

ProfLycee

Quelques *petites* commandes pour \LaTeX (au lycée)

Cédric Pierquet

c pierquet -- at -- outlook . fr


Version 3.01f – 21 janvier 2024

Résumé : Quelques commandes pour faciliter l'utilisation de \LaTeX pour les mathématiques, au lycée.

- ✦ résoudre, de manière approchée, des équations
- ✦ calculer (et représenter) une valeur approchée d'une intégrale
- ✦ tracer *facilement* des repères/grilles/courbes
- ✦ tracer des courbes *lisses* avec gestion des extrema et des dérivées
- ✦ 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, créer la *toile* pour une suite récurrente
- ✦ afficher et utiliser un cercle trigo
- ✦ afficher un petit schéma pour le signe d'une fonction affine ou d'un trinôme
- ✦ travailler sur les statistiques à deux variables (algébriques et graphiques)
- ✦ tracer un histogramme, avec classes régulières ou non
- ✦ 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 »
- ✦ générer des listes d'entiers aléatoires (avec ou sans répétitions)
- ✦ déterminer la mesure principale d'un angle, calculer les lignes trigonométriques d'angles « classiques »
- ✦ résoudre une équation diophantienne « classique »
- ✦ travailler avec un peu de géométrie analytique
- ✦ composer des mathématiques
- ✦ travailler sur des intervalles
- ✦ afficher quelques fractales classiques
- ✦ arbre des diviseurs d'un entier
- ✦ ...

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!

\LaTeX

pdf \LaTeX

Lua \LaTeX

TikZ

TeXLive

MiKTeX

Table des matières

I	Introduction	10
1	Le package ProfLycee	10
1.1	« Philosophie » du package	10
1.2	Chargement du package	10
1.3	Librairies	11
1.4	Gestion des fontes	11
2	Compléments	12
2.1	Le système de « clés/options »	12
2.2	Compilateur(s)	12
2.3	Problèmes éventuels...	12
II	Liste des commandes, par thème	15
III	Écritures d'objets mathématiques	21
3	Introduction	21
4	Commandes	21
4.1	Arrondi	21
4.2	Ensembles et intervalles	21
4.3	Repères et coordonnées	22
4.4	Divers	23
4.5	Probabilités	24
5	Collection d'objets	26
5.1	Idée	26
5.2	Commande et options	26
IV	Outils pour l'analyse	28
6	Résolution approchée d'une équation	28
6.1	Idée	28
6.2	Clés et options	28
7	Présentation d'une solution d'équation par balayage	30
7.1	Idée	30
7.2	Clés et arguments	30
7.3	Interaction avec la commande de résolution approchée	31
8	Suites récurrentes simples	32
8.1	Idées	32
8.2	Clés et arguments	32
8.3	Exemple d'utilisation	33
9	Valeur approchée d'une intégrale	34
9.1	Idée	34
9.2	Clés et arguments	34
9.3	Exemples	35

V Outils graphiques	37
10 Intervalles	37
10.1 Idée	37
10.2 Création de l'environnement	37
10.3 Représentation d'intervalles	38
11 Repérage et tracé de courbes	40
11.1 Idée	40
11.2 Commandes, clés et options	41
11.3 Commandes annexes	44
11.4 Repère non centré en O	45
12 L'outil « SplineTikz »	47
12.1 Courbe d'interpolation	47
12.2 Code, clés et options	47
12.3 Compléments sur les coefficients de « compensation »	47
12.4 Exemples	48
12.5 Avec une gestion plus fine des « coefficients »	49
12.6 Conclusion	49
13 Génération de la courbe d'interpolation	50
13.1 Intro	50
13.2 Exemples et illustrations	51
14 L'outil « TangenteTikz »	53
14.1 Définitions	53
14.2 Exemple et illustration	53
14.3 Exemple avec les deux outils, et « personnalisation »	54
15 Points de discontinuité	55
15.1 Idée	55
15.2 Commandes	55
15.3 Exemples	55
16 Petits schémas pour le signe d'une fonction affine ou d'un trinôme	56
16.1 Idée	56
16.2 Commandes	56
16.3 Intégration avec tkz-tab	58
17 Suites récurrentes et « toile »	59
17.1 Idée	59
17.2 Commandes	59
17.3 Exemples	60
17.4 Influence des paramètres	61
18 Méthodes graphiques et intégrales	62
18.1 Idée	62
18.2 Clés et arguments	62
18.3 Exemples	64
VI Présentation de codes	67
19 Précautions	67

20 Code Python « simple » via le package listings	67
20.1 Introduction	67
20.2 Commande et options	67
20.3 Insertion via un fichier « externe »	68
20.4 Exemples	68
21 Code Python via le package piton	71
21.1 Introduction	71
21.2 Présentation de code Python	71
21.3 Console en partenariat avec Pyluatex	73
22 Code & Console Python, via les packages Pythontex ou Minted	74
22.1 Bibliothèques	74
22.2 Introduction	74
22.3 Présentation de code Python grâce au package pythontex	74
22.4 Présentation de code Python via le package minted	75
22.5 Console d'exécution Python	76
23 Pseudo-Code	78
23.1 Introduction	78
23.2 Présentation de Pseudo-Code	78
23.3 Compléments	79
24 PseudoCode via le package piton	80
24.1 Introduction	80
24.2 Présentation de PseudoCode	80
24.3 Exemples	81
25 Cartouche Capytale	83
25.1 Introduction	83
25.2 Commandes	83
26 Présentation de code \LaTeX	84
26.1 Introduction	84
26.2 Commandes	84
VII Outils pour la géométrie	86
27 Pavé droit « simple »	86
27.1 Introduction	86
27.2 Commandes	86
27.3 Influence des paramètres	87
28 Tétraèdre « simple »	88
28.1 Introduction	88
28.2 Commandes	88
28.3 Influence des paramètres	89
29 Cercle trigo	90
29.1 Idée	90
29.2 Commandes	90
29.3 Équations trigos	91
VIII Outils pour la géométrie analytique	94
30 Conseils d'utilisation	94

31 Affichage de coordonnées	94
31.1 Idée	94
31.2 Options et arguments	95
32 Équation cartésienne d'un plan de l'espace	96
32.1 Idée et commande	96
32.2 Clés et arguments	96
33 Équation paramétrique d'une droite de l'espace	98
33.1 Idée et commande	98
33.2 Clés et arguments	98
34 Équation cartésienne d'une droite du plan	100
34.1 Idée et commande	100
34.2 Clés et arguments	100
35 Norme d'un vecteur, distance entre deux points	102
35.1 Idée et commande	102
35.2 Clés et arguments	102
36 Distance d'un point à un plan	103
36.1 Idée et commande	103
36.2 Clés et arguments	103
37 Équation réduite d'une droite du plan	105
37.1 Idée	105
37.2 Clés et arguments	105
37.3 Exemples	105
IX Outils pour les statistiques	108
38 Paramètres d'une régression linéaire par la méthode des moindres carrés	108
38.1 Idée	108
38.2 Commandes	108
38.3 Intégration dans un environnement TikZ	110
39 Statistiques à deux variables	112
39.1 Idées	112
39.2 Commandes, clés et options	113
39.3 Commandes annexes	116
39.4 Interactions avec CalculsRegLin	117
39.5 Exemple complémentaire, pour illustration	120
40 Boîtes à moustaches	122
40.1 Introduction	122
40.2 Clés et options	122
40.3 Commande pour placer un axe horizontal	123
41 Histogrammes	125
41.1 Introduction	125
41.2 Clés et options	126
41.3 Exemple avec des classes régulières	127
41.4 Exemple avec des classes non régulières	128

42 Courbe des ECC/FCC, paramètres	130
42.1 Introduction	130
42.2 Clés et options	132
42.3 Styles et exemples	132
 X Outils pour les probabilités	 135
43 Calculs de probabilités	135
43.1 Introduction	135
43.2 Calculs « simples »	135
43.3 Complément avec sortie « formatée »	137
44 Arbres de probabilités « classiques »	139
44.1 Introduction	139
44.2 Options et arguments	139
44.3 Exemples complémentaires	140
45 Petits schémas pour des probabilités continues	142
45.1 Idée	142
45.2 Commandes et options	142
45.3 Remarques et compléments	143
46 Nombres aléatoires	144
46.1 Idée	144
46.2 Clés et options	145
47 Combinatoire	146
47.1 Idée	146
47.2 Utilisation	146
48 Fonction de répartition	147
48.1 Idée	147
48.2 Utilisation	147
 XI Outils pour l'arithmétique	 150
49 Division euclidienne	150
49.1 Idée	150
49.2 Clés et options	150
50 Conversions binaire/hexadécimal/décimal	151
50.1 Idée	151
50.2 Conversion décimal vers binaire	151
50.3 Conversion binaire vers hexadécimal	152
50.4 Conversion hexadécimal vers binaire	153
50.5 Conversion binaire ou hexadécimal en décimal	153
51 Conversion « présentée » d'un nombre en base décimale	155
51.1 Idée	155
51.2 Code et clés	155
52 Algorithme d'Euclide pour le PGCD	157
52.1 Idée	157
52.2 Options et clés	157
52.3 Compléments	158

53 Résolution d'une équation diophantienne	159
53.1 Idée	159
53.2 Options et clés	159
54 Diviseurs	163
54.1 Idées	163
54.2 Options et clés	163
55 Chiffrements	166
55.1 Idées	166
55.2 Chiffrement de César	166
55.3 Inverse modulo	167
55.4 Chiffrement affine	167
55.5 Chiffrement de Hill	168
XII Écritures, simplifications	171
56 Simplification sous forme d'une fractions	171
56.1 Idée	171
56.2 Commande et options	171
57 Écriture d'un trinôme, trinôme aléatoire	173
57.1 Idée	173
57.2 Clés et options	173
58 Simplification de racines	174
58.1 Idée	174
58.2 Exemples	174
59 Mesure principale d'un angle	175
59.1 Idée	175
59.2 Exemples	175
60 Lignes trigonométriques	176
60.1 Idée	176
60.2 Commande	176
60.3 Valeurs disponibles	178
61 Écriture sous forme de fraction irréductible d'un décimal périodique	179
61.1 Idées	179
61.2 Options et clés	179
XIII Jeux et récréations	182
62 SudoMaths, en TikZ	182
62.1 Introduction	182
62.2 Clés et options	183
63 Quelques fractales, en TikZ	186
63.1 Introduction	186
63.2 Flocon de Koch et triangle de Sierpinski	186
63.3 Affichage de plusieurs étapes pour les flocons de Koch	188
63.4 Affichage de plusieurs étapes pour les tapis de Sierpinski	189

64 Châteaux de cartes	190
64.1 Introduction	190
64.2 Clés et options	190
65 Allumettes	192
65.1 Introduction	192
65.2 Clés et options	192
66 Machines à transformer, en TikZ	194
66.1 Introduction	194
66.2 Clés et options	195
 XIV Historique	 199

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 les packages :

- `\mathtools`, `\amssymb`, `\xspace`, `\esvect`, `\interval`;
- `\tikz`, `\pgf`, `\pgffor`, `\nicefrac`, `\nicematrix`;
- `\tcolorbox` avec les bibliothèques `breakable`, `fitting`, `skins`, `listings`, `listingsutf8`, `hooks`;
- `\xparse`, `\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 \LaTeX avec `\listofitems`, `\randomlist`, `\simplekv` ou encore `\xstring`!

1.2 Chargement du package



Code \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 \LaTeX

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

1.3 Bibliothèques



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 `tclobox` ou `tikz`, dans le préambule, et une fois le package appelé.



</> Code \LaTeX

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



Les bibliothèques disponibles seront indiquées dans les sections spécifiques. Pour le moment, il existe :

- **python** (page 71);
- **minted** (page 75);
- **pythontex** (page 74);
- **ecritures** (page 21).



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



En compilant (notamment avec les bibliothèques **minted** et **pythontex**) on peut spécifier des répertoires particuliers pour les (ou des) fichiers auxiliaires.

Avec l'option **<build>**, l'utilisateur a la possibilité de placer les fichiers temporaires de **minted** et **pythontex** dans un répertoire **build** du répertoire courant.

Dans ce cas il faut créer au préalable le répertoire **build** avant de compiler un fichier, pour éviter toute erreur!



</> Code \LaTeX

```
...
\usepackage[build]{ProfLycee}
\useproflyclub{...}
...
```



L'option **<build>** charge certains packages (bibliothèques **minted** et **pythontex**) avec les options :

- \LaTeX `\setpythontexoutdir{./build/pythontex-files-\jobname}`
- \LaTeX `\RequirePackage[outputdir=build]{minted}`

1.4 Gestion des fontes



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

Sous \XeTeX & \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**).



</> Code \LaTeX

```
\documentclass{article}
\usepackage[nonamssymb]{ProfLycee}
\usepackage{fourier-otf}
```

2 Compléments

2.1 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*.



Les commandes et environnements présentés seront explicités via leur syntaxe avec les options/clés ou arguments.

Autant que faire se peut, des exemples/illustrations/remarques seront proposés à chaque fois.



À noter que certaines commandes disponibles sont liées à un environnement `tikzpicture`, elles peuvent ne pas être autonomes mais permettent de conserver – en parallèle – toute commande liée à TikZ!

2.2 Compilateur(s)



Le package `ProfLycee` est compatible avec les compilateurs classiques : latex, pdflatex ou encore lua-latex.

En ce qui concerne les codes librairies, il faudra :

- `pythontex` : compiler en chaîne (xxx)latex + pythontex + (xxx)latex;
- `minted` : compiler avec shell-escape (ou write18);
- `piton` : compiler en Lua \LaTeX et shell-escape (ou write18).

2.3 Problèmes éventuels...



Certaines commandes sont à intégrer dans un environnement TikZ, afin de pouvoir rajouter des éléments, elles ont été testées dans des environnements `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!

↔ Bonne(s) découverte(s) ↔

Thème

LISTE DES COMMANDES

Deuxième partie

Liste des commandes, par thème



2.0.0 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.



</> Code \LaTeX

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



</> Code \LaTeX

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

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

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

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

%Valeur approchée d'une intégrale
\IntegraleApprochee[clés]{fonction}{a}{b}
```



</> Code \LaTeX

```
%fenêtre de repérage en tikz et courbe
\GrilleTikz[options][options grille ppale][options grille second.]
\AxesTikz[options] \AxexTikz[options]{valeurs} \AxeYtikz[options]{valeurs}
\FenetreSimpleTikz[options](opt axes)<opt axe Ox>{liste valx}<opt axe Oy>{liste valy}
\DeclareFonctionTikz[nom]{expr}
\CourbeTikz[options]{fonction}{valxmin:valxmax}

%génération du tracé d'une courbe d'interpolation, dans une commande tikz
\GenereSplineTikz[options]{liste}[\nomdutracé]

%courbe d'interpolation, tangente, dans un environnement tikz
\SplineTikz[options]{liste}
\TangenteTikz[options]{liste}
\PtsDiscontinuite[options]{liste}

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

%intégrales et méthodes graphiques
\IntegraleApprocheeTikz[clés]{nom_fonction}{a}{b}
```



</> Code L^AT_EX

```

%présentation de code Python
\begin{CodePythonLst}(*)[clés]{commandes tcbbox}...\end{CodePythonLst}
\begin{CodePythonLstAlt}(*)[clés]{commandes tcbbox}...\end{CodePythonLstAlt}
%:=librairie piton
\begin{CodePiton}[options piton]{commandes tcbbox}<option line-numbers>...\end{CodePiton}
\begin{PitonConsole}<clés>{commandes tcbbox}...\end{PitonConsole}
%:=librairie pythontex
\begin{CodePythontex}[clés]{commandes tcbbox}...\end{CodePythontex}
\begin{CodePythontexAlt}[clés]{commandes tcbbox}...\end{CodePythontexAlt}
\begin{ConsolePythontex}[options]{...}\end{ConsolePythontex}
%:=librairie minted
\begin{CodePythonMinted}(*)[clés]{commandes tcbbox}...\end{CodePythonMinted}
\begin{CodePythonMintedAlt}(*)[largeur][clés]{commandes tcbbox}...\end{CodePythonMintedAlt}

%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}

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

```



</> Code L^AT_EX

```

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

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

```



</> Code L^AT_EX

```

%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)

%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)

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

%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)

%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)

%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}
```



</> Code \LaTeX

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

%stats à 2 variables, dans un environnement tikz
\GrilleTikz[options][options grille ppale][options grille second.]
\AxesTikz[options]
\AxexTikz[options]{valeurs} \AxeYtikz[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}

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

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

%courbe ECC/FCC
\CourbeECC[clés]{liste valeurs}{liste effectifs}
\begin{EnvCourbeECC}[clés]{liste valeurs}{liste effectifs}...\end{EnvCourbeECC}
\MedianeQuartilesECC{liste valeurs}{liste effectifs}
```



</> 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}

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



</> Code \LaTeX

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

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



</> Code \LaTeX

```

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

%loi exponentielle  $E(l)$ 
\CalcExpoC{l}{a}{b}
\ExpoC(*)[prec]{l}{a}{b}

%arbres de probas
\ArbreProbasTikz[options]{donnees}
\begin{EnvArbreProbasTikz}[options]{donnees}...\end{EnvArbreProbasTikz}

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

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

```



</> Code \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}{calcul}
%liste d'entiers aléatoires
\TirageAleatoireEntiers[options]{macro}
%liste des diviseurs
\ListeDiviseurs(*)[option]{nombre}
%arbre des diviseurs
\ArbreDiviseurs[options]{nombre}

```



</> Code \LaTeX

```

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

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

```



</> Code \LaTeX

```

%conversions
\ConversionDecBin(*)[clés]{nombre}
\ConversionBinHex[clés]{nombre}
\ConversionHexBin[clés]{nombre}
\ConversionVersDec[clés]{nombre}
\ConversionBaseDix[clés]{nombre}{base de départ}
\ConversionDepuisBaseDix[options]{nombre en base 10}{base d'arrivée}

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

%Équation diophantienne
\EquationDiophantienne[clés]{equation}

```



</> Code \LaTeX

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

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

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

%Chiffrement affine
\ChiffrementAffine[clés]{message}

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



</> Code \LaTeX

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

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

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

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



</> Code \LaTeX

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

%fractales
\FractaleTikz(*)[clés]<options tikz>
\EtapesFloconKoch[clés]{étapes}
\EtapesTapisSierpinski[clés]{étapes}

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

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



</> Code \LaTeX

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

Thème

ÉCRITURES MATHÉMATIQUES

Troisième partie

Écritures d'objets mathématiques

3 Introduction



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

4 Commandes

4.1 Arrondi



Il est possible de calculer/arrondir/formater un calcul mathématique, grâce aux package `\siunitx` et `\xinteval`.

- 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 `\xinteval`.



Code \LaTeX

```
$1+\dfrac{7}{11} \approx \Arroundi{1+7/11} \approx \Arroundi[5]{1+7/11} \approx \Arroundi*[2]{1+7/11}$
```

```
$\ln\big(1+e^4\big) \approx \Arroundi[6]{\log(1+\exp(4))}$
```

$$1 + \frac{7}{11} \approx 1,636 \approx 1,63636 \approx +1,64$$
$$\ln(1+e^4) \approx 4,018150$$

4.2 Ensembles et intervalles



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

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



Code \LaTeX

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

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


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 \IntervalleF0{\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 $\left]\frac{1}{2}; \sqrt{26}\right]$

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



Code \LaTeX

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

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

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

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



Code \LaTeX

```
%matrices 2x2
\MatDeux{4}{-2}{1}{-5}

\MatDeux{\frac{4}{7}}{-1}{0}{-\frac{1}{7}}
```

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

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



Code \LaTeX

```
%repères classiques
\RepereOij ou \RepereOij* ou \RepereOij[Sep={,}] ou \RepereOij*[Sep={,}]

\RepereOijk ou \RepereOijk* ou \RepereOijk[Sep={,}] ou \RepereOijk*[Sep={,}]

\RepereOuv ou \RepereOuv* ou \RepereOuv[Sep={,}] ou \RepereOuv*[Sep={,}]
```

$$(O; \vec{i}, \vec{j}) \text{ ou } (O; \vec{i}, \vec{j}) \text{ ou } (O, \vec{i}, \vec{j}) \text{ ou } (O, \vec{i}, \vec{j})$$

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

$$(O; \vec{u}, \vec{v}) \text{ ou } (O; \vec{u}, \vec{v}) \text{ ou } (O, \vec{u}, \vec{v}) \text{ ou } (O, \vec{u}, \vec{v})$$



Code \LaTeX

```
%repères personnalisés
\ReperePlan{A}{AB}{AC} ou \ReperePlan*{A}{AB}{AC} ou \ReperePlan*{O}{OI}{OJ} ou
\ReperePlan[Sep={,}]{O}{OI}{OK}

\RepereEspace{A}{AB}{AC}{AD} ou \RepereEspace*{A}{AB}{AC}{k} ou \RepereEspace{D}{i}{j}{k}
```

$$(A; \overrightarrow{AB}, \overrightarrow{AC}) \text{ ou } (A; \overrightarrow{AB}, \overrightarrow{AC}) \text{ ou } (O; \overrightarrow{OI}, \overrightarrow{OJ}) \text{ ou } (O, \overrightarrow{OI}, \overrightarrow{OK})$$

$$(A; \overrightarrow{AB}, \overrightarrow{AC}, \overrightarrow{AD}) \text{ ou } (A; \overrightarrow{AB}, \overrightarrow{AC}, \vec{k}) \text{ ou } (D; \vec{i}, \vec{j}, \vec{k})$$

4.4 Divers



La librairie `\usepackage{ecritures}` 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).

À noter que les commandes sont 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}]
```

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

$$\mathscr{C} \text{ et } \mathscr{C}_f \text{ et } \mathscr{C}_{g^{-1}}$$



</> Code \LaTeX

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

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



</> Code \LaTeX

On a $\$ \backslash j = \backslash jfalg = \backslash jfexp \$$

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



</> Code \LaTeX

%La version étoilée augmente l'espacement, l'argument optionnel change la présentation

$\$ 21 \ \backslash equiv \ 1 \ \backslash Modulo\{5\} \ \backslash equiv \ 1 \ \backslash Modulo[Par]\{5\} \ \backslash equiv \ 1 \ \backslash Modulo[Txt]\{5\} \$$

$\$ 21 \ \backslash equiv \ 1 \ \backslash Modulo*\{5\} \ \backslash equiv \ 1 \ \backslash Modulo*[Par]\{5\} \ \backslash equiv \ 1 \ \backslash Modulo*[Txt]\{5\} \$$

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

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



</> Code \LaTeX

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

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



</> Code \LaTeX

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

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

4.5 Probabilités



La librairie `\usepackage{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.

