  $\overline{A}$/\dots/,
  $B$/\dots/,
  $\overline{B}$/$\frac{1}{3}$$/
}
```

$\backslash\text{ArbreProbabTikz}\{\text{\ArbreDeuxDeux}\}$

%des éléments, en gris, ont été rajoutés pour illustrer certaines options

Code $\text{\LaTeX}$



Les nœuds créés par les commandes sont :

- $\text{\R}$ pour la racine;
- $\text{\A1x}$ pour les nœuds du 1^{er} niveau (de haut en bas);
- $\text{\A2x}$ pour les nœuds du 2^d niveau (de haut en bas).

45.3 Exemples complémentaires

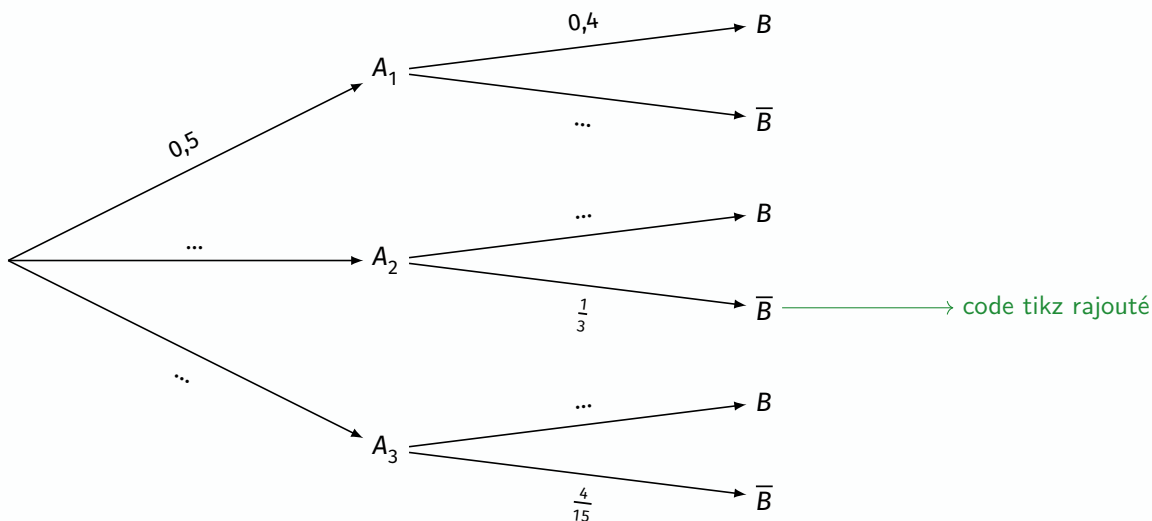


À noter qu'il existe les clés $\langle\text{CouleurTraits}\rangle$, $\langle\text{CouleurNoeuds}\rangle$ et $\langle\text{CouleurProbab}\rangle$ pour personnaliser un peu plus en profondeur l'arbre de probabilités.

Code $\text{\LaTeX}$

```
\def\ArbreTroisDeux{
  $A_1$/\num{0.5}/,
  $B$/\num{0.4}/,
  $\overline{B}$/\numdots/,
  $A_2$/\numdots/,
  $B$/\numdots/,
  $\overline{B}$/$\frac{1}{3}$$/,
  $A_3$/\numdots/,
  $B$/\numdots./,
  $\overline{B}$/$\frac{4}{15}$$/
}

\begin{EnvArbreProbabTikz}
  [Type=3x2,Fleche,EspaceNiveau=5,EspaceFeuille=1.25,PositionProbab=auto]%
  {\ArbreTroisDeux}
  \draw[CouleurVertForet,->] (A24)--($\langle\text{A24}\rangle+(2.5,0)$) node[right,font=\sffamily] {code tikz rajouté} ;
\end{EnvArbreProbabTikz}
```

Code $\LaTeX$ Code $\LaTeX$

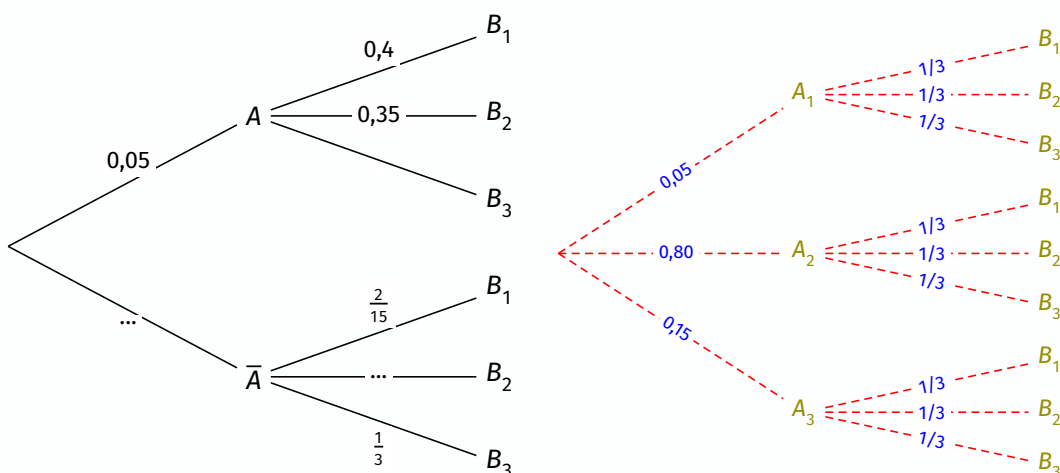
```

\def\ArbreDeuxTrois{
  $A$/\num{0.05}/above,
  $B_1$/\num{0.4}/above,$B_2$/\num{0.35}/,$B_3$/below,
  $\overline{A}$/.../below,
  $B_1$/\frac{2}{15}/above,$B_2$/.../,$B_3$/\frac{1}{3}/below
}
\ArbreProbasTikz[Type=2x3,InclineProbas=false,EspaceFeuille=1.15]{\ArbreDeuxTrois}

\def\ArbreTroisTrois{
  $A_1$/\num{0.05}/,$B_1$/\frac{1}{3}/,$B_2$/\frac{1}{3}/,$B_3$/\frac{1}{3}/,
  $A_2$/\num{0.80}/,$B_1$/\frac{1}{3}/,$B_2$/\frac{1}{3}/,$B_3$/\frac{1}{3}/,
  $A_3$/\num{0.15}/,$B_1$/\frac{1}{3}/,$B_2$/\frac{1}{3}/,$B_3$/\frac{1}{3}/
}

\ArbreProbasTikz%
[Type=3x3,StyleTrait={densely dashed},
CouleurTraits=red,CouleurProbas=blue,CouleurNoeuds=olive,
EspaceFeuille=0.7,PoliceProbas=\scriptsize,Police=\small]%
{\ArbreTroisTrois}

```

Code $\LaTeX$ 

45.4 Arbres de probabilités flexibles



L'idée (expérimentale) est de proposer des arbres de probabilités *de forme libre* (nombre de branches et de niveaux au choix), dessinés en TikZ, avec une description **récursive** : une branche porte une probabilité, un nom de nœud, et éventuellement ses branches filles.

Les étiquettes sont saisies *telles quelles* ($\dots$, $\frac{\dots}{\dots}$, $\text{\num{...}}$), c'est l'utilisateur qui garde la main sur le formatage.

Code $\LaTeX$

```
%environnement
\begin{tkzArbre}[clés]
  \tkzArbreArete[proba]{noeud}{branches filles}  %{branches filles} facultatif
\end{tkzArbre}

%macro (arbres courts)
\tkzArbre[clés]{\tkzArbreArete[proba]{noeud}{...} ...}

%réglages communs à un groupe
\tkzArbreConfig[clés]
```



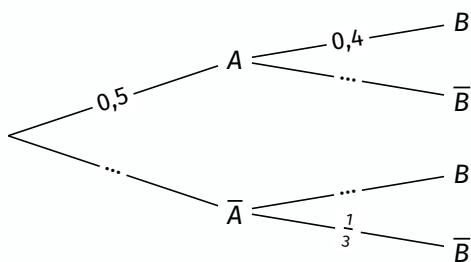
Quelques **<Clés>** principales de l'arbre :

- **<EspaceNiveaux>** : espace horizontal entre les niveaux (**<espace niveaux>**); défaut **<3cm>**
- **<EspaceFeuilles>** : espace vertical entre les feuilles (**<espace feuilles>**); défaut **<1cm>**
- **<PositionProbas>** : **<auto>**, **<dessus>**, **<dessous>** ou **<sur>** (**<position probas>**); défaut **<auto>**
- **<Incline>** : probas inclinées le long des branches (**<incline>**); défaut **<true>**
- **<Fleches>** : pointes de flèche sur les branches (**<fleches>**); défaut **<false>**
- **<Racine>** : texte du nœud racine (**<racine>**); défaut **<vide>**
- **<Resultats>** : produits le long des chemins : **<num>** ou **<prod>** (**<resultats>**); défaut **<false>**
- **<Direction>** : **<droite>** ou **<bas>**; défaut **<droite>**

Pour une branche, entre `[...]` : **<proba>**, **<nom>** (nom TikZ du nœud), **<style>**, **<arete>**, **<position>**, **<sur-brillance>** (ou `[*]` après la commande pour mettre un chemin en évidence).

Code $\LaTeX$

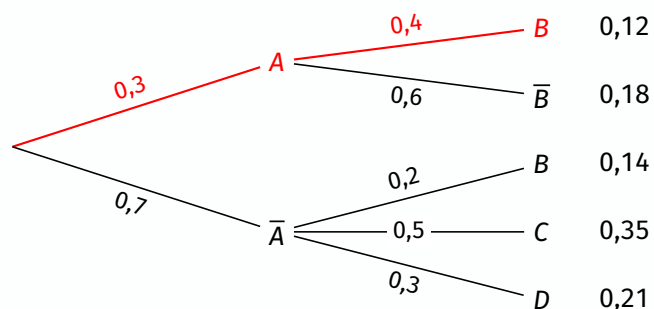
```
\begin{tkzArbre}[position probas=sur,espace niveaux=3cm,espace feuilles=1cm]
  \tkzArbreArete[$0{,}5$]{A$}{\overline{B}$}
  \tkzArbreArete[$0{,}4$]{A$}{B$}
  \tkzArbreArete[$\ldots$]{A$}{\overline{B}$}
}
\tkzArbreArete[$\ldots$]{\overline{A}$}{\overline{B}$}
\tkzArbreArete[$\ldots$]{\overline{A}$}{B$}
\tkzArbreArete[$\frac{1}{3}$]{\overline{A}$}{\overline{B}$}
}
\end{tkzArbre}
```



```

\begin{tkzArbre}[resultats,espace niveaux=3.5cm,espace feuilles=0.9cm]
  \tkzArbreArete*[$0{,}3$]{$A$}{
    \tkzArbreArete*[$0{,}4$]{$B$}
    \tkzArbreArete[$0{,}6$]{$\overline{B}$}
  }
  \tkzArbreArete[$0{,}7$]{$\overline{A}$}{
    \tkzArbreArete[$0{,}2$]{$B$}
    \tkzArbreArete[$0{,}5$]{$C$}
    \tkzArbreArete[$0{,}3$]{$D$}
  }
\end{tkzArbre}

```



Les styles se surchargent avec `\tikzset` (par exemple `tkzArbreProbas arete`, `tkzArbreProbas proba`, `tkzArbreProbas surbrillance arete`), et les nœuds sont nommés `prefixe-chemin` (par défaut `tkzPT-2-1`) pour une exploitation TikZ ultérieure.

46 Petits schémas pour des probabilités continues

46.1 Idée



L'idée est de proposer des commandes pour illustrer, sous forme de schémas en TikZ, des probabilités avec des lois continues (normales et exponentielles).

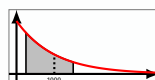
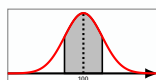
Ces « schémas » peuvent être insérés en tant que graphique explicatif, ou bien en tant que petite illustration rapide!

Code $\LaTeX$

```
\LoiNormaleGraphe[options]<options tikz>\{m\}\{s\}\{a\}\{b\}
```

```
\LoiExpoGraphe[options]<options tikz>\{l\}\{a\}\{b\}
```

Code $\LaTeX$



Les probabilités *illustrables* sont donc des probabilités **Cumulées** ($P(a \leq X \leq b)$).

On peut utiliser $\LaTeX$ `*` comme borne (a ou b), pour les probabilités du type $P(X \leq b)$ et $P(X \geq a)$.

46.2 Commandes et options



Quelques **Clés** sont disponibles pour ces commandes :

- | | |
|---|-------------------------|
| – la clé CouleurAire pour l'aire sous la courbe; | défaut LightGray |
| – la clé CouleurCourbe pour la courbe; | défaut red |
| – la clé Largeur qui sera la largeur (en cm) du graphique; | défaut 2 |
| – la clé Hauteur qui sera la hauteur (en cm) du graphique; | défaut 1 |
| – un booléen AfficheM qui affiche la moyenne; | défaut true |
| – un booléen AfficheCadre qui affiche un cadre pour délimiter le schéma. | défaut true |



Les commandes sont donc des environnements TikZ, sans possibilité de « rajouter » des éléments. Ces petits schémas sont donc vraiment dédiés à *montrer* rapidement une probabilité continue, sans fioriture.

Code $\LaTeX$

Avec centrage vertical sur l'axe des abscisses :

```
\LoiNormaleGraphe  
[AfficheM=false,CouleurCourbe=blue,CouleurAire=cyan]<baseline=0pt>%  
{1000}{100}{950}{*}
```

Avec centrage vertical sur l'axe des abscisses :



Avec quelques modifications :

```
\LoiNormaleGraphe[Largeur=4,Hauteur=2]{150}{12.5}{122}{160}
```

```
\medskip
```

Avec centrage vertical :

```
\LoiNormaleGraphe[Largeur=5,Hauteur=2.5]<baseline=(current bounding box.center)>{200}{5}{204}{*}
```

```
\medskip
```

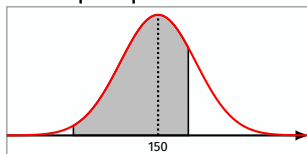
Avec centrage vertical sur l'axe des abscisses :

```
\LoiExpoGraphe  
[AfficheM=false,CouleurCourbe=blue,CouleurAire=cyan]<baseline=0pt>{0.05}{*}{32}
```

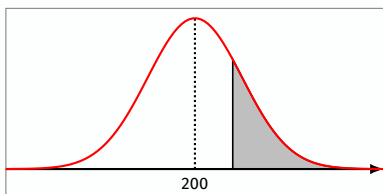
```
\medskip
```

```
\LoiExpoGraphe[Largeur=4,Hauteur=2]{0.00025}{5000}{*}
```

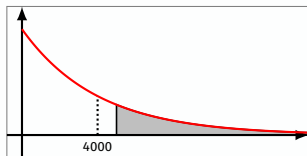
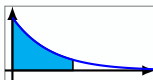
Avec quelques modifications :



Avec centrage vertical :



Avec centrage vertical sur l'axe des abscisses :



46.3 Remarques et compléments



Pour le moment, seules les lois (continues) exponentielles et normales sont disponibles, peut-être que d'autres lois seront ajoutées, mais il ne me semble pas très pertinent de proposer des schémas similaires pour des lois discrètes, qui ont des *représentations* assez variables...

47 Nombres aléatoires

47.1 Idée



2.0.9 L'idée est de proposer des commandes pour générer des nombres aléatoires, pour exploitation ultérieure :

- un entier ou un nombre décimal;
- des nombres entiers, avec ou sans répétitions.



Pour chacune des commandes, le ou les résultats sont stockés dans une macro dont le nom est choisi par l'utilisateur.

Code $\text{\LaTeX}$

```
%entier aléatoire entre a et b
\NbAlea{a}{b}{macro}

%nombre décimal (n chiffres après la virgule) aléatoire entre a et b+1 (exclus)
\NbAlea[n]{a}{b}{macro}

%création d'un nombre aléatoire sous forme d'une macro
\VarNbAlea{macro}{calculs}

%liste d'entiers aléatoires
\TirageAleatoireEntiers[options]{macro}
```

Code $\text{\LaTeX}$

```
%nombre aléatoire entre 1 et 50, stocké dans \PremierNbAlea
Entier entre 1 et 50 : \NbAlea{1}{50}{\PremierNbAlea}\PremierNbAlea \\
%nombre aléatoire créé à partir du 1er, stocké dans \DeuxiemeNbAlea
Entier à partir du précédent : \VarNbAlea{\DeuxiemeNbAlea}{\PremierNbAlea+randint(0,10)}\DeuxiemeNbAlea \\
%nombre aléatoire décimal (au millième) entre 0 et 10+1 (exclus), stocké dans \PremierDecAlea
Décimal entre 0 et $10,999\ldots$ : \NbAlea[3]{0}{10}{\PremierDecAlea}\PremierDecAlea \\
%liste de 6 nombres, sans répétitions, entre 1 et 50
Liste par défaut (6 entre 1 et 50) : \TirageAleatoireEntiers{\PremiereListeAlea}\PremiereListeAlea
```

Entier entre 1 et 50 : 3
Entier à partir du précédent : 13
Décimal entre 0 et 10,999... : 6.326
Liste par défaut (6 entre 1 et 50) : 19,21,34,40,46,2



Les listes créées sont exploitables, *a posteriori*, par le package `listofitems` par exemple!

Code $\text{\LaTeX}$

```
Liste générée : \TirageAleatoireEntiers{\TestListeA}\TestListeA
Liste traitée : \readlist*\LISTEa{\TestListeA}\showitems{\LISTEa}
```

Liste générée : 35,23,4,3,38,12
Liste traitée :

35	23	4	3	38	12
----	----	---	---	----	----

47.2 Clés et options



Quelques clés sont disponibles pour la commande `\TirageAleatoireEntiers` :

- la clé `<ValMin>` pour préciser borne inférieure de l'intervalle; défaut `<1>`
- la clé `<ValMax>` pour préciser borne supérieure de l'intervalle; défaut `<50>`
- la clé `<NbVal>` qui est le nombre d'entiers à générer; défaut `<6>`
- la clé `<Sep>` pour spécifier le séparateur d'éléments; défaut `<,>`
- la clé `<Tri>` parmi `<non/<croissant/<decroissant>` pour trier les valeurs; défaut `<non>`
- le booléen `<Repetition>` pour autoriser la répétition d'éléments. défaut `<false>`

Code $\text{\LaTeX}$

Une liste de 15 valeurs (différentes), entre 10 et 100, stockée dans la macro MaListeA : `\`
Liste : `\TirageAleatoireEntiers[ValMin=10,ValMax=100,NbVal=15]{\MaListeA}\MaListeA \`

Une liste de 12 valeurs (différentes), entre 1 et 50, ordre croissant : `\`
Liste : `\TirageAleatoireEntiers[ValMin=1,ValMax=50,NbVal=12,Tri=croissant]{\MaListeB}\MaListeB \`

Une liste de 12 valeurs (différentes), entre 1 et 50, ordre décroissant : `\`
Liste : `\TirageAleatoireEntiers[ValMin=1,ValMax=50,NbVal=12,Tri=decroissant]{\MaListeC}\MaListeC \`

15 tirages de dé à 6 faces : `\`
↪ `\TirageAleatoireEntiers[ValMin=1,ValMax=6,NbVal=15,Repetition]{\TestDes}\TestDes`

Une liste de 15 valeurs (différentes), entre 10 et 100, stockée dans la macro MaListeA :
Liste : 26,99,93,88,40,64,24,43,59,23,41,66,14,71,38

Une liste de 12 valeurs (différentes), entre 1 et 50, ordre croissant :
Liste : 4,7,10,13,23,24,26,27,32,35,45,48

Une liste de 12 valeurs (différentes), entre 1 et 50, ordre décroissant :
Liste : 49,47,46,32,31,30,29,24,13,11,10,3

15 tirages de dé à 6 faces :
2,2,6,3,3,6,6,5,2,6,3,3,3,2,2

Code $\text{\LaTeX}$

Une liste (10) pour le Keno $\text{\textcopyright}$, ordonnée, et séparée par des `\texttt{-}` :
`\TirageAleatoireEntiers[ValMin=1,ValMax=70,NbVal=10,Tri=croissant,Sep={-}]{\ListeKeno}`
`$\ListeKeno$`

`\setsepchar{-}\readlist*\KENO{\ListeKeno}\showitems{\KENO}`

Une liste (10) pour le Keno $\text{\textcopyright}$, ordonnée, et séparée par des `-` :
10 - 24 - 31 - 35 - 39 - 42 - 44 - 48 - 57 - 63

10	24	31	35	39	42	44	48	57	63
----	----	----	----	----	----	----	----	----	----

48 Combinatoire

48.1 Idée



L'idée est de proposer une commande pour calculer un arrangement ou une combinaison, en utilisant les capacités de calcul du package `\xint` (2.5.4).

Code $\LaTeX$

```
\Arrangement(*)[option]{p}{n}
\Combinaison(*)[option]{p}{n}
\CalculAnp{p}{n} ou \CalculCnp{p}{n} dans un calcul via \xinteval{...}
```

48.2 Utilisation



Peu de paramétrage pour ces commandes qui permettent de calculer A_n^p et $\binom{n}{p}$:

- les versions étoilées ne formatent pas le résultat grâce à `\num` de `\sinuitx`;
- le booléen `\Notation` pour avoir la notation au début; défaut `\false`
- le booléen `\NotationAncien` pour avoir la notation « ancienne » des combinaisons au début; défaut `\false`
- le booléen `\Formule` permet de présenter la formule avant le résultat; défaut `\false`
- le premier argument, *obligatoire*, est la valeur de p ;
- le second argument, *obligatoire*, est la valeur de n .

Code $\LaTeX$

On a $A_{20}^3 = \text{\texttt{\textbackslash Arrangement}\{3\}\{20\}}$ en non formaté,
et $\text{\texttt{\textbackslash Arrangement}\text{\texttt{[Notation]}\{3\}\{20\}}}$ en formaté avec la notation au début.

On a $A_{20}^3 = 6840$ en non formaté, et $A_{20}^3 = 6\,840$ en formaté avec la notation au début.

Code $\LaTeX$

On a $\text{\texttt{\textbackslash displaystyle\binom}\{20\}\{3\}} = \text{\texttt{\textbackslash Combinaison}\{3\}\{20\}}$ en non formaté, ~
et $\text{\texttt{\textbackslash displaystyle\binom}\text{\texttt{[Notation]}\{3\}\{20\}}}$ en formaté avec la notation au début. \\
Et $\text{\texttt{\textbackslash dbinom}\{20\}\{3\}} + \text{\texttt{\textbackslash dbinom}\{20\}\{4\}} = \text{\texttt{\textbackslash num}\{ \text{\texttt{\textbackslash xinteval}\{ \text{\texttt{\textbackslash CalculCnp}\{3\}\{20\}} + \text{\texttt{\textbackslash CalculCnp}\{4\}\{20\}} \}}}$.

On a $\binom{20}{3} = 1140$ en non formaté, et $\binom{20}{3} = 1\,140$ en formaté avec la notation au début.

Et $\binom{20}{3} + \binom{20}{4} = 5\,985$.

Code $\LaTeX$

On a $\text{\texttt{\textbackslash displaystyle\Arrangement}\text{\texttt{[Notation,Formule]}\{3\}\{20\}}}$.

On a $A_{20}^3 = \frac{20!}{17!} = 6\,840$.

Code $\LaTeX$

On a $\text{\texttt{\textbackslash displaystyle\Combinaison}\text{\texttt{[NotationAncien,Formule]}\{3\}\{20\}}}$. %ancienne notation FR

On a $C_{20}^3 = \frac{20!}{3! \times 17!} = 1\,140$.

49 Arbres de dénombrement

49.1 Idée



 3.10c L'idée est de proposer des commandes pour représenter des arbres de choix, pour du dénombrement :

- tirages successifs avec remise ;
- tirages successifs sans remise (permutations).

La présentation globale de l'arbre est fixée, mais quelques éléments de personnalisations sont possibles.



Pour des raisons de *calculs internes*, les permutations sont limitées à des ensembles à entre 2 et 4 éléments.

Pour les choix successifs (avec répétitions), l'affichage des résultats sera limitée à 5 successions !



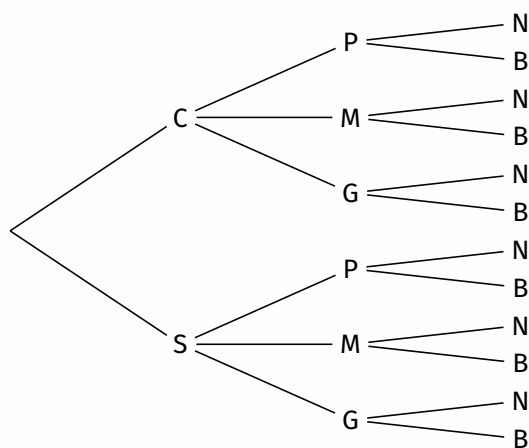
Une aprtie du code interne est issu d'un partage de code, de Mark Wibrow, sous licence CC BY-SA 3.0. [[lien tex.stackexchange](https://tex.stackexchange.com)].

Code $\LaTeX$

```
%arbre "avec remise"  
\ArbreChoix[clés]<options tikzpicture>{liste}  
  
%arbre "sans remise"  
\ArbreChoixSansRemise[clés]<options tikzpicture>{liste}
```

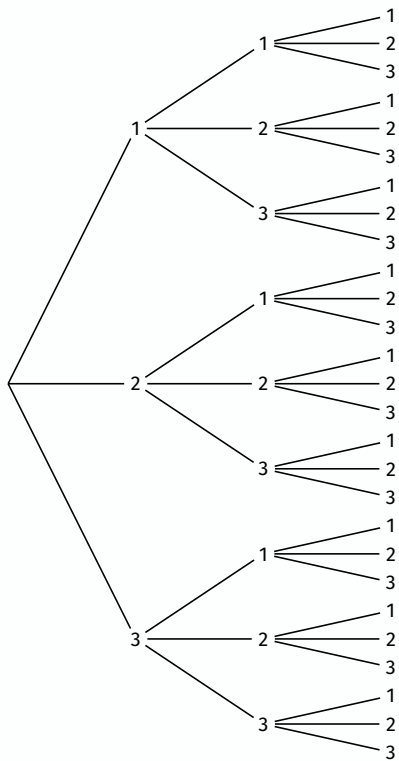
Code $\LaTeX$

```
\ArbreChoix{C/S,P/M/G,N/B}
```



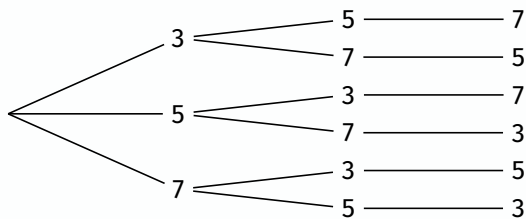
Code $\text{\LaTeX}$

```
\ArbreChoix[Repet=3,Echelle=0.75]{1/2/3}
```



Code $\text{\LaTeX}$

```
\ArbreChoixSansRemise{3,5,7}
```



49.2 Styles et clés



Chaque élément peut être personnalisé via un style, en langage `tikz`.
Les éléments sont donnés ci-après.

Code $\text{\LaTeX}$

```
\tikzset{arbrechoixaretes/.style={semithick}} %traits
\tikzset{arbrechoixsommets/.style={circle,draw=none,inner sep=1pt}} %nœuds
\tikzset{arbrechoixresultats/.style={rectangle,draw,inner sep=1.5pt}} %résultats
```



Le premier argument, optionnel et entre `[...]` propose les clés suivantes :

- la clé `<EspaceNiveaux>` pour l'espacement horizontal des niveaux défaut `<2.25>`
- la clé `<EspaceFeuilles>` pour l'espacement vertical du dernier niveau; défaut `<0.75>`
- la clé `<Echelle>` pour spécifier une échelle **globale**; défaut `<1>`
- la clé `<Repet>` pour préciser le nb de répétitions (pour une *mono-liste*); défaut `<vide>`
- la clé `<Notice>` pour préciser les notices des niveaux; défaut `<vide>`
- le booléen `<TraitsNotice>` afficher des traits de séparation des niveaux; défaut `<false>`
- la clé `<CouleursNiveaux>` pour préciser les éventuelles couleurs; défaut `<black>`
- le booléen `<AffResultats>` pour afficher les résultats (si possible!); défaut `<false>`
- la clé `<SepResultats>` pour préciser la séparation des éléments des résultats. défaut `<vide>`

L'argument optionnel et entre `<...>` permet de passer des options à l'environnement `tikzpicture`.

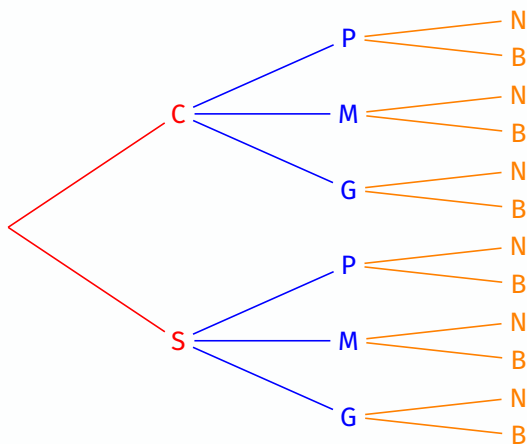
L'argument obligatoire et entre `{...}` permet de spécifier la liste des choix :

- sous la forme `N1a/N1b/... , N2a/N2b/... , ...` pour des successions *distinctes*;
- sous la forme `C1/C2/...` avec `<Repet=nb>` pour des successions *identiques*;
- sous la forme `e1,e2,...` pour des permutations.

49.3 Exemples

Code $\text{\LaTeX}$