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}}

```

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



</> Code \LaTeX

```

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

```

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

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 {}  
$\EcritureEnsemble{ \frac{1}{1+\frac{1}{3}} / b / c / d / \frac{1}{2} }$
```



Sortie \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 Résolution approchée d'une équation

6.1 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 et sortie \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

6.2 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**
- $\langle \text{macro} \rangle d$ pour la valeur approchée par défaut;
- $\langle \text{macro} \rangle e$ pour la valeur approchée par excès;
- $\langle \text{macro} \rangle a$ pour la valeur approchée.



Code \LaTeX et sortie \LaTeX

```

\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  $\text{\text{e}}^{0,5x}+x^2-4=0$  sur
 $\left[0;2\right]$  est  $\beta$  avec :
\begin{itemize}
  \item  $\beta \approx \text{\num[minimum-decimal-digits=4]{\masolutiond}}$  par défaut ;
  \item  $\beta \approx \text{\num[minimum-decimal-digits=4]{\masolutione}}$  par excès ;
  \item  $\beta \approx \text{\num[minimum-decimal-digits=4]{\masolutiona}}$ .
\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  $3t\text{\rm{e}}^{-0,5t+1}=4$  est  $t_1$ 
avec :
\begin{itemize}
  \item  $t_1 \approx \text{\num[minimum-decimal-digits=2]{\SolAd}}$  par défaut ;
  \item  $t_1 \approx \text{\num[minimum-decimal-digits=2]{\SolAe}}$  par excès ;
  \item  $t_1 \approx \text{\num[minimum-decimal-digits=2]{\SolAa}}$ .
\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  $3t\text{\rm{e}}^{-0,5t+1}=4$  est  $t_2$ 
avec :
\begin{itemize}
  \item  $t_2 \approx \text{\num[minimum-decimal-digits=2]{\SolBd}}$  par défaut ;
  \item  $t_2 \approx \text{\num[minimum-decimal-digits=2]{\SolBe}}$  par excès ;
  \item  $t_2 \approx \text{\num[minimum-decimal-digits=2]{\SolBa}}$ .
\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 $e^{0,5x} + x^2 - 4 = 0$ sur $[0; 2]$ est β avec :

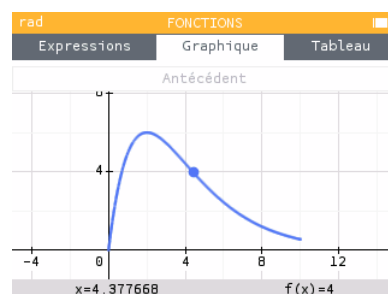
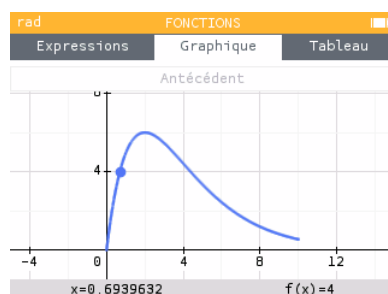
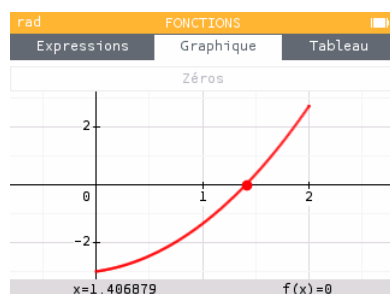
- $\beta \approx 1,4068$ par défaut ;
- $\beta \approx 1,4069$ par excès ;
- $\beta \approx 1,4069$.

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

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



7 Présentation d'une solution d'équation par balayage

7.1 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}
```

7.2 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 et sortie \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 et sortie \LaTeX

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



Avec $\varphi(t) = 3te^{-0,5t+1} = 5$, 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 et sortie \LaTeX

On s'intéresse à $g(x)=\text{num}\{1,5\}$ avec $g(x)=\ln(x)$.
 $\backslash\text{SolutionTVI}\%$
 $[\text{Balayage},\text{Stretch}=1.5,\text{va}=4.48,\text{vb}=4.49,\text{NomFct}=g,\text{Precision}=4,\text{NomSol}=\{x_0\}]{\log(x)}{1.5}.$



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

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

7.3 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 $\backslash\text{ResolutionApprochee}$.



Code \LaTeX et sortie \LaTeX

On s'intéresse à $g(x)=\text{num}\{1,5\}$ avec $g(x)=\ln(x)$ sur l'intervalle $[\text{left}\{3;5\}\text{right}\}$.
 $\backslash\text{ResolutionApprochee}[\text{Intervalle}=3:5]{\log(x)=1.5}[\text{SolLn}]$
 $\backslash\text{SolutionTVI}\%$
 $[\text{Balayage},\text{Stretch}=1.5,\text{va}=\{\text{SolLnd}\},\text{vb}=\{\text{SolLne}\},$
 $\text{NomFct}=g,\text{Precision}=4,\text{NomSol}=\{x_0\}]{\log(x)}{1.5}.$



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

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



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

8 Suites récurrentes simples

8.1 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 L^AT_EX

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

8.2 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 L^AT_EX

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



Sortie L^AT_EX

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

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

On obtient $u_{15} \approx 0,4142$

On obtient $u_{20} \approx 0,414214$

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⟩**
- le booléen **⟨Exact⟩** qui affiche $\frac{a}{b} =$ au lieu de $\frac{a}{b} \approx$; 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 $\frac{a}{b}$ `xint` et avec comme variable x , et le second est le seuil à dépasser.



Code \LaTeX et sortie \LaTeX

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



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

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 $\begin{cases} u_7 \approx 4,8681 \leq 5 \\ u_8 \approx 5,2089 > 5 \end{cases}.$

8.3 Exemple d'utilisation



Code \LaTeX et sortie \LaTeX

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

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



Avec
$$\begin{cases} u_1 = 2 \\ u_{n+1} = 1 + \frac{1+u_n^2}{1+u_n} \end{cases}, \text{ on obtient le tableau de valeurs suivant :}$$

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 $n = 29$)

9 Valeur approchée d'une intégrale

9.1 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 *Trapèzes* seront plutôt utiles pour des résultats obtenus par algorithme par exemple.



Code \LaTeX

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

9.2 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 et sortie \LaTeX

```
On s'intéresse à  $\int_4^{10} f(x) \, dx$  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 : $\int_4^{10} \sqrt{x} \, dx \approx 15,74852$

9.3 Exemples



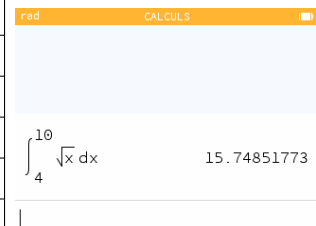
Code \LaTeX

```
%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}
$\displaystyle\IntegraleApprochee[NbSubDiv=10,AffFormule,Expr={\sqrt{x}}]{sqrt(x)}{4}{10}$
```



Sortie \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
$\int_4^{10} \sqrt{x} dx \approx 15,749$	



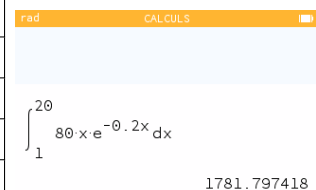
Code \LaTeX

```
%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}
$\displaystyle\IntegraleApprochee[NbSubDiv=100,AffFormule,Expr={80x\,\text{e}^{-0,2x}}]{80*x*exp(-0.2*x)}{1}{20}$
```



Sortie \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
$\int_1^{20} 80x e^{-0,2x} dx \approx 1781,797$	



Thème

OUTILS GRAPHIQUES

Cinquième partie

Outils graphiques

10 Intervalles

10.1 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 \LaTeX

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



Code \LaTeX et sortie \LaTeX

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



10.2 Création de l'environnement



Code \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 :

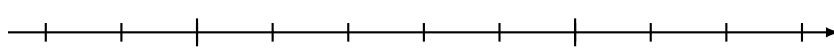
- **<xmin>** : valeur minimale pour l'axe; défaut **<-8>**
- **<xmax>** : valeur maximale pour l'axe; défaut **<8>**
- **<Elargir>** : coefficient d'élargissement de l'axe, en pourcentage de largeur; défaut **<0.05>**
- **<Largeur>** : largeur (en cm) de l'axe (hors élargissement); défaut **<12>**
- **<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.



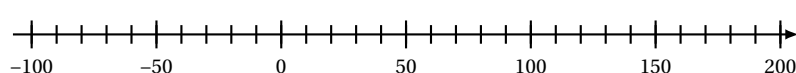
Code \LaTeX et sortie \LaTeX

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



Code \LaTeX et sortie \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}
```



10.3 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 \LaTeX

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



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

- **<Couleur>** : couleur de base des tracés; défaut **<red>**
- **<Type>** (parmi `\tikzFF / \tikzFO / \tikzOO / \tikzOF`) : 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 `\tikzfond / \tikzizzag / \tikzhach` : pour décorer l'intervalle (`\tikzhach/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 `\tikzsiunitx`; défaut **<true>**
- **<NumSup>** : booléen pour afficher la borne sup grâce à `\tikzsiunitx`; défaut **<true>**
- **<PosValeurs>** : choix de la position des valeurs des bornes. défaut **<above>**

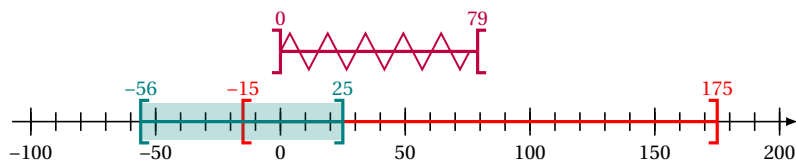
Les deux arguments obligatoires, et entre `\tikz{...}`, permettent de préciser les bornes de l'intervalle (en langage TikZ, et avec `\tikz*` pour l'infini.)

3.00b Les deux arguments optionnels correspondants, et entre `\tikz[...]`, 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>**.



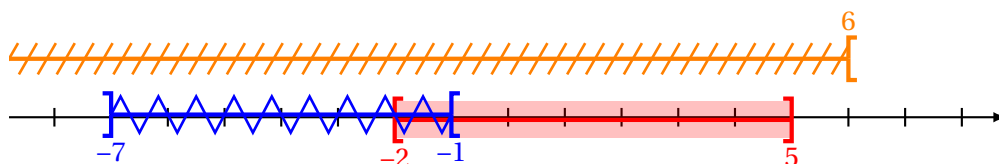
Code \LaTeX et sortie \LaTeX

```
\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=00]{0}{79}
\end{RepIntervalles}
```



Code \LaTeX et sortie \LaTeX

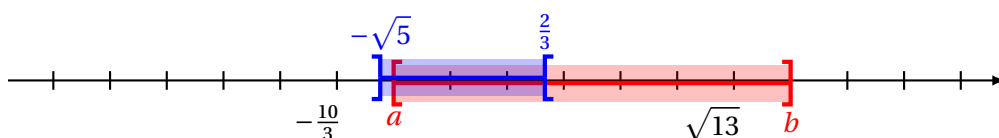
```
\begin{RepIntervalles}[Graduations={-8,-7,...,8}]
  \tkzIntervalle[Decor=fond,Offset=-1pt,AffValeurs,PosValeurs=below]{-2}{5}
  \tkzIntervalle[Type=00,Couleur=blue,Decor=zigzag,Offset=1pt,
    AffValeurs,PosValeurs=below]{-7}{-1}
  \tkzIntervalle[Type=F0,Couleur=orange,NiveauV=1.25,Decor=hach,AffValeurs]{*}{6}
\end{RepIntervalles}
```



Code \LaTeX et sortie \LaTeX

%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)}{2/3}{2/3}{\tfrac{2}{3}}
\end{RepIntervalles}
```



11 Repérage et tracé de courbes

11.1 Idée



2.1.1 L'idée est de proposer des commandes *simplifiées* pour tracer un repère, en TikZ, avec :

- axes et graduations, grille;
- courbe.



Au niveau du code, il y aura donc plusieurs *aspects* :

- le paramétrage de la fenêtre graphique directement dans la déclaration de l'environnement;
- les commandes de tracés avec options et clés.



</> Code \LaTeX

```
%version basique
\begin{tikzpicture}[paramètres]
  %grille et axes
  \GrilleTikz[options][options grille ppale][options grille second.]
  \AxesTikz[options]
  \AxexTikz[options]{valeurs}
  \AxeYtikz[options]{valeurs}
  %courbe
  \CourbeTikz[options]{fonction}{valxmin:valxmax}
\end{tikzpicture}
```

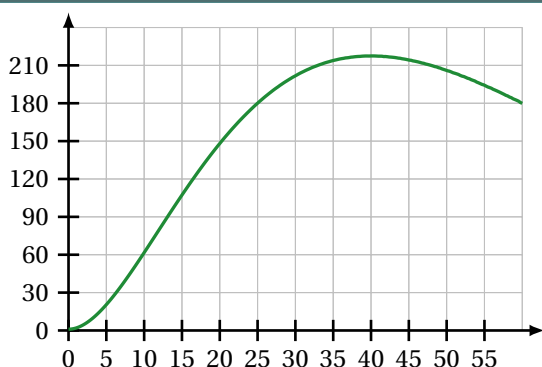


</> Code \LaTeX

```
%version simplifiée
\begin{tikzpicture}[<paramètres>]
  %grille et axes
  \FenetreSimpleTikz[opt](opt axes)<opt axe Ox>{liste valx}<opt axe Oy>{liste valy}
  %courbe
  \CourbeTikz[options]{fonction}{valxmin:valxmax}
\end{tikzpicture}
```



Sortie \LaTeX



11.2 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*.



</> Code \LaTeX

```
%...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>**

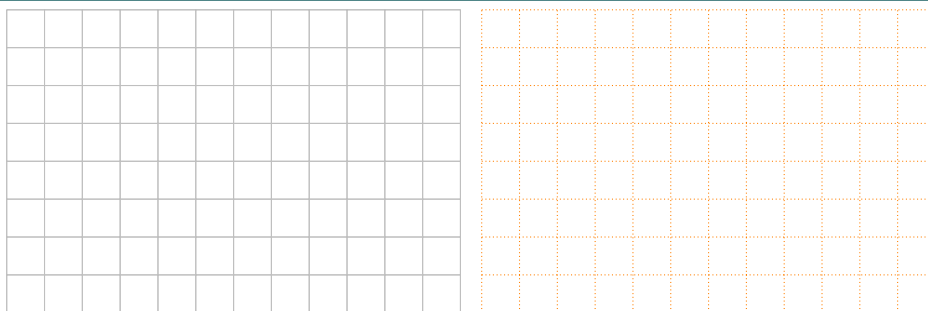


</> Code \LaTeX

```
\begin{tikzpicture}%
[x=0.1cm,y=0.0167cm, %unités
xmin=0,xmax=60,xgrille=5,xgrilles=5, %axe Ox
ymin=0,ymax=240,ygrille=30,ygrilles=30] %axe Oy
\GrilleTikz
\end{tikzpicture}
~~
\begin{tikzpicture}%
[x=0.1cm,y=0.0167cm, %unités
xmin=0,xmax=60,xgrille=5,xgrilles=5, %axe Ox
ymin=0,ymax=240,ygrille=30,ygrilles=30] %axe Oy
\GrilleTikz[Affp=false][orange,densely dotted]
\end{tikzpicture}
```



Sortie \LaTeX





Code \LaTeX

```
%...code tikz
\AxesTikz[options]
```



Cette commande permet de tracer les axes, avec des **clés** :

- **Épaisseur** qui est l'épaisseur des axes; défaut **1pt**
- **Police** qui est le style des labels des axes; défaut **$\backslash\text{normalsize}\backslash\text{normalfont}$**
- **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**



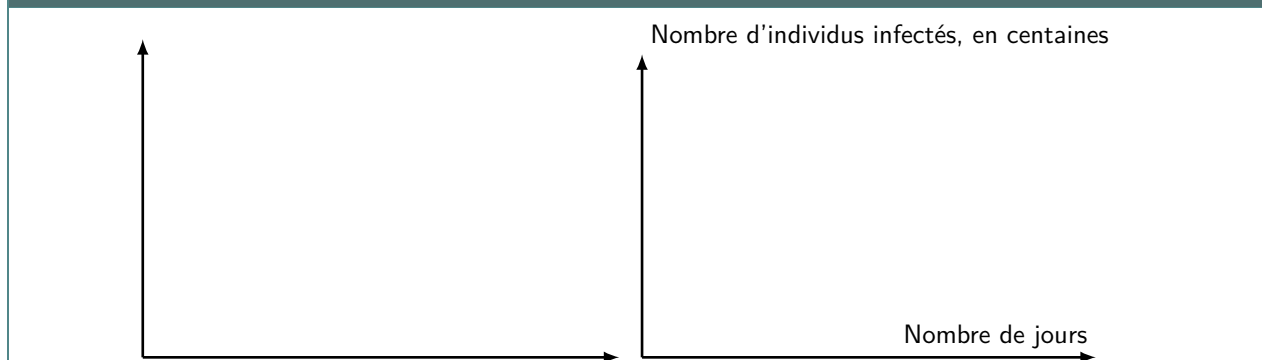
Code \LaTeX

```
%code tikz
\AxesTikz

%code tikz
\AxesTikz%
[AffLabel=xy,Labelx={Nombre de jours},Labely={Nombre d'individus infectés, en centaines},%
PosLabelx={above left},PosLabely={above right},%
Police=\small\sffamily,ElargirOx=0,ElargirOy=0]
```



Sortie \LaTeX





Code \LaTeX

```
%...code tikz
\AxexTikz[options]{valeurs}
\AxyTikz[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 **$\backslash\text{normalsize}\backslash\text{normalfont}$**
- **PosGrad** qui est la position des graduations par rapport à l'axe; défaut **below** et **left**
- **HautGrad** qui est la hauteur 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**
- \maj 2.5.6 le booléen **Trigo** (uniquement pour l'axe (Ox)) pour des graduations libres en radians; défaut **false**
- \maj 2.5.6 le booléen **Dfrac** (uniquement pour l'axe (Ox) en **Trigo**) pour forcer les fractions en *grand*; défaut **false**
- \maj 2.7.0 le booléen **Frac** (uniquement pour l'axe (Oy)) pour forcer les graduations en fraction (taille normale). défaut **false**

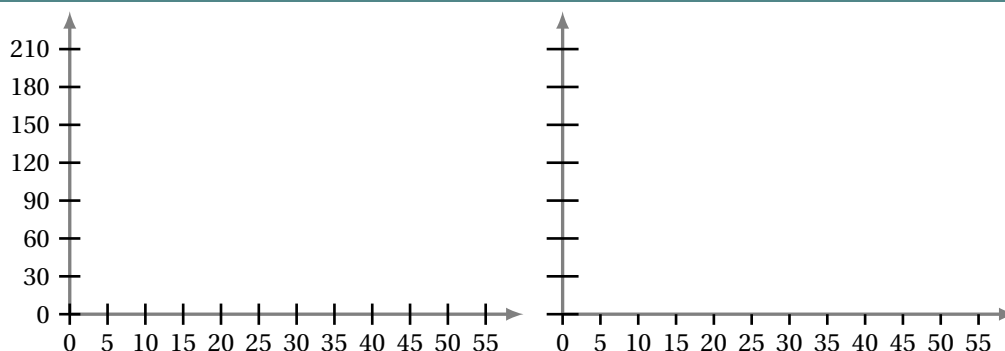


Code \LaTeX

```
%code tikz
\AxexTikz[Police=\small]{0,5,...,55}
\AxyTikz[Police=\small]{0,30,...,210}
%code tikz
\AxexTikz[Police=\small,HautGrad=0pt/4pt]{0,5,...,55}
\AxyTikz[AffGrad=false,HautGrad=6pt]{0,30,...,210}
%des axes fictifs (en gris) sont rajoutés pour la lisibilité du code de sortie
```



Sortie \LaTeX



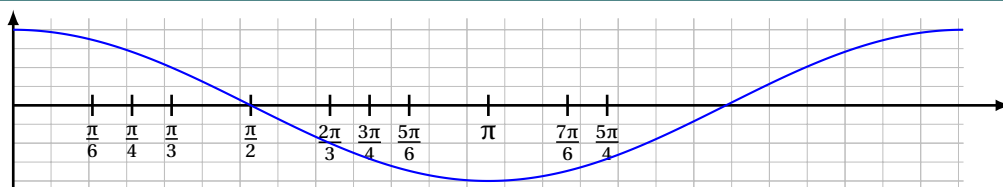


Code \LaTeX

```
\begin{tikzpicture}[x=2cm,y=1cm,xmin=0,xmax={2*pi},xgrille=0.5,xgrilles=0.25,
  ymin=-1.15,ymax=1.15,ygrille=0.5,ygrilles=0.25]
  \GrilleTikz \AxesTikz
  \AxexTikz[Trigo]{{pi/6},{pi/4},{pi/3},{pi/2},{2*pi/3},%
    {3*pi/4},{5*pi/6},pi,{7*pi/6},{5*pi/4}}
  \CourbeTikz[thick,blue,samples=250]{cos(deg(\x))}{0:2*pi}
\end{tikzpicture}
```



Sortie \LaTeX



La clé **Trigo** utilise, en interne, une commande qui permet de *transformer* les abscisses, données en langage TikZ, en fraction en \LaTeX .



Code \LaTeX et sortie \LaTeX

```
$\AffAngleRadian{0}$ \quad $\AffAngleRadian{pi}$ \quad $\AffAngleRadian{pi/4}$ \quad
$\AffAngleRadian{2*pi/3}$ \quad $\AffAngleRadian{-2*pi/3}$ \quad $\AffAngleRadian*{-2*pi/3}$
```



0 π $\frac{\pi}{4}$ $\frac{2\pi}{3}$ $-\frac{2\pi}{3}$ $-\frac{2\pi}{3}$

11.3 Commandes annexes



Il existe, de manière marginale, quelques commandes complémentaires qui ne seront pas trop détaillées mais qui sont existantes :

- `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 \LaTeX

```
%code tikz
\FenetreTikz %on restreint les tracés
\FenetreSimpleTikz%
[options](opt axes)<opt axe Ox>{valeurs Ox}<opt axe Oy>{valeurs Oy}
```



L'idée est de proposer, en *complément*, une commande simplifiée pour tracer une courbe en TikZ.



Code \LaTeX

```
%...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 $\backslash x$ comme variable ;
- le second *obligatoire* est le domaine du tracé, sous la forme $\backslash valxmin:valxmax$.

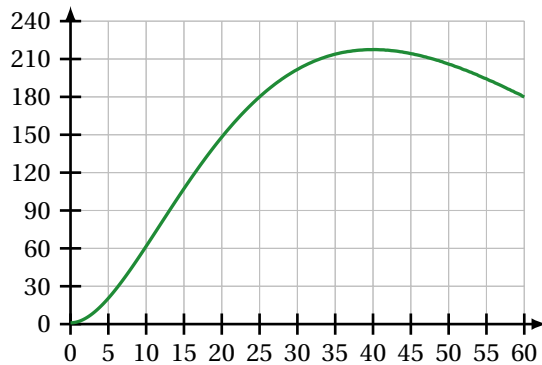


Code \LaTeX

```
\begin{tikzpicture}[x=0.1cm,y=0.0167cm, %unités
  xmin=0,xmax=60,xgrille=5,xgrilles=5, %axe Ox
  ymin=0,ymax=240,ygrille=30,ygrilles=30] %axe Oy
  \FenetreSimpleTikz%
    <Police=\small>{0,5,...,60}%
    <Police=\small>{0,30,...,240} %repère
  \CourbeTikz[line width=1.25pt,CouleurVertForet,samples=250]%
    {\x*\x*exp(-0.05*\x)+1}{0:60} %courbe
\end{tikzpicture}
```



Sortie \LaTeX



11.4 Repère non centré en O

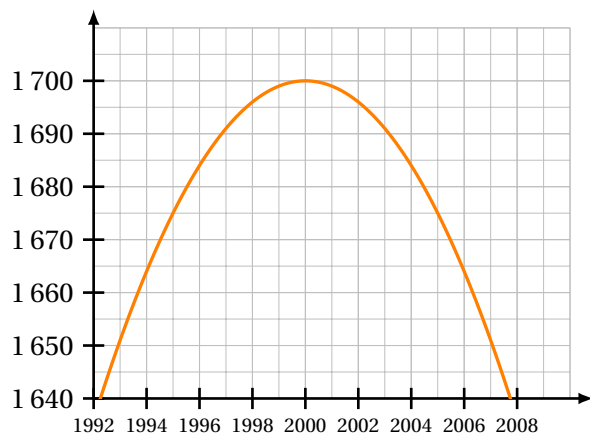


Parfois on est amené à travailler dans des repères qui n'ont pas forcément pour origine (0;0). De ce fait - pour éviter des erreurs de `dimension too large` liées à TikZ - il faut *décaler les axes* pour se ramener à une origine en O. L'idée est donc d'utiliser les commandes précédentes, sans se soucier des éventuelles transformations !



Code \LaTeX et sortie \LaTeX

```
\begin{tikzpicture}[x=0.35cm,y=0.07cm,Ox=1992,xmin=1992,xmax=2010,%  
  xgrille=2,xgrilles=1,Oy=1640,ymin=1640,ymax=1710,ygrille=10,ygrilles=5]  
  \FenetreSimpleTikz<Annee,Police=\scriptsize>{1992,1994,...,2008}{1640,1650,...,1700}  
  \FenetreTikz  
  \CourbeTikz[line width=1.25pt,orange,samples=500]{-(\x-2000)*(\x-2000)+1700}{\xmin:\xmax}  
\end{tikzpicture}
```



12 L'outil « SplineTikz »

12.1 Courbe d'interpolation



On va utiliser les notions suivantes pour paramétrer le tracé « automatique » grâce à `\SplineTikz[options]` :

- il faut rentrer les **points de contrôle**;
- il faut préciser les **pentés des tangentes** (pour le moment on travaille avec les mêmes à gauche et à droite...);
- on peut « affiner » les portions de courbe en paramétrant des **coefficients** (voir un peu plus loin...).

Pour déclarer les paramètres :

- liste des points de contrôle (minimum 2!!) par : $x_1/y_1/d_1$ $x_2/y_2/d_2$... avec les points **(x_i ; y_i)** et **$f'(x_i)=d_i$** ;
- coefficients de contrôle par `coeffs=...` :
 - `coeffs=x` pour mettre tous les coefficients à x ;
 - `coeffs=C1$C2$...` pour spécifier les coefficients par portion (donc il faut avoir autant de \$ que pour les points!);
 - `coeffs=C1G/C1D$...` pour spécifier les coefficients par portion et par partie gauche/-droite;
 - on peut mixer avec `coeffs=C1$C2G/C2D$...`

12.2 Code, clés et options



Code LaTeX

```
\begin{tikzpicture}
...
\SplineTikz[options]{liste}
...
\end{tikzpicture}
```



Certains paramètres et **clés** peuvent être gérés directement dans la commande `\SplineTikz` :

- la couleur de la courbe par la clé **⟨Couleur⟩**; défaut **⟨red⟩**
- l'épaisseur de la courbe par la clé **⟨Epaisseur⟩**; défaut **⟨1.25pt⟩**
- du style supplémentaire pour la courbe peut être rajouté, grâce à la clé **⟨Style⟩**; défaut **⟨vide⟩**
- les coefficients de *compensation* gérés par la clé **⟨Coeffs⟩**; défaut **⟨3⟩**
- les points de contrôle, affichés ou non par la clé booléenne **⟨AffPoints⟩**; défaut **⟨false⟩**
- la taille des points de contrôle est gérée par la clé **⟨TaillePoints⟩**. défaut **⟨2pt⟩**

12.3 Compléments sur les coefficients de « compensation »



Le choix a été fait ici, pour *simplifier* le code, le travailler sur des courbes de Bézier.

Pour *simplifier* la gestion des nombres dérivés, les points de contrôle sont gérés par leurs coordonnées *polaires*, les coefficients de compensation servent donc – grosso modo – à gérer la position radiale.

Le coefficient **⟨3⟩** signifie que, pour une courbe de Bézier entre $x = a$ et $x = b$, les points de contrôle seront situés à une distance radiale de $\frac{b-a}{3}$.

Pour *écarter* les points de contrôle, on peut du coup *réduire* le coefficient de compensation!

Pour des intervalles *étroits*, la *pente* peut paraître abrupte, et donc le(s) coefficient(s) peuvent être modifiés, de manière fine.

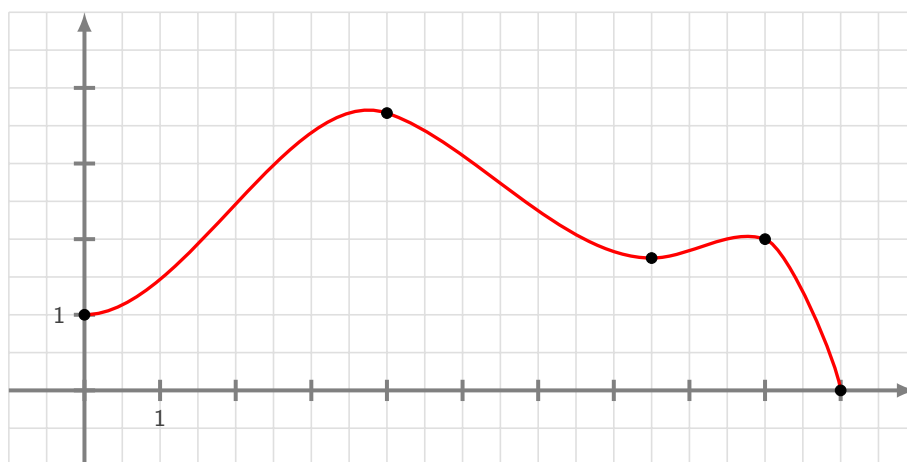
Si jamais il existe (un ou) des points *anguleux*, le plus simple est de créer les splines en plusieurs fois.

12.4 Exemples

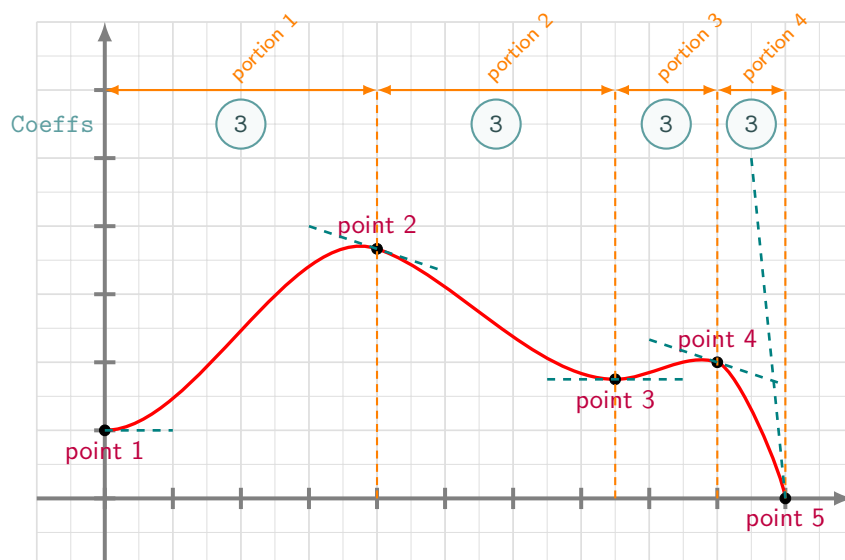


Code \LaTeX et sortie \LaTeX

```
%code tikz
\def\x{0.9cm}\def\y{0.9cm}
\def\xmin{-1}\def\xmax{11}\def\xgrille{1}\def\xgrilles{0.5}
\def\ymin{-1}\def\ymax{5}\def\ygrille{1}\def\ygrilles{0.5}
%axes et grilles
\draw[xstep=\xgrilles,ystep=\ygrilles,line width=0.6pt,lightgray!50] (\xmin,\ymin) grid
(\xmax,\ymax);
\draw[line width=1.5pt,->,gray,>=latex] (\xmin,0)--(\xmax,0) ;
\draw[line width=1.5pt,->,gray,>=latex] (0,\ymin)--(0,\ymax) ;
\foreach \x in {0,1,...,10} {\draw[gray,line width=1.5pt] (\x,4pt) -- (\x,-4pt) ;}
\foreach \y in {0,1,...,4} {\draw[gray,line width=1.5pt] (4pt,\y) -- (-4pt,\y) ;}
\draw[darkgray] (1,-4pt) node[below,font=\sffamily] {1} ;
\draw[darkgray] (-4pt,1) node[left,font=\sffamily] {1} ;
%splines
\def\LISTE{0/1/0$4/3.667/-0.333$7.5/1.75/0$9/2/-0.333$10/0/-10}
\SplineTikz[AffPoints,Coeffs=3,Couleur=red]{\LISTE}
```



Avec des explications utiles à la compréhension :



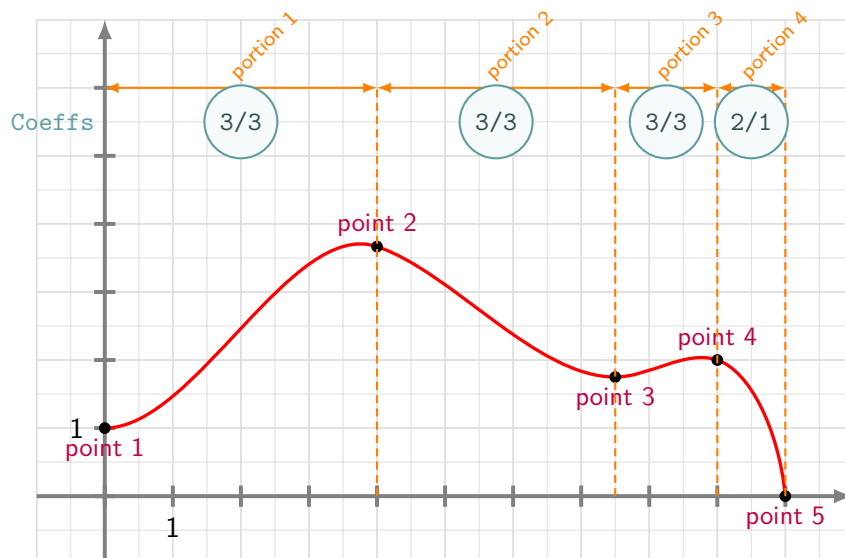
12.5 Avec une gestion plus fine des « coefficients »



Dans la majorité des cas, le *coefficient* ③ permet d'obtenir une courbe (ou une portion) très satisfaisante!

Dans certains cas, il se peut que la portion paraisse un peu trop « abrupte ».

On peut dans ce cas *jouer* sur les coefficients de cette portion pour *arrondir* un peu tout cela (ie diminuer le coeff...)!

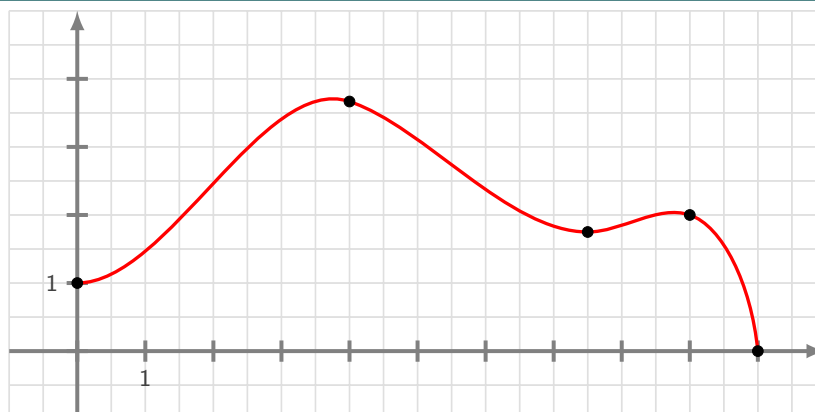


Code \LaTeX

```
...
%splines
\def\LISTE{0/1/0$4/3.667/-0.333$7.5/1.75/0$9/2/-0.333$10/0/-10}
\SplineTikz[AffPoints,Coeffs=3$3$3$2/1]{\LISTE}
...
```



Sortie \LaTeX



12.6 Conclusion



Le plus « simple » est donc :

- de déclarer la liste des points de contrôle, grâce à `\def\LISTE{x1/y1/d1$x2/y2/d2$...}`;
- de saisir la commande `\SplineTikz[...]{\LISTE}`;
- d'ajuster les options et coefficients en fonction du rendu!

13 Génération de la courbe d'interpolation

13.1 Intro



3.01c Dans le cas où la courbe d'interpolation est destinée à être (ré)utilisée, comme pour illustrer des intégrales par exemple, il est possible de *générer* une macro pour le tracé, afin de faire la représentation ou l'exploitation *à la main*.



</> Code \LaTeX

```
\begin{tikzpicture}
  ...
  \GenererSplineTikz[options]{liste}[\nomdutracé]
  \draw[options] \nomdutracé ;
  ...
\end{tikzpicture}
```



Cela permet de générer le tracé de la courbe d'interpolation, avec le même format de données que pour la commande de *tracé* :

- la liste des points de contrôle (minimum 2!!) définie par : $x_1/y_1/d_1 x_2/y_2/d_2 \dots$ avec les points $(x_i; y_i)$ et $f'(x_i) = d_i$;
- les coefficients de contrôle par `coeffs=...` :
 - `coeffs=x` pour mettre tous les coefficients à x ;
 - `coeffs=C1SC2S...` pour spécifier les coefficients par portion (donc il faut avoir autant de S que pour les points!);
 - `coeffs=C1G/C1DS...` pour spécifier les coefficients par portion et par partie gauche/droite;
 - on peut mixer avec `coeffs=C1SC2G/C2DS...`

Les **<clés>** optionnelles, et entre **<...>**, permettent de spécifier :

- le numéro du point de départ du tracé, via la clé **<NumDebut>** (valant **<1>** par défaut);
- le numéro du point d'arrivée du tracé, via la clé **<NumFin>** (valant **<dernier>** par défaut).