```
%coloration
\ArbreChoix[CouleursNiveaux={red,blue,orange}]{C/S,P/M/G,N/B}
```

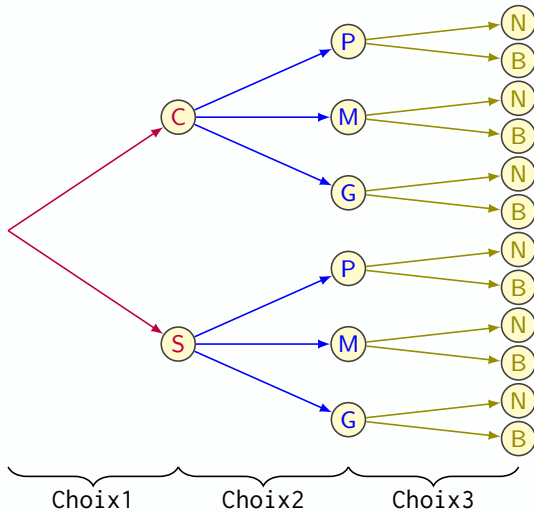


Code $\LaTeX$

%avec personnalisations

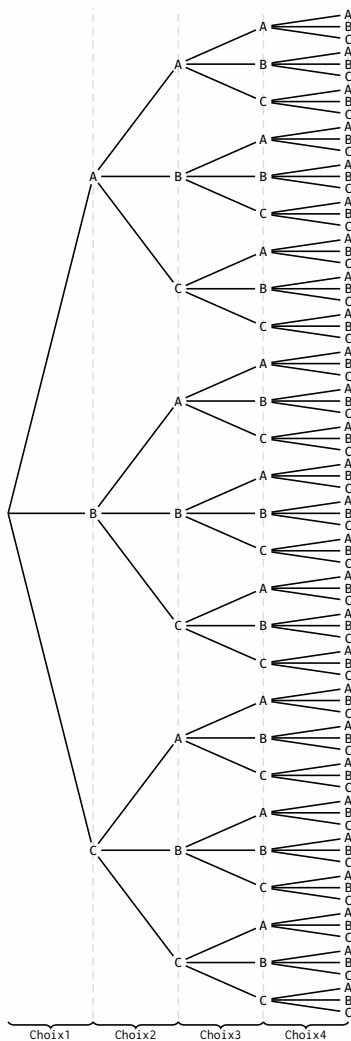
```
\tikzset{arbrechoixsommets/.style={%
  circle,semithick,draw=darkgray,fill=yellow!25,minimum size=\fpeval{0.9*\LISTECH0IXinterfeuille}cm,
  inner sep=1pt,font=\small\sffamily}}
\tikzset{arbrechoixaretes/.style={semithick,->,>=latex}}
```

```
\ArbreChoix[Notice={Choix1,Choix2,Choix3},CouleursNiveaux={purple,blue,olive}]{C/S,P/M/G,N/B}
```



Code $\LaTeX$

```
\ArbreChoix[EspaceFeuilles=0.33,Repet=4,Echelle=0.5,Notice={Choix1,Choix2,Choix3,Choix4},TraitsNotice]{A/B/C}%
```

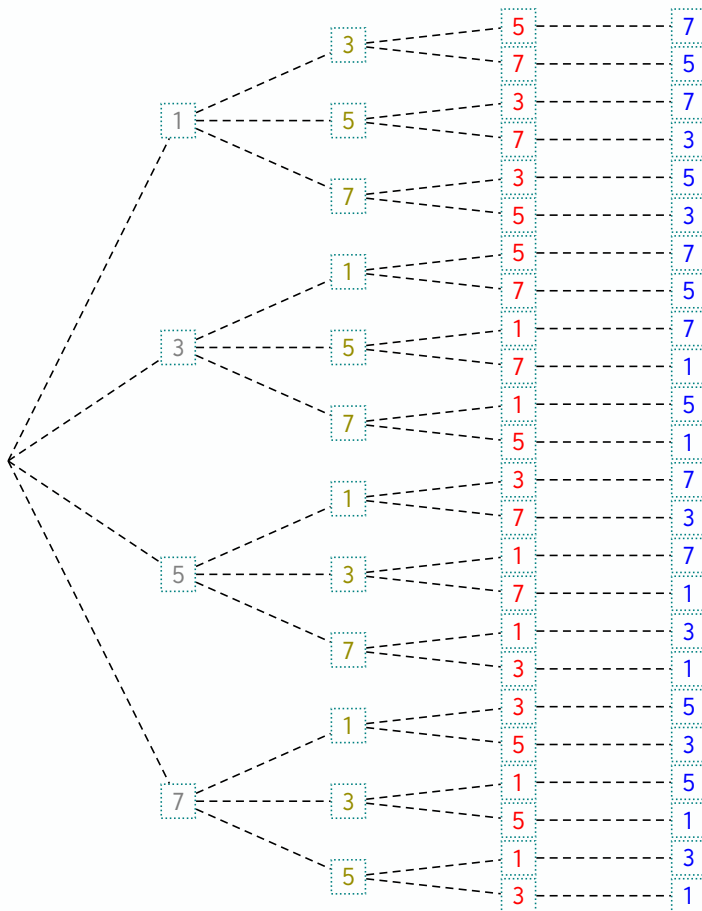


```

\tikzset{arbrechoixsommets/.style={%
  rectangle,semithick,draw=teal,densely dotted,minimum size=\fpeval{0.9*\LISTECHOIXinterfeuille}cm,
  inner sep=1pt,font=\small\ttfamily}}
\tikzset{arbrechoixaretes/.style={semithick,densely dashed}}

\ArbreChoixSansRemise[CouleursNiveaux={gray,olive,red,blue}]{1,3,5,7}

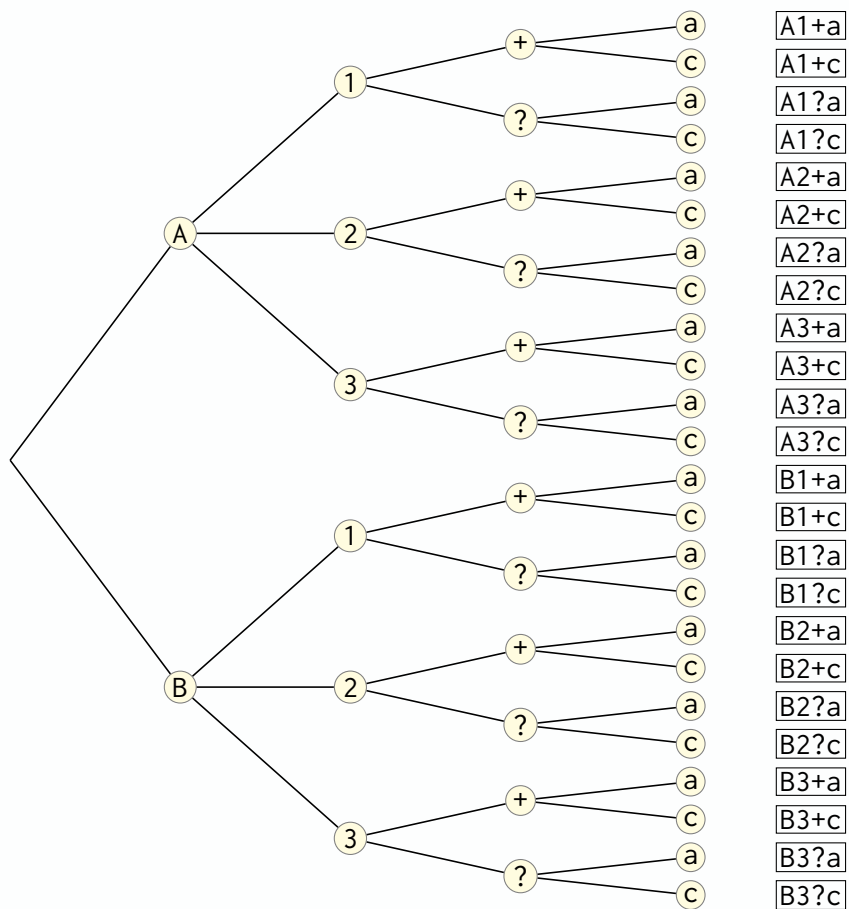
```



```

\tikzset{arbrechoixsommets/.style={circle,draw=gray,fill=yellow!15,inner sep=1pt,font=\ttfamily}}
\tikzset{arbrechoixresultats/.style={rectangle,draw,inner sep=1.5pt,font=\ttfamily}}
\ArbreChoix[AffResultats]{A/B,1/2/3,+/? ,a/c}%,6/9}

```



50 Fonction de répartition

50.1 Idée



2.7.0 L'idée est de proposer une commande (en accord avec les commandes de repérage, page ??) pour tracer la représentation graphique d'une fonction de répartition discrète.

Code $\LaTeX$

```
\begin{tikzpicture}[paramètres de la fenêtre]
%commandes pour la fenêtre graphique
\FonctionRepartTikz[clés]{liste des probas,borneinf,bornesup}
\end{tikzpicture}
```

50.2 Utilisation



Le premier argument, optionnel et entre **[...]** propose les clés suivantes :

- la clé **<Couleur>** pour la couleur du tracé; défaut **<red>**
- la clé **<Epaisseur>** pour gérer l'épaisseur des tracés (en raccourci TikZ); défaut **<thick>**
- le booléen **<Pointilles>** pour afficher les pointillés horizontaux; défaut **<true>**
- la clé **<Extremite>** parmi **<crochet/>point>** pour gérer les extrémités des segments. défaut **<crochet>**

L'argument obligatoire et entre **{...}** permet de spécifier la liste des probas-intervalles :

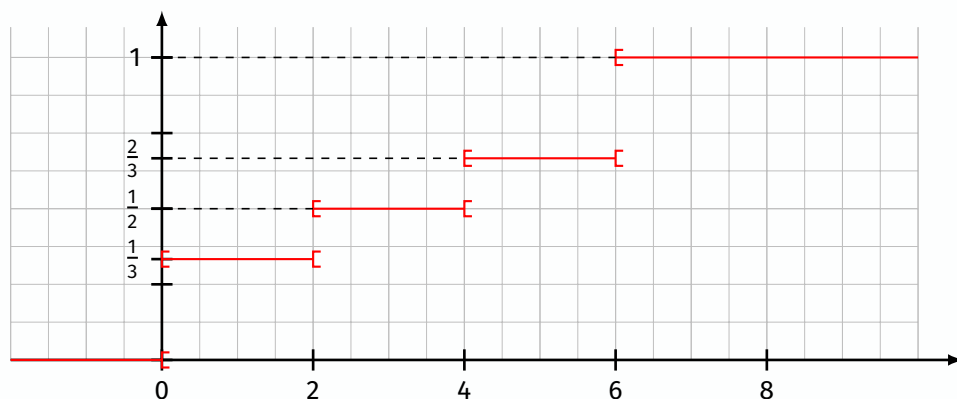
- avec ***** pour remplacer ∞ ;
- sous la forme **proba,borneinf,bornesup / proba,borneinf,bornesup / ...**.



Le code remplace ***** par les valeurs stockées dans **\xmin** ou **\xmax**, d'où l'intérêt d'utiliser la commande en *partenariat* des commandes de repérage de Proflycee.

Code $\LaTeX$

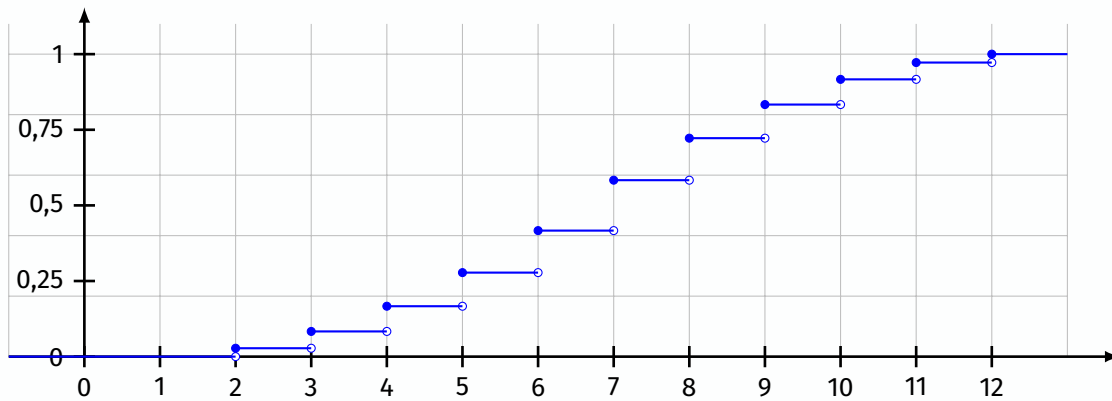
```
\begin{tikzpicture}[y=4cm,xmin=-2,xmax=10,ymin=0,ymax=1.1,
  xgrille=1,xgrilles=0.5,ygrille=0.5,ygrilles=0.125]
\GrilleTikz %grille
\AxesTikz %axes
\AxexTikz{0,2,4,6,8} %graduations de (0x)
\AxeXTikz[AffGrad=false]{0,0.25,...,1} %graduations de (0y) sans valeurs
\AxeYTikz[Frac]{1/3,1/2,2/3,1} %valeurs des probas, en fraction
%les probas étant données en fraction, on protège par des {...}
\FonctionRepartTikz{0,*,0 / {1/3},0,2 / {1/2},2,4 / {2/3},4,6 / 1,6,*}
\end{tikzpicture}
```



```

\begin{tikzpicture}[y=4cm,xmin=-1,xmax=13,ymin=0,ymax=1.1,
  ↳ xgrille=1,xgrilles=0.5,ygrille=0.2,ygrilles=0.125]
  \GrilleTikz[Affs=false]
  \AxesTikz
  \AxeYtikz{0,0.25,...,1}
  \AxeXTikz{0,1,...,12}
  \FonctionRepartTikz[Extremite=point,Couleur=blue,Pointilles=false]%
    {0,*,2 / {1/36},2,3 / {3/36},3,4 / {6/36},4,5 / {10/36},5,6 / {15/36},6,7 / {21/36},7,8 / {26/36},8,9
    ↳ / {30/36},9,10 / {33/36},10,11 / {35/36},11,12 / 1,12,*}
\end{tikzpicture}

```



51 Loi binomiale et graphiques

51.1 Idée



3.04b L'idée est de proposer une commande ou un environnement pour travailler sur la représentation graphique (en TikZ) d'une loi binomiale.

Il est également possible de rajouter la loi normale « associée ».

Code $\text{\LaTeX}$

```
%commande autonome
\HISTOGRAMMEBINOMIALE[clés]{n}{p}

%dans un environnement, pour rajout(s) éventuel(s)
\begin{HISTOGRAMMEBINOMIALE[clés]{n}{p}
%code(s) tikz
\TraceLoiNormale(*)[options]<plage>{param1}{param2}
\end{HISTOGRAMMEBINOMIALE}
```

51.2 Utilisation de la commande autonome et de l'environnement



Le fonctionnement est similaire pour les deux possibilités, l'environnement permet de rajouter des éléments graphiques *a posteriori*.

Le premier argument, optionnel et entre `[...]` propose les clés suivantes :

- **<Largeur>** : largeur, en cm, de la fenêtre graphique (hors labels); défaut **<10>**
- **<Hauteur>** : hauteur, en cm, de la fenêtre graphique (hors labels); défaut **<5>**
- **<PasX>** : graduation de l'axe horizontal; défaut **<5>**
- **<PasY>** : graduation de l'axe vertical; défaut **<0.01>**
- **<Plage>** : plage, sous la forme a-b du coloriage éventuel; défaut **<vide>**
- **<CouleurPlage>** : couleur de la plage éventuelle; défaut **<teal!50>**
- **<Epaisseur>** : épaisseur des tracés (tous); défaut **<0.8pt>**
- **<ClipX>** : restriction de l'axe Ox, sous la forme a-b; défaut **<vide>**
- **<AffNormale>** : booléen pour rajouter la loi normale; défaut **<true>**
- **<CouleurNormale>** : couleur pour la loi normale; défaut **<red>**
- **<Police>** : police des labels des axes. défaut **<\normalfont\normalsize>**

Les arguments obligatoires et entre `{...}` permettent de spécifier les paramètres de la loi binomiale.



Les unités graphiques ne sont pas à préciser, le code se charge de les déterminer en fonction des dimensions voulues!

Pour des lois binomiales avec un aspect « dissymétrique », il est conseillé d'utiliser la clé **<ClipX>**, pour réduire la zone d'affichage, sous la forme :

- **<ClipX=a-b>** pour une déclaration des données en *dur*;
- avec `*` pour remplacer les bornes par 0 et/ou n .

51.3 Le tracé de la loi normale



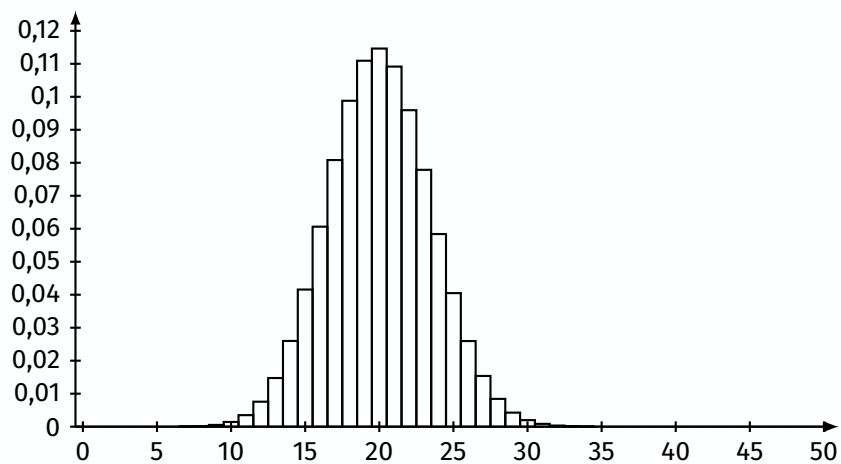
Concernant la possibilité de rajouter la *loi normale* :

- la version étoilée permet de préciser les paramètres de la loi binomiale, tandis que la version non étoilée permet de préciser les paramètres de la loi normale (directement);
- l'argument obligatoire, entre `[...]`, correspond aux options (en langage TikZ) du tracé;
- l'argument obligatoire, entre `<...>`, correspond au domaine du tracé (en langage `xint`), sous la forme `deb...[pas]...fin`;
- les arguments correspondent aux paramètres (n et p ou μ et σ) suivant le mode étoilé ou non.

51.4 Exemples

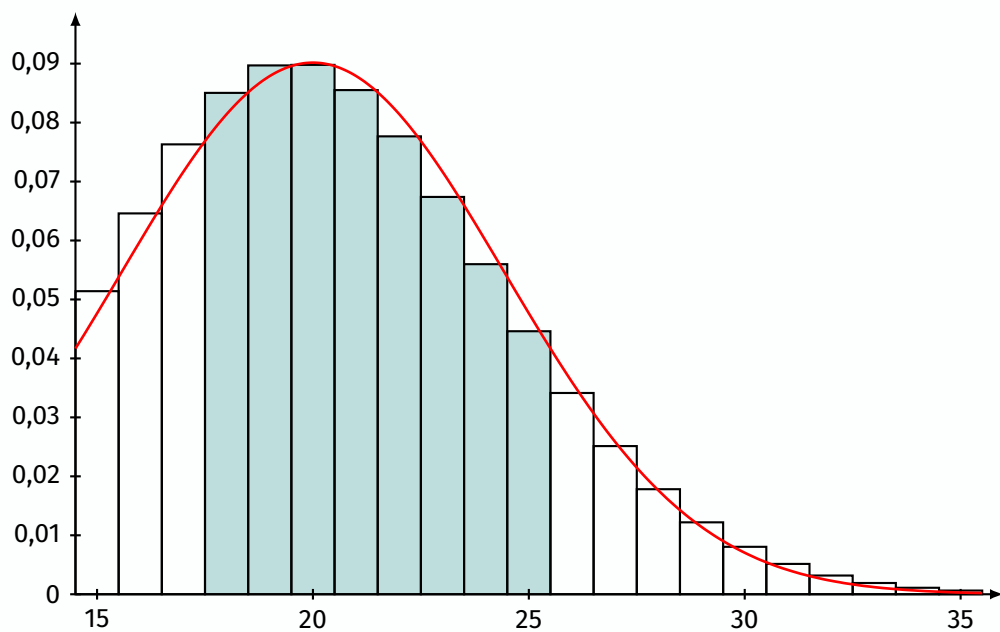
Code $\LaTeX$

```
\HistogrammeBinomiale{50}{0.4}
```



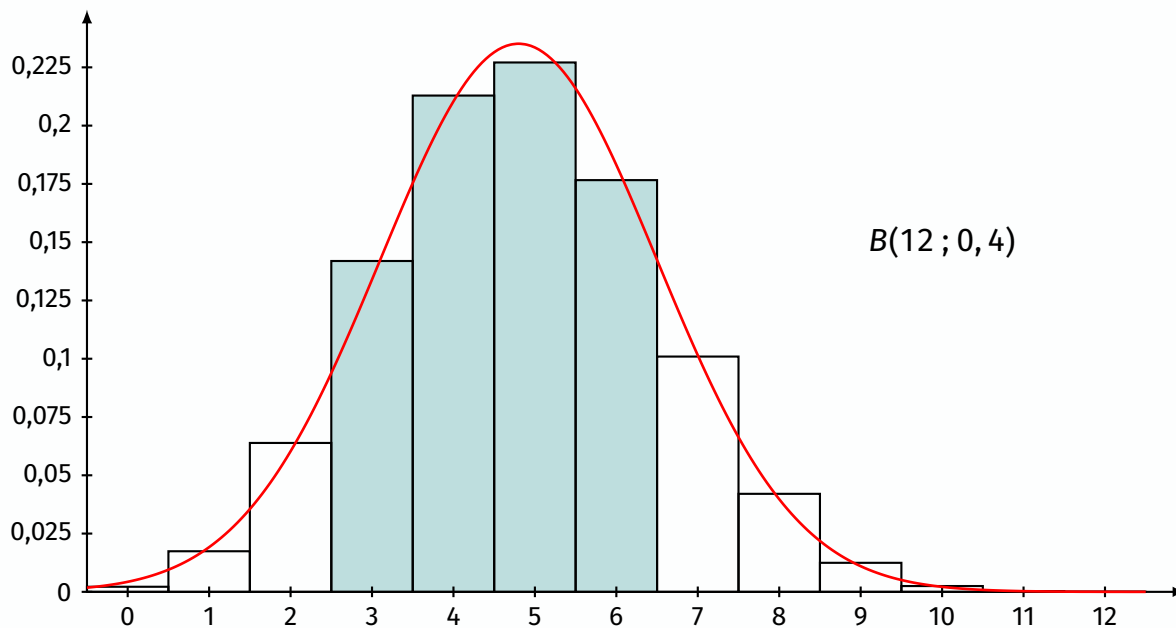
Code $\LaTeX$

```
\HistogrammeBinomiale[Largeur=12,Hauteur=7,ClipX=15-35,Plage=18-25,AffNormale]{1000}{0.02}
```



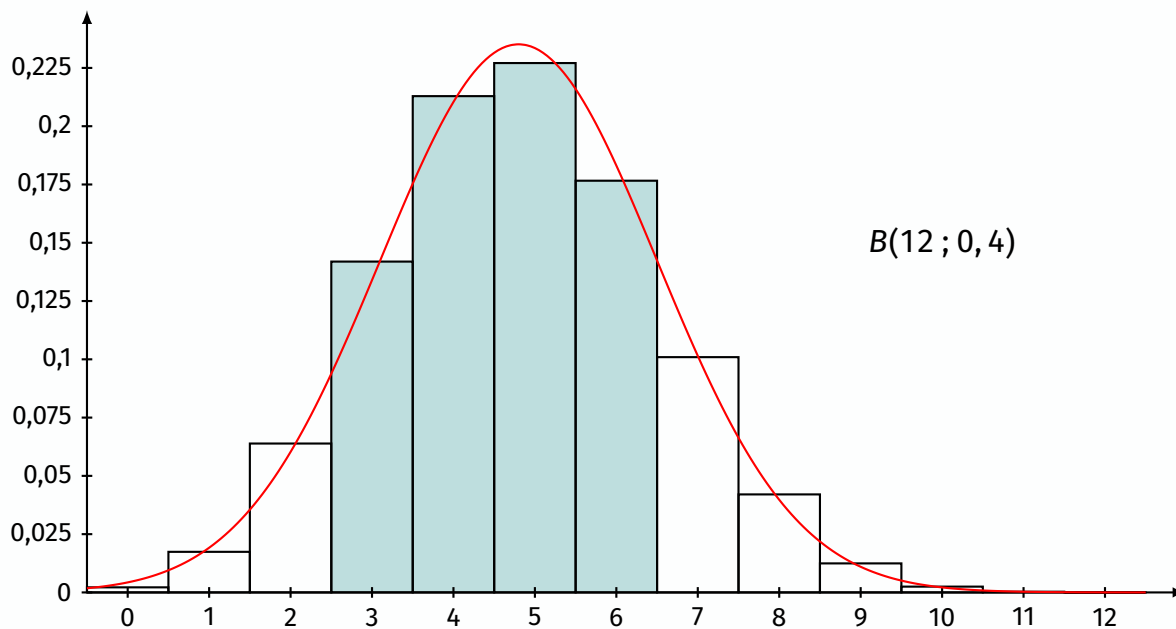
Code $\LaTeX$

```
\begin{HistoBinomiale}[PasX=1,Largeur=14,Hauteur=7,Plage=3-6,PasY=0.025,AffNormale]{12}{0.4}
\draw (10,0.15) node[font=\large] {$\mathcal{B}(12;0,4)$} ;
\end{HistoBinomiale}
```



Code $\LaTeX$

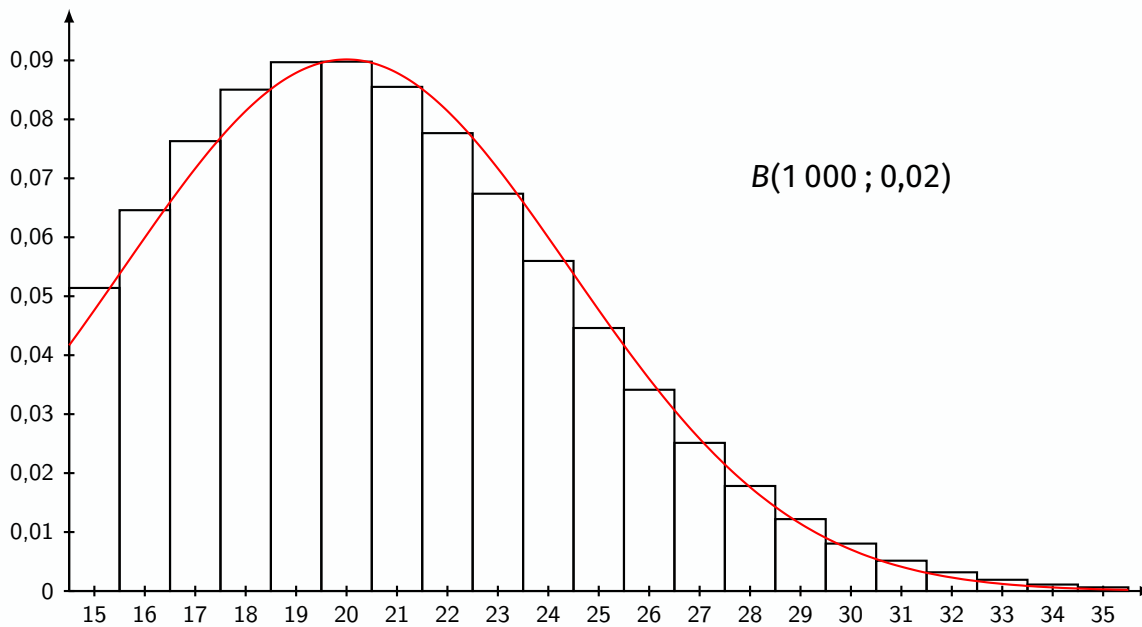
```
\begin{HistoBinomiale}[PasX=1,Largeur=14,Hauteur=7,Plage=3-6,PasY=0.025]{12}{0.4}
\draw (10,0.15) node[font=\large] {$\mathcal{B}(12;0,4)$} ;
\TraceLoiNormale*[thick,red]<-0.5..[0.1]..12.5>{12}{0.4}
\end{HistoBinomiale}
```