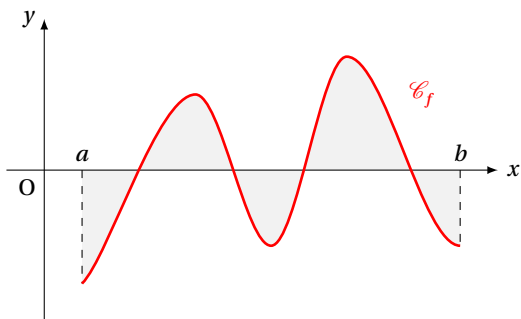
Le dernier argument, optionnel et entre **<[...]>**, est quant à lui la macro dans laquelle on stocke le tracé (**<\CourbeSplineTikz>** par défaut).

13.2 Exemples et illustrations



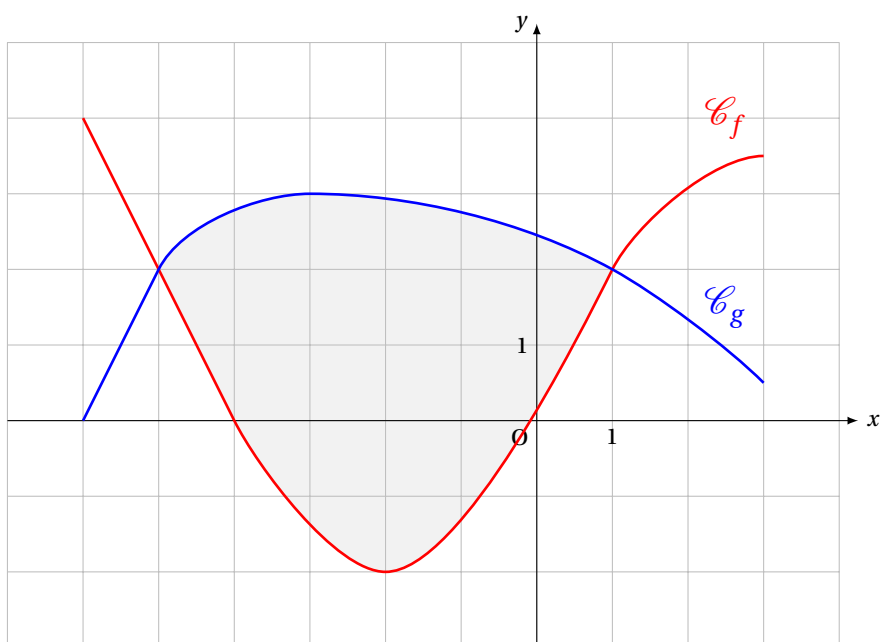
Code \LaTeX et sortie \LaTeX

```
\begin{tikzpicture}
  %splines
  \def\LISTE{0.5/-1.5/1$2/1/0$3/-1/0$4/1.5/0$5.5/-1/0}
  \GenererSplineTikz{\LISTE} %on créé la courbe, stockée dans \CourbeSplineTikz
  %intégrale
  \draw[draw=none,fill=lightgray!20] (0.5,0) -- \CourbeSplineTikz -- (5.5,0) -- cycle ;
  %tracés
  \draw[->,>=latex] (-0.5,0) -- (6,0) node[right] {$x$};
  \draw[->,>=latex] (0,-2) -- (0,2) node[left] {$y$};
  \draw (0,0) node[color=black,below left] {$0$};
  \draw (5,1) node[color=red] {$\mathscr{C}_f$};
  \draw[dashed] (0.5,0)--(0.5,-1.5); \draw[dashed] (5.5,0)--(5.5,-1);
  \draw (0.5,0) node[color=black, above]{$a$};
  \draw (5.5,0) node[color=black, above]{$b$};
  %spline complet
  \draw[line width=1pt,red] \CourbeSplineTikz ; %la courbe
\end{tikzpicture}
```





```
\begin{tikzpicture}
  %splines
  \def\LISTEF{-6/4/-2§-5/2/-2§-4/0/-2§-2/-2/0§1/2/2§3/3.5/0}
  \GenererSplineTikz{\LISTEF}{\CourbeDeF}
  \def\LISTEG{-6/0/2§-5/2/2§-3/3/0§1/2/-0.55§3/0.5/-1}
  \GenererSplineTikz{\LISTEG}{\CourbeDeG}
  %splines partielles
  \GenererSplineTikz<NumDebut=2,NumFin=5>{\LISTEF}{\BoutCourbeDeF}
  \GenererSplineTikz<NumDebut=2,NumFin=4>{\LISTEG}{\BoutCourbeDeG}
  %intégrale entre les deux courbes entre -5 et 1
  \draw[draw=none,fill=lightgray!20] (-5,2) -- \BoutCourbeDeG -- \BoutCourbeDeF ;
  %graphique
  \draw[thin,lightgray] (-7,-3) grid (4,5) ;
  \draw[->,>=latex] (-7,0) -- (4.25,0) node[right] {$x$} ;
  \draw[->,>=latex] (0,-3) -- (0,5.25) node[left] {$y$} ;
  \draw (0,0) node[color=black,below left] {$0$} ;
  \draw (1,0) node[color=black,below] {$1$} ;
  \draw (0,1) node[color=black,left] {$1$} ;
  \draw (2.5,4) node[color=red,font=\Large] {$\mathscr{C}_f$} ;
  \draw (2.5,1.5) node[color=blue,font=\Large] {$\mathscr{C}_g$} ;
  %splines 'complets'
  \draw[line width=1pt,red] \CourbeDeF ;
  \draw[line width=1pt,blue] \CourbeDeG ;
\end{tikzpicture}
```



14 L'outil « TangenteTikz »

14.1 Définitions



En parallèle de l'outil `\SplineTikz`, il existe l'outil `\TangenteTikz` qui va permettre de tracer des tangentes à l'aide de la liste de points précédemment définie pour l'outil `\SplineTikz`.

NB : il peut fonctionner indépendamment de l'outil `\SplineTikz` puisque la liste des points de travail est gérée de manière autonome !



Code \LaTeX

```
\begin{tikzpicture}
...
  \TangenteTikz[options]{liste}
...
\end{tikzpicture}
```



Cela permet de tracer la tangente :

- au point numéro **Point** de la liste **liste**, de coordonnées x_i/y_i avec la pente d_i ;
- avec une épaisseur de **Epaisseur**, une couleur **Couleur** et un style additionnel **Style**;
- en la traçant à partir de **xl** avant x_i et jusqu'à **xr** après x_i .

14.2 Exemple et illustration



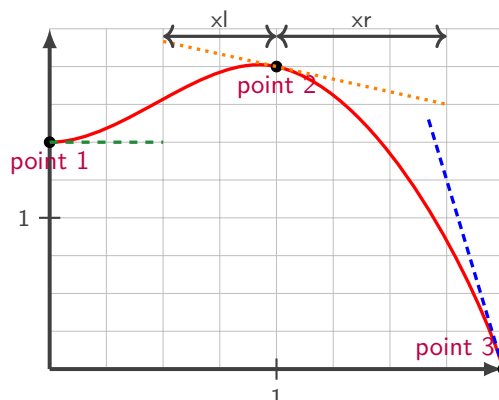
Code \LaTeX

```
\begin{tikzpicture}
...
  \def\LISTE{0/1.5/0$1/2/-0.333$2/0/-5}
  %spline
  \SplineTikz[AffPoints,Coeffs=3$2,Couleur=red]{\LISTE}
  %tangente
  \TangenteTikz[xl=0,xr=0.5,Couleur=CouleurVertForet,Style=dashed]{\LISTE}
  \TangenteTikz[xl=0.5,xr=0.75,Couleur=orange,Style=dotted,Point=2]{\LISTE}
  \TangenteTikz[xl=0.33,xr=0,Couleur=blue,Style=densely dashed,Point=3]{\LISTE}
...
\end{tikzpicture}
```



Sortie \LaTeX

On obtient le résultat suivant (avec les éléments rajoutés utiles à la compréhension) :



14.3 Exemple avec les deux outils, et « personnalisation »



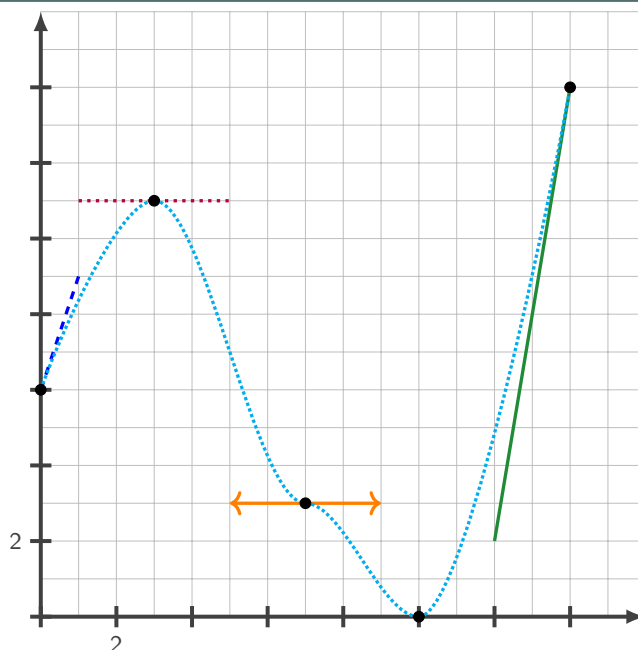
</> Code \LaTeX

```
\tikzset{%
  xmin/.store in=\xmin,xmin/.default=-5,xmin=-5,
  xmax/.store in=\xmax,xmax/.default=5,xmax=5,
  ymin/.store in=\ymin,ymin/.default=-5,ymin=-5,
  ymax/.store in=\ymax,ymax/.default=5,ymax=5,
  xgrille/.store in=\xgrille,xgrille/.default=1,xgrille=1,
  xgrilles/.store in=\xgrilles,xgrilles/.default=0.5,xgrilles=0.5,
  ygrille/.store in=\ygrille,ygrille/.default=1,ygrille=1,
  ygrilles/.store in=\ygrilles,ygrilles/.default=0.5,ygrilles=0.5,
  xunit/.store in=\xunit,unit/.default=1,xunit=1,
  yunit/.store in=\yunit,unit/.default=1,yunit=1
}

\begin{tikzpicture}[x=0.5cm,y=0.5cm,xmin=0,xmax=16,xgrilles=1,ymin=0,ymax=16,ygrilles=1]
  \draw[xstep=\xgrilles,ystep=\ygrilles,line width=0.3pt,lightgray] (\xmin,\ymin) grid
    (\xmax,\ymax) ;
  \draw[line width=1.5pt,->,darkgray,>=latex] (\xmin,0)--(\xmax,0) ;
  \draw[line width=1.5pt,->,darkgray,>=latex] (0,\ymin)--(0,\ymax) ;
  \foreach \x in {0,2,...,14} {\draw[darkgray,line width=1.5pt] (\x,4pt) -- (\x,-4pt) ;}
  \foreach \y in {0,2,...,14} {\draw[darkgray,line width=1.5pt] (4pt,\y) -- (-4pt,\y) ;}
  %la liste pour la courbe d'interpolation
  \def\liste{0/6/3$3/11/0$7/3/0$10/0/0$14/14/6}
  %les tangentes "stylisées"
  \TangenteTikz[xl=0,xr=1,Couleur=blue,Style=dashed]{\liste}
  \TangenteTikz[xl=2,xr=2,Couleur=purple,Style=dotted,Point=2]{\liste}
  \TangenteTikz[xl=2,xr=2,Couleur=orange,Style=<->,Point=3]{\liste}
  \TangenteTikz[xl=2,xr=0,Couleur=CouleurVertForet,Point=5]{\liste}
  %la courbe en elle-même
  \SplineTikz[AffPoints,Coeffs=3,Couleur=cyan,Style=densely dotted]{\liste}
\end{tikzpicture}
```



Sortie \LaTeX



15 Points de discontinuité

15.1 Idée



2.7.7 L'idée est de présenter, en marge de la création de *splines cubiques*, des points de discontinuité. Pour des raisons *internes* au code, cette possibilité n'est pas offerte (encore?) directement dans la commande de création des splines.



</> Code \LaTeX

```
%dans un environnement tikz  
\PtsDiscontinuite
```

15.2 Commandes



</> Code \LaTeX

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



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

- la clé **<Couleur>** qui permet de définir la couleur du symbole; défaut **<red>**
- la clé **<Epaisseur>** qui est relative à l'épaisseur du symbole; défaut **<1.25pt>**
- la clé **<Pos>** pour choisir la position de la discontinuité (parmi **<G/D>**); défaut **<D>**
- la clé **<Echelle>** pour modifier l'échelle du symbole; défaut **<1>**
- la clé **<Type>** pour choisir le type de symbole, parmi **<par/cro/rond/demirond>**. défaut **<par>**

Le second argument, obligatoire et entre {...} permet de préciser (comme pour les commandes des paragraphes précédents) la liste des points en lesquels le symbole de discontinuité sera positionné, sous la forme $x_1/y_1/d_1 \S x_2/y_2/d_2 \S \dots$ avec les points $(x_i; y_i)$ et $f'(x_i)=d_i$.

15.3 Exemples

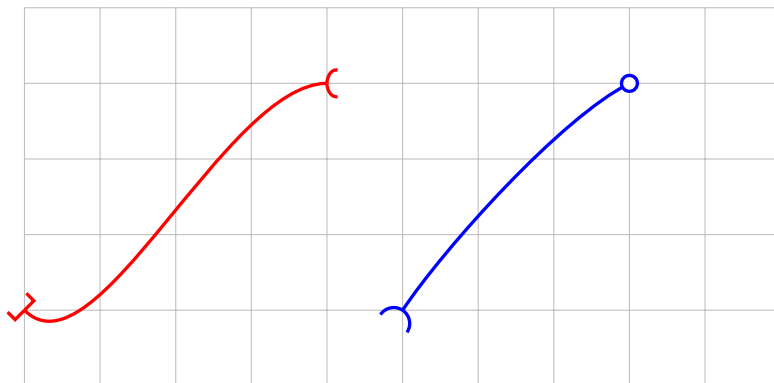


</> Code \LaTeX

```
\begin{tikzpicture}  
  \draw[lightgray] (0,0) grid (10,5) ;  
  \SplineTikz{0/1/-1 \S 4/4/0}  
  \PtsDiscontinuite{4/4/0}  
  \PtsDiscontinuite[Pos=G,Type=cro]{0/1/-1}  
  \SplineTikz[Couleur=blue]{5/1/1.5 \S 8/4/0.5}  
  \PtsDiscontinuite[Couleur=blue,Type=rond]{8/4/0.5}  
  \PtsDiscontinuite[Couleur=blue,Pos=G,Type=demirond,Echelle=2]{5/1/1.5}  
\end{tikzpicture}
```



</> Code \LaTeX



16 Petits schémas pour le signe d'une fonction affine ou d'un trinôme

16.1 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.

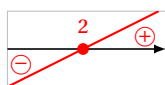
Le code est largement inspiré de celui du package `tnsana` même si la philosophie est un peu différente. Comme pour les autres commandes TikZ, l'idée est de laisser la possibilité à l'utilisateur de définir et créer son environnement TikZ, et d'insérer la commande `\MiniSchemaSignes` pour afficher le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

```
\MiniSchemaSignes*
```



16.2 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

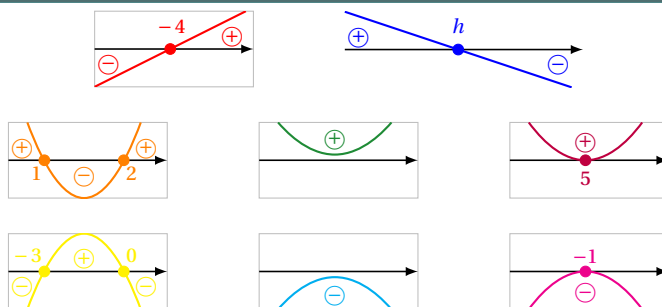


</> Code \LaTeX

```
\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}
```



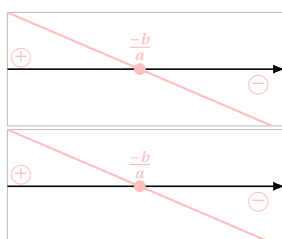
Sortie \LaTeX



Code \LaTeX et sortie \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>
```



16.3 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 \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 !



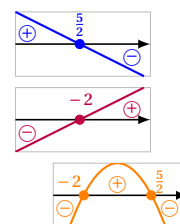
</> Code \LaTeX

```
\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}
```



Sortie \LaTeX

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



17 Suites récurrentes et « toile »

17.1 Idée



L'idée est d'obtenir une commande pour tracer (en TikZ) la « toile » permettant d'obtenir – graphiquement – les termes d'une suite récurrente définie par une relation $u_{n+1} = f(u_n)$.

Comme pour les autres commandes TikZ, l'idée est de laisser l'utilisateur définir et créer son environnement TikZ, et d'insérer la commande `\ToileReccurrence` pour afficher la « toile ».

17.2 Commandes



Code \LaTeX

```
...
\begin{tikzpicture}[options]
...
\ToileReccurrence[clés][options du tracé][options supplémentaires des termes]
...
\end{tikzpicture}
```



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

- le premier argument optionnel définit les **Clés** de la commande :
 - la clé **Fct** qui définit la fonction f ; défaut **vide**
 - la clé **Nom** qui est le *nom* de la suite; défaut **u**
 - la clé **No** qui est l'indice initial; défaut **0**
 - la clé **Uno** qui est la valeur du terme initial; défaut **vide**
 - la clé **Nb** qui est le nombre de termes à construire; défaut **5**
 - la clé **PosLabel** qui est le placement des labels par rapport à l'axe (Ox); défaut **below**
 - la clé **DecalLabel** qui correspond au décalage des labels par rapport aux abscisses; défaut **6pt**
 - la clé **TailleLabel** qui correspond à la taille des labels; défaut **small**
 - un booléen **AffTermes** qui permet d'afficher les termes de la suite sur l'axe (Ox). défaut **true**
- le deuxième argument optionnel concerne les **options** du tracé de l'*escalier* en *langage TikZ*; défaut **thick,color=magenta**;
- le troisième argument optionnel concerne les **options** du tracé des termes en *langage TikZ*. défaut **dotted**.



Il est à noter que le code n'est pas autonome, et doit être intégré dans un environnement `tikzpicture`.

L'utilisateur est donc libre de définir ses styles pour l'affichage des éléments de son graphique, et il est libre également de rajouter des éléments en plus du tracé de la *toile*!

La macro ne permet – pour le moment – ni de tracer la bissectrice, ni de tracer la courbe...

En effet, il y aurait trop d'options pour ces deux éléments, et l'idée est quand même de conserver une commande *simple*! Donc l'utilisateur se chargera de tracer et de personnaliser sa courbe et sa bissectrice!

17.3 Exemples



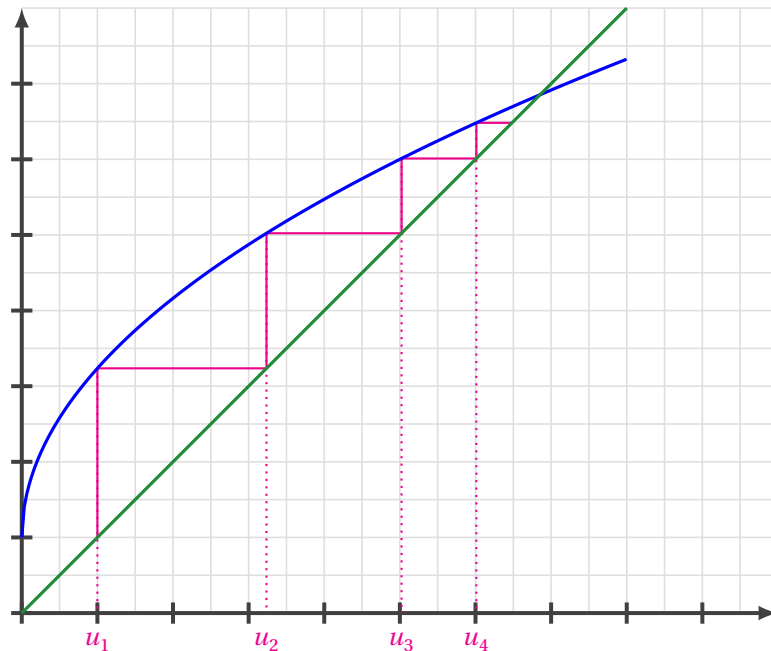
On va tracer la *toile* des 4 premiers termes de la suite récurrente :

$$\begin{cases} u_1 = 1 \\ u_{n+1} = \sqrt{5u_n} + 1 \text{ pour tout entier } n \geq 1 \end{cases}$$



Code \LaTeX et sortie \LaTeX

```
%code tikz
\def\x{1.5cm}\def\y{1.5cm}
\def\xmin{0}\def\xmax{10}\def\xgrille{1}\def\xgrilles{0.5}
\def\ymin{0}\def\ymax{8}\def\ygrille{1}\def\ygrilles{0.5}
%axes et grilles
\draw[xstep=\xgrilles,ystep=\ygrilles,line width=0.6pt,lightgray!50] (\xmin,\ymin) grid
(\xmax,\ymax);
\draw[line width=1.5pt,->,darkgray,>=latex] (\xmin,0)--(\xmax,0) ;
\draw[line width=1.5pt,->,darkgray,>=latex] (0,\ymin)--(0,\ymax) ;
\foreach \x in {0,1,...,9} {\draw[darkgray,line width=1.5pt] (\x,4pt) -- (\x,-4pt) ;}
\foreach \y in {0,1,...,7} {\draw[darkgray,line width=1.5pt] (4pt,\y) -- (-4pt,\y) ;}
%fonction définie et réutilisable
\def\f{sqrt(5*\x)+1}
%toile
\ToileRecurrence[Fct={\f},No=1,Uno=1,Nb=4,DecalLabel=4pt]
%éléments supplémentaires
\draw[very thick,blue,domain=0:8,samples=250] plot (\x,{\f}) ;
\draw[very thick,CouleurVertForet,domain=0:8,samples=2] plot (\x,\x) ;
```



Peut-être que – ultérieurement – des options *booléennes* seront disponibles pour un tracé *générique* de la courbe et de la bissectrice, mais pour le moment la macro ne fait *que* l'escalier.

17.4 Influence des paramètres

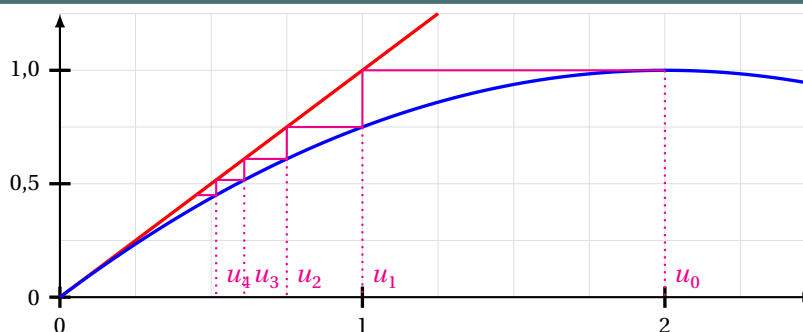


Code \LaTeX

```
\begin{center}
\begin{tikzpicture}[x=4cm,y=3cm]
%axes + grilles + graduations
...
%fonction
\def\f{-0.25*\x*\x+\x}
%tracés
\begin{scope}
\clip (0,0) rectangle (2.5,1.25) ;
\draw[line width=1.25pt,blue,domain=0:2.5,samples=200] plot (\x,{\f}) ;
\end{scope}
\ToileRecurrence[Fct={\f},No=0,Uno=2,Nb=5,PosLabel=above right,DecalLabel=0pt]
\end{tikzpicture}
\end{center}
```



Sortie \LaTeX

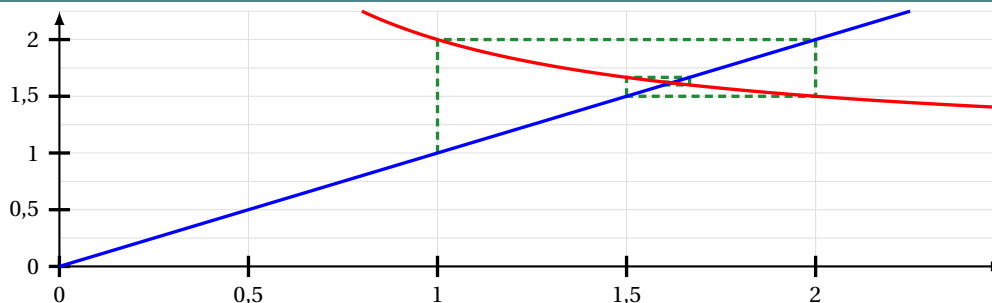


Code \LaTeX

```
\begin{center}
\begin{tikzpicture}[x=5cm,y=1.5cm]
...
\def\f{1+1/\x}
\ToileRecurrence%
[Fct={\f},No=0,Uno=1,Nb=7,PosLabel=above right,DecalLabel=0pt,AffTermes=false]%
[line width=1.25pt,CouleurVertForet,densely dashed] []
\draw[line width=1.25pt,blue,domain=0:2.25,samples=2] plot (\x,{\x});
\draw[line width=1.25pt,red,domain=0.8:2.5,samples=250] plot (\x,{\f});
\end{tikzpicture}
\end{center}
```



Sortie \LaTeX



18 Méthodes graphiques et intégrales

18.1 Idée



2.6.1 L'idée est de proposer plusieurs méthodes graphiques pour illustrer graphiquement une intégrale, via :

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

La commande n'est pas autonome, elle est de ce fait à être placée dans un environnement `tikzpicture`.



</> Code \LaTeX

```
%commande pour déclarer une fonction réutilisable  
\DeclareFonctionTikz[nom]{expr}
```



</> Code \LaTeX

```
%environnement tikz  
\IntegraleApprocheeTikz[clés]{nom_fonction}{a}{b}
```

18.2 Clés et arguments



Plusieurs **<Clés>** sont disponibles pour la commande :

- la clé **<Epaisseur>** pour l'épaisseur des « figures » ; défaut : **<semithick>**
- la clé **<Couleur>** pour la couleur des « figures » ; défaut : **<red>**
- le booléen **<Remplir>**, pour remplir les « figures » ; défaut : **<true>**
- la clé **<Opacite>** pour l'opacité du remplissage des « figures » ; défaut : **<0.25>**
- la clé **<CouleurRemplissage>** pour la couleur de remplissage des « figures » ; défaut : **<Couleur !25>**
- la clé **<Methode>**, parmi **<RectanglesGauche / RectanglesDroite / RectanglesMilieu / Trapezes>** pour spécifier la méthode utilisée ; défaut : **<RectanglesGauche>**
- la clé **<NbSubDiv>** précise le nombre de « figures ». défaut : **<10>**

Concernant les arguments obligatoires :

- le premier est la fonction , déclarée au préalable ;
- les deux autres arguments sont les bornes de l'intégrale.

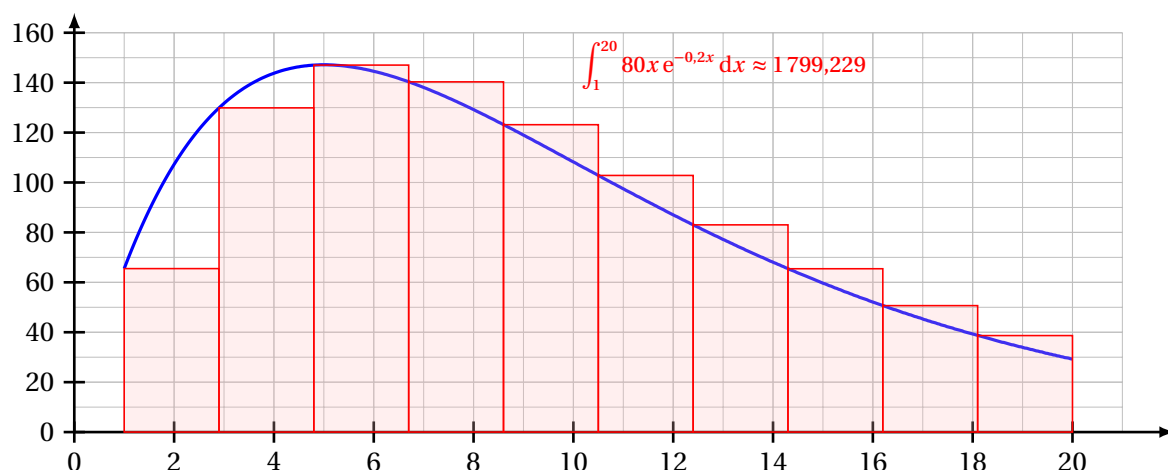
Les commandes graphiques de `Proflycee` peuvent être utilisées pour configurer la fenêtre!



Code \LaTeX et sortie \LaTeX

```
\begin{tikzpicture}%

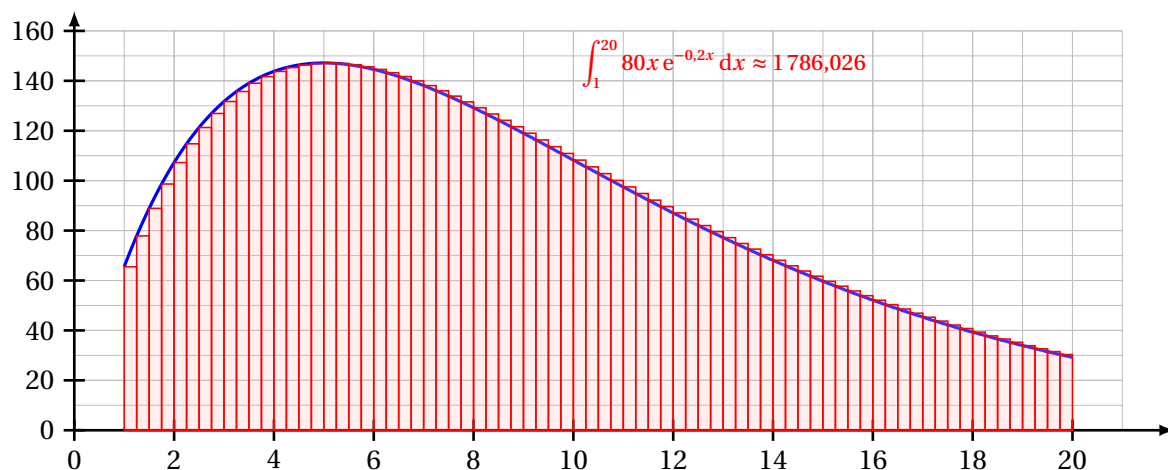
[x=0.66cm,y=0.033cm,xmin=0,xmax=21,xgrille=2,xgrilles=1,ymin=0,ymax=160,ygrille=20,ygrilles=10]
\DeclareFonctionTikz{80*\x*exp(-0.2*\x)}
\FenetreSimpleTikz{0,2,...,20}{0,20,...,160}
\CourbeTikz[very thick,samples=500,blue]{f(\x)}{1:20}
\IntegraleApprocheeTikz{f}{1}{20}
\draw[red] (10,160) node[below right]
  {\displaystyle%
    \IntegraleApprochee[Methode=RectanglesGauche,AffFormule,Expr={80x\,\text{e}^{-0,2x}}]}%
    {80*\x*exp(-0.2*\x)}{1}{20}$} ;
\end{tikzpicture}
```



Code \LaTeX et sortie \LaTeX

```
\begin{tikzpicture}%

[x=0.66cm,y=0.033cm,xmin=0,xmax=21,xgrille=2,xgrilles=1,ymin=0,ymax=160,ygrille=20,ygrilles=10]
\DeclareFonctionTikz{80*\x*exp(-0.2*\x)}
\FenetreSimpleTikz{0,2,...,20}{0,20,...,160}
\CourbeTikz[very thick,samples=500,blue]{f(\x)}{1:20}
\IntegraleApprocheeTikz[NbSubDiv=76]{f}{1}{20}
\draw[red] (10,160) node[below right]
  {\displaystyle\IntegraleApprochee%
    [NbSubDiv=76,Methode=RectanglesGauche,AffFormule,Expr={80x\,\text{e}^{-0,2x}}]}%
    {80*\x*exp(-0.2*\x)}{1}{20}$} ;
\end{tikzpicture}
```



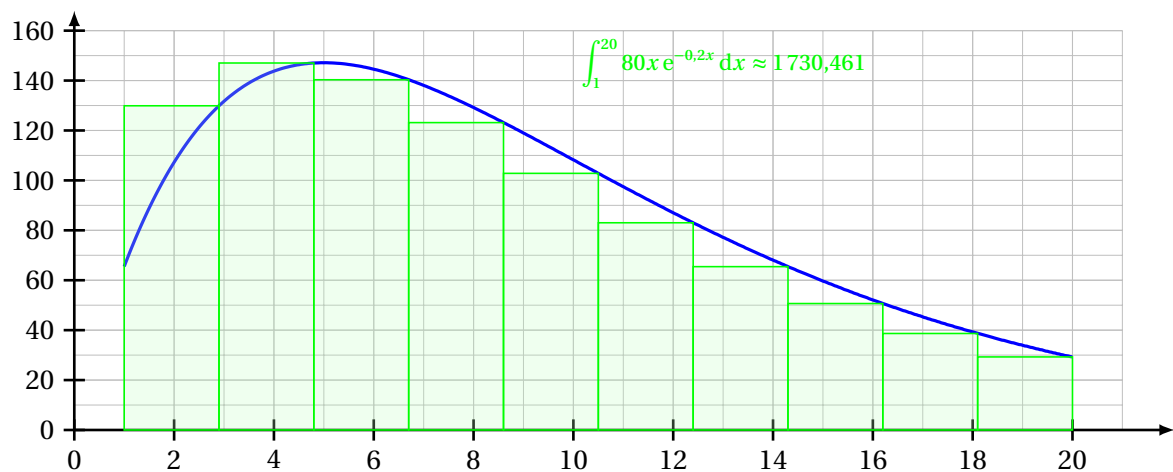
18.3 Exemples



Code \LaTeX et sortie \LaTeX

```
\begin{tikzpicture}%

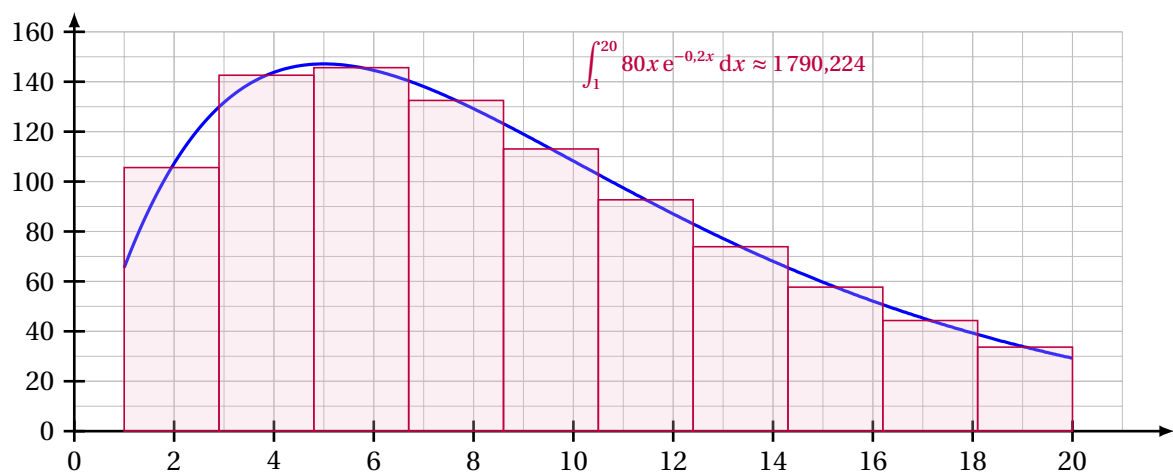
[x=0.66cm,y=0.033cm,xmin=0,xmax=21,xgrille=2,xgrilles=1,ymin=0,ymax=160,ygrille=20,ygrilles=10]
\DeclareFonctionTikz{80*x*exp(-0.2*x)}
\FenetreSimpleTikz{0,2,...,20}{0,20,...,160}
\CourbeTikz[very thick,samples=500,blue]{f(x)}{1:20}
\IntegraleApprocheeTikz[Methode=RectanglesDroite,Couleur=green]{f}{1}{20}
\draw[green] (10,160) node[below right]
{\$\displaystyle\IntegraleApprochee%
[Methode=RectanglesDroite,AffFormule,Expr={80x\,\text{e}^{-0,2x}}}%
{80*x*exp(-0.2*x)}{1}{20}\$} ;
\end{tikzpicture}
```



Code \LaTeX et sortie \LaTeX

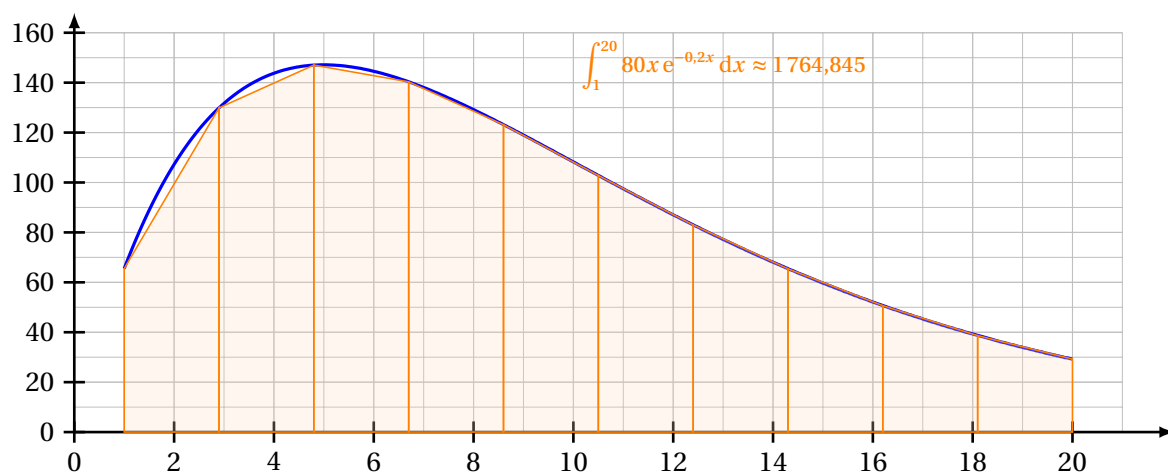
```
\begin{tikzpicture}%

[x=0.66cm,y=0.033cm,xmin=0,xmax=21,xgrille=2,xgrilles=1,ymin=0,ymax=160,ygrille=20,ygrilles=10]
\DeclareFonctionTikz{80*x*exp(-0.2*x)}
\FenetreSimpleTikz{0,2,...,20}{0,20,...,160}
\CourbeTikz[very thick,samples=500,blue]{f(x)}{1:20}
\IntegraleApprocheeTikz[Methode=RectanglesMilieu,Couleur=purple]{f}{1}{20}
\draw[purple] (10,160) node[below right]
{\$\displaystyle\IntegraleApprochee%
[Methode=RectanglesMilieu,AffFormule,Expr={80x\,\text{e}^{-0,2x}}}%
{80*x*exp(-0.2*x)}{1}{20}\$} ;
\end{tikzpicture}
```





```
\begin{tikzpicture}%  
  
  [x=0.66cm,y=0.033cm,xmin=0,xmax=21,xgrille=2,xgrilles=1,ymin=0,ymax=160,ygrille=20,ygrilles=10]  
  \DeclareFonctionTikz{80*\x*\exp(-0.2*\x)}  
  \FenetreSimpleTikz{0,2,...,20}{0,20,...,160}  
  \CourbeTikz[very thick,samples=500,blue]{f(\x)}{1:20}  
  \IntegraleApprocheeTikz[Methode=Trapezes,Couleur=orange]{f}{1}{20}  
  \draw[orange] (10,160) node[below right]  
  {\$ \displaystyle \IntegraleApprochee%  
    [Methode=Trapezes,AffFormule,Expr={80x\,\text{e}^{-0,2x}}] %  
    {80*\x*\exp(-0.2*\x)}{1}{20}$} ;  
\end{tikzpicture}
```



Thème

PRÉSENTATION DE CODES

Présentation de codes

19 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 `piton` et `pythontex`), les environnements seront composés :

- dans une boîte `tcolorbox`;
- de deux styles : `CodeXXX` ou `CodeXXXAlt`;
- 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.

20 Code Python « simple » via le package listings

20.1 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 `minted` ou `piton`) 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 `minted` ou `piton`.

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

20.2 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 tcbbox}
...
\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 \LaTeX `tcolorbox` (**<\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 \LaTeX `tcolorbox` en *langage tcolorbox*, comme l'alignement.



Les environnements créés par \LaTeX `tcolorbox` et \LaTeX `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 !

20.3 Insertion via un fichier « externe »



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



</> Code \LaTeX

```

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

```

20.4 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}

```



Sortie \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 L^AT_EX

```
\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}
```



Sortie L^AT_EX

```
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 Python



</> Code L^AT_EX

```
\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}
```



Sortie L^AT_EX

</> 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 L^AT_EX

```

\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}

```



Sortie L^AT_EX

Code Python

```

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
22     # valeurs
23     plt.plot(X,Y,".b")
24     fin=time.clock()
25     return "Temps : " + str(fin-depart) + "
26     s."

```

21 Code Python via le package `piton`

21.1 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 \LaTeX

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



La librairie `piton` (qui charge `piton`, est compatible uniquement avec \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!

Depuis la version `0.95` de `piton`, `<left-margin=auto>` est disponible et activée dans `ProfLycee`.

Depuis la version `0.99` de `piton`, `<break-lines>` est disponible et activée dans `ProfLycee`.

Depuis la version `1.0` de `piton`, `<tabs-auto-gobble>` est disponible et activée dans `ProfLycee`.

Depuis la version `2.2a` de `piton`, des options pour `<line-numbers>` sont disponibles.



Le package `piton` nécessite donc obligatoirement l'emploi de \LaTeX !

Ce package n'est chargé que si la compilation détectée est en \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.

21.2 Présentation de code Python




</> Code \LaTeX

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



Plusieurs **clés** sont disponibles :

- la clé booléenne **Lignes** pour afficher ou non les numéros de lignes; défaut **true**
- la clé booléenne **Gobble** pour activer les options liées au gobble; défaut **true**
- 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**
- `3.01f` la clé **Style** (parmi **Moderne / Classique**) 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**
- `2.5.9` la clé **CouleurNombres** pour la couleur des nombres. défaut **orange**



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`)!



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



</> Code L^AT_EX

```
\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]{}
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}[Largeur=12cm,Style=Classique,Cadre=false]{}
      def f(x) :
        return x**2
    \end{CodePiton}
  \item Et avec un autre \texttt{itemize} :
    %
    \begin{CodePiton}[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 :

+ Code Python

```

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

```

21.3 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.