```

\begin{HistoBinomiale}%
  [PasX=1, Largeur=14, Hauteur=7, PasY=0.01, ClipX=15-35, Police=\small\sffamily]{1000}{0.02}
  \draw (30,0.07) node[font=\large] {$\mathcal{B}(\text{\num{1000}};\text{\num{0.02}})$};
  \TraceLoiNormale[thick,red]<14.5..[0.1]..35.5>{20}{4.427}
\end{HistoBinomiale}

```



Thème

OUTILS POUR L'ARITHMÉTIQUE

Onzième partie

Outils pour l'arithmétique

52 Réduction modulo

52.1 Idée



 3.02c L'idée est de proposer une commande pour travailler sur les réductions *modulo* :

- en donnant la *plus petite positive*;
- ou la *plus petite négative*.

Code $\LaTeX$

```
%reste/quotient modulo de a par n
\ResteMod(*){a}{n}
\QuoMod{a}{n}
```

Code $\LaTeX$

```
%avec la librairie 'ecritures' pour les modulus
$\num{1234} \equiv \ResteMod{1234}{26} \Modulo{26} \equiv \ResteMod*{1234}{26} \Modulo{26}$
```

1 234 $\equiv$ 12 [26] $\equiv$ -14 [26]

52.2 Clés et options



En ce qui concerne la commande, la version *étoilée* permet d'afficher la *plus petite congruence négative*. L'argument $\text{\code{a}}$ peut être donné via un *calcul*, en langage $\text{\code{xint}}$.

Code $\LaTeX$

```
$\num{98} \equiv \ResteMod{98}{9} \Modulo{9}$\\
$\num{98} \equiv \ResteMod*{98}{9} \Modulo{9}$\\
$7^{50} \equiv \ResteMod{7**50}{8} \Modulo{8}$\\
```

98 $\equiv$ 8 [9]
98 $\equiv$ -1 [9]
7⁵⁰ $\equiv$ 1 [8]

53 Conversions entre bases

53.1 Idée



 L'idée est de proposer une commande effectuer et présenter des conversions entre bases :

- avec affichage des bases;
- avec regroupements par paquets de pour le binaire.

Code $\LaTeX$

```
%mode sans affichage des bases
\ConversionEntreBases[espace blocs 4 pour le binaire]{dep->arrivée}{nb}

%mode avec affichage des bases
\ConversionEntreBases*[espace blocs 4 pour le binaire]{dep->arrivée}{nb}
```

53.2 Exemples

Code $\LaTeX$

```
\ConversionEntreBases{2->16}{11111110}\par
\ConversionEntreBases*{2->16}{11111110}\par
\ConversionEntreBases{16->10}{ACDC}\par
\ConversionEntreBases*{16->2}{ACDC}\par
\ConversionEntreBases*{16->5}{A47E}\par
\ConversionEntreBases*{8->7}{7777}
```

1111 1110 = FE
 $1111\ 1110_2 = FE_{16}$
ACDC = 44252
 $ACDC_{16} = 1010110011011100_2$
 $A47E_{16} = 2321420_5$
 $7777_8 = 14640_7$

54 Division euclidienne

54.1 Idée



`3.01b` L'idée est de proposer une commande pour travailler sur les divisions euclidiennes :

- présentation classique;
- présentation avec vérification pour le reste;
- présentation avec pointillés pour compléter;
- `3.03b` présentation *brute*.



Le package `xinttools` est la base de cette macro. C'est lui qui s'occupe de la partie *calculs*.

Code $\text{\LaTeX}$

```
%mode brut
\DivisionEucl{a}{b}

%mode présentation
\DivEucl{*}[clés]{a}{b}
```

Code $\text{\LaTeX}$

```
\DivisionEucl{145}{7}\par
\DivisionEucl{-40}{7}\par
\DivEucl{145}{7}\par
\DivEucl*{145}{7}\par
\DivEucl*[Quotient=false]{-145}{7}\par
\DivisionEucl{145}{-7}\par
\DivEucl[Reste=false]{145}{-7}\par
\DivEucl[Vide]{-145}{-7}
```

$7 \times 20 + 5$
 $7 \times (-6) + 2$
 $145 = 7 \times 20 + 5$
 $145 = 7 \times 20 + 5$ avec $0 \leq 5 < 7$
 $-145 = 7 \times \dots + 2$ avec $0 \leq 2 < 7$
 $-7 \times (-20) + 5$
 $145 = -7 \times (-20) + \dots$
 $-145 = -7 \times \dots + \dots$

54.2 Clés et options



En ce qui concerne la commande pour la division euclidienne *présentée* :

- la version *étoilée* permet d'afficher la *vérification* du reste;
- le booléen `\Quotient` permet d'afficher le quotient; défaut `\true`
- le booléen `\Reste` permet d'afficher le reste; défaut `\true`
- le booléen `\Vide` permet de ne pas afficher le quotient et le reste; défaut `\false`
- la clé `\Pointilles` permet de spécifier les éventuels pointillés; défaut `\ldots`
- les arguments obligatoires, et entre `\{...\}`, sont les entiers avec lesquels on travaille.

À noter que la commande est incluse dans un bloc `\ensuremath` (sans la vérification du reste), donc les `\ldots` ne sont pas nécessaires.

```
\DivEucl*{123456789}{8547}\par  
\DivEucl[Vide,Pointilles={\makebox[1cm]{\dotfill}}]{123456789}{8547}
```

123 456 789 = 8 547 $\times$ 14 444 + 3 921 avec $0 \leq 3\,921 < 8\,547$

123 456 789 = 8 547 $\times$ + ...

55 Conversions binaire/hexadécimal/décimal

55.1 Idée



L'idée est de *compléter* les possibilités offertes par le package `xintbinhex`, en mettant en forme quelques conversions :

- décimal en binaire avec blocs de 4 chiffres en sortie;
- hexadécimal en binaire avec blocs de 4 chiffres en sortie;
- conversion binaire ou hexadécimal en décimal avec écriture polynomiale.



Le package `xintbinhex` est la base de ces macros, puisqu'il permet de faire des conversions directes ! Les macros présentées ici ne font que les intégrer dans un environnement adapté à une correction ou une présentation !

Code $\text{\LaTeX}$

```
\xintDecToHex{100}  
\xintDecToBin{51}  
\xintHexToDec{A4C}  
\xintBinToDec{110011}  
\xintBinToHex{11111111}  
\xintHexToBin{ACDC}  
\xintCHexToBin{3F}
```

Code $\text{\LaTeX}$

```
64  
110011  
2636  
51  
FF  
1010110011011100  
00111111
```

55.2 Conversion décimal vers binaire

Code $\text{\LaTeX}$

```
\ConversionDecBin(*)[clés]{nombre}
```



Concernant la commande en elle même, peu de paramétrage :

- la version *étoilée* qui permet de ne pas afficher de zéros avant pour « compléter »;
- le booléen `<AffBase>` qui permet d'afficher ou non la base des nombres; défaut `<true>`
- l'argument, *obligatoire*, est le nombre entier à convertir.

Le formatage est géré par `sinuitx`, le mieux est donc de positionner la commande dans un environnement mathématique.

Les nombres écrits en binaire sont, par défaut, présentés en bloc(s) de 4 chiffres.

Code $\text{\LaTeX}$

```
% Conversion avec affichage de la base et par bloc de 4  
$\ConversionDecBin{415}$  
% Conversion avec affichage de la base et sans forcément des blocs de 4  
$\ConversionDecBin*{415}$  
% Conversion sans affichage de la base et par bloc de 4  
$\ConversionDecBin[AffBase=false]{415}$  
% Conversion sans affichage de la base et sans forcément des blocs de 4  
$\ConversionDecBin*[AffBase=false]{415}$
```

Code $\LaTeX$

```
415_{10} = 0001\,1001\,1111_2
415_{10} = 1\,1001\,1111_2
415 = 0001\,1001\,1111
415 = 1\,1001\,1111
```

55.3 Conversion binaire vers hexadécimal



L'idée est ici de présenter la conversion, grâce à la conversion « directe » par blocs de 4 chiffres :

- la macro rajoute éventuellement les zéros pour compléter;
- elle découpe par blocs de 4 chiffres binaires;
- elle présente la conversion de chacun des blocs de 4 chiffres binaires;
- elle affiche la conversion en binaire.

Code $\LaTeX$

```
\ConversionBinHex[clés]{nombre}
```



Quelques **clés** sont disponibles pour cette commande :

- le booléen **<AffBase>** qui permet d'afficher ou non la base des nombres; défaut **<true>**
- le booléen **<Details>** qui permet d'afficher ou le détail par bloc de 4. défaut **<true>**

Le formatage est géré par le package `sinuitx`, le mieux est de positionner la commande dans un environnement mathématique.

Code $\LaTeX$

```
%conversion avec détails et affichage de la base
$\ConversionBinHex{110011111}$
%conversion sans détails et affichage de la base
$\ConversionBinHex[Details=false]{110011111}$
%conversion sans détails et sans affichage de la base
$\ConversionBinHex[AffBase=false,Details=false]{110011111}$
```

Code $\LaTeX$

```
1\,1001\,1111_2 = 0001\,1001\,1111 = \underbrace{0001}_1 \underbrace{1001}_9 \underbrace{1111}_F = 19F_{16}
1\,1001\,1111_2 = 19F_{16}
1\,1001\,1111 = 19F
```

55.4 Conversion hexadécimal vers binaire



2.7.8 L'idée est ici de présenter la conversion, grâce à la conversion « directe » par blocs de 4 chiffres :

- la macro découpe chaque caractère hexa en bloc de 4 chiffres binaires;
- elle affiche la conversion en binaire.

Code $\text{\LaTeX}$

```
\ConversionHexBin[clés]{nombre}
```



Quelques **clés** sont disponibles pour cette commande :

- le booléen **AffBase** qui permet d'afficher ou non la base des nombres; défaut **true**
- le booléen **Details** qui permet d'afficher ou le détail par bloc de 4. défaut **true**

La commande est à insérer dans un environnement mathématique.

Code $\text{\LaTeX}$

```
%conversion avec détails et affichage de la base
$\ConversionHexBin{ACDC}$
%conversion sans détails et affichage de la base
$\ConversionHexBin[Details=false]{ACDC}$
%conversion sans détails et sans affichage de la base
$\ConversionHexBin[AffBase=false,Details=false]{ACDC}$
```

Code $\text{\LaTeX}$

```
ACDC16 = 1010 1100 1101 11002
ACDC16 = A1010 C1100 D1101 C11002
ACDC = 1010 1100 1101 1100
```

55.5 Conversion binaire ou hexadécimal en décimal



L'idée est ici de présenter la conversion, grâce à l'écriture polynômiale :

- écrit la somme des puissances;
- convertir si besoin les *chiffres* hexadécimal;
- peut ne pas afficher les monômes de coefficient 0.

Code $\text{\LaTeX}$

```
\ConversionVersDec[clés]{nombre}
```



Quelques **clés** sont disponibles pour cette commande :

- la clé **BaseDep** qui est la base de départ (2 ou 16!); défaut **2**
- le booléen **AffBase** qui permet d'afficher ou non la base des nombres; défaut **true**
- le booléen **Details** qui permet d'afficher ou le détail par bloc de 4; défaut **true**
- le booléen **Zeros** qui affiche les chiffres 0 dans la somme. défaut **true**

Le formatage est toujours géré par le package `sinuitx`, le mieux est de positionner la commande dans un environnement mathématique.

Code $\LaTeX$

```
%conversion 16->10 avec détails et affichage de la base et zéros
 $\backslash$ ConversionVersDec[BaseDep=16]{19F}$
%conversion 2->10 avec détails et affichage de la base et zéros
 $\backslash$ ConversionVersDec{110011}$
%conversion 2->10 avec détails et affichage de la base et sans zéros
 $\backslash$ ConversionVersDec[Zeros=false]{110011}$
%conversion 16->10 sans détails et affichage de la base et avec zéros
 $\backslash$ ConversionVersDec[BaseDep=16,Details=false]{AC0DC}$
%conversion 16->10 avec détails et sans affichage de la base et sans zéros
 $\backslash$ ConversionVersDec[Zeros=false,BaseDep=16]{AC0DC}$
```

Code $\LaTeX$

```
19F16 = 1 × 162 + 9 × 161 + 15 × 160 = 41510
1100112 = 1 × 25 + 1 × 24 + 0 × 23 + 0 × 22 + 1 × 21 + 1 × 20 = 5110
1100112 = 1 × 25 + 1 × 24 + 1 × 21 + 1 × 20 = 5110
AC0DC16 = 70473210
AC0DC16 = 10 × 164 + 12 × 163 + 13 × 161 + 12 × 160 = 70473210
```

56 Conversion « présentée » d'un nombre en base décimale

56.1 Idée



L'idée est de proposer une « présentation » par divisions euclidiennes pour la conversion d'un entier donné en base 10 dans une base quelconque.

Les commandes de la section précédente donne *juste* les résultats, dans cette section il y a en plus la présentation de la conversion.

56.2 Code et clés

Code $\LaTeX$

```
%conversion basique  
\ConversionDepuisBaseDix{78}{2}
```

$$\left\{ \begin{array}{rcl} 78 & = & 2 \times 39 + 0 \\ 39 & = & 2 \times 19 + 1 \\ 19 & = & 2 \times 9 + 1 \\ 9 & = & 2 \times 4 + 1 \\ 4 & = & 2 \times 2 + 0 \\ 2 & = & 2 \times 1 + 0 \\ 1 & = & 2 \times 0 + 1 \end{array} \right. \Rightarrow 78_{10} = 1001110_2$$



La « tableau », qui est géré par `\array` est inséré dans un `\ensuremath`, donc les `\$...\$` ne sont pas utiles.

Code $\LaTeX$

```
\ConversionDepuisBaseDix[options]{nombre en base 10}{base d'arrivée}
```



Quelques options pour cette commande :

- la clé **<Couleur>** pour la couleur du « rectangle » des restes; défaut **<red>**
- la clé **<DecalH>** pour gérer le décalage H du « rectangle », qui peut être donné soit sous la forme **<Esp>** ou soit sous la forme **<espgauche/espdroite>**; défaut **<2pt>**
- la clé **<DecalV>** pour le décalage vertical du « rectangle »; défaut **<3pt>**
- la clé **<Noeud>** pour le préfixe du nœud du premier et du dernier reste (pour utilisation en TikZ); défaut **<EEE>**
- le booléen **<Rect>** pour afficher ou non le « rectangle » des restes; défaut **<true>**
- le booléen **<CouleurRes>** pour afficher ou non la conversion en couleur (identique au rectangle). défaut **<false>**

Code $\LaTeX$

```
%conversion avec changement de couleur  
\ConversionDepuisBaseDix[Couleur=blue]{45}{2}  
  
%conversion sans le rectangle  
Par divisions euclidiennes successives, \ConversionDepuisBaseDix[Rect=false]{54}{3}.  
  
%conversion avec gestion du decalh pour le placement précis du rectangle  
\ConversionDepuisBaseDix[Couleur=brown,DecalH=6pt/2pt]{1012}{16}  
  
%conversion avec noeud personnalisé et réutilisation  
\ConversionDepuisBaseDix[Couleur=CouleurVertForet,CouleurRes,Noeud=TEST]{100}{9}.  
\begin{tikzpicture}  
\draw[overlay,CouleurVertForet,thick,->] (TEST2.south east) to[bend right] ++ (3cm,-1cm) node[right]  
{test } ;  
\end{tikzpicture}
```

$$\left\{ \begin{array}{l} 45 = 2 \times 22 + 1 \\ 22 = 2 \times 11 + 0 \\ 11 = 2 \times 5 + 1 \\ 5 = 2 \times 2 + 1 \\ 2 = 2 \times 1 + 0 \\ 1 = 2 \times 0 + 1 \end{array} \right. \Rightarrow 45_{10} = 101101_2$$

Par divisions euclidiennes successives, $\left\{ \begin{array}{l} 54 = 3 \times 18 + 0 \\ 18 = 3 \times 6 + 0 \\ 6 = 3 \times 2 + 0 \\ 2 = 3 \times 0 + 2 \end{array} \right. \Rightarrow 54_{10} = 2000_3.$

$$\left\{ \begin{array}{l} 1012 = 16 \times 63 + 4 \\ 63 = 16 \times 3 + 15 \\ 3 = 16 \times 0 + 3 \end{array} \right. \Rightarrow 1012_{10} = 3F4_{16}$$

On obtient donc $\left\{ \begin{array}{l} 100 = 9 \times 11 + 1 \\ 11 = 9 \times 1 + 2 \\ 1 = 9 \times 0 + 1 \end{array} \right. \Rightarrow 100_{10} = 121_9.$

→ test

57 Algorithme d'Euclide pour le PGCD

57.1 Idée



L'idée est de proposer une « présentation » de l'algorithme d'Euclide pour le calcul du PGCD de deux entiers.

Le package `\xintgcd` permet déjà de le faire, il s'agit ici de travailler sur la *mise en forme*.

Code $\LaTeX$

```
\PresentationPGCD[options]{a}{b}
```

Code $\LaTeX$

```
\tikzstyle{every picture}+=[remember picture]
...
\PresentationPGCD{150}{27}
```

Code $\LaTeX$

```
\PresentationPGCD{150}{27}
```

$$\left\{ \begin{array}{l} 150 = 27 \times 5 + 15 \\ 27 = 15 \times 1 + 12 \\ 15 = 12 \times 1 + \textcolor{red}{3} \\ 12 = 3 \times 4 + 0 \end{array} \right. \Rightarrow \text{PGCD}(150; 27) = 3$$



La mise en valeur du dernier reste non nul est gérée par du code TikZ, en mode `\tikzstyle{every picture}+=[remember picture]`, donc il faut bien penser à déclarer dans le préambule :

57.2 Options et clés



Quelques options disponibles pour cette commande :

- la clé `<Couleur>` qui correspond à la couleur pour la mise en valeur; défaut `<red>`
- la clé `<DecalRect>` qui correspond à l'écartement du rectangle de mise en valeur; défaut `<2pt>`
- le booléen `<Rectangle>` qui gère l'affichage ou non du rectangle de mise en valeur; défaut `<true>`
- la clé `<Noeud>` qui gère le préfixe du nom du nœud TikZ du rectangle (pour exploitation ultérieure); défaut `<FFF>`
- le booléen `<CouleurResultat>` pour mettre ou non en couleur de PGCD; défaut `<false>`
- le booléen `<AfficheConclusion>` pour afficher ou non la conclusion; défaut `<true>`
- le booléen `<AfficheDelimiteurs>` pour afficher ou non les délimiteurs (accolade gauche et trait droit). défaut `<true>`

Le rectangle de mise en valeur est donc un nœud TikZ qui sera nommé, par défaut `FFF1`.

La présentation est dans un environnement `\ensuremath` donc les `\$... \$` ne sont pas indispensables.

Code $\LaTeX$

```
\PresentationPGCD[CouleurResultat]{150}{27}
```

$$\left\{ \begin{array}{l} 150 = 27 \times 5 + 15 \\ 27 = 15 \times 1 + 12 \\ 15 = 12 \times 1 + \textcolor{red}{3} \\ 12 = 3 \times 4 + 0 \end{array} \right. \Rightarrow \text{PGCD}(150; 27) = \textcolor{red}{3}$$

```

\PresentationPGCD[CouleurResultat,Couleur=CouleurVertForet]{1250}{450}.

\PresentationPGCD[CouleurResultat,Couleur=blue]{13500}{2500}.

\PresentationPGCD[Rectangle=false]{420}{540}. \\\

D'après l'algorithme d'Euclide, on a $\left[ \right] \PresentationPGCD[Couleur=lime,AfficheConclusion=false,
\AfficheDelimiteurs=false]{123456789}{9876} \right.$
\begin{tikzpicture}
\draw[overlay,lime,thick,<-] (FFF1.east) to[bend right] ++ (1cm,0.75cm) node[right] {dernier reste non
nul} ;
\end{tikzpicture}

```

$$\left[\begin{array}{l} 1\,250 = 450 \times 2 + 350 \\ 450 = 350 \times 1 + 100 \\ 350 = 100 \times 3 + 50 \\ 100 = 50 \times 2 + 0 \end{array} \right] \Rightarrow \text{PGCD}(1\,250; 450) = 50.$$

$$\left[\begin{array}{l} 13\,500 = 2\,500 \times 5 + 1\,000 \\ 2\,500 = 1\,000 \times 2 + 500 \\ 1\,000 = 500 \times 2 + 0 \end{array} \right] \Rightarrow \text{PGCD}(13\,500; 2\,500) = 500.$$

$$\left[\begin{array}{l} 420 = 540 \times 0 + 420 \\ 540 = 420 \times 1 + 120 \\ 420 = 120 \times 3 + 60 \\ 120 = 60 \times 2 + 0 \end{array} \right] \Rightarrow \text{PGCD}(420; 540) = 60.$$

$$\begin{array}{l|l} \text{D'après l'algorithme d'Euclide, on a} & \begin{array}{r} 123\,456\,789 = 9\,876 \times 12\,500 + 6\,789 \\ 9\,876 = 6\,789 \times 1 + 3\,087 \\ 6\,789 = 3\,087 \times 2 + 615 \\ 3\,087 = 615 \times 5 + 12 \\ 615 = 12 \times 51 + 3 \\ 12 = 3 \times 4 + 0 \end{array} \end{array}$$

dernier reste non nul

57.3 Compléments



La présentation des divisions euclidiennes est gérée par un tableau du type `\array`, avec alignement vertical de symboles $=$ et $+$.

Par défaut, les délimiteurs choisis sont donc l'accolade gauche et le trait droit, mais la clé booléenne `\AfficheDelimiteurs=false` permet de choisir des délimiteurs différents.

```

$\left[ \PresentationPGCD[AfficheConclusion=false,AfficheDelimiteurs=false]{1234}{5} \right]$

```

$$\left[\begin{array}{l} 1\,234 = 5 \times 246 + 4 \\ 5 = 4 \times 1 + 1 \\ 4 = 1 \times 4 + 0 \end{array} \right]$$

58 Résolution d'une équation diophantienne

58.1 Idée



L'idée est de proposer une résolution d'équation diophantienne du type $ax + by = c$ avec $(a; b; c) \in \mathbb{Z}^3$. Le code se charge de tester les différentes conditions d'existence, et d'adapter la rédaction (fixée et non modifiable...) aux différentes situations :

- cas où $\text{PGCD}(a; b) = 1$; existence de solutions
- cas où $\text{PGCD}(a; b) \neq 1$ et $\text{PGCD}(a; b) \mid c$; existence de solutions
- cas où $\text{PGCD}(a; b) \neq 1$ et $\text{PGCD}(a; b) \nmid c$. pas de solution



Logiquement le code se charge de *parenthéser* de manière automatique pour les nombres négatifs, mais il se peut que certains cas particuliers puissent donner des résultats « non esthétiques »...

Code $\LaTeX$

```
\EquationDiophantienne[Clés]{equation}
```

58.2 Options et clés



Concernant les Clés disponibles pour cette commande, à donner entre $\boxed{\text{...}}$:

- la clé **<Lettre>** pour spécifier le *nom* de l'équation; défaut **<E>**
- la clé **<Inconnues>** qui paramètre les noms des inconnues, sous la forme **<x/y>**; défaut **<x/y>**
- la clé **<Entier>** qui gère le nom de l'entier dans la solution; défaut **<k>**
- le booléen **<Cadres>** pour mettre en valeur les solutions; défaut **<false>**
- le booléen **<PresPGCD>** présenter le calcul du PGCD de $|a|$ et de $|b|$. défaut **<true>**

L'argument obligatoire, et entre $\boxed{\text{...}}$ est quant à lui l'équation, en langage « naturel » du type $\boxed{ax+by=c}$ (le code se charge d'extraire les coefficients, donc pas besoin des signes *).

Code $\LaTeX$

```
\EquationDiophantienne{48x+18y=3}
```

On cherche à résoudre l'équation diophantienne :

$$48x + 18y = 3 \quad (E)$$

D'après l'algorithme d'Euclide : $\left\{ \begin{array}{l} 48 = 18 \times 2 + 12 \\ 18 = 12 \times 1 + 6 \\ 12 = 6 \times 2 + 0 \end{array} \right. \Rightarrow \text{PGCD}(48; 18) = 6.$

Le PGCD de 48 et 18 ne divise pas 3, donc l'équation (E) n'admet aucune solution.

Code $\LaTeX$