22 Code & Console Python, via les packages Pythontex ou Minted

22.1 Librairies



2.5.0 Cette section nécessite de charger les librairies `\usepackage{minted}` et/ou `\usepackage{pythontex}` dans le préambule.



</> Code \LaTeX

```
\usepackage{minted}
\usepackage{pythontex}
%ou
\usepackage{minted,pythontex}
```

22.2 Introduction



2.5.0 La librairie `\usepackage{pythontex}` permet d'insérer et d'exécuter du code Python. On peut :

- **2.5.8** présenter du code Python (deux styles disponibles);
- exécuter du code Python dans un environnement type « console »;
- charger du code Python, et éventuellement l'utiliser dans la console.



Attention : il faut dans ce cas une compilation en plusieurs étapes, comme par exemple pdf_latex puis pythontex puis pdf_latex!

Voir par exemple <http://lesmathsduyeti.fr/fr/informatique/latex/pythontex/>!



Compte tenu de la *relative complexité* pour gérer les options (par paramètres/clés...) des *tcbbox* et des *fancyvrb*, les style sont « fixés » tels quels, et seules la taille et la position de la *tcbbox* sont modifiables. Si toutefois vous souhaitez personnaliser davantage, il faudra prendre le code correspondant et appliquer vos modifications!

Cela peut donner – en tout cas – des idées de personnalisation en ayant une base *préexistante*!

22.3 Présentation de code Python grâce au package pythontex



L'environnement `\usepackage{CodePythontex}` est donc lié à `\usepackage{pythontex}` (chargé par `\usepackage{ProfLycee}`, avec l'option *autogobble*) permet de présenter du code Python, dans une `\usepackage{tcolorbox}` avec deux styles particuliers (**2.5.8**).



</> Code \LaTeX

```
\begin{CodePythontex}[clés]{ } %les {} vides sont nécessaires
...
\end{CodePythontex}
```



</> Code \LaTeX

```
\begin{CodePythontexAlt}[clés]{ } %les {} vides sont nécessaires
...
\end{CodePythontexAlt}
```



Comme précédemment, des **<Clés>** qui permettent de *légèrement* modifier le style :

- **<Largeur>** : largeur de la *tcbbox*; défaut **<\linewidth>**
- **<PremLigne>** : numéro initial des lignes; défaut **<1>**
- **<TaillePolice>** : taille des caractères; défaut **<\footnotesize>**
- **<EspaceVertical>** : option (*stretch*) pour l'espacement entre les lignes; défaut **<1>**
- **<Lignes>** : booléen pour afficher ou non les numéros de ligne. défaut **<true>**



</> Code \LaTeX

```
\begin{CodePythontex}{} %bien mettre les {} !!
#environnement Python(tex) par défaut
def f(x) :
    return x**2
\end{CodePythontex}
```



Sortie \LaTeX

```
1 #environnement Python(tex) par défaut
2 def f(x) :
3     return x**2
```

Code Python



</> Code \LaTeX

```
\begin{CodePythontexAlt}[Largeur=12cm,Centre,Lignes=false]{}
#environnement Python(tex) classique, centré, sans lignes
def f(x) :
    return x**2
\end{CodePythontexAlt}
```



Sortie \LaTeX

</> Code Python

```
#environnement Python(tex) classique, centré, sans lignes
def f(x) :
    return x**2
```

22.4 Présentation de code Python via le package `minted`



Pour celles et ceux qui ne sont pas à l'aise avec le package `pythontex` et notamment sa spécificité pour compiler, il existe le package `minted` qui permet de présenter du code, et notamment Python.

2.5.8 Deux styles sont désormais disponibles.

2.5.0 C'est donc la librairie `minted` qu'il faudra charger.



Le package `minted` nécessite quand même une compilation avec l'option `--shell-escape` ou `-write18`!



</> Code \LaTeX

```
\begin{CodePythonMinted}(*)[clés]{options tcbbox}
...
\end{CodePythonMinted}
```



</> Code \LaTeX

```
\begin{CodePythonMintedAlt}(*)[clés]{options tcbbox}
...
\end{CodePythonMintedAlt}
```



Plusieurs **<arguments>** 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 `\tcbox` (**<\linewidth>** par défaut) et la clé **<PremLigne>** (**<1>** par défaut) ;
- le 2nd argument *obligatoire* concerne les **<options>** de la `\tcbox` en langage *tcbox*. défaut **<vide>**



</> Code \LaTeX

```
\begin{CodePythonMinted}[Largeur=13cm,PremLigne=10]{center}
#environnement Python(minted) centré avec numéros, de largeur 13cm
def f(x) :
    return x**2
\end{CodePythonMinted}
```



Sortie \LaTeX

```
10 #environnement Python(minted) centré avec numéros
11 def f(x) :
12     return x**2
```

Code Python



</> Code \LaTeX

```
\begin{CodePythonMintedAlt}*[Largeur=0.8\linewidth]{}
#environnement Python(minted), style alt, sans numéro, de largeur 0.8\linewidth
def f(x) :
    return x**2
\end{CodePythonMintedAlt}
```



Sortie \LaTeX

</> Code Python

```
#environnement Python(minted), style alt, sans numéro, 0.8\linewidth
def f(x) :
    return x**2
```

22.5 Console d'exécution Python



`\pythontex` permet également de *simuler* (en exécutant également!) du code Python dans une *console*, avec la librairie `\pythontex` du coup!

C'est l'environnement `\ConsolePythontex` qui permet de le faire.



</> Code \LaTeX

```
\begin{ConsolePythontex}[clés]{} %les {} vides sont nécessaires
...
\end{ConsolePythontex}
```



Les **<Clés>** disponibles sont :

- **<Largeur>** : largeur de la *console*; défaut **<\linewidth>**
- **<Centre>** : booléen pour centrer ou non la *console*; défaut **<false>**
- **<TaillePolice>** : taille des caractères; défaut **<\footnotesize>**
- **<EspacementVertical>** : option (*stretch*) pour l'espacement entre les lignes; défaut **<1>**
- **<Label>** : booléen pour afficher ou non le titre. défaut **<true>**



</> Code \LaTeX

```
\begin{ConsolePythontex}{}
  #console Python(tex) par défaut
  from math import sqrt
  1+1
  sqrt(12)
\end{ConsolePythontex}
```



Sortie \LaTeX

----- Début de la console python -----

```
>>> #console Python(tex) par défaut
>>> from math import sqrt
>>> 1+1
2
>>> sqrt(12)
3.4641016151377544
```

----- Fin de la console python -----



</> Code \LaTeX

```
\begin{ConsolePythontex}[Largeur=14cm,Label=false,Centre={}
  #console Python(tex) centrée sans label, 14cm
  table = [[1,2],[3,4]]
  table[0][0]

  from random import randint
  tableau = [[randint(1,20) for j in range(0,6)] for i in range(0,3)]
  tableau
  len(tableau), len(tableau[0]), tableau[1][4]
\end{ConsolePythontex}
```



Sortie \LaTeX

```
>>> #console Python(tex) centrée sans label, 14cm
>>> table = [[1,2],[3,4]]
>>> table[0][0]
1

>>> from random import randint
>>> tableau = [[randint(1,20) for j in range(0,6)] for i in range(0,3)]
>>> tableau
[[17, 6, 11, 17, 12, 1], [8, 20, 13, 1, 12, 13], [4, 18, 20, 16, 9, 18]]
>>> len(tableau), len(tableau[0]), tableau[1][4]
(3, 6, 12)
```



Le package `pythontex` peut donc servir à présenter du code Python, comme `minted` ou `piton`, sa particularité est toutefois de pouvoir *exécuter* du code Python pour une présentation de type *console*.

23 Pseudo-Code

23.1 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**.

23.2 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 tcbbox}
%attention à l'indentation, gobble ne fonctionne pas...
...
\end{PseudoCode}
```



</> Code \LaTeX

```
\begin{PseudoCodeAlt}(*)[clés]{options tcbbox}
%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 `tcbbox` (`\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 `tcbbox`.



</> 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}
```



Sortie \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}
```



Sortie \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
```

23.3 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}
```



Sortie \LaTeX

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





Pseudo-Code

24 PseudoCode via le package piton

24.1 Introduction




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



</> Code \LaTeX

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






Le package  nécessite donc obligatoirement l'emploi de $\text{Lua}\mathcal{T}\mathcal{E}\mathcal{X}$!
Ce package n'est chargé que si la compilation détectée est en $\text{Lua}\mathcal{T}\mathcal{E}\mathcal{X}$!

24.2 Présentation de PseudoCode



Plusieurs **<clés>** sont disponibles :

- la clé booléenne **<Lignes>** pour afficher ou non les numéros de lignes; défaut **<true>**
- la clé booléenne **<Gobble>** pour activer les options liées au gobble; défaut **<true>**
- 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>**



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  (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]
\SetPitonIdentifieur[minimal]%
{
  Algorithme,Fonction,Début,Paramètre,Paramètres,Faire,
  Fin,Si,Pour,Tant,Que,alors,Alors,Sinon,SinonSi,
  FinSi,FinPour,FinTantQue,TantQue,Variable,Variables,Procédure
}
{\color{MotsClesPseudoCodeA}}

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


24.3 Exemples



</> Code L^AT_EX

```
\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}
```

PseudoCode

```
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
```



</> Code L^AT_EX

```
\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}
```

PseudoCode

```
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
```

25 Cartouche Capytale

25.1 Introduction



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

25.2 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
```



Sortie \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}}
```



Sortie \LaTeX

abcd-12345 

26 Présentation de code \LaTeX

26.1 Introduction



2.0.6 L'idée est de proposer un environnement pour présenter du code \LaTeX . Ce n'est pas forcément lié à l'enseignement en Lycée mais pourquoi pas!

Il s'agit d'un environnement créé en `\tcolorbox`, et utilisant la présentation *basique* de code via `\listings`.

26.2 Commandes



Code \LaTeX

```
\begin{PresentationCode}[Couleur]{options tcolorbox}
...
\end{PresentationCode}
```



Peu de personnalisations pour ces commandes :

- le premier argument, *optionnel*, permet de préciser la *couleur* de la présentation; défaut `\CouleurVertForet`
- le second, *obligatoire*, correspond aux éventuelles options liées à la `\tcolorbox`.



Il est à noter que, même dans le cas d'option vide pour la `\tcolorbox`, les `\{ }` sont nécessaires.

On peut par exemple utiliser l'option `\listing only` pour ne présenter *que* le code source.



Code \LaTeX et sortie \LaTeX

```
\begin{PresentationCode}{}
\edef\ValAleaA{\fpeval{randint(1,100)}}
\edef\ValAleaB{\fpeval{randint(1,100)}}

Avec $A=\ValAleaA$ et $B=\ValAleaB$, on a $A\times B=\inteval{\ValAleaA * \ValAleaB}$.
\end{PresentationCode}

\begin{PresentationCode}[blue]{}
On peut faire beaucoup de choses avec \LaTeX{} !
\end{PresentationCode}
```



```
\xdef\ValAleaA{\fpeval{randint(1,100)}}
\xdef\ValAleaB{\fpeval{randint(1,100)}}

Avec $A=\ValAleaA$ et $B=\ValAleaB$, on a $A\times B=\inteval{\ValAleaA *
\ValAleaB}$.

Avec A = 97 et B = 46, on a A  $\times$  B = 4462.
```

Code \LaTeX

On peut faire beaucoup de choses avec `\LaTeX{} !`

Code \LaTeX

On peut faire beaucoup de choses avec \LaTeX !

Thème

OUTILS POUR LA GÉOMÉTRIE

Outils pour la géométrie

27 Pavé droit « simple »

27.1 Introduction



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

27.2 Commandes



Code \LaTeX

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



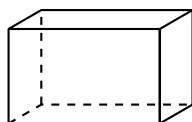
Quelques **clés** sont disponibles pour cette commande :

- **⟨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⟩**



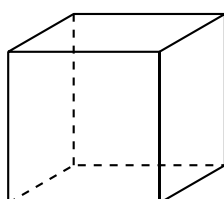
Code \LaTeX et sortie \LaTeX

```
%code tikz
\PaveTikz
```



Code \LaTeX et sortie \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!

27.3 Influence des paramètres

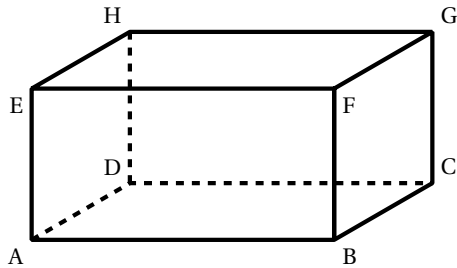


</> Code \LaTeX

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



Sortie \LaTeX

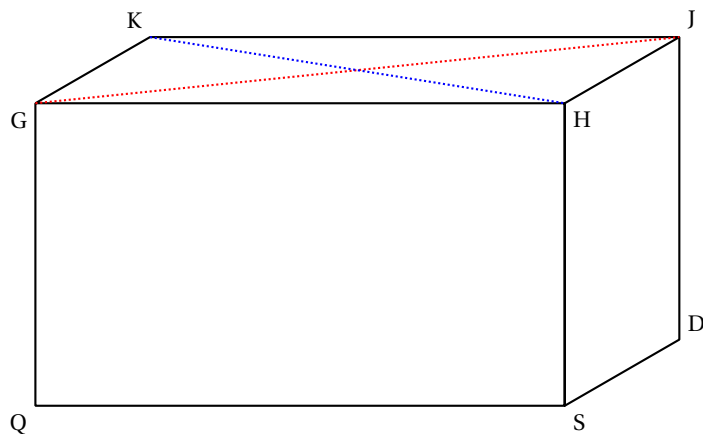


</> Code \LaTeX

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



Sortie \LaTeX



28 Tétraèdre « simple »

28.1 Introduction



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

28.2 Commandes



Code \LaTeX

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



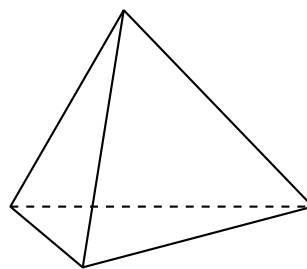
Quelques **clés** sont disponibles pour cette commande :

- **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**



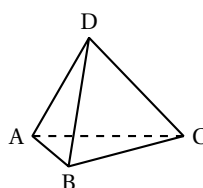
Code \LaTeX et sortie \LaTeX

```
%code tikz
\TetraedreTikz
```



Code \LaTeX et sortie \LaTeX

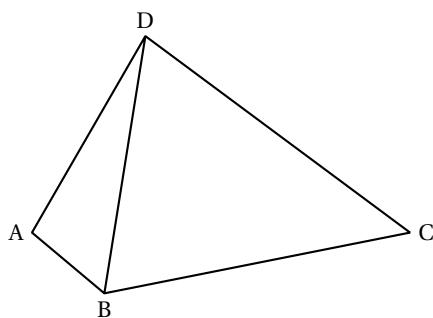
```
%code tikz
\TetraedreTikz[Aff,Largeur=2,Profondeur=0.625,Hauteur=1.5]
```





Code \LaTeX et sortie \LaTeX

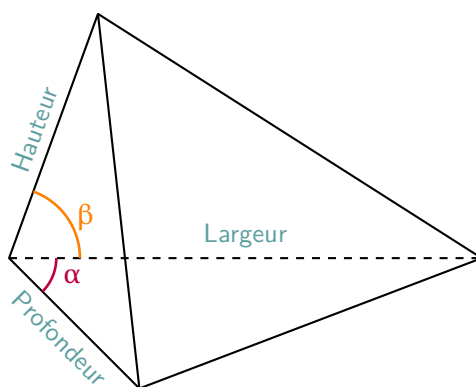
```
%code tikz
\TetraedreTikz[Plein,Aff,Largeur=5,Beta=60]
```



28.3 Influence des paramètres



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

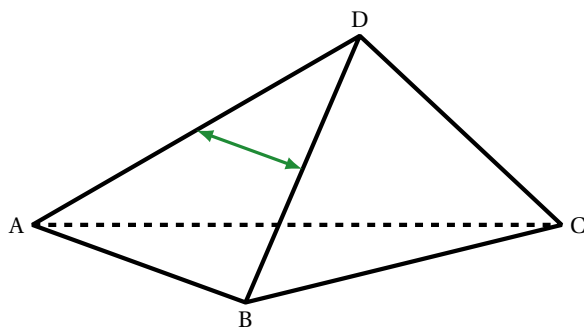


Code \LaTeX

```
\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}
```



Sortie \LaTeX



29 Cercle trigo

29.1 Idée



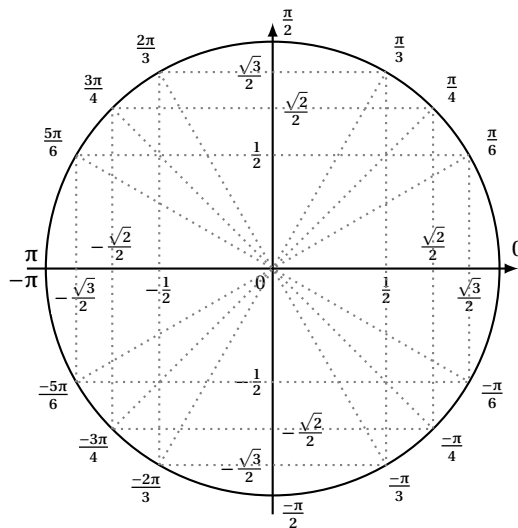
L'idée est d'obtenir une commande pour tracer (en TikZ) un cercle trigonométrique, avec personnalisation des affichages.