```
\EquationDiophantienne[PresPGCD=false]{48x+18y=-5}
```

On cherche à résoudre l'équation diophantienne :

$$48x + 18y = -5 \quad (E)$$

Le PGCD de 48 et de 18 vaut 6.

Le PGCD de 48 et 18 ne divise pas -5, donc l'équation (E) n'admet aucune solution.

$\backslash\text{EquationDiophantienne}\{3x+4y=1\}$

On cherche à résoudre l'équation diophantienne :

$$3x + 4y = 1 \quad (E)$$

D'après l'algorithme d'Euclide : $\begin{cases} 3 = 4 \times 0 + 3 \\ 4 = 3 \times 1 + 1 \\ 3 = 1 \times 3 + 0 \end{cases} \Rightarrow \text{PGCD}(3; 4) = 1.$

Les entiers 3 et 4 sont premiers entre eux, donc l'équation (E) admet une infinité de solutions.
On détermine une solution particulière de (E) :

$$3 \times (-1) + 4 \times 1 = 1 \quad (E_0)$$

Par soustraction :

$$\begin{array}{rcl} 3 \times x & + & 4 \times y = 1 \\ - & 3 \times (-1) & + 4 \times 1 = 1 \\ \hline 3 \times (x + 1) & + & 4 \times (y - 1) = 0 \end{array}$$

On en déduit que $3 \times \underbrace{(x + 1)}_{\text{entier}} = -4 \times (y - 1)$, et donc que $3 \mid -4 \times (y - 1)$.

Or 3 et 4 sont premiers entre eux, donc d'après le théorème de Gauss, on a $3 \mid y - 1$.
Il existe donc un entier k tel que $y - 1 = 3 \times k$, ce qui donne $y = 1 + 3k$.

$$\begin{aligned} 3 \times (x + 1) = -4 \times (y - 1) &\implies 3 \times (x + 1) = -4 \times \underbrace{(1 + 3k - 1)}_y \\ &\implies 3 \times (x + 1) = -4 \times (3k) \\ &\implies x + 1 = -4k \\ &\implies x = -1 - 4k \end{aligned}$$

Ainsi, si x et y sont solutions de (E), alors il existe un entier k tel que $x = -1 - 4k$ et $y = 1 + 3k$.

Réciproquement, soit k un entier quelconque :

$$\begin{aligned} 3 \times (-1 - 4k) + 4 \times (1 + 3k) &= 3 \times (-1) + \cancel{3 \times (-4)k} + 4 \times 1 + \cancel{4 \times 3k} \\ &= 3 \times (-1) + 4 \times 1 \\ &\quad \underbrace{\hspace{1cm}}_{= 1 \text{ d'après } (E_0)} \\ &= 1 \end{aligned}$$

On en déduit que $(-1 - 4k; 1 + 3k)$ est solution de (E).

En conclusion, les solutions de (E) sont donc les couples $(-1 - 4k; 1 + 3k)$, avec k un entier relatif.

$\backslash\text{EquationDiophantienne}[\text{Cadres}, \text{Inconnues}=u/v, \text{Entier}=1][48u+18v=12]$

On cherche à résoudre l'équation diophantienne :

$$48u + 18v = 12$$

D'après l'algorithme d'Euclide : $\begin{cases} 48 = 18 \times 2 + 12 \\ 18 = 12 \times 1 + 6 \\ 12 = 6 \times 2 + 0 \end{cases} \Rightarrow \text{PGCD}(48; 18) = 6.$

Le PGCD de 48 et 18 divise 12, donc on peut simplifier l'équation diophantienne par 6.

$$48u + 18v = 12 \xrightarrow{\div 6} 8u + 3v = 2 \quad (E)$$

Les entiers 8 et 3 sont premiers entre eux, donc l'équation (E) admet une infinité de solutions. On détermine une solution particulière de (E) :

$$8 \times (-1) + 3 \times 3 = 1 \xrightarrow{\times 2} 8 \times (-2) + 3 \times 6 = 2 \quad (E_0)$$

Par soustraction :

$$\begin{array}{rcl} 8 \times u & + & 3 \times v = 2 \\ - & 8 \times (-2) & + 3 \times 6 = 2 \\ \hline 8 \times (u + 2) & + & 3 \times (v - 6) = 0 \end{array}$$

On en déduit que $8 \times \underbrace{(u + 2)}_{\text{entier}} = -3 \times (v - 6)$, et donc que $8 \mid -3 \times (v - 6)$.

Or 8 et 3 sont premiers entre eux, donc d'après le théorème de Gauss, on a $8 \mid v - 6$.

Il existe donc un entier l tel que $v - 6 = 8 \times l$, ce qui donne $\boxed{v = 6 + 8l}$.

$$\begin{aligned} 8 \times (u + 2) &= -3 \times (v - 6) \implies 8 \times (u + 2) = -3 \times \underbrace{(6 + 8l - 6)}_v \\ &\implies 8 \times (u + 2) = -3 \times (8l) \\ &\implies u + 2 = -3l \\ &\implies \boxed{u = -2 - 3l} \end{aligned}$$

Ainsi, si u et v sont solutions de (E), alors il existe un entier l tel que $u = -2 - 3l$ et $v = 6 + 8l$.

Réciproquement, soit l un entier quelconque :

$$\begin{aligned} 8 \times (-2 - 3l) + 3 \times (6 + 8l) &= 8 \times (-2) + \cancel{8 \times (-3)l} + 3 \times 6 + \cancel{3 \times 8l} \\ &= 8 \times (-2) + 3 \times 6 \\ &\quad \underbrace{\hspace{1cm}}_{= 2 \text{ d'après } (E_0)} \\ &= 2 \end{aligned}$$

On en déduit que $(-2 - 3l; 6 + 8l)$ est solution de (E).

En conclusion, les solutions de (E) sont donc les couples $(-2 - 3l; 6 + 8l)$, avec l un entier relatif.

$\backslash\text{EquationDiophantienne}\{47x-18y=1\}$

On cherche à résoudre l'équation diophantienne :

$$47x + (-18)y = 1 \quad (E)$$

D'après l'algorithme d'Euclide :

$$\left\{ \begin{array}{l} 47 = 18 \times 2 + 11 \\ 18 = 11 \times 1 + 7 \\ 11 = 7 \times 1 + 4 \\ 7 = 4 \times 1 + 3 \\ 4 = 3 \times 1 + 1 \\ 3 = 1 \times 3 + 0 \end{array} \right. \Rightarrow \text{PGCD}(47; 18) = 1.$$

Les entiers 47 et 18 sont premiers entre eux, donc l'équation (E) admet une infinité de solutions.
On détermine une solution particulière de (E) :

$$47 \times 5 + (-18) \times 13 = 1 \quad (E_0)$$

Par soustraction :

$$\begin{array}{rcl} 47 \times x & + & (-18) \times y = 1 \\ - & 47 \times 5 & + (-18) \times 13 = 1 \\ \hline 47 \times (x - 5) & + & (-18) \times (y - 13) = 0 \end{array}$$

On en déduit que $47 \times \underbrace{(x - 5)}_{\text{entier}} = 18 \times (y - 13)$, et donc que $47 \mid 18 \times (y - 13)$.

Or 47 et 18 sont premiers entre eux, donc d'après le théorème de Gauss, on a $47 \mid y - 13$.
Il existe donc un entier k tel que $y - 13 = 47 \times k$, ce qui donne $y = 13 + 47k$.

$$\begin{aligned} 47 \times (x - 5) &= 18 \times (y - 13) \Rightarrow 47 \times (x - 5) = 18 \times \underbrace{(13 + 47k - 13)}_y \\ &\Rightarrow 47 \times (x - 5) = 18 \times (47k) \\ &\Rightarrow x - 5 = 18k \\ &\Rightarrow x = 5 + 18k \end{aligned}$$

Ainsi, si x et y sont solutions de (E), alors il existe un entier k tel que $x = 5 + 18k$ et $y = 13 + 47k$.

Réciproquement, soit k un entier quelconque :

$$\begin{aligned} 47 \times (5 + 18k) + (-18) \times (13 + 47k) &= 47 \times 5 + \cancel{47 \times 18k} + (-18) \times 13 + \cancel{(-18) \times 47k} \\ &= 47 \times 5 + (-18) \times 13 \\ &\quad \underbrace{\hspace{1.5cm}}_{= 1 \text{ d'après } (E_0)} \\ &= 1 \end{aligned}$$

On en déduit que $(5 + 18k; 13 + 47k)$ est solution de (E).

En conclusion, les solutions de (E) sont donc les couples $(5 + 18k; 13 + 47k)$, avec k un entier relatif.

59 Nombres premiers

59.1 Idées



 3.10f L'idée est de proposer des commandes pour travailler sur les nombres premiers :

- macro booléenne pour tester la primalité d'un entier;
- macro pour afficher le résultat d'un test de primalité;
- macro pour écrire la décomposition (formatée) en facteurs premiers d'un entier.

Les macros ont pour base du code provenant de la documentation de  xint et d'un code interne de  ProfCollege.

Code  $\TeX$

```
\pflboolestpremier{nb}  
\pflestpremier[code si vrai][code si faux]{nb}
```

Code  $\TeX$

```
\pfldecompnb(*){nb}
```

59.2 Test de primalité



Concernant les commandes de test de primalité :

- la commande  \pflboolestpremier renvoie 1 si l'entier est premier, 0 sinon;
- la commande  \pflestpremier renvoie code si vrai si l'entier est premier, code si faux.

L'argument obligatoire, et entre  {...} est quant à lui le nombre ou le calcul à traiter.

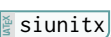
Code  $\TeX$

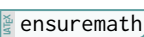
```
\pflboolestpremier{25457} et \pflboolestpremier{25458} \\  
%version manuelle  
\num{25457} \xintifboolexpr{\pflboolestpremier{25457} == 1}{est premier}{n'est pas premier} \\  
%version automatique  
\pflestpremier{25457} \\  
\pflestpremier[est un nombre premier][n'est pas un nombre premier]{25458}
```

1 et 0
25 457 est premier
25 457 est premier
25 458 n'est pas un nombre premier

59.3 Décomposition en facteurs premiers



La commande se charge de décomposer, avec un formatage effectué par  siunitx, un entier en produit de facteurs premiers.

La commande est protégée par un  ensuremath, donc le mode math n'est pas obligatoire.

La version étoilée force l'affichage des exposants égaux à 1.

Code  $\TeX$

```
\pfldecompnb{9999} et \pfldecompnb*{9999} \\  
\pfldecompnb{24458} et \pfldecompnb*{24458} \\  
$\num{24458}=\pfldecompnb{24458}=\pfldecompnb*{24458}$
```

$3^2 \times 11 \times 101$ et $3^2 \times 11^1 \times 101^1$
 $2 \times 7 \times 1\,747$ et $2^1 \times 7^1 \times 1\,747^1$
 $24\,458 = 2 \times 7 \times 1\,747 = 2^1 \times 7^1 \times 1\,747^1$

Code $\text{\LaTeX}$

%avec un calcul comme argument :-)

```
 $\$ \backslash \text{num} \{ \backslash \text{xintfloateval} \{ 2 ** 5 * 3 ** 3 * 17 ** 1 \} \} = \backslash \text{pfldecompb} \{ 2 ** 5 * 3 ** 3 * 17 ** 1 \} \$$ 
```

$$14\,688 = 2^5 \times 3^3 \times 17$$

60 Diviseurs

60.1 Idées



2.7.8 L'idée est de proposer des commandes pour travailler sur les diviseurs d'un entier :

- afficher la liste des diviseurs sous forme d'un ensemble ordonné;
- créer un arbre pour retrouver les diviseurs par la décomposition en facteurs premiers.

Le code se charge de déterminer les diviseurs et de mettre en forme l'arbre.

Code $\text{\LaTeX}$

```
\ListeDiviseurs(*)[Clé]{nombre}
```

Code $\text{\LaTeX}$

```
\ArbreDiviseurs[Clés]{nombre}
```

60.2 Options et clés



En ce qui concerne la commande pour la liste des diviseurs :

- la version *étoilée* permet d'afficher le nom de l'ensemble via $\text{\LaTeX}$ `\mathscr`, $\text{\LaTeX}$ `\mathcal` sinon;
- le booléen `<AffNom>` permet d'afficher le nom de l'ensemble; défaut `<true>`
- l'argument obligatoire, et entre $\text{\LaTeX}$ `{...}`, est quant lui le nombre, y compris en langage $\text{\LaTeX}$ `xint`.

À noter que la commande est incluse dans un bloc $\text{\LaTeX}$ `\ensuremath`, donc les $\text{\LaTeX}$ `...` ne sont pas nécessaires.

Code $\text{\LaTeX}$

```
%sortie par défaut
\ListeDiviseurs{150}

%sortie avec \mathscr (si chargé)
\ListeDiviseurs*{300}

%sortie sans libellé
\ListeDiviseurs[AffNom=false]{60}
```

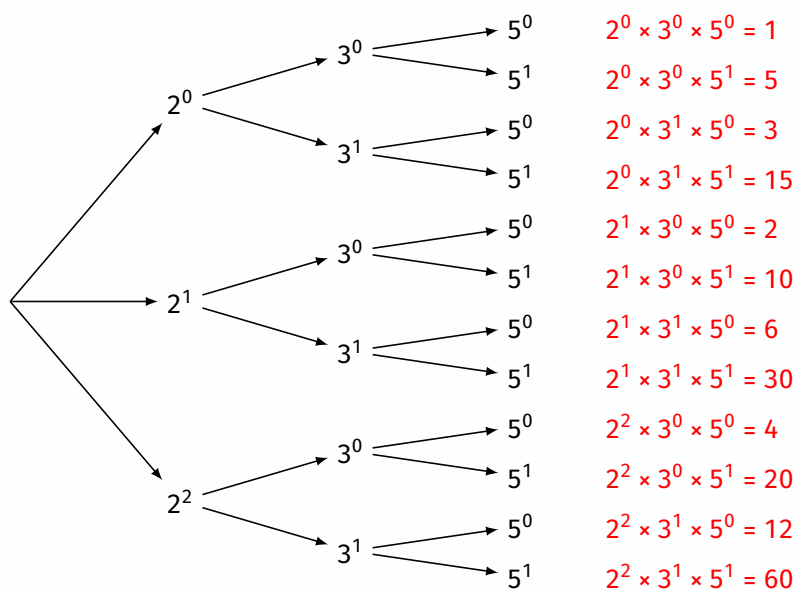
$$D_{150} = \{1; 2; 3; 5; 6; 10; 15; 25; 30; 50; 75; 150\}$$
$$\mathcal{D}_{300} = \{1; 2; 3; 4; 5; 6; 10; 12; 15; 20; 25; 30; 50; 60; 75; 100; 150; 300\}$$
$$\{1; 2; 3; 4; 5; 6; 10; 12; 15; 20; 30; 60\}$$


En ce qui concerne la commande pour l'arbre (créé en TikZ) permettant d'obtenir la liste des diviseurs (je remercie Christophe Poulain pour un *bout de code* que j'ai adapté et pour son aide précieuse) :

- les clés disponible sont :
 - la clé `<EspaceNiveau>` qui est l'espace horizontal (en cm) entre les étages; défaut `<2.25>`
 - la clé `<EspaceFeuille>` qui est l'espace vertical entre les feuilles; défaut `<0.66>`
 - le booléen `<Details>` pour afficher les calculs des diviseurs; défaut `<true>`
 - la clé `<CouleurDetails>` pour la couleur des détails; défaut `<red>`
 - la clé `<Echelle>` pour spécifier une échelle globale (y compris le texte) de la figure; défaut `<1>`
 - le booléen `<Fleches>` pour afficher une flèche sur les branches. défaut `<true>`
- l'argument obligatoire, et entre $\text{\LaTeX}$ `{...}`, est quant lui le nombre, y compris en langage $\text{\LaTeX}$ `xint`.

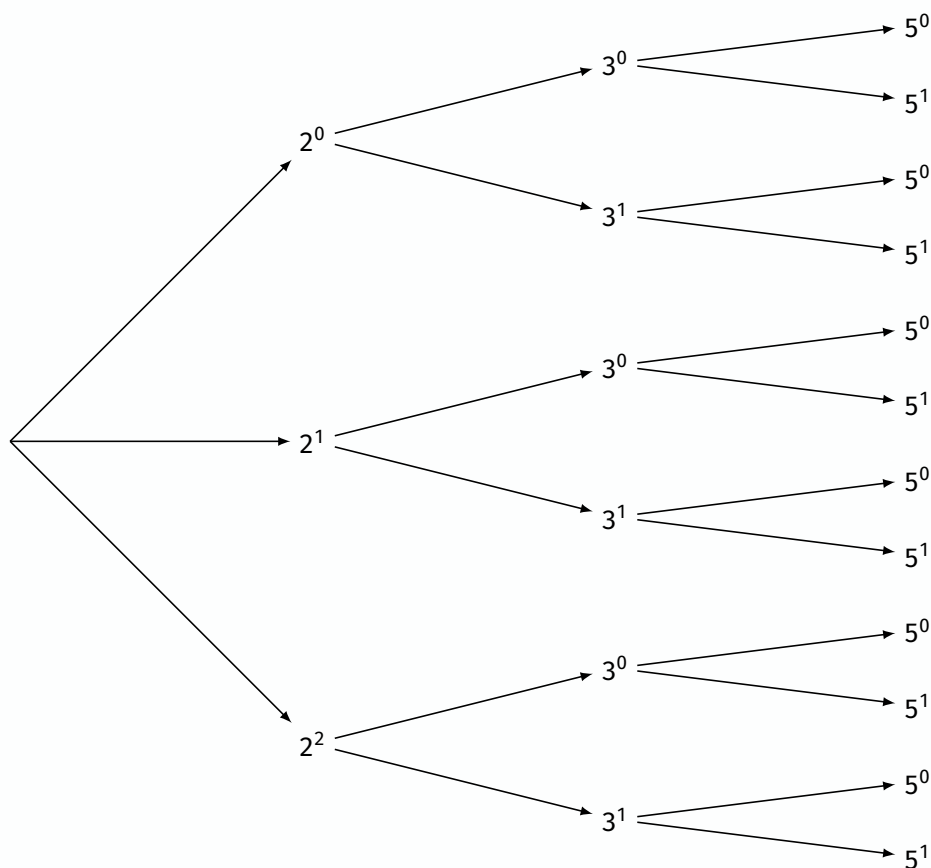
Code $\LaTeX$

```
%sortie par défaut
\ArbreDiviseurs{60}
```

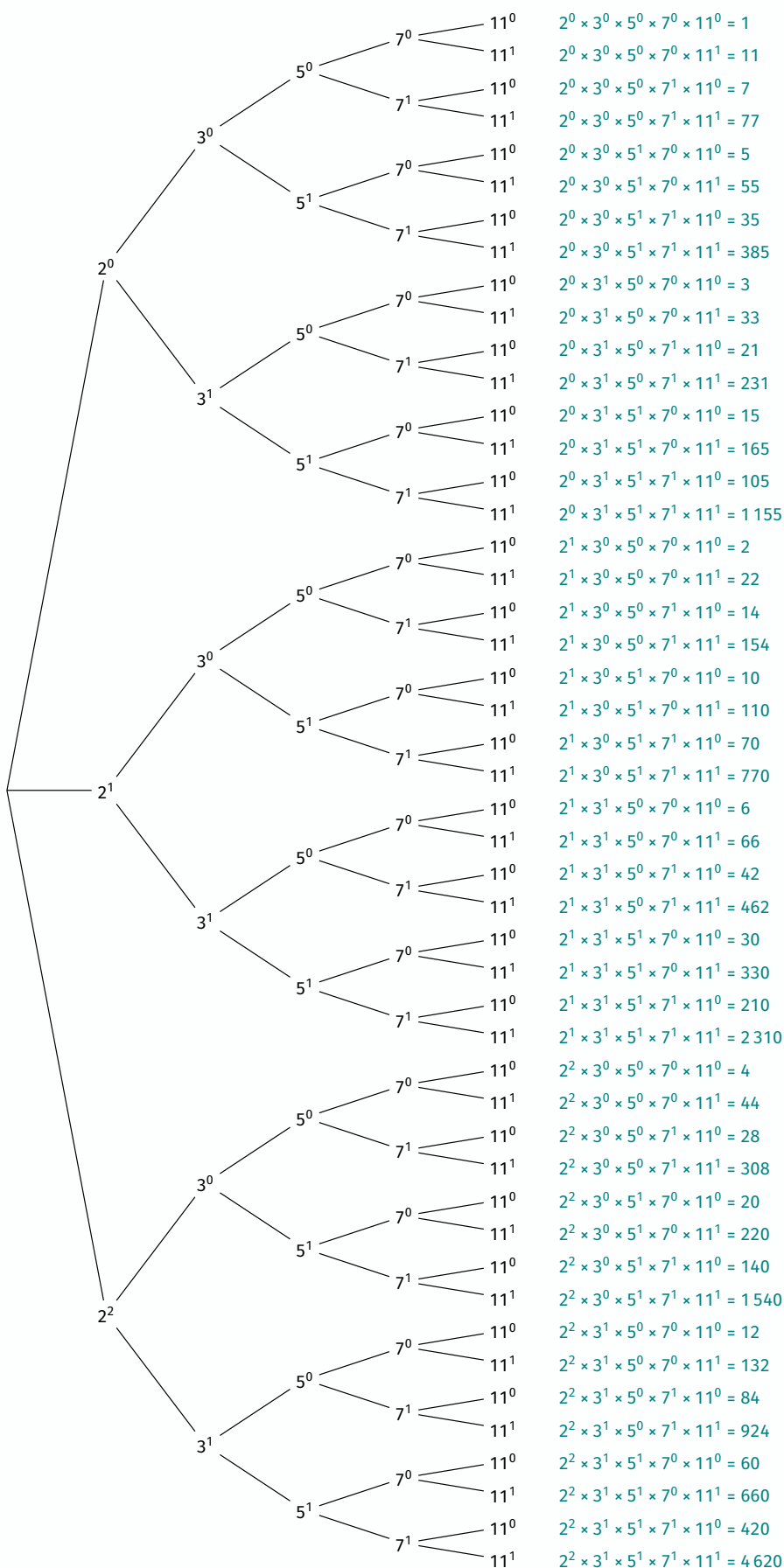


Code $\LaTeX$

```
%sortie personnalisée
\ArbreDiviseurs[EspaceNiveau=4,EspaceFeuille=1,Details=false]{60}
```



```
\ArbreDiviseurs[EspaceNiveau=2,Echelle=0.75,CouleurDetails=teal,Fleches=false]{2^2*3*5*7*11}
```



60.3 Compléments



 L'idée est de proposer des commandes complémentaires pour travailler sur les facteurs premiers, y compris avec de *grands* entiers :

- décomposition en facteurs premiers;
- simplification de fractions.

Le mode mathématique est activé par défaut pour ces commandes.

Code $\LaTeX$

```
\DecompFactPremiers[Clés]{nombre}
% Longue      = false
% PuissanceUn = false
```

Code $\LaTeX$

```
\PresFactPremiers[Clés]{nombre}
% EpaisseurTrait = 0.4pt
% EncadreFin     = false
```

Code $\LaTeX$

```
\SimplFracDecomp[Clés]{num}{denom}
% CouleurCommun = red!90!black
% Barrer        = false
% d             = false (dfrac)
% t             = false (tfrac)
```

Code $\LaTeX$

```
%décomposition, compatible avec des grands entiers
\DecompFactPremiers{2781240}
\DecompFactPremiers[PuissanceUn]{2781240}
\DecompFactPremiers[Longue]{2781240}
```

```
23 × 3 × 5 × 72 × 11 × 43
23 × 31 × 51 × 72 × 111 × 431
2 × 2 × 2 × 3 × 5 × 7 × 7 × 11 × 43
```

Code $\LaTeX$

```
%décomposition, présentation verticale
\PresFactPremiers{2781240}
```

```
2 781 240 | 2
1 390 620 | 2
 695 310 | 2
 347 655 | 3
 115 885 | 5
  23 177 | 7
   3 311 | 7
    473 | 11
     43 | 43
      1 |
```

%simplification de fractions

$\text{\SimplFracDecomp}\{1320\}\{1248\}$

$\text{\SimplFracDecomp}[d,\text{Barrer}]\{1320\}\{1248\}$

$$\frac{1320}{1248} = \frac{2 \times 2 \times 2 \times 3 \times 5 \times 11}{2 \times 2 \times 2 \times 2 \times 2 \times 3 \times 13} = \frac{55}{52}$$

$$\frac{1320}{1248} = \frac{\cancel{2} \times \cancel{2} \times \cancel{2} \times \cancel{3} \times 5 \times 11}{\cancel{2} \times \cancel{2} \times \cancel{2} \times 2 \times 2 \times \cancel{3} \times 13} = \frac{55}{52}$$

61 Chiffrements

61.1 Idées



 3.00c L'idée est de proposer des commandes pour travailler sur des chiffrement/déchiffrements classiques :

- chiffrement de César;
- chiffrement affine $ax + b$ (avec détermination d'un inverse modulo);
- chiffrement de Hill avec une matrice 2×2 .

Code $\LaTeX$

```
%inverse modulo
\InverseModulo(*){a}{modulo}

%Chiffrement de César
\ChiffrementCesar[clés]{message}

%Chiffrement affine
\ChiffrementAffine[clés]{message}

%Chiffrement de Hill
\ChiffrementHill[clés]{message}
```

61.2 Chiffrement de César

Code $\LaTeX$

```
%Chiffrement de César
\ChiffrementCesar[Decal=4]{ABCDEFGH IJKLMNOPQRSTUVWXYZ}
```

EFGHIJKLMNOPQRSTUVWXYZABCD



Les clés disponibles sont :

- la clé **<Decal>** qui spécifie le décalage à appliquer;
- le booléen **<Dechiffr>** pour déchiffrer.

défaut **<5>**

défaut **<false>**

L'argument obligatoire, entre  {...} est le message, avec espace et/ou majuscules et/ou minuscules.

Code $\LaTeX$

```
%Chiffrement de César, décalage de 5
\ChiffrementCesar{TEXTE A CHIFFRER}
```

YJCYJ F HMNKKWJW

Code $\LaTeX$

```
%Chiffrement de César, décalage de 5
\ChiffrementCesar[Decal=7]{TEXTE A CHIFFRER}
```

ALEAL H JOPMMYLY

Code $\LaTeX$

```
%Déchiffrement de César, décalage de 5
\ChiffrementCesar[Dechiffr]{IJHTIJW HJXFW}
```

DECODER CESAR

61.3 Inverse modulo

Code $\LaTeX$

```
%Inverse modulo, version non étoilée, résultat stocké dans \resinvmod  
\InverseModulo{3}{26}\resinvmod
```

9

Code $\LaTeX$

```
%Inverse modulo, version étoilée, avec rédaction (fixée)  
\InverseModulo*{3}{26}
```

On a $\text{PGCD}(3; 26) = 1$. Le PGCD étant égal à 1, on en déduit que 3 admet un inverse modulo 26.
De plus on a $3 \times 9 = 27 \equiv 1 [26]$, donc 9 est l'inverse de 3 modulo 26.



La version étoilée présente une rédaction sommaire, tandis que la version non étoilée détermine l'inverse (éventuelle) et stocke le résultat dans la macro `\resinvmod`.
Les deux arguments, obligatoires et entre `{...}`, correspondent aux entiers, le second étant l'entier *du modulo*.

Code $\LaTeX$

```
\InverseModulo{2}{26}\resinvmod
```

2 n'est pas inversible modulo 26.

Code $\LaTeX$

```
\InverseModulo*{2}{26}
```

On a $\text{PGCD}(2; 26) = 2$. Le PGCD étant différent de 1, on en déduit que 2 n'est pas inversible modulo 26.

61.4 Chiffrement affine



Le principe est ici de travailler sur un chiffrement affine, du type $ax + b [n]$, avec gestion des cas de non possibilité de déchiffrement.
Le code permet de chiffrer et déchiffrer un message constitué des caractères majuscules et/ou minuscules et éventuellement des espaces.

Code $\LaTeX$

```
%Chiffrement affine  
\ChiffrementAffine[a=3,b=12]{Texte a chiffrer}
```

Rydry m shkbblyl



Les clés disponibles sont :

- la clé `<a>` qui spécifie le coefficient a ; défaut `<3>`
- la clé `<b>` qui spécifie le coefficient b ; défaut `<12>`
- le booléen `<Dechiffr>` qui propose le déchiffrement. défaut `<false>`

L'argument obligatoire, entre `{...}` est le message, avec espace et/ou majuscules et/ou minuscules.

Code $\LaTeX$

```
\ChiffrementAffine[a=5,b=17]{BTSSIO}
```

WIDDFJ

Code $\LaTeX$

```
\ChiffrementAffine[Dechiffr,a=5,b=17]{widdfj}
```

btssio

Code $\LaTeX$

```
\ChiffrementAffine[Dechiffr,a=2,b=13]{WIDDFJ}
```

Le message ne peut pas être déchiffré car PGCD(2 ; 26) $\neq$ 1 !

61.5 Chiffrement de Hill



Le principe est ici de travailler sur un chiffrement de Hill, du type $A \times X[n]$, avec A une matrice 2×2 . Le code permet de chiffrer et déchiffrer un message constitué des caractères majuscules et/ou minuscules, complétés éventuellement avec le caractère A pour avoir un nombre pair de caractères. Le code se charge également de gérer les cas particuliers de non possibilité de déchiffrement.

Code $\LaTeX$

```
\ChiffrementHill[Matrice={3,5,1,2}]{ELECTION}
```

PAWITJDO



Les clés disponibles sont :

- la clé **<Matrice>** qui spécifie les coefficients de la matrice 2×2 défaut **<1,2,3,5>**
- le booléen **<Dechiffr>** qui propose le déchiffrement. défaut **<false>**

L'argument obligatoire, entre `{...}` est le message, avec espace et/ou majuscules et/ou minuscules.

Code $\LaTeX$

```
\ChiffrementHill[Matrice={3,5,6,17}]{TEXTEACRYPTER}
```

ZAITMYNPRJZAZY

Code $\LaTeX$

```
\ChiffrementHill[Matrice={3,5,1,2},Dechiffr]{pawitjdo}
```

election

Code $\LaTeX$

```
\ChiffrementHill[Matrice={1,2,3,4},Dechiffr]{PAWITJDO}
```

Le déterminant de la matrice $\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$ (qui vaut -2) n'est pas inversible modulo 26, donc pas de déchiffrement !

Code $\LaTeX$

```
\ChiffrementHill[Matrice={1,2,3,6},Dechiffr]{PAWITJDO}
```

La matrice $\begin{pmatrix} 1 & 2 \\ 3 & 6 \end{pmatrix}$ n'est pas inversible, donc pas de déchiffrement possible !

62 Primorielle

62.1 Idée



L'idée est de proposer de quoi travailler avec les primorielles (produit des nombres premiers inférieurs ou égaux à un entier donné).

Les calculs sont effectués par le package `xint`, et le formatage peut être effectué par le package `siunitx`.

Code $\LaTeX$

```
\Primorielle(*)[clés]{nb}
```

Code $\LaTeX$

```
\Primorielle{9} \\
\Primorielle{17}
```

210
510 510

62.2 Clés



La version *étoilée* ne formate pas les résultats par `siunitx`.

Les **clés** disponibles pour cette commande sont :

- **Enonce** : booléen pour afficher l'énoncé; défaut **false**
- **Complet** : booléen pour afficher la formule complète; défaut **false**
- **Grand** : booléen pour les grands résultats; défaut **false**
- **Sens** : préciser l'ordre pour les détails **d/m**; défaut **m**
- **ChSignif** : nombre de chiffres significatifs pour les grands résultats. défaut **9**

Code $\LaTeX$

```
\Primorielle[Enonce,Complet]{9}
```

9# = $2 \times 3 \times 5 \times 7 = 210$

Code $\LaTeX$

```
\Primorielle[Enonce,Complet,Sens=d]{23}
```

23# = $23 \times 19 \times 17 \times 13 \times 11 \times 7 \times 5 \times 3 \times 2 = 223\,092\,870$

63 Opérations posées

63.1 Idée



`3.02c` L'idée est de proposer une commande poser une opération :

- addition/soustraction/multiplication;
- avec retenues pour l'addition et la soustraction (`3.02d`).



La commande est disponible avec tout compilateur, et seule l'addition est compatible avec l'affichage des retenues.

Les commandes présentées dans la partie projets sont cependant toujours disponibles, pour le moment.

Code $\LaTeX$

```
%opération posée  
\OperationPosee[clés]{opération}
```

Code $\LaTeX$

```
%Addition  
\OperationPosee{9999+851}
```

$$\begin{array}{r} \overset{1}{} \overset{1}{} \overset{1}{} \overset{1}{} \\ 9999 \\ + 851 \\ \hline = 10850 \end{array}$$

Code $\LaTeX$

```
%Soustraction binaire  
\OperationPosee[Base=bin]{10001-1101}
```

$$\begin{array}{r} 10001 \\ - 1101 \\ \hline = 100 \end{array}$$

Code $\LaTeX$

```
%Multiplication hexa  
\OperationPosee[Base=hex]{ACDC*AF4}
```

$$\begin{array}{r} A C D C \\ \times A F 4 \\ \hline 2 B 3 7 0 \\ + A 2 0 E 4 . \\ + 6 C 0 9 8 . . \\ \hline = 7 6 5 5 9 B 0 \end{array}$$

63.2 Clés et options



En ce qui concerne la commande, les **clés**, disponibles entre `[...]`, sont :

- `<Base>` pour spécifier la base (`<dec>` par défaut);
- `<LimiteCapac>` pour fixer une limite de chiffres (`<0>` pour aucune limite par défaut);
- `<SymbDecal>` pour le symbole du décalages des multiplications (`<.>` par défaut);
- `<Offset>` pour l'espacement horizontal entre les chiffres (`<6pt>` par défaut);
- `<CouleurRetenue>` pour la couleur des retenues dans les additions (`<red>` par défaut);
- `<Interm>` := booléen pour afficher les étapes intermédiaires des multiplications (`<true>` par défaut);
- `<AffRetenues>` := booléen pour afficher les retenues des additions/soustractions (`<true>` par défaut);
- `<AffEgal>` := booléen pour afficher le signe = du résultat (`<true>` par défaut).

63.3 Exemples

Code $\LaTeX$

```
%Addition décimale
\OperationPosee{8475+6520}
```

$$\begin{array}{r} \overset{1}{} 8 4 7 5 \\ + 6 5 2 0 \\ \hline = 1 4 9 9 5 \end{array}$$

Code $\LaTeX$

```
%Addition binaire
\OperationPosee[Base=bin]{1111+111} et
\OperationPosee[Base=bin,AffRetenues=false]{1111+111}
```

$$\begin{array}{r} \overset{1}{} \overset{1}{} \overset{1}{} \overset{1}{} 1 1 1 1 \\ + 1 1 1 \\ \hline = 1 0 1 1 0 \end{array} \quad \text{et} \quad \begin{array}{r} 1 1 1 1 \\ + 1 1 1 \\ \hline = 1 0 1 1 0 \end{array}$$

Code $\LaTeX$

```
%Addition hexa
\OperationPosee[Base=hex]{ABC+DE}
```

$$\begin{array}{r} \overset{1}{} A B C \\ + D E \\ \hline = B 9 A \end{array}$$

Code $\LaTeX$

```
%Addition hexa, personnalisée
{\Huge\ttfamily\OperationPosee[Base=hex,AffEgal=false,Offset=0pt]{ABCD+FE}}
```

$$\begin{array}{r} \overset{1}{} \overset{1}{} A B C D \\ + F E \\ \hline A C C B \end{array}$$

Code $\LaTeX$

```
%Addition binaire, limité à 4 bits
\OperationPosee[Base=bin,LimiteCapac=4,Offset=2pt]{1111+111}
```

$$\begin{array}{r} \overset{1}{1} \overset{1}{1} \overset{1}{1} \overset{1}{1} \\ 1111 \\ + \quad 111 \\ \hline = 0110 \end{array}$$

Code $\LaTeX$

```
%Soustraction binaire
\OperationPosee[Base=bin]{11000-111}
```

$$\begin{array}{r} 11\overset{1}{0}\overset{1}{0}\overset{1}{0} \\ - \quad \overset{1}{1}\overset{1}{1}\overset{1}{1} \\ \hline = 10001 \end{array}$$

Code $\LaTeX$

```
%Multiplication binaire
\OperationPosee[Base=bin,LimiteCapac=6]{110011*101}
```

$$\begin{array}{r} 110011 \\ \times 101 \\ \hline 110011 \\ + 000000 \\ + 110011 \\ \hline = 111111 \end{array}$$

Code $\LaTeX$

```
%Mutiplication hexa, symbole de décalage '0'
\textsf{\OperationPosee[Base=hex,SymbDecal=0]{ABCD*FE}}
```

$$\begin{array}{r} A B C D \\ \times F E \\ \hline 9 6 5 3 6 \\ + A 1 1 0 3 0 \\ \hline = A A 7 5 6 6 \end{array}$$

Code $\LaTeX$

```
%Mutiplication hexa
\texttt{\OperationPosee[Base=hex]{ABCD*FAFA}}
```

$$\begin{array}{r} A B C D \\ \times F A F A \\ \hline 6 B 6 0 2 \\ + A 1 1 0 3 \\ + 6 B 6 0 2 \\ + A 1 1 0 3 \\ \hline = A 8 6 D F 8 3 2 \end{array}$$

Code $\LaTeX$

```
%Mutiplication hexa, limité à 4 'chiffres'
\texttt{\OperationPosee[Base=hex,LimiteCapac=4]{ABCD*FAFA}}
```

$$\begin{array}{r} \text{A B C D} \\ \times \quad \text{F A F A} \\ \hline \text{B 6 0 2} \\ + \quad \text{1 1 0 3 .} \\ + \quad \text{B 6 0 2 . .} \\ + \quad \text{1 1 0 3 . . .} \\ \hline = \quad \text{F 8 3 2} \end{array}$$

Code $\LaTeX$

```
%Mutiplication hexa, sans lignes intermédiaires
\textsf{\OperationPosee[Base=hex,Interm=false]{ABCD*FE}}
```

$$\begin{array}{r} \text{A B C D} \\ \times \quad \text{F E} \\ \hline = \text{A A 7 5 6 6} \end{array}$$

Code $\LaTeX$

```
%Soustraction hexa, sans signe '=', sans espacement horizontal
{\Huge\ttfamily\OperationPosee[Base=hex,AffEgal=false]{ABCD-FE}}
```

$$\begin{array}{r} \text{A B C D} \\ - \quad \text{F E} \\ \hline \text{A A C F} \end{array}$$

Thème

ÉCRITURES, SIMPLIFICATIONS

Douzième partie

Écritures, simplifications

64 Simplification sous forme d'une fractions

64.1 Idée



L'idée est d'obtenir une commande pour *simplifier* un calcul sous forme de fraction irréductible.

Code $\LaTeX$

```
\ConversionFraction(*)[option de formatage]{calcul}
```

64.2 Commande et options



Quelques explications sur cette commande :

- `\ConversionFraction[2.5.1]` la version *étoilée* force l'écriture du signe « - » sur le numérateur;
- le premier argument, *optionnel* et entre [...] permet de spécifier un formatage du résultat :
 - `<t>` pour l'affichage de la fraction en mode `tfrac`;
 - `<d>` pour l'affichage de la fraction en mode `dfrac`;
 - `<n>` pour l'affichage de la fraction en mode `nicefrac`;
 - `<dec>` pour l'affichage du résultat en mode décimal (sans arrondi!);
 - `<dec=k>` pour l'affichage du résultat en mode décimal arrondi à 10^{-k} ;
- le second argument, *obligatoire*, est quant à lui, le calcul en syntaxe `xint`.

À noter que la macro est dans un bloc `\ensuremath` donc les `\$...$` ne sont pas nécessaires.

Code $\LaTeX$

```
\ConversionFraction{-10+1/3*(-5/16)} %sortie par défaut
\ConversionFraction*{-10+1/3*(-5/16)} %sortie fraction avec - sur numérateur
\ConversionFraction[d]{-10+1/3*(-5/16)} %sortie en displaystyle
\ConversionFraction[n]{-10+1/3*(-5/16)} %sortie en nicefrac
\ConversionFraction[dec=4]{-10+1/3*(-5/16)} %sortie en décimal arrondi à 0,0001
\ConversionFraction{2+91/7} %entier formaté
\ConversionFraction{111/2145}
\ConversionFraction{111/3}
```

Code $\LaTeX$

```
- \frac{485}{48}
- \frac{485}{48}
- \frac{485}{48}
- \frac{485}{48}
-485/48
-10,104 2
15
\frac{37}{715}
37
```

Code $\LaTeX$

```
\frac{111}{2145}=\ConversionFraction{111/2145}$ \\  
\frac{3}{15}=\ConversionFraction[]{3/15}$ \\  
\tfrac{3}{15}=\ConversionFraction[t]{3/15}$ \\  
\dfrac{3}{15}=\ConversionFraction[d]{3/15}$ \\  
\dfrac{0,42}{0,015}=\ConversionFraction[d]{0.42/0.015}$ \\  
\dfrac{0,41}{0,015}=\ConversionFraction[d]{0.41/0.015}$ \\  
\dfrac{1}{7}-\dfrac{3}{8}=\ConversionFraction[d]{1/7-3/8}$ \\  
\ConversionFraction[d]{1+1/2}$ \\  
\ConversionFraction{0.1/0.7+30/80}$
```

$$\frac{111}{2145} = \frac{37}{715}$$

$$\frac{3}{15} = \frac{1}{5}$$

$$\frac{3}{15} = \frac{1}{5}$$

$$\frac{3}{15} = \frac{1}{5}$$

$$\frac{0,42}{0,015} = 28$$

$$\frac{0,41}{0,015} = \frac{82}{3}$$

$$\frac{1}{7} - \frac{3}{8} = -\frac{13}{56}$$

$$\frac{3}{2}$$

$$\frac{29}{56}$$



A priori le package `xint` permet de s'en sortir pour des calculs « simples », je ne garantis pas que tout calcul ou toute division donne un résultat *satisfaisant* !

65 Écriture d'un trinôme, trinôme aléatoire

65.1 Idée



L'idée est de proposer une commande pour écrire, sous forme développée réduite, un trinôme en fonction de ses coefficients a , b et c (avec $a \neq 0$), avec la gestion des coefficients nuls ou égaux à ± 1 .

En combinant avec le package `\xfp` et fonction de générateur d'entiers aléatoires, on peut de ce fait proposer une commande pour générer aléatoirement des trinômes à coefficients entiers (pour des fiches d'exercices par exemple).

L'affichage des monômes est géré par le package `\siunitx` et le tout est dans un environnement `\ensuremath`.

Code $\LaTeX$

```
\EcritureTrinome[options]{a}{b}{c}
```

Code $\LaTeX$

```
\EcritureTrinome{1}{7}{0} \\
\EcritureTrinome{1.5}{7.3}{2.56} \\
\EcritureTrinome{-1}{0}{12} \\
\EcritureTrinome{-1}{-5}{0}
```

$x^2 + 7x$
 $1,5x^2 + 7,3x + 2,56$
 $-x^2 + 12$
 $-x^2 - 5x$

65.2 Clés et options



Quelques clés et options sont disponibles :

- la clé booléenne `\Alea` pour autoriser les coefficients aléatoires; défaut `\false`
- la clé booléenne `\Anegatif` pour autoriser a à être négatif. défaut `\true`



La clé `\Alea` va modifier la manière de saisir les coefficients, il suffira dans ce cas de préciser les bornes, sous la forme `\valmin, \valmax`, de chacun des coefficients. C'est ensuite le package `\xfp` qui va se charger de générer les coefficients.

Code $\LaTeX$

```
Avec $a$ entre 1 et 5 (et signe aléatoire) puis $b$ entre $-5$ et 5 puis $c$ entre $-10$ et 20 : \\
$f(x)=\EcritureTrinome[Alea]{1,5}{-5,5}{-10,10}$ \\
$g(x)=\EcritureTrinome[Alea]{1,5}{-5,5}{-10,10}$ \\
$h(x)=\EcritureTrinome[Alea]{1,5}{-5,5}{-10,10}$ \\
Avec $a$ entre 1 et 10 (forcément positif) puis $b$ entre $-2$ et 2 puis $c$ entre 0 et 4 : \\
\EcritureTrinome[Alea,Anegatif=false]{1,10}{-2,2}{0,4} \\
\EcritureTrinome[Alea,Anegatif=false]{1,10}{-2,2}{0,4} \\
\EcritureTrinome[Alea,Anegatif=false]{1,10}{-2,2}{0,4}
```

Avec a entre 1 et 5 (et signe aléatoire) puis b entre -5 et 5 puis c entre -10 et 20 :

$f(x) = 4x^2 + 4$
 $g(x) = x^2 - 3x + 6$
 $h(x) = 4x^2 + 4x + 8$

Avec a entre 1 et 10 (forcément positif) puis b entre -2 et 2 puis c entre 0 et 4 :

$8x^2 + x + 3$
 $7x^2 + 1$
 $4x^2 + x + 3$

66 Simplification de racines

66.1 Idée



2.1.0 L'idée est de proposer une commande pour simplifier *automatiquement* une racine carrée, sous la forme $\frac{a\sqrt{b}}{c}$ avec $\frac{a}{c}$ irréductible et b le « plus petit possible ».

Code $\LaTeX$

```
\SimplificationRacine{expression ou calcul}
```

Code $\LaTeX$

```
\SimplificationRacine{48} \\\SimplificationRacine{100/34}\\\SimplificationRacine{99999} \\\SimplificationRacine{1500*0.31*(1-0.31)}\\
```

$$\begin{array}{l} 4\sqrt{3} \\ \frac{5\sqrt{34}}{17} \\ 3\sqrt{11111} \\ \frac{3\sqrt{3565}}{10} \end{array}$$



C'est – comme souvent – le package `\xint` qui s'occupe en interne des calculs, et qui devrait donner des résultats satisfaisants dans la majorité des cas (attention aux *grands nombres*...)

La commande ne fait pas office de *calculatrice*, elle ne permet *que* de simplifier *une* racine carrée (donc transformer si besoin!).

66.2 Exemples

Code $\LaTeX$

```
%Simplification d'un module de complexe
$\left| 4+6\text{i}\right| = \sqrt{4^2+6^2} =
\rightarrow \sqrt{\xinteval{4**2+6**2}}=\SimplificationRacine{4**2+6**2}$

%Simplification n°1
$\frac{1}{\sqrt{6}}=\left(\sqrt{\frac{1}{6}}\right)=\SimplificationRacine{1/6}$

%Simplification n°2
$\frac{42}{\sqrt{5}}=\left(\sqrt{\frac{42^2}{5}}\right)=\SimplificationRacine{(42*42)/5}$

%Écart-type d'une loi binomiale
$\sqrt{\num{150}\times\num{0.35}\times(1-\num{0.35})} =
\rightarrow \displaystyle\SimplificationRacine{150*0.35*(1-0.35)}$
```

$$\begin{array}{l} |4 + 6i| = \sqrt{4^2 + 6^2} = \sqrt{52} = 2\sqrt{13} \\ \frac{1}{\sqrt{6}} = \left(\sqrt{\frac{1}{6}}\right) = \frac{\sqrt{6}}{6} \\ \frac{42}{\sqrt{5}} = \left(\sqrt{\frac{42^2}{5}}\right) = \frac{42\sqrt{5}}{5} \\ \sqrt{150 \times 0,35 \times (1 - 0,35)} = \frac{\sqrt{546}}{4} \end{array}$$

67 Mesure principale d'un angle

67.1 Idée



2.1.2 L'idée est de proposer (sur une suggestion de Marylyne Vignal) une commande pour déterminer la mesure principale d'un angle en radian.

Code $\text{\LaTeX}$

```
\MesurePrincipale[booléens]{angle} %dans un mode mathématique
```



La commande est à insérer dans un environnement mathématique, via $\text{\$...\$}$ ou $\text{\[...\]}$.
L'angle est donné sous forme *explicite* avec la chaîne $\text{\pi}$.

67.2 Exemples



Pour cette commande :

- le booléen $\langle d \rangle$ permet de forcer l'affichage en $\text{\displaystyle}$; défaut $\langle false \rangle$
- le booléen $\langle Crochets \rangle$ permet d'afficher le *modulo* entre crochets (sinon parenthèses); défaut $\langle false \rangle$
- **2.6.0** le booléen $\langle Brut \rangle$ pour afficher uniquement la mesure principale; défaut $\langle false \rangle$
- l'argument *obligatoire* est en écriture *en ligne*.

Code $\text{\LaTeX}$

```
\$ \MesurePrincipale[d]{54\pi/7}$  
\$ \MesurePrincipale[d]{-128\pi/15}$  
\$ \MesurePrincipale{3\pi/2}$  
\$ \MesurePrincipale[Crochets]{5\pi/2}$  
\$ \MesurePrincipale{-13\pi}$  
\$ \MesurePrincipale{28\pi}$  
\$ \MesurePrincipale[d]{14\pi/4}$  
\$ \MesurePrincipale[Crochets]{14\pi/7}$  
\$ \dfrac{121\pi}{12} = \MesurePrincipale[Brut]{121\pi/12}$ à  $2\pi$  près
```

Code $\text{\LaTeX}$

$$\frac{54\pi}{7} = \frac{-2\pi}{7} (2\pi)$$
$$\frac{-128\pi}{15} = \frac{-8\pi}{15} (2\pi)$$
$$\frac{3\pi}{2} = \frac{-\pi}{2} (2\pi)$$
$$\frac{5\pi}{2} = \frac{\pi}{2} [2\pi]$$
$$-13\pi = \pi (2\pi)$$
$$28\pi = 0 (2\pi)$$
$$\frac{14\pi}{4} = \frac{-\pi}{2} (2\pi)$$
$$\frac{14\pi}{7} = 0 [2\pi]$$
$$\frac{121\pi}{12} = \frac{\pi}{12} \text{ à } 2\pi \text{ près}$$

68 Lignes trigonométriques

68.1 Idée



2.6.0 L'idée est de proposer pour déterminer les lignes trigonométriques (cos, sin et tan) d'angles classiques, formés des « π » et « π sur 2 ; 3 ; 4 ; 5 ; 6 ; 8 ; 10 ; 12 ».

La commande détermine – et affiche si demandée la réduction – et la valeur exacte de la ligne trigonométrique demandée.

Code $\LaTeX$

```
\LigneTrigo(*)[booléens]{cos/sin/tan}(angle)
```

68.2 Commande



Pour cette commande :

- la version *étoilée* n'affiche pas l'angle initial;
- le booléen **<d>** permet de forcer l'affichage en `\displaystyle`; défaut **<false>**
- le booléen **<Etapes>** permet d'afficher la réduction avant le résultat; défaut **<false>**
- le premier argument *obligatoire*, entre `{...}` est le type de calcul demandé, parmi **<cos / sin / tan>**;
- le second argument *obligatoire*, entre `(...)` est l'angle, donné en ligne, avec `\pi`.

Code $\LaTeX$

```
$_\LigneTrigo{cos}(56\pi/3)$ et $_\LigneTrigo{sin}(56\pi/3)$ et $_\LigneTrigo{tan}(56\pi/3)$
```

$$-\frac{1}{2} \text{ et } \frac{\sqrt{3}}{2} \text{ et } -\sqrt{3}$$

Code $\LaTeX$

```
$_\LigneTrigo[d,Etapes]{cos}(56\pi/3)$ et $_\LigneTrigo[d,Etapes]{sin}(56\pi/3)$
```

$$\cos\left(\frac{56\pi}{3}\right) = \cos\left(\frac{2\pi}{3}\right) = -\frac{1}{2} \text{ et } \sin\left(\frac{56\pi}{3}\right) = \sin\left(\frac{2\pi}{3}\right) = \frac{\sqrt{3}}{2}$$

Code $\LaTeX$

```
$_\LigneTrigo*[d,Etapes]{cos}(2\pi/3)$ et $_\LigneTrigo*[d,Etapes]{sin}(2\pi/3)$
```

$$\cos\left(\frac{2\pi}{3}\right) = -\frac{1}{2} \text{ et } \sin\left(\frac{2\pi}{3}\right) = \frac{\sqrt{3}}{2}$$

Code $\LaTeX$

```
$_\LigneTrigo[d,Etapes]{cos}(146\pi)$ et $_\LigneTrigo[d,Etapes]{sin}(146\pi)$
```

$$\cos(146\pi) = \cos(0) = 1 \text{ et } \sin(146\pi) = \sin(0) = 0$$

Code $\LaTeX$

```
$_\LigneTrigo[d,Etapes]{cos}(-551\pi/12)$ et $_\LigneTrigo[d,Etapes]{sin}(-551\pi/12)$
```

$$\cos\left(\frac{-551\pi}{12}\right) = \cos\left(\frac{\pi}{12}\right) = \frac{\sqrt{6} + \sqrt{2}}{4} \text{ et } \sin\left(\frac{-551\pi}{12}\right) = \sin\left(\frac{\pi}{12}\right) = \frac{\sqrt{6} - \sqrt{2}}{4}$$

Code $\LaTeX$

```
 $\backslash$ LigneTrigo[d,Etapes]{cos}(447pi/8)$ et  $\backslash$ LigneTrigo[d,Etapes]{sin}(447pi/8)$
```

$$\cos\left(\frac{447\pi}{8}\right) = \cos\left(\frac{-\pi}{8}\right) = \frac{\sqrt{2+\sqrt{2}}}{2} \text{ et } \sin\left(\frac{447\pi}{8}\right) = \sin\left(\frac{-\pi}{8}\right) = -\frac{\sqrt{2-\sqrt{2}}}{2}$$

Code $\LaTeX$

```
 $\backslash$ LigneTrigo*[d,Etapes]{cos}(-pi/8)$ et  $\backslash$ LigneTrigo*[d,Etapes]{sin}(-pi/8)$
```

$$\cos\left(\frac{-\pi}{8}\right) = \frac{\sqrt{2+\sqrt{2}}}{2} \text{ et } \sin\left(\frac{-\pi}{8}\right) = -\frac{\sqrt{2-\sqrt{2}}}{2}$$

Code $\LaTeX$

```
 $\backslash$ LigneTrigo[d,Etapes]{cos}(-595pi/12)$ et  $\backslash$ LigneTrigo[d,Etapes]{sin}(-595pi/12)$ et  
↪  $\backslash$ LigneTrigo[d,Etapes]{tan}(-595pi/12)$
```

$$\cos\left(\frac{-595\pi}{12}\right) = \cos\left(\frac{5\pi}{12}\right) = \frac{\sqrt{6}-\sqrt{2}}{4} \text{ et } \sin\left(\frac{-595\pi}{12}\right) = \sin\left(\frac{5\pi}{12}\right) = \frac{\sqrt{6}+\sqrt{2}}{4} \text{ et } \tan\left(\frac{-595\pi}{12}\right) = \tan\left(\frac{5\pi}{12}\right) = 2+\sqrt{3}$$

Code $\LaTeX$

```
 $\backslash$ LigneTrigo[d,Etapes]{cos}(33pi/10)$ et  $\backslash$ LigneTrigo[d,Etapes]{sin}(33pi/10)$\\  
 $\backslash$ LigneTrigo[d,Etapes]{tan}(33pi/10)$
```

$$\cos\left(\frac{33\pi}{10}\right) = \cos\left(\frac{-7\pi}{10}\right) = -\frac{\sqrt{10-2\sqrt{5}}}{4} \text{ et } \sin\left(\frac{33\pi}{10}\right) = \sin\left(\frac{-7\pi}{10}\right) = -\frac{1+\sqrt{5}}{4}$$

$$\tan\left(\frac{33\pi}{10}\right) = \tan\left(\frac{-7\pi}{10}\right) = \frac{\sqrt{25+10\sqrt{5}}}{5}$$

Code $\LaTeX$

```
 $\backslash$ LigneTrigo[d,Etapes]{cos}(-14pi/5)$ et  $\backslash$ LigneTrigo[d,Etapes]{sin}(-14pi/5)$\\  
 $\backslash$ LigneTrigo[d,Etapes]{tan}(-14pi/5)$
```

$$\cos\left(\frac{-14\pi}{5}\right) = \cos\left(\frac{-4\pi}{5}\right) = \frac{-1-\sqrt{5}}{4} \text{ et } \sin\left(\frac{-14\pi}{5}\right) = \sin\left(\frac{-4\pi}{5}\right) = -\frac{\sqrt{10-2\sqrt{5}}}{4}$$

$$\tan\left(\frac{-14\pi}{5}\right) = \tan\left(\frac{-4\pi}{5}\right) = \sqrt{5-2\sqrt{5}}$$

68.3 Valeurs disponibles



Les valeurs disponibles sont :

angle	0	$\pi/6$	$\pi/4$	$\pi/3$	$\pi/2$	$2\pi/3$	$3\pi/4$	$5\pi/6$	π
cos	1	$\frac{\sqrt{3}}{2}$	$\frac{\sqrt{2}}{2}$	$\frac{1}{2}$	0	$-\frac{1}{2}$	$-\frac{\sqrt{2}}{2}$	$-\frac{\sqrt{3}}{2}$	-1
sin	0	$\frac{1}{2}$	$\frac{\sqrt{2}}{2}$	$\frac{\sqrt{3}}{2}$	1	$\frac{\sqrt{3}}{2}$	$\frac{\sqrt{2}}{2}$	$\frac{1}{2}$	0
tan	0	$\frac{\sqrt{3}}{3}$	1	$\sqrt{3}$		$-\sqrt{3}$	-1	$-\frac{\sqrt{3}}{3}$	0

angle		$-\pi/6$	$-\pi/4$	$-\pi/3$	$-\pi/2$	$-2\pi/3$	$-3\pi/4$	$-5\pi/6$	
cos		$\frac{\sqrt{3}}{2}$	$\frac{\sqrt{2}}{2}$	$\frac{1}{2}$	0	$-\frac{1}{2}$	$-\frac{\sqrt{2}}{2}$	$-\frac{\sqrt{3}}{2}$	
sin		$-\frac{1}{2}$	$-\frac{\sqrt{2}}{2}$	$-\frac{\sqrt{3}}{2}$	-1	$-\frac{\sqrt{3}}{2}$	$-\frac{\sqrt{2}}{2}$	$-\frac{1}{2}$	
tan		$-\frac{\sqrt{3}}{3}$	-1	$-\sqrt{3}$		$\sqrt{3}$	1	$\frac{\sqrt{3}}{3}$	

angle	$\pi/8$	$3\pi/8$	$5\pi/8$	$7\pi/8$	$\pi/12$	$5\pi/12$	$7\pi/12$	$11\pi/12$
cos	$\frac{\sqrt{2+\sqrt{2}}}{2}$	$\frac{\sqrt{2-\sqrt{2}}}{2}$	$-\frac{\sqrt{2-\sqrt{2}}}{2}$	$-\frac{\sqrt{2+\sqrt{2}}}{2}$	$\frac{\sqrt{6+\sqrt{2}}}{4}$	$\frac{\sqrt{6-\sqrt{2}}}{4}$	$-\frac{\sqrt{6+\sqrt{2}}}{4}$	$-\frac{\sqrt{6-\sqrt{2}}}{4}$
sin	$\frac{\sqrt{2-\sqrt{2}}}{2}$	$\frac{\sqrt{2+\sqrt{2}}}{2}$	$\frac{\sqrt{2+\sqrt{2}}}{2}$	$\frac{\sqrt{2-\sqrt{2}}}{2}$	$\frac{\sqrt{6-\sqrt{2}}}{4}$	$\frac{\sqrt{6+\sqrt{2}}}{4}$	$\frac{\sqrt{6+\sqrt{2}}}{4}$	$\frac{\sqrt{6-\sqrt{2}}}{4}$
tan	$-1 + \sqrt{2}$	$1 + \sqrt{2}$	$-1 - \sqrt{2}$	$1 - \sqrt{2}$	$2 - \sqrt{3}$	$2 + \sqrt{3}$	$-2 - \sqrt{3}$	$-2 + \sqrt{3}$

angle	$-\pi/8$	$-3\pi/8$	$-5\pi/8$	$-7\pi/8$	$-\pi/12$	$-5\pi/12$	$-7\pi/12$	$-11\pi/12$
cos	$\frac{\sqrt{2+\sqrt{2}}}{2}$	$\frac{\sqrt{2-\sqrt{2}}}{2}$	$-\frac{\sqrt{2-\sqrt{2}}}{2}$	$-\frac{\sqrt{2+\sqrt{2}}}{2}$	$\frac{\sqrt{6+\sqrt{2}}}{4}$	$\frac{\sqrt{6-\sqrt{2}}}{4}$	$-\frac{\sqrt{6+\sqrt{2}}}{4}$	$-\frac{\sqrt{6-\sqrt{2}}}{4}$
sin	$-\frac{\sqrt{2-\sqrt{2}}}{2}$	$-\frac{\sqrt{2+\sqrt{2}}}{2}$	$-\frac{\sqrt{2+\sqrt{2}}}{2}$	$-\frac{\sqrt{2-\sqrt{2}}}{2}$	$-\frac{\sqrt{6+\sqrt{2}}}{4}$	$-\frac{\sqrt{6-\sqrt{2}}}{4}$	$-\frac{\sqrt{6-\sqrt{2}}}{4}$	$-\frac{\sqrt{6+\sqrt{2}}}{4}$
tan	$1 - \sqrt{2}$	$-1 - \sqrt{2}$	$1 + \sqrt{2}$	$-1 + \sqrt{2}$	$-2 + \sqrt{3}$	$-2 - \sqrt{3}$	$2 + \sqrt{3}$	$2 - \sqrt{3}$

angle	$-4\pi/5$	$-3\pi/5$	$-2\pi/5$	$-\pi/5$	$\pi/5$	$2\pi/5$	$3\pi/5$	$4\pi/5$
cos	$\frac{-1-\sqrt{5}}{4}$	$\frac{1-\sqrt{5}}{4}$	$\frac{-1+\sqrt{5}}{4}$	$\frac{1+\sqrt{5}}{4}$	$\frac{1+\sqrt{5}}{4}$	$\frac{-1+\sqrt{5}}{4}$	$\frac{1-\sqrt{5}}{4}$	$\frac{-1-\sqrt{5}}{4}$
sin	$-\frac{\sqrt{10-2\sqrt{5}}}{4}$	$-\frac{\sqrt{10+2\sqrt{5}}}{4}$	$-\frac{\sqrt{10+2\sqrt{5}}}{4}$	$-\frac{\sqrt{10-2\sqrt{5}}}{4}$	$\frac{\sqrt{10-2\sqrt{5}}}{4}$	$\frac{\sqrt{10+2\sqrt{5}}}{4}$	$\frac{\sqrt{10+2\sqrt{5}}}{4}$	$\frac{\sqrt{10-2\sqrt{5}}}{4}$
tan	$\sqrt{5-2\sqrt{5}}$	$\sqrt{5+2\sqrt{5}}$	$-\sqrt{5+2\sqrt{5}}$	$-\sqrt{5-2\sqrt{5}}$	$\sqrt{5-2\sqrt{5}}$	$\sqrt{5+2\sqrt{5}}$	$-\sqrt{5+2\sqrt{5}}$	$-\sqrt{5-2\sqrt{5}}$

angle	$-9\pi/10$	$-7\pi/10$	$-3\pi/10$	$-\pi/10$	$\pi/10$	$3\pi/10$	$7\pi/10$	$9\pi/10$
cos	$-\frac{\sqrt{10+2\sqrt{5}}}{4}$	$-\frac{\sqrt{10-2\sqrt{5}}}{4}$	$\frac{\sqrt{10-2\sqrt{5}}}{4}$	$\frac{\sqrt{10+2\sqrt{5}}}{4}$	$\frac{\sqrt{10+2\sqrt{5}}}{4}$	$\frac{\sqrt{10-2\sqrt{5}}}{4}$	$-\frac{\sqrt{10-2\sqrt{5}}}{4}$	$-\frac{\sqrt{10+2\sqrt{5}}}{4}$
sin	$\frac{1-\sqrt{5}}{4}$	$-\frac{1+\sqrt{5}}{4}$	$-\frac{1+\sqrt{5}}{4}$	$\frac{1-\sqrt{5}}{4}$	$\frac{-1+\sqrt{5}}{4}$	$\frac{1+\sqrt{5}}{4}$	$\frac{1+\sqrt{5}}{4}$	$\frac{-1+\sqrt{5}}{4}$
tan	$\frac{\sqrt{25-10\sqrt{5}}}{5}$	$\frac{\sqrt{25+10\sqrt{5}}}{5}$	$-\frac{\sqrt{25+10\sqrt{5}}}{5}$	$-\frac{\sqrt{25-10\sqrt{5}}}{5}$	$\frac{\sqrt{25-10\sqrt{5}}}{5}$	$\frac{\sqrt{25+10\sqrt{5}}}{5}$	$-\frac{\sqrt{25+10\sqrt{5}}}{5}$	$-\frac{\sqrt{25-10\sqrt{5}}}{5}$

69 Écriture sous forme de fraction irréductible d'un décimal périodique

69.1 Idées



3.00f L'idée est de proposer une commande pour travailler sur l'écriture fractionnaire d'un nombre à écriture décimale périodique.

À partir de l'écriture sous la forme « $A.B\overline{C}$ », le code se charge de :

- faire les calculs pour obtenir le numérateur et le dénominateur de la fraction (sans simplification);
- ou de présenter le résultat de manière basique;
- ou de rédiger complètement la résolution.

Code $\LaTeX$