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 \LaTeX et sortie \LaTeX

```
%code tikz
\CercleTrigo
```



29.2 Commandes



Code \LaTeX

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



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

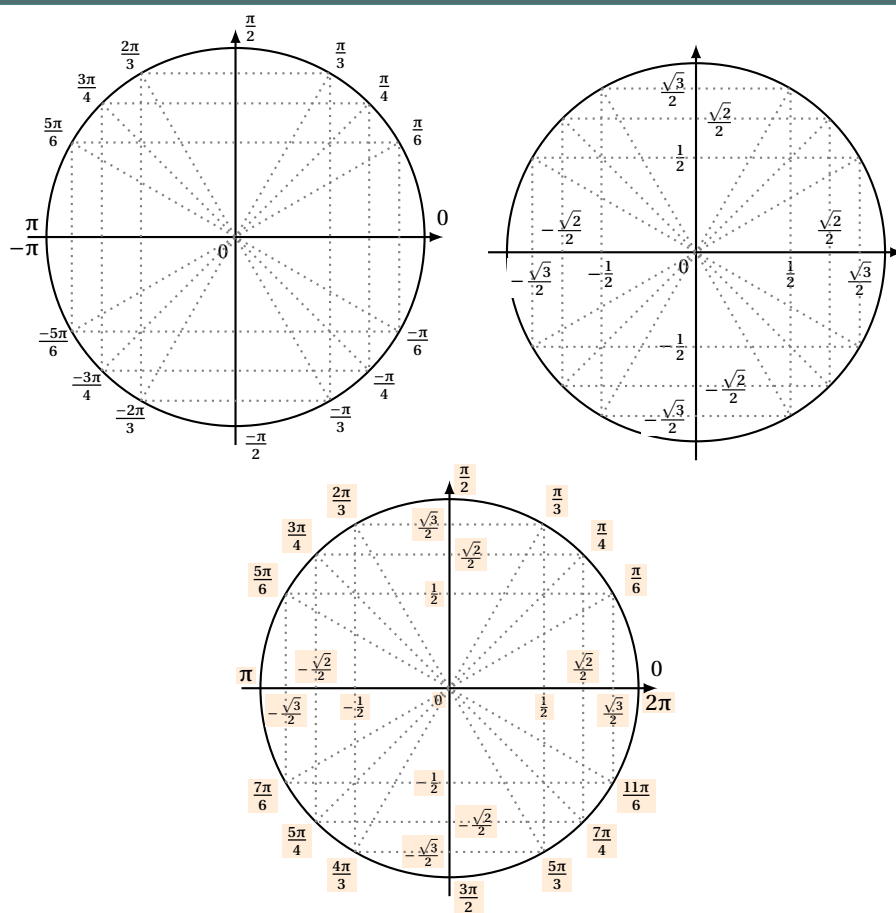
- la clé **<Rayon>** qui définit le rayon du cercle; défaut **<3>**
- la clé **<Epaisseur>** qui donne l'épaisseur des traits de base; défaut **<thick>**
- la clé **<Marge>** qui est l'écartement de axes; défaut **<0.25>**
- la clé **<TailleValeurs>** qui est la taille des valeurs remarquables; défaut **<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 « -pipi » à « zerodeuxpi »; 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 **<AffValeurs>** qui permet d'afficher les valeurs remarquables. défaut **<true>**

Code \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}

```

Sortie \LaTeX 

29.3 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>**

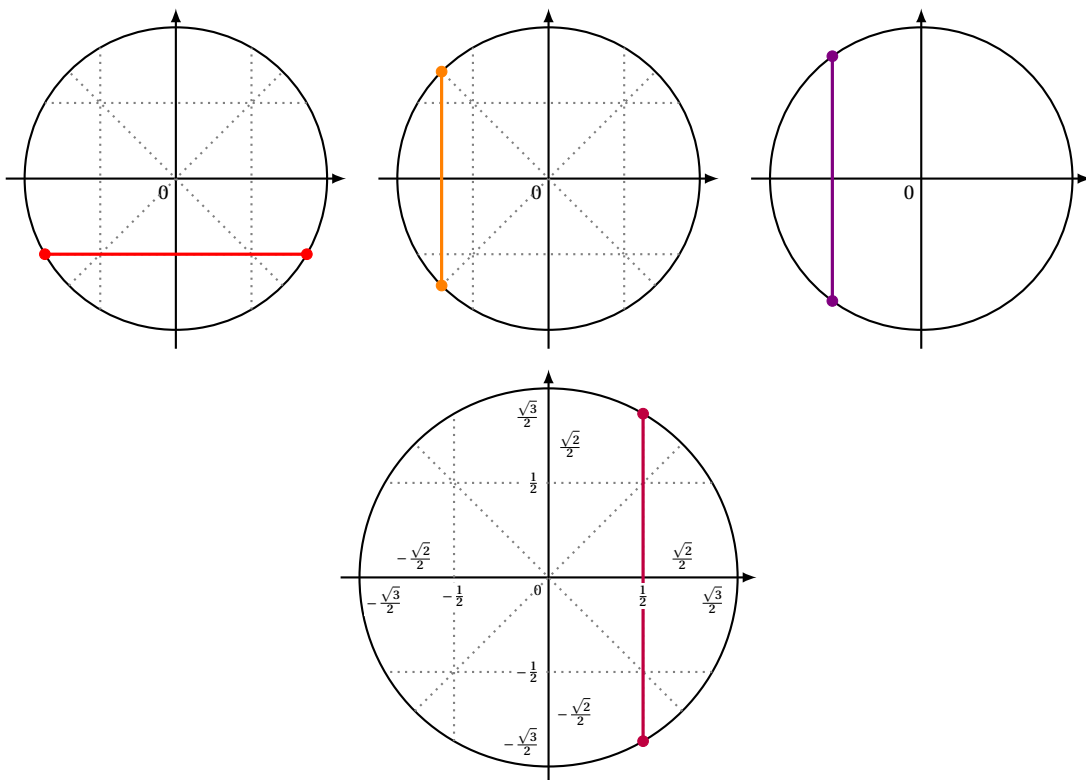


</> Code L^AT_EX

```
\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}
```



Sortie L^AT_EX



Thème

OUTILS POUR LA GÉOMÉTRIE ANALYTIQUE

Huitième partie

Outils pour la géométrie analytique

30 Conseils d'utilisation



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

31 Affichage de coordonnées

31.1 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 \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!

31.2 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` et sortie `\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 calculs et affichage classique
$\AffVecteur((2-(-3));(5-6);(1-1))$
```



```
(1; 2/3)
(-0,5; 1; 2/3)
(1) et (2)
(2)
(0,5)
(2/3)
(0,5)
(0,6)
(0,75)
(5)
(-1)
(0)
```

32 Équation cartésienne d'un plan de l'espace

32.1 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 \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)
```

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>`
 - `<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 `(...)` correspondent aux données utilisées (entre 2 et 3).

À noter que les séparateurs `,` ou `;` permettent de spécifier point ou vecteur.



Code \LaTeX et sortie \LaTeX

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



Une équation cartésienne du plan \mathcal{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 \mathcal{P} :
$$x + 2y + 3z - 32 = 0$$



Code \LaTeX et sortie \LaTeX

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



Une équation cartésienne du plan \mathcal{P} de vecteur normal $\vec{n} \begin{pmatrix} 1/2 \\ 2/3 \\ 3/5 \end{pmatrix}$ et passant par le point A de coordonnées $(4;5;6)$ est \mathcal{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 et sortie \LaTeX

Une équation cartésienne du plan \mathcal{P} de vecteur normal $\vec{n}(1; 2/3; 0)$ et passant par le point A de coordonnées $A(0.75; 0.56; 1/3)$ est :

```
\AffVecteur[n](1;2/3;0) \AffPoint[dec,dec,d](0.75,0.56,1/3) \Leftrightarrow \TrouveEqCartPlan(1;2/3;0)(0.75,0.56,1/3) \SimplifyCoeffs(1;2/3;0)(0.75,0.56,1/3)
```



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

$$\mathcal{P}: x + \frac{2}{3}y - \frac{337}{300} = 0 \Leftrightarrow 300x + 200y - 337 = 0$$


Code \LaTeX et sortie \LaTeX

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

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



Une équation cartésienne du plan \mathcal{P}_3 passant par les points A(2; 0; 1), B(3; 1; 1) et C(1; -2; 0) est

$$\mathcal{P}_3: -x + y - z + 3 = 0$$


Code \LaTeX et sortie \LaTeX

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

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



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

$$\mathcal{R}: -x - 3y + 5z - 5 = 0$$

$$\mathcal{R}: x + 3y - 5z + 5 = 0$$


Code \LaTeX et sortie \LaTeX

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

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



Une équation cartésienne du plan \mathcal{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 :

$$\mathcal{P}_0: 9x + 25y + 32z - 38 = 0$$

33 Équation paramétrique d'une droite de l'espace

33.1 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)
```

33.2 Clés et arguments



Concernant les arguments des commandes :

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

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



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 et sortie \LaTeX

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



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 et sortie \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



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 et sortie \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



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

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

34 Équation cartésienne d'une droite du plan

34.1 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)
```

34.2 Clés et arguments



Concernant les arguments des commandes :

- le premier argument, optionnel et entre $\boxed{\dots}$ contient les clés :
 - $\langle \text{OptionCoeffs} \rangle$ pour spécifier un formatage *global* des coefficients; défaut : $\langle \text{d} \rangle$
 - le booléen $\langle \text{SimplifCoeffs} \rangle$ pour forcer des coeffs simples (entiers et premiers entre eux); défaut : $\langle \text{false} \rangle$
 - $\langle \text{Facteur} \rangle$ pour spécifier un facteur personnalisé aux simplifications; défaut : $\langle 1 \rangle$
 - le booléen $\langle \text{VectDirecteur} \rangle$ pour préciser que le vecteur utilisé est directeur. défaut : $\langle \text{false} \rangle$
- les arguments suivants, entre $\boxed{(\dots)}$ correspondent aux données utilisées.

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



Code \LaTeX et sortie \LaTeX

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



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



Code \LaTeX et sortie \LaTeX

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



Une équation cartésienne de la droite \mathcal{D} de vecteur directeur $\vec{u}\left(\frac{1}{2}, \frac{2}{3}\right)$ et passant par le point A de coordonnées (5;6) est \mathcal{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 et sortie \LaTeX

Une équation cartésienne de la droite \mathcal{D} passant par les points $A(2,4)$ et $B(-4,2)$ est $\mathcal{D} : 2x - 6y + 20 = 0 \Leftrightarrow x - 3y + 10 = 0$



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

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

35 Norme d'un vecteur, distance entre deux points

35.1 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.

35.2 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 et sortie \LaTeX

La distance AB avec $A\text{\texttt{\textbackslashAffPoint}}(-5,2)$ et $B\text{\texttt{\textbackslashAffPoint}}(4,-3)$ vaut
 $d = \text{\texttt{\textbackslashdisplaystyle}\textbackslashTrouveNorme}(-5,2)(4,-3)$



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



Code \LaTeX et sortie \LaTeX

La distance AB avec $A\text{\texttt{\textbackslashAffPoint}}(2,1,2)$ et $B\text{\texttt{\textbackslashAffPoint}}(-4,1,1)$ vaut
 $d = \text{\texttt{\textbackslashdisplaystyle}\textbackslashTrouveNorme}(2,1,2)(-4,1,1)$



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



Code \LaTeX et sortie \LaTeX

La norme de $\text{\texttt{\textbackslashAffVecteur}}(2;4)$ vaut
 $d = \text{\texttt{\textbackslashdisplaystyle}\textbackslashTrouveNorme}(2;4)$



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



Code \LaTeX et sortie \LaTeX

La norme de $\text{\texttt{\textbackslashAffVecteur}}[d;d;n](2;4;0.5)$ vaut
 $d = \text{\texttt{\textbackslashdisplaystyle}\textbackslashTrouveNorme}(2;4;0.5)$



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

36 Distance d'un point à un plan

36.1 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.

36.2 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 AffPoint , ou AffVecteur permettent de spécifier point ou vecteur pour les arguments 1 et 3.



Code \LaTeX et sortie \LaTeX

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



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 et sortie \LaTeX

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



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

$$d = 2$$



Code \LaTeX et sortie \LaTeX

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



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 et sortie \LaTeX

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



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 et sortie \LaTeX

La distance entre le point $H(\text{AffPoint}(0,0,5))$ et le plan d'équation $z-1=0$ vaut
 $\left[d = \displaystyle \text{TrouveDistancePtPlan}(0,0,5)(z-1=0) \right]$



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

$$d = 4$$

37 Équation réduite d'une droite du plan

37.1 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 \LaTeX

```
\EquationReduite[option]{A/xa/ya,B/xb/yb}
```

37.2 Clés et arguments



Concernant le fonctionnement de la commande :

- le premier argument, optionnel et entre `[...]` et valant `<[d]>` par défaut, permet de formater les fractions éventuelles en mode `\displaystyle`;
- le second argument, obligatoire et entre `{...}`, permet de donner les coordonnées des points concernés.



Code \LaTeX et sortie \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 :

$$y_C = 2 \times x_C + p \Rightarrow 0 = 2 \times 2 + p \Rightarrow p = 0 - (2 \times 2) \Rightarrow p = -4$$

Donc l'équation réduite de (CD) est $y = 2x - 4$.

37.3 Exemples



Code \LaTeX et sortie \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 et sortie \LaTeX

$\backslash\text{EquationReduite}\{U/-4/5,V/-4/5\}$



Les deux points donnés sont identiques, donc pas de droite...



Code \LaTeX et sortie \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 et sortie \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 $L(\frac{1}{3}; 2,5)$ 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 et sortie \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 $y = -x$.



Code \LaTeX et sortie \LaTeX

$\backslash\text{EquationReduite}\{G/-4/5,H/10/4\}$



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 :

$$y_G = -\frac{1}{14} \times x_G + p \Rightarrow 5 = -\frac{1}{14} \times (-4) + p \Rightarrow p = 5 - \left(-\frac{1}{14} \times (-4)\right) \Rightarrow p = \frac{33}{7}$$

Donc l'équation réduite de (GH) est $y = -\frac{1}{14}x + \frac{33}{7}$.

Thème

OUTILS POUR LES STATISTIQUES

Outils pour les statistiques

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 \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 `\NomCoeffa` qui permet de définir la variable qui contiendra a ; défaut `\COEFFa`
- la clé `\NomCoeffb` qui permet de définir la variable qui contiendra b ; défaut `\COEFFb`
- la clé `\NomCoeffr` qui permet de définir la variable qui contiendra r ; défaut `\COEFFr`
- la clé `\NomCoeffrd` qui permet de définir la variable qui contiendra r^2 ; défaut `\COEFFrd`
- la clé `\NomXmin` qui permet de définir la variable qui contiendra x_{\min} ; défaut `\LXmin`
- la clé `\NomXmax` qui permet de définir la variable qui contiendra x_{\max} ; défaut `\LXmax`



Code \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 \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 $a=\text{\COEFFa}$.
Coefficient $b=\text{\COEFFb}$.
Coefficient $r=\text{\COEFFr}$.
Coefficient $r^2=\text{\COEFFrd}$.
```



Sortie \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.

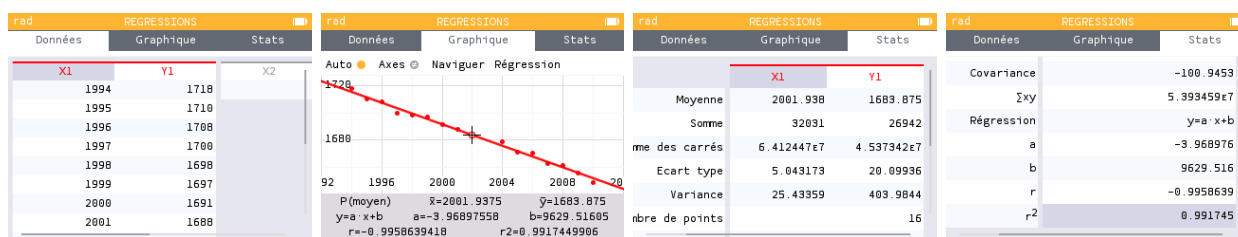
Variance des X := 25.43359375 et variance des Y := 403.984375

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 éq. est $y=\arrond[3]{\TESTa}x
\arrond*[3]{\TESTb}$.
Le coeff. de corrélation est $r=\arrond[4]{\TESTr}$, et son carré est
$r^2=\arrond[4]{\TESTrd}$.
```



Sortie \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 $r^2 = 0,7954$.



rad. REGRESSIONS			rad. REGRESSIONS		
Données	Graphique	Stats	Données	Graphique	Stats
X1	Y1	X2	Covariance		-3.13333
0	-35		Σxy		-742.7
1	-37.4		Régression	$y=a \cdot x+b$	
3	-37.7		a		-0.7006211
4	-39.9		b		-35.88137
5	-39		r		-0.891847
6	-39.6		r^2		0.7953911

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

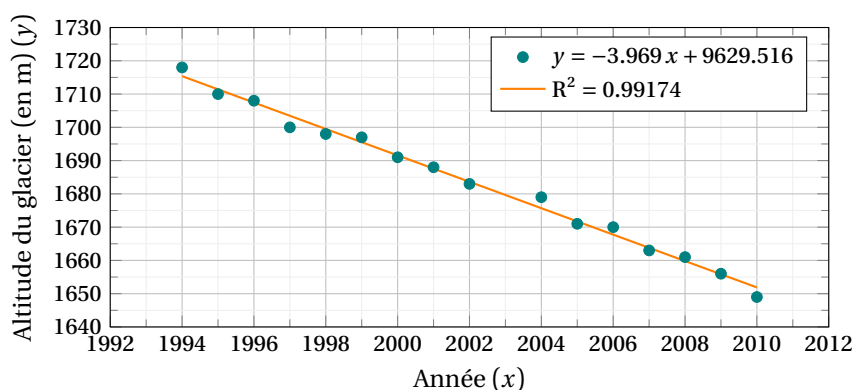
```

\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=\LLXmin:\LLXmax,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}

```



Sortie \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>**

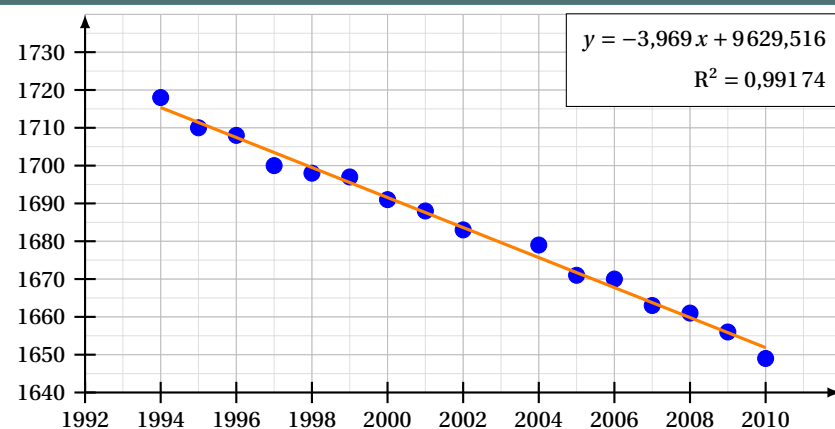


Code \LaTeX

```
\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 {${y}=\text{num}\{\text{fpeval}\{\text{round}(\COEFFa,3)\}}\,x+\text{num}\{\text{fpeval}\{\text{round}(\COEFFb,3)\}}\}$} ; \\
\node {${R^2}=\text{num}\{\text{fpeval}\{\text{round}(\COEFFrd,5)\}}\}$} ; \\
};
\end{tikzpicture}
```



Sortie \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 O.

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 \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 \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
\AxeYtikz{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 \LaTeX

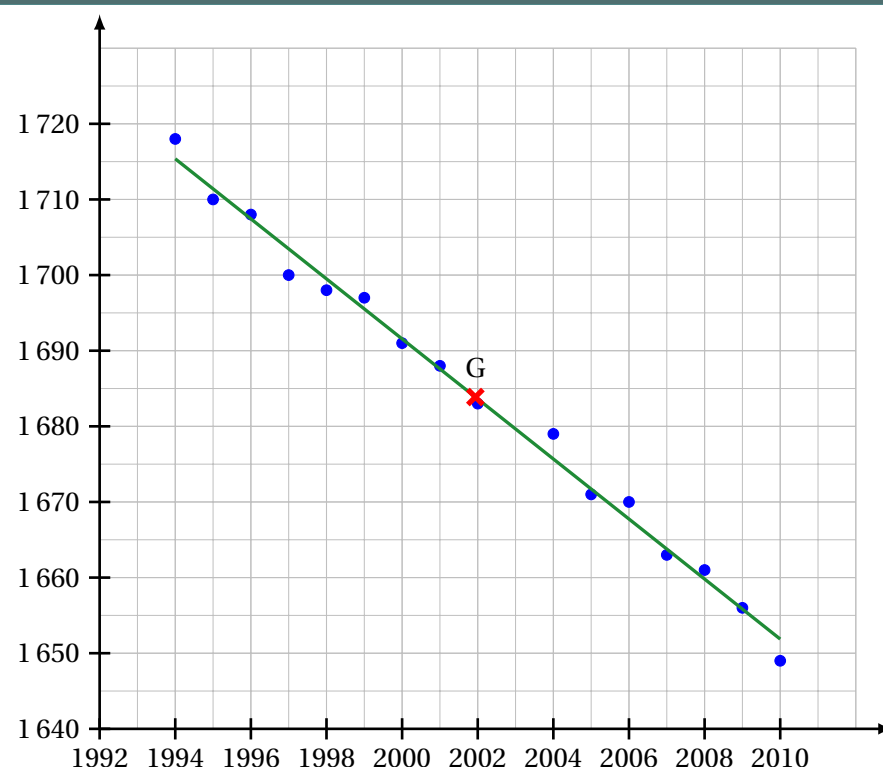
```