```
%commande générale
\FractionPeriode{clés}{partie avant la période}{période}
%résultats bruts, pour utilisation 'externe'
\FractionPeriode{partie avant la période}{période}
%présentation simple := nombre + fraction brute + fraction simplifiée
\FractionPeriode[Simple]{partie avant la période}{période}
%présentation complète := résolution
\FractionPeriode[Solution]{partie avant la période}{période}
```

69.2 Options et clés



Les clés disponibles pour cette commande sont :

- le booléen **<Brut>** qui permet de stocker le numérateur et dénominateur :
 - dans les variables `\FracPerNum` et `\FracPerDenom` pour la version *complète*;
 - dans les variables `\FracPerNumSimpl` et `\FracPerDenomSimpl` pour la version *irréductible*;
- le booléen **<d>** pour forcer l'afficher avec `\displaystyle`;
- la clé **<Inconnue>** qui gère l'inconnue dans la rédaction;
- le booléen **<Solution>** qui affiche la résolution *complète*;
- le booléen **<Simple>** qui affiche uniquement la *conclusion*;
- le booléen **<Enonce>** (en mode **<Simple=true>**) qui affiche le nombre de départ (formaté).

défaut **<true>**

défaut **<true>**

défaut **<x>**

défaut **<false>**

défaut **<false>**

défaut **<false>**

À noter que la commande gère automatiquement le mode mathématique éventuel.

Code $\LaTeX$

```
%version brute, pour extraire numérateur et dénominateur puis mise en forme manuelle
\FractionPeriode{45.1}{23}Numérateur : \FracPerNum{} et \FracPerNumSimpl\par Dénominateur =
  \FracPerDenom{} et \FracPerDenomSimpl\par
\medskip
On a  $\frac{45.1}{\overline{23}} = \frac{\num{\FracPerNum}}{\num{\FracPerDenom}} =$ 
 $\frac{\num{\FracPerNumSimpl}}{\num{\FracPerDenomSimpl}} =$ 
 $\frac{\num{\FracPerNumSimpl}}{\num{\FracPerDenomSimpl}}.$ 
```

Numérateur : 44672 et 22336

Dénominateur = 990 et 495

$$\text{On a } 45,1\overline{23} = \frac{44\,672}{990} = \frac{22\,336}{495} = 22\,336/495.$$

Code $\LaTeX$

```
\FractionPeriode[Simple]{45.1}{23}
```

$$45,1\overline{23} = \frac{44\,672}{990} = \frac{22\,336}{495}$$

Code $\LaTeX$

```
\FractionPeriode[Solution]{45.1}{23}
```

On note $x = 45,1\overline{23}$.

On ramène la période près de la virgule en multipliant par 10^1 :

$$10^1 \times x = 451,\overline{23} \quad (1)$$

On décale la période avant la virgule, en multipliant l'égalité (1) par 10^2 :

$$10^2 \times 10^1 \times x = 10^2 \times 451,\overline{23} \Rightarrow 10^3 \times x = 45\,123,\overline{23} \quad (2)$$

On soustrait les deux égalités, (2) - (1), ce qui permet d'enlever la partie décimale :

$$\begin{aligned} 10^3 \times x - 10^1 \times x &= 45\,123,\overline{23} - 451,\overline{23} \Rightarrow (10^3 - 10^1) \times x = 45\,123 - 451 \\ &\Rightarrow 990 \times x = 44\,672 \\ &\Rightarrow x = \frac{44\,672}{990} \end{aligned}$$

Ainsi on a $45,1\overline{23} = \frac{44\,672}{990} = \frac{22\,336}{495}$.

Code $\LaTeX$

```
\FractionPeriode[Simple]{0.}{4}
```

$$0,\overline{4} = \frac{4}{9}$$

Code $\LaTeX$

```
\FractionPeriode[Solution,Inconnue=n]{154.99}{174}
```

On note $n = 154,99\overline{174}$.

On ramène la période près de la virgule en multipliant par 10^2 :

$$10^2 \times n = 15\,499,\overline{174} \quad (1)$$

On décale la période avant la virgule, en multipliant l'égalité (1) par 10^3 :

$$10^3 \times 10^2 \times n = 10^3 \times 15\,499,\overline{174} \Rightarrow 10^5 \times n = 15\,499\,174,\overline{174} \quad (2)$$

On soustrait les deux égalités, (2) - (1), ce qui permet d'enlever la partie décimale :

$$\begin{aligned} 10^5 \times n - 10^2 \times n &= 15\,499\,174,\overline{174} - 15\,499,\overline{174} \Rightarrow (10^5 - 10^2) \times n = 15\,499\,174 - 15\,499 \\ &\Rightarrow 99\,900 \times n = 15\,483\,675 \\ &\Rightarrow n = \frac{15\,483\,675}{99\,900} \end{aligned}$$

Ainsi on a $154,99\overline{174} = \frac{15\,483\,675}{99\,900} = \frac{206\,449}{1\,332}$.

Thème

JEUX ET RÉCRÉATIONS

Treizième partie

Jeux et récréations

70 SudoMaths, en TikZ

70.1 Introduction



L'idée est de *proposer* un environnement `TikZ`, une commande permettant de tracer des grilles de SudoMaths.

L'environnement créé, lié à `TikZ`, trace la grille de SudoMaths (avec les blocs démarqués), et peut la remplir avec une liste d'éléments.

Code $\text{\LaTeX}$

```
%grille classique non remplie, avec légendes H/V, {} nécessaires pour préciser que les cases seront  
↪ "vides"  
\SudoMaths{}
```

Code $\text{\LaTeX}$

```
\SudoMaths{}
```

	a	b	c	d	e	f	g	h	i
A									
B									
C									
D									
E									
F									
G									
H									
I									



La commande `\SudoMaths` crée donc la grille (remplie ou non), dans un environnement `TikZ`, c'est c'est tout!

On peut également utiliser l'environnement `\EnvSudoMaths` dans lequel on peut rajouter du code `TikZ`!

Code $\text{\LaTeX}$

```
%grille "toute seule"  
\SudoMaths[clés]{liste}  
  
%grille avec ajout de code  
\begin{EnvSudoMaths}[clés]{grille}  
  %commandes tikz  
\end{EnvSudoMaths}
```

70.2 Clés et options



Quelques **clés** sont disponibles pour cette commande :

- la clé **<Epaisseur>** pour gérer l'épaisseur des traits épais; défaut **<1.5pt>**
- la clé **<Epaisseur>** pour gérer l'épaisseur des traits fins; défaut **<0.5pt>**
- la clé **<Unite>** qui est l'unité graphique de la figure; défaut **<1cm>**
- la clé **<CouleurCase>** pour la couleur (éventuelles) des cases; défaut **<cyan !50>**
- la clé **<CouleurTexte>** pour gérer la couleur du label des cases; défaut **<blue>**
- la clé **<NbCol>** qui est le nombre de colonnes; défaut **<9>**
- la clé **<NbSubCol>** qui est le nombre de sous-colonnes; défaut **<3>**
- la clé **<NbLig>** qui est le nombre de lignes; défaut **<9>**
- la clé **<NbSubLig>** qui est le nombre de sous-colonnes; défaut **<3>**
- la clé **<Police>** qui formate le label des cases; défaut **<\normalfont\normalsize>**
- le booléen **<Legendes>** qui affiche ou non les légendes (H et V) des cases; défaut **<true>**
- la clé **<PoliceLeg>** qui formate le label des légendes; défaut **<\normalfont\normalsize>**
- la clé **<ListeLegV>** qui est la liste de la légende verticale; défaut **<ABCD...WXYZ>**
- la clé **<ListeLegH>** qui est la liste de la légende horizontale; défaut **<abcd...wxyz>**
- la clé **<DecalLegende>** qui est le décalage de la légende par rapport à la grille. défaut **<0.45>**



La liste éventuelle des éléments à rentrer dans le tableau est traitée par le package `listofitems`, et se présente sous la forme suivante : `/ / / ... / / $ / / / ... / / $ ... $ / / / ... / /`

Il peut donc être intéressant de *déclarer* la liste au préalable pour simplifier la saisie de la commande!



La **<CouleurCase>** est gérée – en interne – par le caractère `*` qui permet de préciser qu'on veut que la case soit coloriée.

Code $\text{\LaTeX}$

```
%grille 6x6 avec blocs 2x3, avec coloration de cases (présentée sous forme de "cases")
\def\grilleSuMa{%
  (a)* / (b)* /      /      / (c)* / (d)* $%
  (e)* /      /      / (f)* / (g)* / (h)* $%
      /      / (i)* /      /      / (j)* $%
      /      / (k)* /      / (l)* / (m)* $%
  (n)* /      / (o)* /      /      / (p)* $%
      /      /      / (q)* /      /      $%
}

\SudoMaths[Unite=0.75cm,NbCol=6,NbSubCol=2,NbLig=6,NbSubLig=3,%
  Police=\small\bfseries\ttfamily,CouleurTexte=red,CouleurCase=yellow!50,%
  Legendes=false]{\grilleSuMa}
```

Code $\text{\LaTeX}$

(a)	(b)			(c)	(d)
(e)			(f)	(g)	(h)
		(i)			(j)
		(k)		(l)	(m)
(n)		(o)			(p)
			(q)		



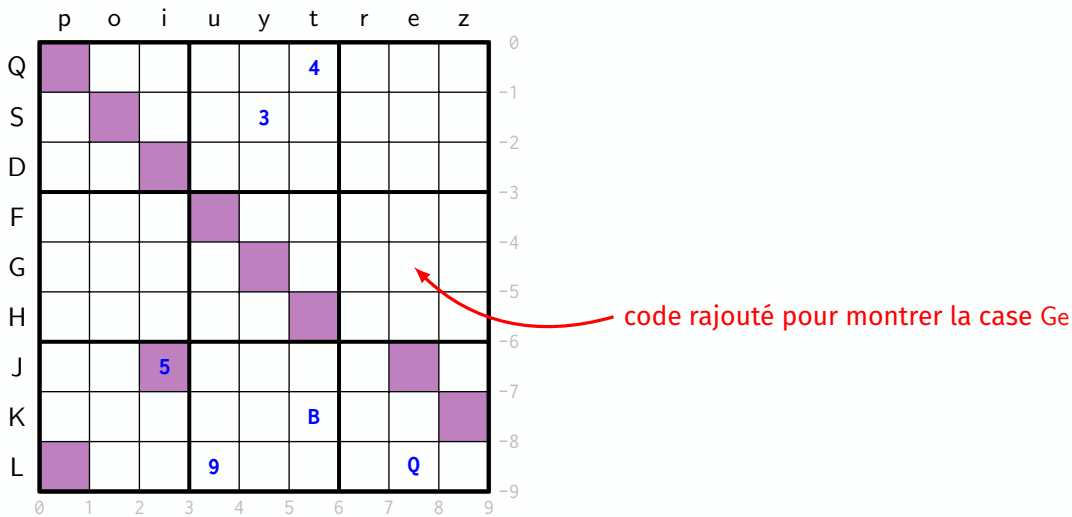
La grille, créée en TikZ, est portée par le rectangle de « coins » (0 ; 0) et (nbcol ; -nblig), de sorte que les labels des cases sont situés au nœuds de coordonnées (x, 5 ; -y, 5).

Code $\LaTeX$

```
%grille classique avec coloration de cases et commande tikz
%graduations rajoutées pour la lecture des coordonnées
\def\grilleSuMaB{%
  *////4///$%
  /*///3///$%
  //*////$%
  ///*////$%
  ////*////$%
  /////*///$%
  //5*////*$%
  ////B///*$%
  *///9///Q/$%
}

\begin{EnvSudoMaths}[%
  Unite=0.66cm,Police=\footnotesize\bfseries\ttfamily,CouleurCase=violet!50,%
  ListeLegV=QSDFGHJKL,ListeLegH=poiuytrez]{\grilleSuMaB}
  \draw[red,very thick,<-,>=latex] (7.5,-4.5) to[bend right] ++ (4,-1) node[right] {code rajouté...} ;
\end{EnvSudoMaths}
```

Code $\LaTeX$



71 Quelques fractales, en TikZ

71.1 Introduction



2.7.9 L'idée est de proposer de quoi représenter quelques fractales, créées avec la librairie `lindenmayersystems`.

Pour le moment, il est possible de :

- tracer un flocon de Koch à une étape donnée;
- tracer un triangle de Sierpinski à une étape donnée;
- présenter différentes étapes successives des flocons de Koch ou des triangles de Sierpinski.



Les figures sont créées en TikZ, et peuvent être autonomes (sans environnement `tikzpicture`). Pour le triangle de Sierpinski, la forme *générale* est *bloquée* pour avoir un rendu *classique*.

71.2 Flocon de Koch et triangle de Sierpinski



La commande pour créer un flocon de Koch ou un triangle de Sierpinski est `\FractaleTikz`. Les éléments de personnalisation sont présentés un peu plus bas.

Code $\text{\LaTeX}$

```
%Flocon de Koch, autonome
\FractaleTikz[Type=Koch,clés]<options tikz>

%Flocon de Koch, dans un environnement tikz
\begin{tikzpicture}
  \FractaleTikz*[Type=Koch,clés]
\end{tikzpicture}
```

Code $\text{\LaTeX}$

```
%Triangle de Sierpinski, autonome
\FractaleTikz[Type=Sierp,clés]<options tikz>

%Flocon de Koch, dans un environnement tikz
\begin{tikzpicture}
  \FractaleTikz[Type=Sierp,clés]
\end{tikzpicture}
```

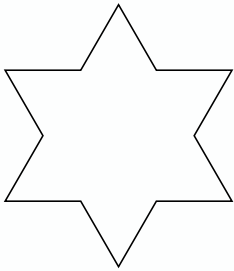


Les **clés** disponibles pour cette commande sont :

- | | |
|--|-------------------------|
| – la clé Épaisseur pour fixer l'épaisseur des tracés; | défaut 0.6pt |
| – la clé Type , parmi Koch / Sierp pour choisir le type de fractale; | défaut Koch |
| – la clé Couleur pour fixer la couleur des tracés; | défaut black |
| – la clé LongueurCote (en cm) pour fixer la longueur des côtés; | défaut 3 |
| – la clé Étape (pour Type=Koch elle est limitée à 7) pour fixer l'étape; | défaut 1 |
| – le booléen remplir pour remplir la fractale; | défaut false |
| – la clé Remplissage pour fixer la couleur de remplissage; | défaut lightgray |
| – la clé Départ pour fixer le point de départ; | défaut (0,0) |
| – le booléen AlignV (pour Type=Koch) pour forcer l'alignement de la base. | défaut false |

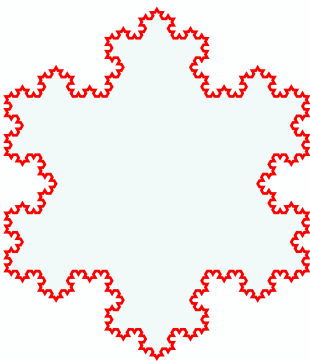
Code $\LaTeX$

```
%Koch par défaut  
\FractaleTikz
```



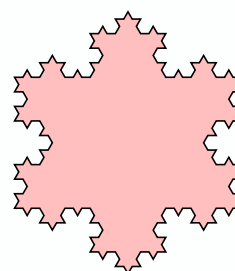
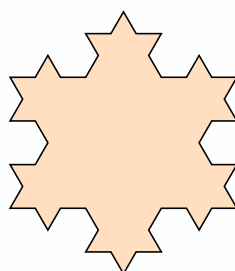
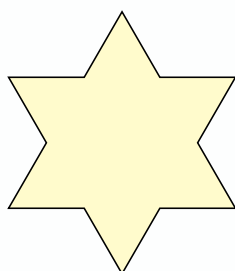
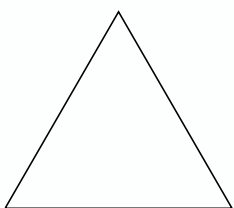
Code $\LaTeX$

```
%Koch par défaut  
\FractaleTikz[Etape=4,LongueurCote=4,Remplir,Remplissage=teal!5,Couleur=red,Epaisseur=1pt]
```



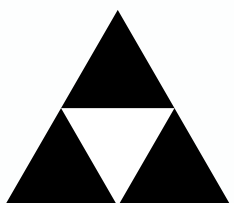
Code $\LaTeX$

```
%dans un environnement tikz  
\begin{tikzpicture}  
  \FractaleTikz*[Etape=0]  
  \FractaleTikz*[Depart={(4,0)},Etape=1,Remplir,Remplissage=yellow!25]  
  \FractaleTikz*[Depart={(8,0)},Etape=2,Remplir,Remplissage=orange!25]  
  \FractaleTikz*[Depart={(12,0)},Etape=3,Remplir,Remplissage=red!25]  
\end{tikzpicture}
```



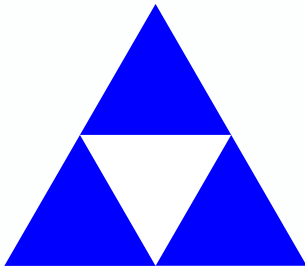
Code $\LaTeX$

```
%Sierpinski par défaut  
\FractaleTikz[Type=Sierp]
```



Code $\LaTeX$

```
%Sierpinski par défaut  
\FractaleTikz[Type=Sierp,LongueurCote=4,Couleur=blue]
```



71.3 Affichage de plusieurs étapes pour les flocons de Koch



L'idée est de présenter des étapes successives pour le flocon de Koch.
À noter que les bases des flocons sont, dans ce cas, correctement alignées!

Code $\LaTeX$

```
%commande autonome, l'environnement tikz est créé  
\EtapesFloconKoch[clés]{étapes}
```



Les **clés** disponibles sont reprises (pour celles dépendant de **Type=Koch**!) de la commande `\FractaleTikz`, avec en plus :

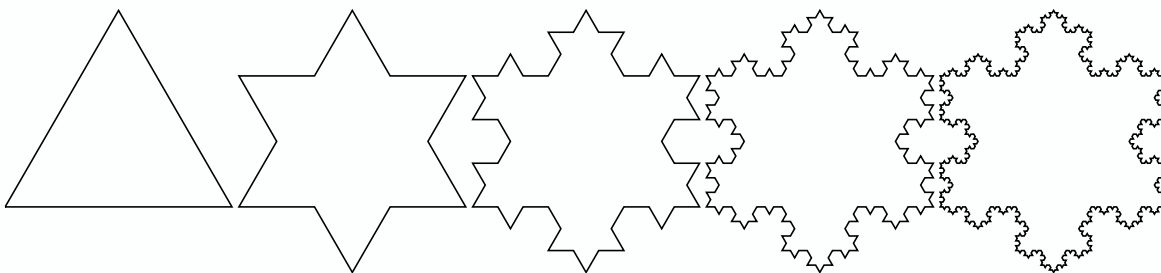
- la clé **Offset** pour fixer un espacement horizontal entre les figures. défaut **2pt**

L'argument obligatoire, et entre `{...}`, permet de spécifier les étapes à afficher, sous la forme TikZ :

- `n1,n2,n3` pour spécifier une liste d'étapes ;
- `n1,...,n2` pour spécifier une plage d'étapes.

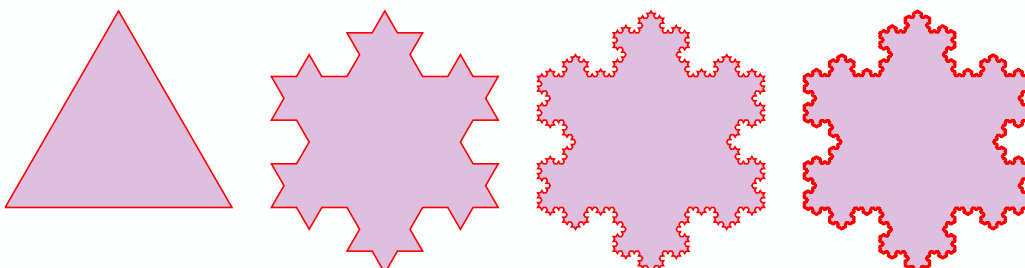
Code $\LaTeX$

```
\EtapesFloconKoch{0,...,4}
```



Code $\LaTeX$

```
\EtapesFloconKoch[Offset=5mm,Couleur=red,Remplir,Remplissage=violet!25]{0,2,4,6}
```



71.4 Affichage de plusieurs étapes pour les tapis de Sierpinski



L'idée est de présenter des étapes successives pour les tapis de Sierpinski.
À noter que les bases des flocons sont correctement alignées!

Code $\LaTeX$

```
%commande autonome, l'environnement tikz est créé  
\EtapesTapisSierpinski[clés]{étapes}
```



Les **(clés)** disponibles sont reprises (pour celles dépendant de **(Type=Sierp)**!) de la commande `\FractaleTikz`, avec en plus :

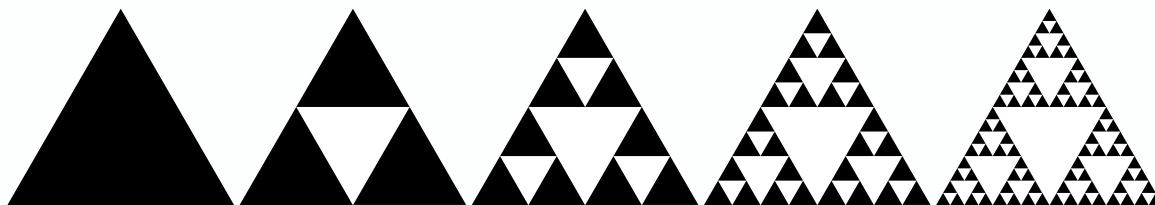
- la clé **(Offset)** pour fixer un espacement horizontal entre les figures. défaut **(2pt)**

L'argument obligatoire, et entre `{...}`, permet de spécifier les étapes à afficher, sous la forme TikZ :

- `n1,n2,n3` pour spécifier une liste d'étapes;
- `n1,...,n2` pour spécifier une plage d'étapes.

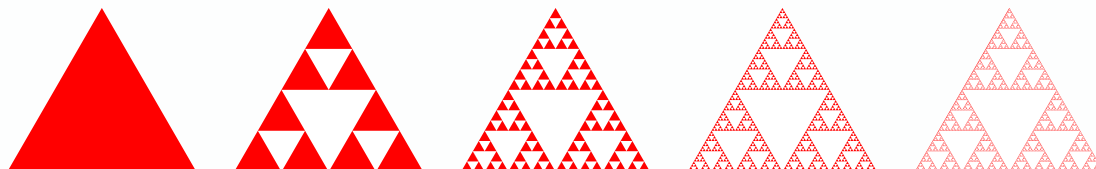
Code $\LaTeX$

```
\EtapesTapisSierpinski{0,...,4}
```



Code $\LaTeX$

```
\EtapesTapisSierpinski[LongueurCote=2.5,Offset=5mm,Couleur=red]{0,2,4,6,8}
```



72 Châteaux de cartes

72.1 Introduction



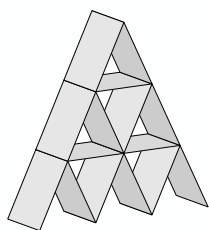
`3.00d` L'idée est de présenter des empilements type *châteaux de cartes* pour illustrer un travail sur des suites, par exemple.

Code $\LaTeX$

```
\ChateauCartes[clés]{niveau}<options tikz>
```

Code $\LaTeX$

```
%clés par défaut  
\ChateauCartes{3}
```



72.2 Clés et options



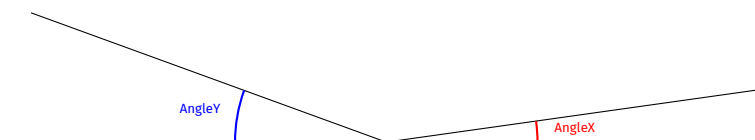
Les **clés** disponibles pour cette commande sont :

- la clé **⟨Echelle⟩** := modifier les dimensions des cartes (par défaut elle est fixée à **⟨1⟩**)
- la clé **⟨Deco⟩** := choix du type de décoration parmi **⟨remplir / vide / hachures⟩** (par défaut elle vaut **⟨remplir⟩**)
- la clé **⟨CouleurDeco⟩** := choix de la couleur de décoration (par défaut elle est à base de **⟨black⟩**)
- le booléen **⟨Arrondi⟩** := créer un arrondi pour les cartes (par défaut **⟨true⟩**)
- les clés **⟨AngleX⟩** et **⟨AngleY⟩** := modification de la *vue* (à utiliser avec précaution, **⟨8⟩** et **⟨20⟩** par défaut)
- le booléen **⟨Bas⟩** := afficher les cartes horizontales du bas (par défaut **⟨false⟩**)
- le booléen **⟨Legende⟩** := afficher les valeurs de n (par défaut **⟨true⟩**)
- la clé **⟨PoliceLegende⟩** := police de la légende.

L'argument obligatoire, entre `{...}`, correspond au nombre d'étages voulus.

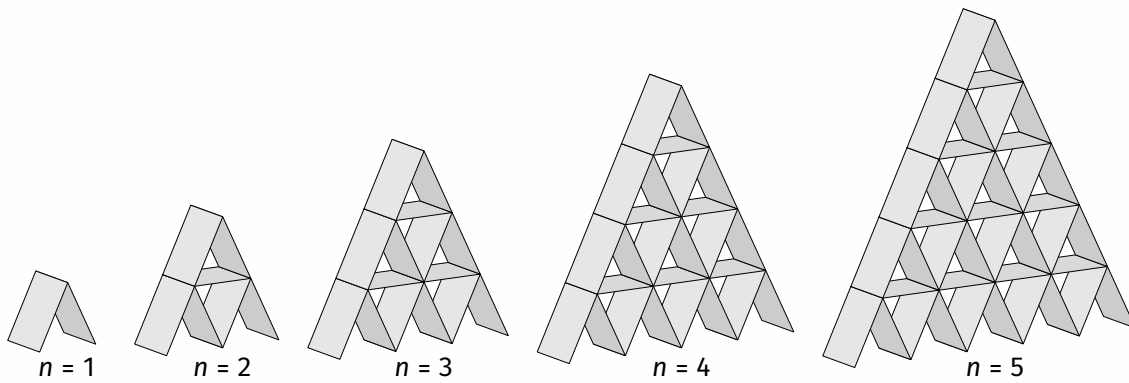


Concernant les clés **⟨AngleX⟩** et **⟨AngleY⟩**, elles correspondent à l'*inclinaison* des axes pour la vue 3D :



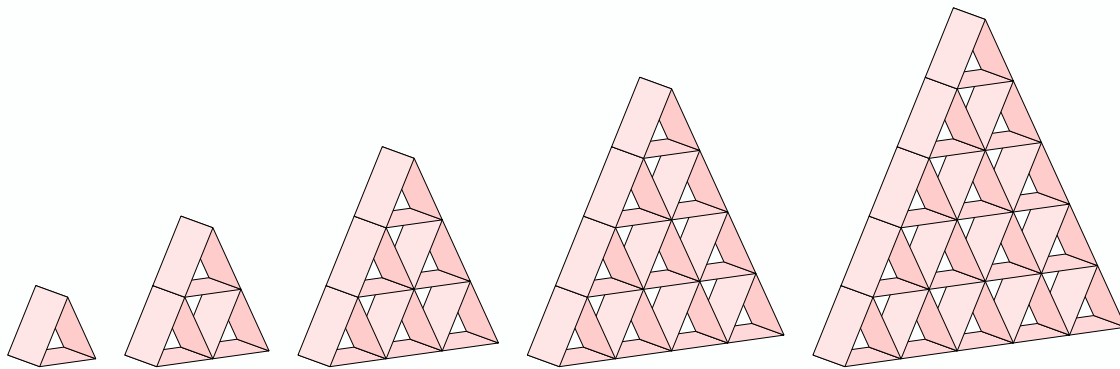
Code $\LaTeX$

```
\xintFor* #1 in {\xintSeq{1}{5}}\do{\ChateauCartes[Legende]{#1}}
```



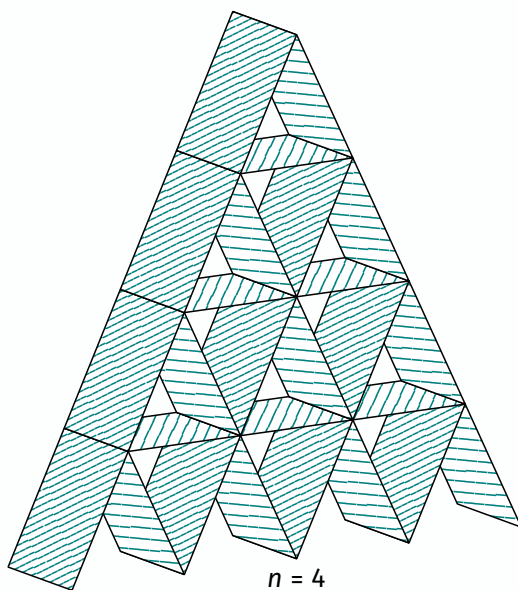
Code $\LaTeX$

```
\xintFor* #1 in {\xintSeq{1}{5}}\do{\ChateauCartes[Bas,Legende=false,CouleurDeco=red]{#1}}
```



Code $\LaTeX$

```
\ChateauCartes[CouleurDeco=teal,Deco=hachures,Echelle=2,Legende]{4}
```



73 Allumettes

73.1 Introduction



`3.00d` L'idée est de proposer une commande pour représenter des allumettes, créées (et à insérer) dans un environnement `tikzpicture`.

L'idée est de :

- déclarer les points *support* pour le placement des allumettes ;
- positionner les allumettes par rapport aux point *support* existants.

Le code se charge de placer correctement l'allumette, en fonction de la distance et de l'angle entre les points *support*.

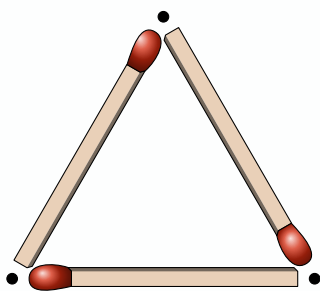
La forme générale de l'allumette est fixée, et les dimensions (internes) sont données en cm.

Code $\LaTeX$

```
%dans un environnement tikzpicture
\PFLAllumettes[clés]{liste des chemins}
```

Code $\LaTeX$

```
\begin{tikzpicture}
  \coordinate (A) at (0,0) ; \coordinate (B) at (60:4) ; \coordinate (C) at (4,0) ;
  \filldraw (A) circle[radius=2pt] (B) circle[radius=2pt] (C) circle[radius=2pt] ; %points de contrôle
  \PFLAllumettes{A>B B>C C>A}
\end{tikzpicture}
```



Pour cette commande, il est conseillé de ne pas utiliser d'option type `scale` pour l'environnement `tikzpicture`, mais plutôt de spécifier les unités en *dur*, via `x=...,y=...`.

73.2 Clés et options



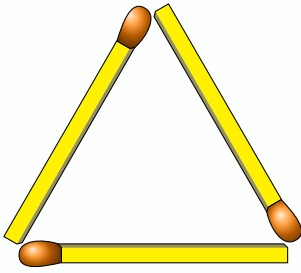
Les **clés** disponibles pour cette commande sont :

- la clé **CouleurBois** := couleur du *manche*, et valant **BoisAllumette** par défaut ;
- la clé **CouleurBout** := couleur du *bout*, et valant **GratteAllumette** par défaut ;
- la clé **Decal** := espacement par rapport aux point *support*, et valant **0.224cm** par défaut ;
- le booléen **NoirBlanc** := forcer un affichage N&B (par défaut **false**).

L'argument obligatoire, entre `{...}`, correspond à la liste des chemins, sous la forme `A1>B1 A2>B2 ...`.

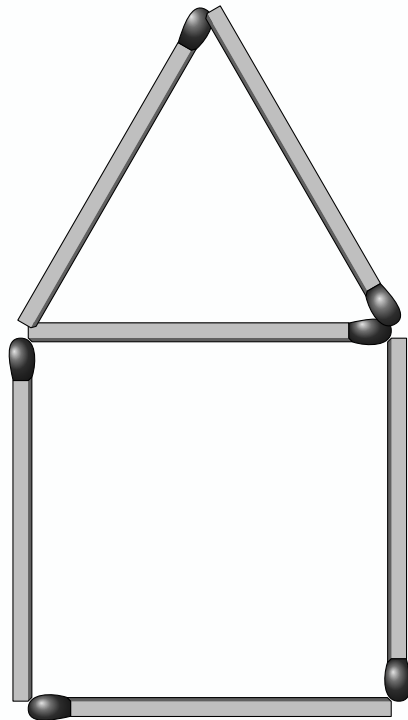
Code $\LaTeX$

```
\begin{tikzpicture}
  \coordinate (A) at (0,0) ; \coordinate (B) at (60:4) ; \coordinate (C) at (4,0) ;
  \PflAllumettes[CouleurBout=orange,CouleurBois=yellow]{A>B B>C C>A}
\end{tikzpicture}
```



Code $\LaTeX$

```
\begin{tikzpicture}[x=1.25cm,y=1.25cm]
  \coordinate (A) at (0,0) ; \coordinate (B) at (0,4) ; \coordinate (C) at (4,4) ; \coordinate (D) at
  (4,0) ;
  \coordinate (E) at ($(B)+(60:4)$) ;
  \PflAllumettes[Decal=1mm,NoirBlanc]{A>B B>C C>D D>A B>E E>C}
\end{tikzpicture}
```



74 Pyramide de sphères

74.1 Introduction



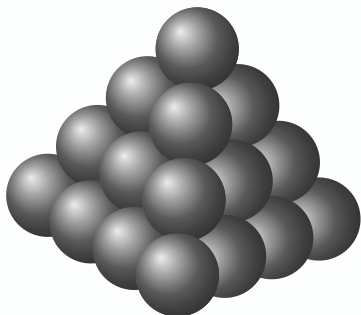
`3.10b` L'idée est de proposer une commande pour représenter, de manière basique, un empilement de sphères.

La visualisation est fixée, mais quelques éléments de personnalisations sont possibles.

Code $\text{\LaTeX}$

```
\EmpilementBalles[clés]{étage(s)}
```

Code $\text{\LaTeX}$



74.2 Clés et options



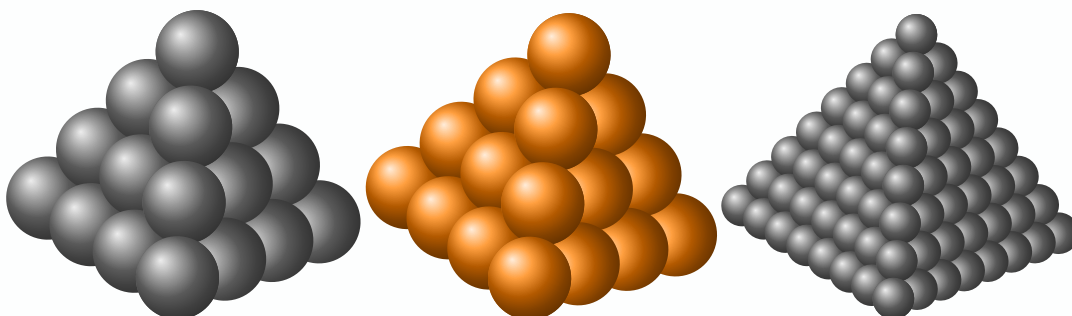
Les **clés** disponibles pour cette commande sont :

- la clé **Couleur** := couleur des boules (**gray** par défaut);
- la clé **Echelle** := échelle globale (**1** par défaut);
- la clé **Rotation** := rotation, à manipuler avec précaution (**-5** par défaut).

L'argument obligatoire correspond au nombre d'étages à afficher, sous la forme `n` ou `deb-fin`.

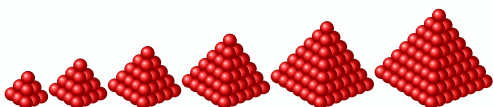
Code $\text{\LaTeX}$

```
\EmpilementBalles{4}%  
\EmpilementBalles[Rotation=-10,Couleur=orange]{4}%  
\EmpilementBalles[Echelle=0.5]{8}
```



Code $\text{\LaTeX}$

```
\EmpilementBalles[Echelle=0.33,Couleur=red]{3-8}
```



75 Commandes pour les dates

75.1 Introduction



3.10b L'idée est de proposer, en partenariat avec le package `datetime2` (qui sera, le cas échéant, chargé avec l'option `[fr-FR]`) des commandes sur les dates :

- afficher le jour (en littéral) correspondant à une date donnée;
- afficher la date complète avec le nom du jour;
- afficher le nombre de jours entre deux dates;
- afficher la date augmentée d'un nombre de jours.

Code $\LaTeX$

```
%version étoilée pour une majuscule au début
\JourSelonDate*{jj/mm/aaaa}

%version étoilée pour une majuscule au début
\DateComplete*{jj/mm/aaaa}

%version étoilée pour afficher
%résultat stocké dans \nbjoursentre par défaut
\NbJoursEntreDates*{jj/mm/aaaa}{jj/mm/aaaa}[\macro]

%version étoilée pour un format jj/mm/aaaa
\AjoutJoursDate*{jj/mm/aaaa}{nb}
```

75.2 Exemples

Code $\LaTeX$

```
Le 14 mai 1952 était un \JourSelonDate{14/05/1952}.
\JourSelonDate*{04/10/1954} 4 octobre 1954.
\DateComplete*{15/08/1980} est un grand jour pour moi !
```

Le 14 mai 1952 était un mercredi.
Lundi 4 octobre 1954.
Vendredi 15 août 1980 est un grand jour pour moi!

Code $\LaTeX$

```
\NbJoursEntreDates{14/05/1952}{15/08/1980}%stockée dans \nbjoursentre
Entre le 14 mai 1952 et le 15 août 1980, il y a \num{\nbjoursentre} jours.
```

Entre le 14 mai 1952 et le 15 août 1980, il y a 10 320 jours.

Code $\LaTeX$

```
100 jours après le 04 octobre 2024, nous serons le \AjoutJoursDate{04/10/2024}{100}.
20 jours après le 25 décembre 2025, nous serons le \AjoutJoursDate*{25/12/2025}{20}.
```

100 jours après le 04 octobre 2024, nous serons le 12 janvier 2025.
20 jours après le 25 décembre 2025, nous serons le 14/01/2026.

76 Machines à transformer, en TikZ

76.1 Introduction



`3.00e` L'idée est de *proposer* une commande pour travailler avec une « machine à transformer ». La figure est réalisée en TikZ, et quelques éléments de personnalisations sont disponibles, bien que la forme *globale* soit fixée.

Code $\text{\LaTeX}$

$\backslash\text{MachineTransformer}[clés]\{e/s\}<\text{options tikz}>$

Code $\text{\LaTeX}$

Code $\text{\LaTeX}$

$\backslash\text{MachineTransformer}[Bordure,Tableau,Fct=\$2x-5\$]\%$
 $\{3/1,7/9,10/15,1/\$-3\$, \$-4\$/\$-13\}$

Entrée	Sortie
3	1
7	9
10	15
1	-3
-4	-13

Code $\text{\LaTeX}$

$\backslash\text{MachineTransformer}[Auto,Formule=\{-2*X+7\},Bordure,Tableau,Fct=\$-2x+7\$]\%$
 $\{3,5,7,-4,0\}$

Entrée	Sortie
3	1
5	-3
7	-7
-4	15
0	7

76.2 Clés et options



Les **clés** disponibles pour cette commande sont :

- la clé **Couleur** := couleur de la machine (**lightgray** par défaut);
- la clé **CouleurFct** := couleur de la flèche (**white** par défaut);
- le booléen **Bordure** := afficher une bordure (**true** par défaut);
- le booléen **AffFleche** := afficher la flèche (**true** par défaut);
- la clé **Hauteur** := hauteur de la machine (**3** par défaut);
- la clé **Largeur** := largeur de la machine (**2** par défaut);
- la clé **Offset** := décalage (H) des blocs (**4pt** par défaut);
- la clé **CouleurBloc** := couleur des blocs (**red** par défaut);
- le booléen **Tableau** := afficher le tableau des valeurs (**false** par défaut);
- la clé **PoliceTbl** := police des éléments du tableau (**\footnotesize** par défaut);
- le booléen **Logo** := afficher le logo dans la machine (**true** par défaut);
- la clé **Fct** := fonction à afficher (si non vide);
- le booléen **Auto** := calcul automatique des images (**false** par défaut);
- la clé **Formule** := formule (variable x) dans le cas **Auto=true**
- le booléen **ES** := afficher des blocs vides, si pas de tableau (**false** par défaut);
- la clé **Echelle** := modifier l'échelle globale, y compris du texte (**1** par défaut).

L'argument obligatoire, entre $\{ \dots \}$, correspond à la liste des entrées/sorties :

- sous la forme $x_1/y_1, x_2/y_2, \dots$ dans le cas *manuel* (**Auto=false**);
- sous la forme $x_1, x_2, \dots$ dans le cas *automatique* (**Auto=true**).

À noter que dans le cas **Auto**, les valeurs sont en mode mathématique, alors qu'en mode **Auto=false**, il faut le rajouter avec des $\$ \dots \$$.

L'argument optionnel, et entre $\langle \dots \rangle$, correspond à des options spécifiques, en langage TikZ, à passer à l'environnement créé.

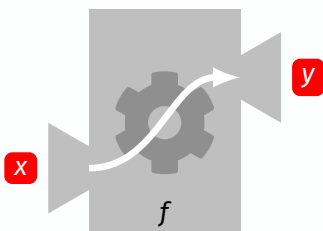


Dans le cas où les calculs seraient faits en automatique :

- attention aux valeurs fractionnaires et/ou approchées (pas de gestion de ces cas de figure);
- l'expression (**Formule**) de la fonction doit être en langage x_{int} , avec x comme variable!

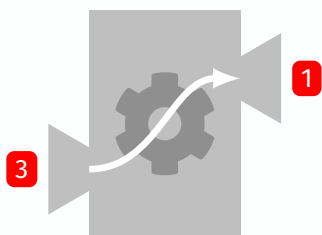
Code $\LaTeX$

```
\MachineTransformer[Fct=$f$]{ $x$ / $y$ }
```



Code $\text{\LaTeX}$

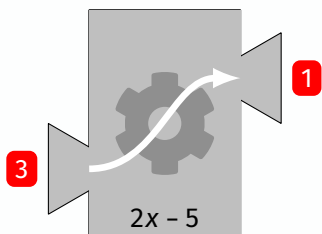
```
\MachineTransformer[Tableau]{3/1,7/9,10/15,1/\$-3$, \$-4$/\$-13\$}
```



Entrée	Sortie
3	1
7	9
10	15
1	-3
-4	-13

Code $\text{\LaTeX}$

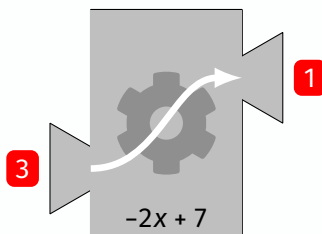
```
\MachineTransformer[Bordure,Tableau,Fct=\$2x-5\$]{3/1,7/9,10/15,1/\$-3$, \$-4$/\$-13\$}
```



Entrée	Sortie
3	1
7	9
10	15
1	-3
-4	-13

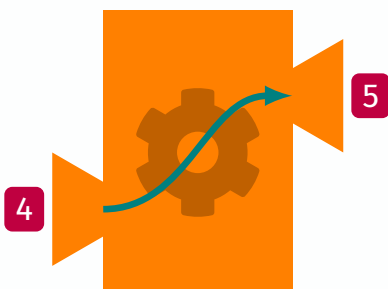
Code $\text{\LaTeX}$

```
\MachineTransformer[Auto,Formule={-2*X+7},Bordure,Tableau,Fct=\$-2x+7\$]{3,5,7,-4,0}
```



Entrée	Sortie
3	1
5	-3
7	-7
-4	15
0	7

```
\MachineTransformer[%  
  Echelle=1.25,Tableau,Couleur=orange,CouleurFct=teal,%  
  CouleurBloc=purple,Offset=1mm  
]%  
{4/5}
```



Entrée	Sortie
4	5

Thème

QUELQUES CLIPARTS

Quatorzième partie

Quelques cliparts

77 Idées



3.05a L'idée est de proposer des petits cliparts vectoriels, en licence libre, pour insertion dans un document :

- en mode *normal*;
- en mode *en ligne* avec taille et ajustement automatiques.



Les pdf ont été obtenus depuis des fichiers *svg* des sites <https://openclipart.org> et <https://www.svgrepo.com>, les auteurs étant Firkin, Juhele, qubodup, TzeenieWheenie, Twitter et jetxee.

Code $\text{\LaTeX}$

```
%insertion classique
\InsererClipart*[options includegraphics]{clipart}

%insertion en ligne
\InsererClipart{clipart}
```

Code $\text{\LaTeX}$

```
%médaille d'or, insertion manuelle
\InsererClipart*[height=3cm]{gold-medal}

%insertion en ligne, avec hauteur et profondeur adaptées
En mode ligne, on peut donner cet exemple (\InsererClipart{notok}) pour illustrer.
```



En mode ligne, on peut donner cet exemple (❌) pour illustrer.



3.10e Un petit logo, pour illustrer une interdiction de calculatrice, peut être utilisé.

Il est créé à partir de **fontawesome5** et est destiné à être utilisé en mode *en ligne*, car il s'adapte à la taille de la fonte courante (hauteur et profondeur).

La version étoilée augmente légèrement les espaces horizontaux autour du logo, et l'argument optionnel permet de spécifier la couleur.

Code $\text{\LaTeX}$

```
\pflnocalc(*)[couleur]
```

78 Liste des cliparts, et aliases



Chaque clipart est nommé sous la forme `pfl-<nom>`, et des aliases sont disponibles, pour insertion *plus rapide*.

Code $\LaTeX$

NOM (pfl-)	MACRO	RENDU
bronze-cup	<code>\pflimgcoupeb</code>	
bronze-medal	<code>\pflimgmedb</code>	
bronze-medal-flat	<code>\pflimgmedbp</code>	
silver-cup	<code>\pflimgcoupea</code>	
silver-medal	<code>\pflimgmeda</code>	
silver-medal-flat	<code>\pflimgmedap</code>	
gold-cup	<code>\pflimgcoupeo</code>	
gold-medal	<code>\pflimgmedo</code>	
gold-medal-flat	<code>\pflimgmedop</code>	
NOM (pfl-)	MACRO	RENDU
checked	<code>\pflimgverif</code>	
crossed	<code>\pflimgnonverif</code>	
ok	<code>\pflimgok</code>	
notok	<code>\pflimgko</code>	
mouse	<code>\pflimgsouris</code>	
mouse-left	<code>\pflimgsourisg</code>	
mouse-center	<code>\pflimgsourisc</code>	
mouse-right	<code>\pflimgsourisd</code>	
bullseye-3d	<code>\pflimgcible</code>	
bullseye-dart	<code>\pflimgflechette</code>	

79 Aperçu du logo "calculatrice interdite"

Code $\LaTeX$

```
En ligne \pflnocalc\ avec un petit texte\\
\textcolor{blue}{\sffamily\LARGE En ligne \pflnocalc[blue] avec un petit texte}\\
\textcolor{red}{\ttfamily\Huge En ligne \pflnocalc[red] avec un petit texte}\\
\scalebox{3.25}[3.25]{Avec une échelle perso : \pflnocalc}
```

En ligne  avec un petit texte

En ligne  avec un petit texte

En ligne  avec un petit texte

Avec une échelle perso : 

80 Cliparts 'Zone Bac/Brevet/BTS'



3.12a L'idée est de proposer des petits cliparts vectoriels (sur une idée du site <https://mathsapiens.fr>), en licence libre, pour illustrer des documents type Zone Examen.

Les cliparts ne sont pas proposés en version individuelle, mais sont à insérer (via `\includegraphics[options,page=...]{pfl-pictosbac.pdf}`) ou via la macro du package.

Une insertion en ligne n'est pas (directement) proposée, mais il est possible d'insérer la commande dans un `\raisebox` ou autre!

Centre	Alias	$\pflpictobac^{*}\{\dots\}$	$\pflpictobac\{\dots\}$	Pages
Amérique du Nord	amnord			1-2
Amérique du Sud	amsud.v1			3-4
Amérique du Sud	amsud.v2			5-6
Asie	asie			7-8
Asie	asie.v2			9-10
Centres Étrangers	ce			10-11
Centres Étrangers	ce.v2			12-13
Centres Étrangers	ce.v3			14-15
Métropole	fr			16-17
Métropole	fr.v2			18-19
Liban	liban			20-21
Nouvelle Calédonie	nouvcal			22-23
Polynésie	poly			24-25
Polynésie	poly.v2			28-29
La Réunion	reunion			26-27
La Réunion	reunion.v2			30-31



3.12a À noter que ces cliparts peuvent être insérés sans forcément charger tout le package `ProfLycee`, mais en chargeant l'une de ses dépendances, à savoir le package `ProfLycee-Pictosbac` (inutile si `ProfLycee` est chargé!)

```
\usepackage{ProfLycee-Pictosbac}
```

81 Cliparts flat 'Zone Bac/Brevet/BTS'



3.12d L'idée est de proposer des petits cliparts 'flat' vectoriels (sur une idée du site <https://mathsapiens.fr>), en licence libre, pour illustrer des documents type Zone Examen.

Les cliparts ne sont pas proposés en version individuelle, mais sont à insérer (via `\includegraphics[options,page=...]{pfl-flatpictosbac.pdf}`) ou via la macro du package.

Une insertion en ligne n'est pas (directement) proposée, mais il est possible d'insérer la commande dans un `\raisebox` ou autre!

Code $\LaTeX$

Centre	Alias	<code>\pflflatpictobac{...}</code>	<code>\pflflatpictobac*{...}</code>	Pages
Amérique du Nord	amnord			1-2
Amérique du Sud	amsud			3-4
Asie	asie.v1			5-6
Asie	asie.v2			7-8
Centres Étrangers	ce.v1			9-10
Centres Étrangers	ce.v2			11-12
Métropole	fr.v1			12-14
Métropole	fr.v2			15-16
Polynésie	poly			17-18

Thème

COMPÉTENCES EN LYCÉE

Quinzième partie

Compétences au lycée

82 Introduction

82.1 Fonctionnement global



L'idée est de proposer des commandes pour insérer, *facilement*, les compétences mathématiques au lycée général ou professionnel.

Les compétences sont celles des documents de travail de l'inspection, et elles peuvent être insérées :

- de manière individuelle ou par type ou toutes;
- en version raccourcie;
- avec libellé ou puce.

82.2 Commandes

Code $\text{\LaTeX}$

```
%compétences LEGT
\CompMathsLyc[clés]{choix}

%compétences LPRO
\CompMathsLycPro[clés]{choix}
```



Les $\langle \text{clés} \rangle$ disponibles pour ces commandes sont :

- | | |
|---|--|
| — $\langle \text{AffCateg} \rangle$: affichage de la catégorie avant; | défaut $\langle \text{false} \rangle$ |
| — $\langle \text{AffNumero} \rangle$: affichage du numéro avant; | défaut $\langle \text{false} \rangle$ |
| — $\langle \text{Court} \rangle$: booléen pour raccourcir la compétence; | défaut $\langle \text{false} \rangle$ |
| — $\langle \text{Puce} \rangle$: booléen pour afficher une puce avant; | défaut $\langle \text{false} \rangle$ |
| — $\langle \text{TypePuce} \rangle$: paramétrage de la puce éventuelle. | défaut $\langle \text{textbullet} \rangle$ |

L'argument obligatoire permet quant à lui de choisir les compétences à afficher :

- individuellement via le code $\langle \text{CATEG/Item} \rangle$ ou $\langle \text{LYCnumitem} \rangle$;
- globalement via le code $\langle \text{CATEG/Tout} \rangle$ ou $\langle \text{LYCnum} \rangle$;
- toutes via le code $\langle \text{LYC} \rangle$.

82.3 Liste des compétences lycée général et technologique



Le tableau suivant propose les compétences, avec leurs *appels* pour la commande d'insertion.

Chercher :		
CH/Analyser	LYC11	Analyser un problème.
CH/Extraire	LYC12	Extraire, organiser et traiter l'information utile.
CH/Observer	LYC13	Observer, s'engager dans une démarche, expérimenter en utilisant éventuellement des outils logiciels, chercher des exemples ou des contre-exemples, simplifier ou particulariser une situation, reformuler un problème, émettre une conjecture.
CH/Valider	LYC14	Valider, corriger une démarche, ou en adopter une nouvelle.
Modéliser :		
MO/Traduire	LYC21	Traduire en langage mathématique une situation réelle (à l'aide d'équations, de suites, de fonctions, de configurations géométriques, de graphes, de lois de probabilité, d'outils statistiques, ...).
MO/Utiliser	LYC22	Utiliser, comprendre, élaborer une simulation numérique ou géométrique prenant appui sur la modélisation et utilisant un logiciel.
MO/Valider	LYC23	Valider ou invalider un modèle.
Représenter :		
RE/Cadre	LYC31	Choisir un cadre (numérique, algébrique, géométrique, ...) adapté pour traiter un problème ou pour représenter un objet mathématique.
RE/Passer	LYC32	Passer d'un mode de représentation à un autre.
RE/Registre	LYC33	Changer de registre.
Calculer :		
CA/Effectuer	LYC41	Effectuer un calcul automatisable à la main ou à l'aide d'un instrument (calculatrice, logiciel).
CA/Mettre	LYC42	Mettre en œuvre des algorithmes simples.
CA/Intellig	LYC43	Exercer l'intelligence du calcul : organiser les différentes étapes d'un calcul complexe, choisir des transformations, effectuer des simplifications.
CA/Controler	LYC44	Contrôler les calculs (au moyen d'ordres de grandeur, de considérations de signe ou d'encadrement).
Raisonner :		
RA/Logique	LYC51	Utiliser les notions de la logique élémentaire (conditions nécessaires ou suffisantes, équivalences, connecteurs) pour bâtir un raisonnement.
RA/Differencier	LYC52	Différencier le statut des énoncés mis en jeu : définition, propriété, théorème démontré, théorème admis...
RA/Raisonnement	LYC53	Utiliser différents types de raisonnement (par analyse et synthèse, par équivalence, par disjonction de cas, par l'absurde, par contraposée, par récurrence...).
RA/Inference	LYC54	Effectuer des inférences (inductives, déductives) pour obtenir de nouveaux résultats, conduire une démonstration, confirmer ou infirmer une conjecture, prendre une décision.
Communiquer :		
CO/Operer	LYC61	Opérer la conversion entre le langage naturel et le langage symbolique formel.
CO/Developper	LYC62	Développer une argumentation mathématique correcte à l'écrit ou à l'oral.
CO/Critiquer	LYC63	Critiquer une démarche ou un résultat.
CO/Exprimer	LYC64	S'exprimer avec clarté et précision à l'oral et à l'écrit.

82.4 Liste des compétences lycée professionnel



Le tableau suivant propose les compétences, avec leurs *appels* pour la commande d'insertion.

S'approprier :		
AP/Rechercher	LYC11	Rechercher, extraire et organiser l'information.
AP/Traduire	LYC12	Traduire des informations, des codages.
Analyser/Raisonner :		
AR/Emettre	LYC21	Émettre des conjectures, formuler des hypothèses.
AR/Proposer	LYC22	Proposer une méthode de résolution.
AR/Modele	LYC23	Choisir un modèle ou des lois pertinentes.
AR/Algorithme	LYC24	Élaborer un algorithme.
AR/Choisir	LYC25	Choisir, élaborer un protocole.
AR/Grandeur	LYC26	Évaluer des ordres de grandeur.
Réaliser :		
RE/Etapes	LYC31	Mettre en œuvre les étapes d'une démarche.
RE/Modele	LYC32	Utiliser un modèle.
RE/Représenter	LYC33	Représenter (tableau, graphique, ...), changer de registre.
RE/Calculer	LYC34	Calculer (calcul littéral, calcul algébrique, calcul numérique exact ou approché, instrumenté ou à la main).
RE/Algorithmes	LYC35	Mettre en œuvre des algorithmes.
RE/Expérimenter	LYC36	Expérimenter – en particulier à l'aide d'outils numériques (logiciels ou des dispositifs d'acquisition de données...).
RE/Simulation	LYC37	Faire une simulation.
RE/Effectuer	LYC38	Effectuer des procédures courantes (représentations, collectes de données, utilisation du matériel, etc).
RE/Protocole	LYC39	Mettre en œuvre un protocole expérimental en respectant les règles de sécurité à partir d'un schéma ou d'un descriptif.
RE/Organiser	LYC3A	Organiser son poste de travail.
Valider :		
VA/Exploiter	LYC41	Exploiter et interpréter les résultats obtenus ou les observations effectuées afin de répondre à une problématique.
VA/Valider	LYC42	Valider ou invalider un modèle, une hypothèse en argumentant.
VA/Controler	LYC43	Contrôler la vraisemblance d'une conjecture.
VA/Critiquer	LYC44	Critiquer un résultat (signe, ordre de grandeur, identification des sources d'erreur), argumenter.
VA/Raisonnement	LYC45	Conduire un raisonnement logique et suivre des règles établies pour parvenir à une conclusion (démontrer, prouver).
Communiquer :		
CO/Rendre	LYC51	Rendre compte d'un résultat en utilisant un vocabulaire adapté et choisir des modes de représentation appropriés.
CO/Expliquer	LYC22	Expliquer une démarche.

82.5 Exemples

Code $\LaTeX$

```
\CompMathsLyc[Court,Puce]{RE/Tout}
```

- Choisir un cadre adapté [...].
- Passer d'un mode de représentation à un autre.
- Changer de registre.

Code $\LaTeX$

```
\CompMathsLyc[Court,Puce,TypePuce={---~}]{LYC4}
```

- Effectuer un calcul automatisable à la main ou à l'aide d'un instrument [...].
- Mettre en œuvre des algorithmes simples.
- Exercer l'intelligence du calcul [...].
- Contrôler les calculs [...].

Code $\LaTeX$

```
\CompMathsLycPro[Court,AffNumero]{LYC3}
```

- LYC31.** Mettre en œuvre les étapes d'une démarche.
LYC32. Utiliser un modèle.
LYC33. Représenter (tableau, graphique, ...), changer de registre.
LYC34. Calculer (calcul littéral, calcul algébrique, [...]).
LYC35. Mettre en œuvre des algorithmes.
LYC36. Expérimenter – en particulier à l'aide d'outils numériques [...].
LYC37. Faire une simulation.
LYC38. Effectuer des procédures courantes (représentations, [...]).
LYC39. Mettre en œuvre un protocole expérimental en respectant les règles de sécurité [...].
LYC3A. Organiser son poste de travail.

Code $\LaTeX$

```
\CompMathsLyc[Court,Puce,TypePuce={---~~}]{LYC4}
```

- Effectuer un calcul automatisable à la main ou à l'aide d'un instrument [...].
- Mettre en œuvre des algorithmes simples.
- Exercer l'intelligence du calcul [...].
- Contrôler les calculs [...].

Thème

EXAMENS

Outils pour les sujets d'examen

83 Page de garde d'un sujet de baccalauréat

83.1 Idée



 L'idée est de proposer une commande pour créer une page de garde « Type Bac » ou « Type Bac EAM ».

La présentation est celle que l'on peut trouver sur les sujets officiels de baccalauréat.

Les *libellés* et *espacements verticaux* sont fixés, mais quelques points de personnalisation sont possibles. Il s'agit essentiellement de *booléens* pour activer ou désactiver des *portions*.

Code $\text{\LaTeX}$ $\text{\PageGardeSujetBac[clés]}$ Code $\text{\LaTeX}$ $\text{\PageGardeSujetBacEAM[clés]}$

83.2 Clés



Les $\langle \text{clés} \rangle$, à donner entre $\text{\LaTeX} [\dots]$ sont :

- $\langle \text{Matiere} \rangle$: paramétrage de matière; défaut $\langle \text{MATHÉMATIQUES} \rangle$
- $\langle \text{Police} \rangle$: paramétrage de la police de la page de garde; défaut $\langle \text{\normalfont} \rangle$
- $\langle \text{Session} \rangle$: paramétrage de la zone *session*; défaut $\langle \text{SESSION \the\year} \rangle$
- $\langle \text{AffSession} \rangle$: booléen pour afficher la zone de *session*; défaut $\langle \text{true} \rangle$
- $\langle \text{Serie} \rangle$: paramétrage de la *série*, $\langle \text{GÉN/TECHNO/PRO} \rangle$; défaut $\langle \text{GÉN} \rangle$
- $\langle \text{Voie} \rangle$: paramétrage de la *voie*, $\langle \text{SPÉ/OBLI/TECHNO} \rangle$; défaut $\langle \text{SPÉ} \rangle$
- $\langle \text{Filiere} \rangle$: paramétrage de la *filière* pour la série technologique; défaut $\langle \text{Sciences et...} \rangle$
- $\langle \text{Jour} \rangle$: paramétrage de la zone *jour*; défaut $\langle \text{Jour 1} \rangle$
- $\langle \text{AffJour} \rangle$: booléen pour afficher la zone *jour*; défaut $\langle \text{true} \rangle$
- $\langle \text{Duree} \rangle$: paramétrage de la durée; défaut $\langle \text{4 heures} \rangle$
- $\langle \text{ModeExamen} \rangle$: booléen pour le message *calculatrice*; défaut $\langle \text{true} \rangle$
- $\langle \text{CalculatriceInterdite} \rangle$: booléen pour le message *calculatrice interdite*; défaut $\langle \text{false} \rangle$
- $\langle \text{DernierePage} \rangle$: méthode pour déterminer la dernière page $\langle \text{lastpage/zref/<num>} \rangle$; défaut $\langle \text{lastpage} \rangle$
- $\langle \text{Justification} \rangle$: booléen pour afficher la zone de *justification*; défaut $\langle \text{true} \rangle$
- $\langle \text{TraceRecherche} \rangle$: booléen pour afficher la zone de *trace de recherche*; défaut $\langle \text{true} \rangle$
- $\langle \text{Clarte} \rangle$: booléen pour afficher la zone de *clarté*; défaut $\langle \text{true} \rangle$
- $\langle \text{NbExos} \rangle$: nombre d'exercices, qui sera converti en toutes lettres; défaut $\langle \text{4} \rangle$
- $\langle \text{Traiter} \rangle$: booléen pour afficher la zone des *exos à traiter*; défaut $\langle \text{false} \rangle$
- $\langle \text{EspaceAvant} \rangle$: espace vertical en haut de la page de garde; défaut $\langle \text{1cm} \rangle$
- $\langle \text{LargeurPar} \rangle$: largeur des derniers paragraphes. défaut $\langle \text{0.85\linewidth} \rangle$



En ce qui concerne la clé **<DernierePage>**, en fonction du choix retenu, le package adéquat devra être chargé par l'utilisateur :

- **<lastpage>** := utilisation du package `lastpage`
- **<zref>** := utilisation du package `zref` avec les options `[lastpage,user]`;
- **<num>** := pour spécifier manuellement le nombre de pages (non calculé automatiquement).

Les exemples proposés ci-dessous utiliseront la version *manuelle* du nombre de pages.

83.3 Exemples

Code $\LaTeX$

`\PageGardeSujetBac[DernierePage=5]`

BACCALAURÉAT GÉNÉRAL

ÉPREUVE D'ENSEIGNEMENT DE SPÉCIALITÉ

SESSION 2026

MATHÉMATIQUES

Jour 1

Durée de l'épreuve : **4 heures**

L'usage de la calculatrice avec mode examen actif est autorisé.

L'usage de la calculatrice sans mémoire « type collègue » est autorisé.

Dès que ce sujet vous est remis, assurez-vous qu'il est complet.

Ce sujet comporte 5 pages numérotées de 1/5 à 5/5.

Sauf mention contraire, toute réponse devra être justifiée.

Le candidat est invité à faire figurer sur la copie toute trace de recherche, même incomplète ou non fructueuse, qu'il aura développée.

La qualité de la rédaction, la clarté et la précision des raisonnements seront prises en compte dans l'appréciation de la copie. Les traces de recherche, même incomplètes ou infructueuses seront valorisées.

```
\PageGardeSujetBac%  
[DernierePage=12,CalculatriceInterdite,Traiter,Police=\sffamily,  
Jour={Bac Blanc n°2 - Vendredi 19 avril 2024},Duree={3 heures},  
LargeurPar=0.75\linewidth]
```

BACCALAURÉAT GÉNÉRAL

ÉPREUVE D'ENSEIGNEMENT DE SPÉCIALITÉ

SESSION 2026

MATHÉMATIQUES

Bac Blanc n°2 - Vendredi 19 avril 2024

Durée de l'épreuve : **3 heures**

L'usage de la calculatrice n'est pas autorisé.

Dès que ce sujet vous est remis, assurez-vous qu'il est complet.

Ce sujet comporte 12 pages numérotées de 1/12 à 12/12.

Sauf mention contraire, toute réponse devra être justifiée.

Le candidat doit traiter les quatre exercices proposés.

Le candidat est invité à faire figurer sur la copie toute trace de recherche, même incomplète ou non fructueuse, qu'il aura développée.

La qualité de la rédaction, la clarté et la précision des raisonnements seront prises en compte dans l'appréciation de la copie. Les traces de recherche, même incomplètes ou infructueuses seront valorisées.

BACCALAURÉAT TECHNOLOGIQUE

ÉPREUVE D'ENSEIGNEMENT DE SPÉCIALITÉ

SESSION 2026

**SCIENCES ET TECHNOLOGIES DE L'INDUSTRIE ET
DU DÉVELOPPEMENT DURABLE**

MATHÉMATIQUES

Jour 1

Durée de l'épreuve : **4 heures**

L'usage de la calculatrice avec mode examen actif est autorisé.
L'usage de la calculatrice sans mémoire « type collège » est autorisé.

Dès que ce sujet vous est remis, assurez-vous qu'il est complet.
Ce sujet comporte 3 pages numérotées de 1/3 à 3/3.

Sauf mention contraire, toute réponse devra être justifiée.

Le candidat est invité à faire figurer sur la copie toute trace de recherche, même incomplète ou non fructueuse, qu'il aura développée.
La qualité de la rédaction, la clarté et la précision des raisonnements seront prises en compte dans l'appréciation de la copie. Les traces de recherche, même incomplètes ou infructueuses seront valorisées.

```
\PageGardeSujetBac%  
[Police=\sfamily,Serie=PRO,  
AffJour=false,DernierePage=8,%  
Duree={2 heures},%  
Filiere={SECTEUR TERTIAIRE},  
Matiere={ÉPREUVE DE MATHÉMATIQUES},%  
DernierePage=4]
```

BACCALAURÉAT PROFESSIONNEL

SESSION 2026

SECTEUR TERTIAIRE

ÉPREUVE DE MATHÉMATIQUES

Durée de l'épreuve : **2 heures**

L'usage de la calculatrice avec mode examen actif est autorisé.

L'usage de la calculatrice sans mémoire « type collègue » est autorisé.

Dès que ce sujet vous est remis, assurez-vous qu'il est complet.

Ce sujet comporte 4 pages numérotées de 1/4 à 4/4.

Sauf mention contraire, toute réponse devra être justifiée.

Le candidat est invité à faire figurer sur la copie toute trace de recherche, même incomplète ou non fructueuse, qu'il aura développée.

La qualité de la rédaction, la clarté et la précision des raisonnements seront prises en compte dans l'appréciation de la copie. Les traces de recherche, même incomplètes ou infructueuses seront valorisées.

BACCALAURÉAT GÉNÉRAL

SESSION 2026

MATHÉMATIQUES

ÉPREUVE ANTICIPÉE

Voie générale : candidats suivant l'enseignement de spécialité de mathématiques.

Durée de l'épreuve : **2 heures**

L'usage de la calculatrice n'est pas autorisé.

Dès que ce sujet vous est remis, assurez-vous qu'il est complet.

Ce sujet comporte 5 pages numérotées de 1/5 à 5/5.

Sauf mention contraire, toute réponse devra être justifiée.

Le candidat est invité à faire figurer sur la copie toute trace de recherche, même incomplète ou non fructueuse, qu'il aura développée.

La qualité de la rédaction, la clarté et la précision des raisonnements seront prises en compte dans l'appréciation de la copie. Les traces de recherche, même incomplètes ou infructueuses seront valorisées.

Thème

PROJETS, EN TEST

Dix-septième partie

En projet

84 Préambule, avertissement(s)



Dans cette section se trouvent des commandes en phases de test, donc non fonctionnelles à 100 %, mais qui pourront être utilisées (avec parcimonie) et qui pourront être amenées à être modifiées et/ou améliorées au fil du temps...donc il vaut mieux éviter de trop baser ses documents sur ces commandes...



Les commandes ne seront pas détaillées complètement, et seuls quelques exemples seront donnés, à titre indicatif.

Une fois les commandes *stabilisées*, elles passeront de manière *officielle*, avec potentiellement quelques modifications en vue !

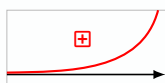


`\4.02a` : la section « Racines carrées, complexes » a été déplacée dans sa propre partie « Nombres complexes », les commandes concernées étant en réalité stables et chargées par défaut (`\useproflyclib{complexes}` fait partie du chargement standard) – elles n’avaient plus leur place ici.

85 Mini schémas de signe

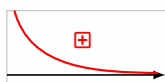
Code $\LaTeX$

```
\MiniSchemaSignes*[Code=expo+]
```



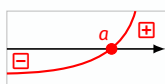
Code $\LaTeX$

```
\MiniSchemaSignes*[Code=expo-]
```



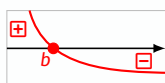
Code $\LaTeX$

```
\MiniSchemaSignes*[Code=exposol+,Racines={a}]
```

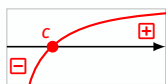


Code $\LaTeX$

```
\MiniSchemaSignes*[Code=exposol-,Racines={b}]
```



```
\MiniSchemaSignes*[Code=lnsol+,Racines={c}]
```



```
\PoseAddition[Base=bin]{110110}{1111}
```

$$\begin{array}{r} \overset{1}{} \overset{1}{} \overset{1}{} \overset{1}{} \overset{1}{} \\ 1 1 1 1 0 \\ + 1 1 1 1 \\ \hline = 1 0 0 1 0 1 \end{array}$$

```
\PoseAddition[Base=bin,Police=\sffamily]{110101}{1001}
```

$$\begin{array}{r} \\ \\ + \\ \hline = 1 1 1 1 \end{array}$$

```
\PoseAddition[Base=bin,Espacement=0pt]{1000000}{10000}
```

$$\begin{array}{r} 1000000 \\ + 10000 \\ \hline = 1010000 \end{array}$$

```
\PoseAddition
[Base=bin,Police=\Huge\ttfamily,CouleurRetenue=teal]
{1111}{111}
```

$$\begin{array}{r} \overset{1}{1} \overset{1}{1} \overset{1}{1} \overset{1}{1} \\ + \\ \hline = 1 \ 0 \ 1 \ 1 \ 0 \end{array}$$

```
\PoseAddition[Base=hex]{9ACD}{47F}
```

$$\begin{array}{r} 9 \text{ A C D} \\ + 4 \text{ 7 F} \\ \hline = 9 \text{ F 4 C} \end{array}$$

Code $\LaTeX$

```
\PoseAddition  
[Base=hex,Police=\LARGE\ttfamily,CouleurRetenue=olive]  
{F7}{D2}
```

$$\begin{array}{r} 1 \\ F\ 7 \\ + D\ 2 \\ \hline = 1\ C\ 9 \end{array}$$

Code $\LaTeX$

```
\PoseAddition  
[Base=hex,Police=\sffamily,CouleurRetenue=magenta,EchelleRetenue=0.75]  
{ACDC}{FFFF}
```

$$\begin{array}{r} 1\ 1\ 1\ 1 \\ A\ C\ D\ C \\ + F\ F\ F\ F \\ \hline = 1\ A\ C\ D\ B \end{array}$$

Code $\LaTeX$

```
\PoseAddition{9999}{999}
```

$$\begin{array}{r} 1\ 1\ 1\ 1 \\ 9\ 9\ 9\ 9 \\ + 9\ 9\ 9 \\ \hline = 1\ 0\ 9\ 9\ 8 \end{array}$$

Code $\LaTeX$

```
\PoseAddition  
[Police=\LARGE\ttfamily,CouleurRetenue=olive]  
{654}{123}
```

$$\begin{array}{r} 6\ 5\ 4 \\ + 1\ 2\ 3 \\ \hline = 7\ 7\ 7 \end{array}$$

86.2 Commande compatible en LUA

Code $\LaTeX$

```
\PoseAdditionLua{437+856+47}
```

$$\begin{array}{r} 1\ 1\ 2 \\ 4\ 3\ 7 \\ + 8\ 5\ 6 \\ + 4\ 7 \\ \hline = 1\ 3\ 4\ 0 \end{array}$$

```
\PoseAdditionLua{9999+999+99+9}
```

$$\begin{array}{r}
 \text{1} \quad \text{2} \quad \text{3} \quad \text{3} \\
 9999 \\
 + \quad 999 \\
 + \quad 99 \\
 + \quad 9 \\
 \hline
 = 11106
 \end{array}$$

```
\PoseAdditionLua[Base=bin]
{1111+1011+1111}
```

$$\begin{array}{r}
 \begin{array}{cccccc}
 & & 1 & 10 & 10 & 10 & 1 \\
 & & & 1 & 1 & 1 & 1 \\
 + & & & 1 & 0 & 1 & 1 \\
 + & & & 1 & 1 & 1 & 1 \\
 \hline
 = & 1 & 0 & 1 & 0 & 0 & 1
 \end{array}
 \end{array}$$

```
\PoseAdditionLua[Base=bin,Police=\LARGE\sffamily]
{1111+1011+1111+10}
```

$$\begin{array}{rcccccc}
 & & 1 & 10 & 10 & 10 & 1 \\
 & & 1 & 1 & 1 & 1 \\
 + & & 1 & 0 & 1 & 1 \\
 + & & 1 & 1 & 1 & 1 \\
 + & & & & 1 & 0 \\
 \hline
 = & 1 & 0 & 1 & 0 & 1 & 1
 \end{array}$$

```
\PoseAdditionLua
[Base=bin,Police=\Huge\ttfamily,CouleurRetenue=teal,
Egal=false,EchelleRetenue=0.3]
{1111+111}
```

$$\begin{array}{r} \overset{1}{1} \overset{1}{1} \overset{1}{1} \overset{1}{1} \\ + \\ \hline 1 0 1 1 0 \end{array}$$

```
\DoubleFactorielle[Enonce,Complet]{14}
```

$$14!! = 2 \times 4 \times 6 \times 8 \times 10 \times 12 \times 14 = 645\,120$$

Code $\LaTeX$

```
\DoubleFactorielle[Enonce,Complet,Sens=d]{14}
```

$$14!! = 14 \times 12 \times 10 \times 8 \times 6 \times 4 \times 2 = 645\,120$$

Code $\LaTeX$

```
\DoubleFactorielle[Enonce,Complet]{13}
```

$$13!! = 1 \times 3 \times 5 \times 7 \times 9 \times 11 \times 13 = 135\,135$$

Code $\LaTeX$

```
\DoubleFactorielle[Enonce,Complet,Sens=d]{13}
```

$$13!! = 13 \times 11 \times 9 \times 7 \times 5 \times 3 \times 1 = 135\,135$$

Code $\LaTeX$

```
\HyperFactorielle[Enonce,Complet]{7}
```

$$H(7) = 1^1 \times 2^2 \times 3^3 \times 4^4 \times 5^5 \times 6^6 \times 7^7 = 3\,319\,766\,398\,771\,200\,000$$

Code $\LaTeX$

```
\HyperFactorielle[Enonce,Partiel,Sens=d,Grand]{7}
```

$$H(7) = 7^7 \times 6^6 \times \dots \times 2^2 \times 1^1 \approx 3,319\,766\,40 \times 10^{18}$$

Code $\LaTeX$

```
\SuperFactorielle[Enonce]{4}
```

$$sf(4) = 288$$

Code $\LaTeX$

```
\SuperFactorielle[Enonce]{6}
```

$$sf(6) = 24\,883\,200$$

Code $\LaTeX$

```
\SuperFactorielle[Enonce,Partiel]{7}
```

$$sf(7) = 1! \times 2! \times \dots \times 6! \times 7! = 125\,411\,328\,000$$

Code $\LaTeX$

```
\SuperFactorielle[Enonce,Complet,Sens=d]{7}
```

$$sf(7) = 7! \times 6! \times 5! \times 4! \times 3! \times 2! \times 1! = 125\,411\,328\,000$$

88 Binôme de Newton

Code $\LaTeX$

```
\BinomeNewton{a}{b}{n}
```

$$(a + b)^n = \sum_{k=0}^n \binom{n}{k} a^k b^{n-k} = a^n + \binom{n}{1} a^{n-1} b + \binom{n}{2} a^{n-2} b^2 + \dots + \binom{n}{n-2} a^2 b^{n-2} + \binom{n}{n-1} a b^{n-1} + b^n$$

Code $\LaTeX$

`\BinomeNewton*[Complet]{a}{b}{n}`

$$(a+b)^n = \sum_{k=0}^n \binom{n}{k} a^k b^{n-k} = \binom{n}{0} a^n b^0 + \binom{n}{1} a^{n-1} b^1 + \binom{n}{2} a^{n-2} b^2 + \dots + \binom{n}{n-2} a^2 b^{n-2} + \binom{n}{n-1} a^1 b^{n-1} + \binom{n}{n} a^0 b^n$$

Code $\LaTeX$

`\BinomeNewton*[Complet,NbTermes=2,Indice=i]{a}{b}{n}`

$$(a+b)^n = \sum_{i=0}^n \binom{n}{i} a^i b^{n-i} = \binom{n}{0} a^n b^0 + \binom{n}{1} a^{n-1} b^1 + \dots + \binom{n}{n-1} a^1 b^{n-1} + \binom{n}{n} a^0 b^n$$

Code $\LaTeX$

`\BinomeNewton*[NbTermes=4,Somme=false]{x}{y}{u}`

$$(x+y)^u = x^u + \binom{u}{1} x^{u-1} y + \binom{u}{2} x^{u-2} y^2 + \binom{u}{3} x^{u-3} y^3 + \dots + \binom{u}{u-3} x^3 y^{u-3} + \binom{u}{u-2} x^2 y^{u-2} + \binom{u}{u-1} x y^{u-1} + y^u$$

Code $\LaTeX$

`\BinomeNewton*[Tout,Simplif]{a}{b}{5}`

$$(a+b)^5 = \sum_{k=0}^5 \binom{5}{k} a^k b^{5-k} = a^5 + 5a^4 b + 10a^3 b^2 + 10a^2 b^3 + 5ab^4 + b^5$$

Code $\LaTeX$

`\BinomeNewton[Tout,Simplif,Somme=false]{x}{y}{8}`

$$(x+y)^8 = x^8 + 8x^7 y + 28x^6 y^2 + 56x^5 y^3 + 70x^4 y^4 + 56x^3 y^5 + 28x^2 y^6 + 8xy^7 + y^8$$

Code $\LaTeX$

`\BinomeNewtonDetails{a}{b}{5}`

$$\begin{aligned} (a+b)^5 &= \sum_{k=0}^5 \binom{5}{k} a^k b^{5-k} \\ &= \binom{5}{0} a^5 b^0 + \binom{5}{1} a^4 b^1 + \binom{5}{2} a^3 b^2 + \binom{5}{3} a^2 b^3 + \binom{5}{4} a^1 b^4 + \binom{5}{5} a^0 b^5 \\ &= a^5 + 5a^4 b + 10a^3 b^2 + 10a^2 b^3 + 5ab^4 + b^5 \end{aligned}$$

Code $\LaTeX$

`\BinomeNewtonDetails*[Indice=\zeta]{u}{v}{9}`

$$\begin{aligned} (u+v)^9 &= \sum_{\zeta=0}^9 \binom{9}{\zeta} u^{\zeta} v^{9-\zeta} \\ &= \binom{9}{0} u^9 v^0 + \binom{9}{1} u^8 v^1 + \binom{9}{2} u^7 v^2 + \binom{9}{3} u^6 v^3 + \binom{9}{4} u^5 v^4 + \binom{9}{5} u^4 v^5 + \binom{9}{6} u^3 v^6 + \binom{9}{7} u^2 v^7 + \binom{9}{8} u^1 v^8 + \binom{9}{9} u^0 v^9 \\ &= u^9 + 9u^8 v + 36u^7 v^2 + 84u^6 v^3 + 126u^5 v^4 + 126u^4 v^5 + 84u^3 v^6 + 36u^2 v^7 + 9uv^8 + v^9 \end{aligned}$$

Thème

HISTORIQUE

Dix-huitième partie

Historique, depuis la version 4.00

v4.02c : Arbres de probas flexibles (154)

v4.02b : Mise en conformité avec siunitx + correction de bugs mineurs

v4.02a : Nettoyage des librairies et du code de chargement

v4.01d : Correction de bugs mineurs

v4.01c : Modifications mineures de clés piton

v4.01b : Correction de bugs mineurs

v4.01a : Correction de bugs mineurs

v4.00f : Modifications mineures de styles

v4.00e : Correction de bugs mineurs

v4.00d : Compléments sur les pages de garde des examens (247)

v4.00c : Compléments sur les facteurs premiers (196)

v4.00b : Améliorations dans la traduction + ajout en géométrie analytique (112)

v4.00a : Consolidation de macros en $\text{\LaTeX}$ 3 + Support *multilangue* en test (13)