%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}

```



Sortie \LaTeX



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 \LaTeX

```
%...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**

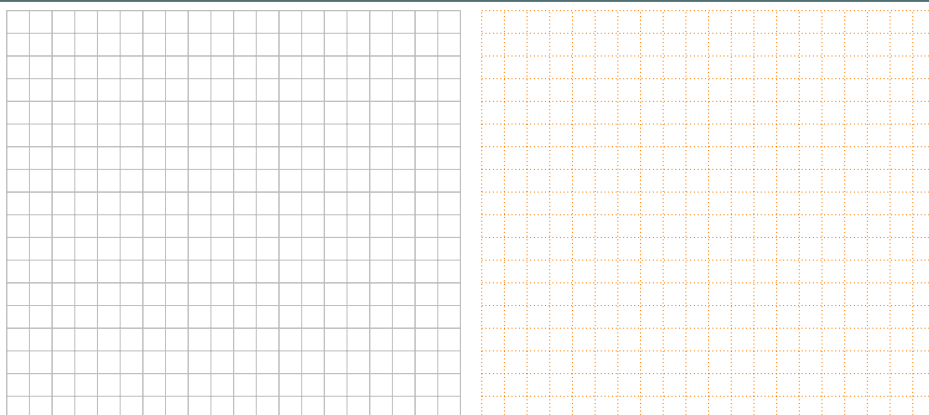


Code \LaTeX

```
\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}
```



Sortie \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**



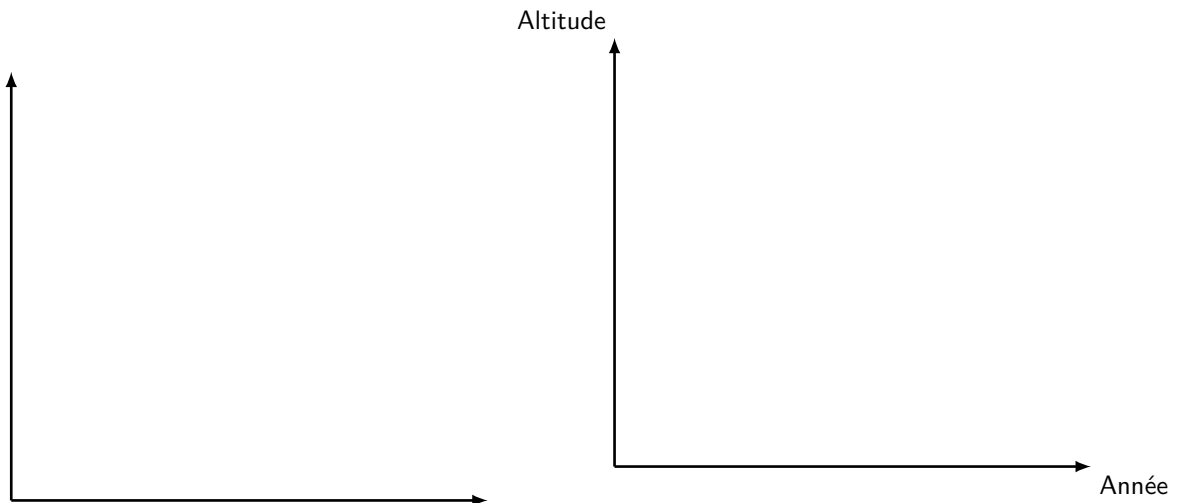
</> Code \LaTeX

```
%code tikz
\AxesTikz

%code tikz
\AxesTikz%
[AffLabel=xy,Labelx={Année},Labely={Altitude},%
PosLabelx={below right},PosLabely={above left},%
Police=\small\sffamily]
```



Sortie \LaTeX



</> Code \LaTeX

```
%...code tikz
\AxeTikz[options]{valeurs}
\AxeTikz[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 `\normalsize\normalfont`
- **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 **lgtb**); 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 L^AT_EX

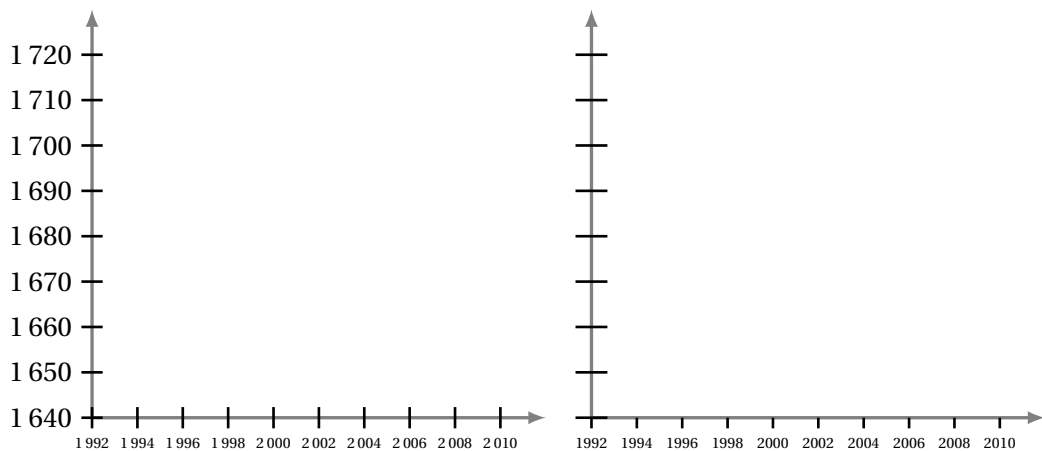
```
%code tikz
\AxexTikz[Police=\small]{1992,1994,...,2010}
\AxexTikz{1640,1650,...,1720}

%code tikz
\AxyTikz[Police=\small,Annee,HautGrad=0pt/4pt]{1992,1994,...,2010}
\AxyTikz[AffGrad=false,HautGrad=6pt]{1640,1650,...,1720}

%des axes fictifs (en gris) sont rajoutés pour la lisibilité du code de sortie
```



Sortie L^AT_EX



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 L^AT_EX

```
%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 \LaTeX

```
%...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**



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}
```

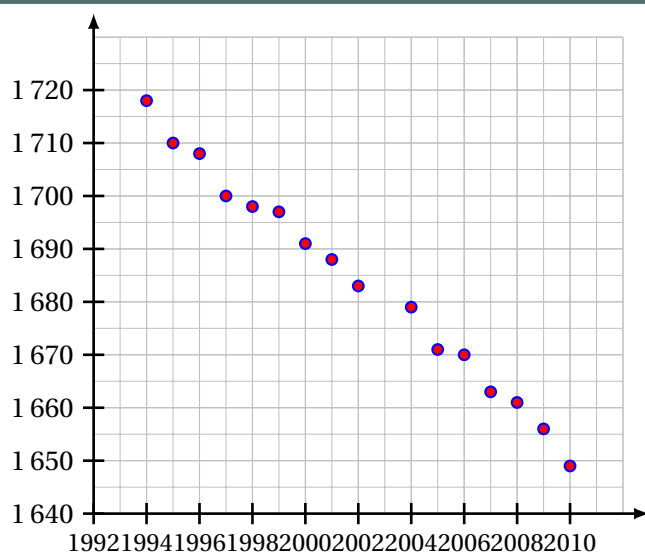


Code \LaTeX

```
\begin{tikzpicture}[...]  
\NuagePointsTikz[Couleur=blue/red]{\LLX}{\LLY}  
\end{tikzpicture}
```

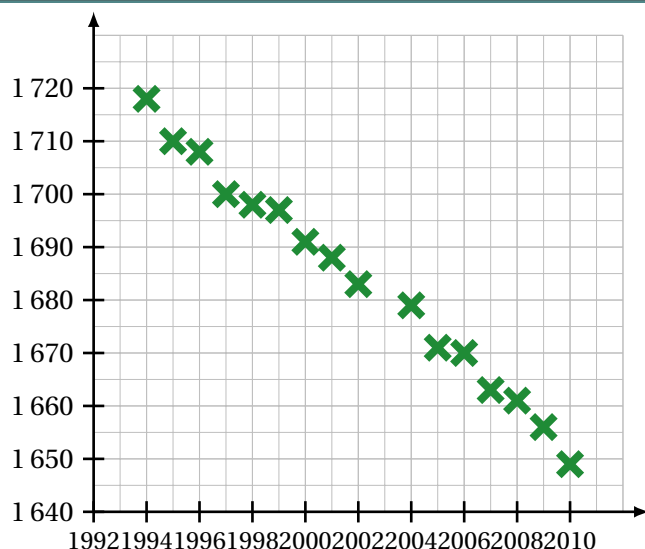


Sortie \LaTeX



Code \LaTeX

```
\begin{tikzpicture}[...]  
\NuagePointsTikz[Couleur=CouleurVertForet,Style=x,Taille=6pt]{\LLX}{\LLY}  
\end{tikzpicture}
```

Sortie \LaTeX Code \LaTeX

```
%...code tikz
\PointMoyenTikz[options]
```



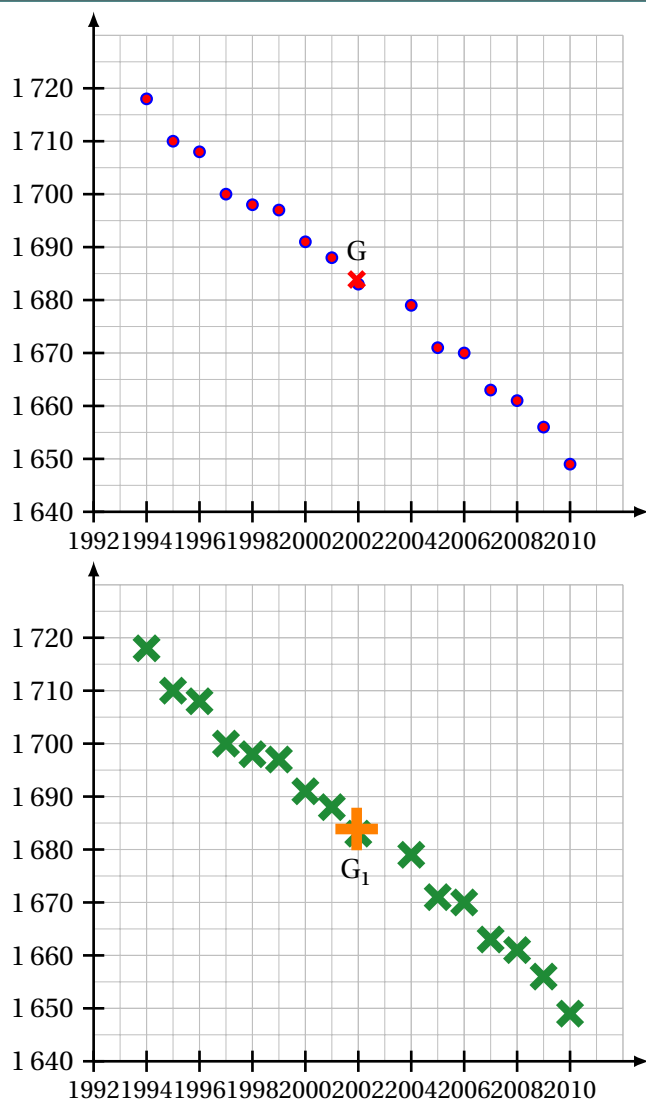
Cette commande permet de rajouter le point moyen du nuage, calculé par la commande `\CalculsRegLin`, avec les **clés** :

- **<Police>**, comme précédemment; défaut `<\normalsize\normalfont>`;
- **<Taille>**, taille du point moyen; défaut `<4pt>`
- **<Couleur>**, couleur du point moyen; défaut `<red>`
- **<Style>** parmi **<o>** (rond) ou **<x>** (croix) ou **<+>** (plus); défaut `<o>`
- **<xg>**, abscisse du point moyen, récupérable via `\CalculsRegLin`; défaut `<\LXmoy>`
- **<yg>**, ordonnée du point moyen, récupérable via `\CalculsRegLin`; défaut `<\LYmoy>`
- **<Nom>**, label du point moyen; défaut `<G>`
- **<Pos>** qui est la position du label par rapport au point; défaut `<above>`
- **<Decal>** qui est l'éloignement de la position du label par rapport au point; défaut `<0pt>`
- la booléen **<AffNom>** qui affiche ou non le libellé. défaut `<>true>`

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
\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}
```

Sortie \LaTeX Code \LaTeX

```
%...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 .



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

```

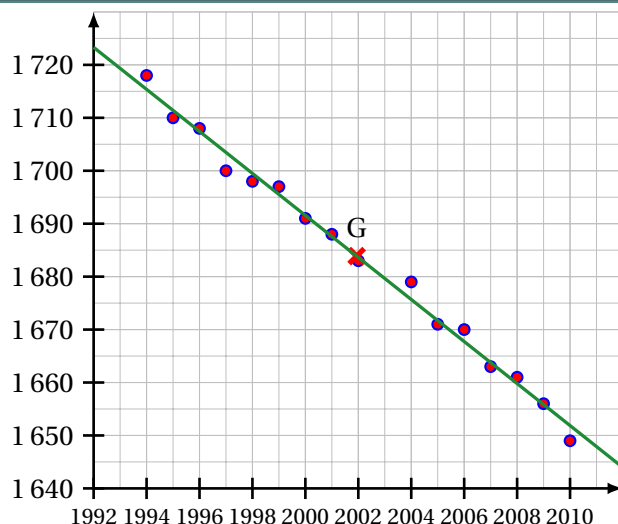
\def\LLX{1994,1995,1996,1997,1998,1999,2000,2001,2002,2004,2005,2006,2007,2008, 2009,2010}
\def\LLY{1718,1710,1708,1700,1698,1697,1691,1688,1683,1679,1671,1670,1663,1661, 1656,1649}
\CalculsRegLin{\LLX}{\LLY}

\begin{tikzpicture}[...]
\NuagePointsTikz[Couleur=blue/red]{\LLX}{\LLY} \PointMoyenTikz
\CourbeTikz[line width=1.25pt,CouleurVertForet,samples=2]{\COEFFa*\x+\COEFFb}{\xmin:\xmax}
\end{tikzpicture}

```



Sortie \LaTeX



39.5 Exemple complémentaire, pour illustration



</> Code \LaTeX

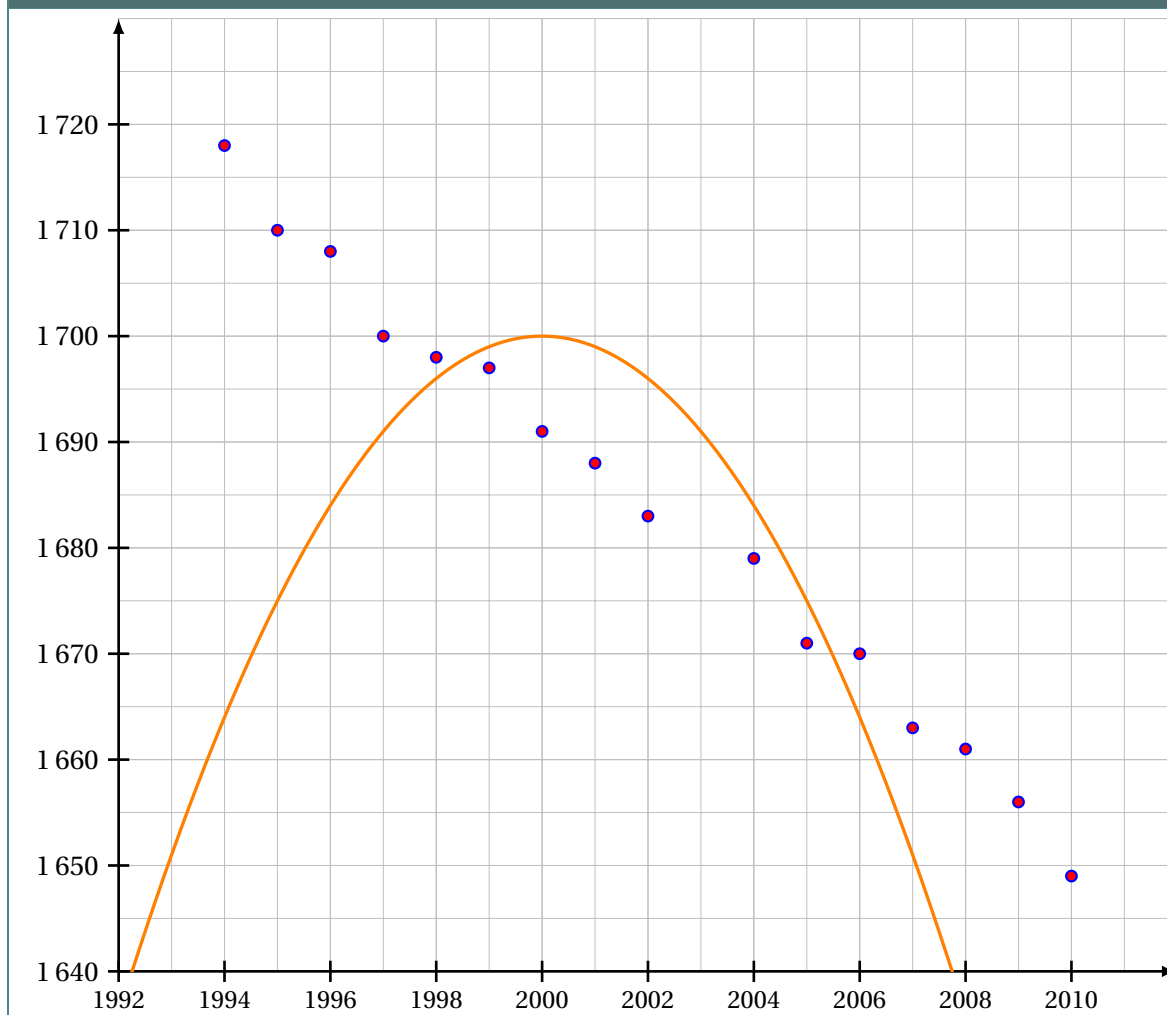
```

\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



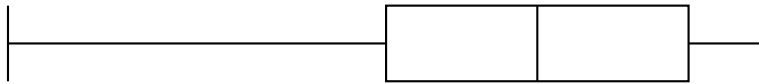
L'idée est de proposer une commande, à intégrer dans un environnement TikZ, pour tracer une boîte à moustaches grâce aux paramètres, saisis par l'utilisateur.

Le code ne calcule pas les paramètres, il ne fait *que* tracer la boîte à moustaches!



Code \LaTeX et sortie \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



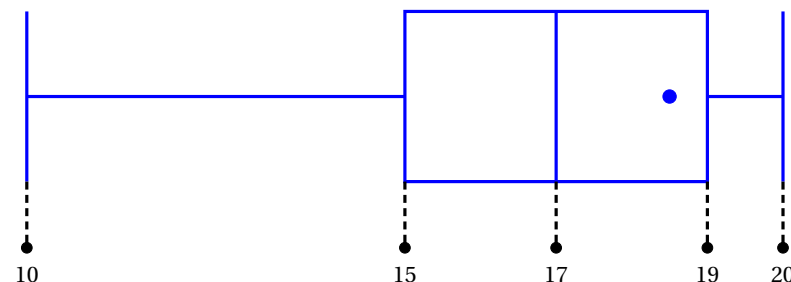
Quelques **clés** sont disponibles pour cette commande :

- 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 \LaTeX et sortie \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}
```



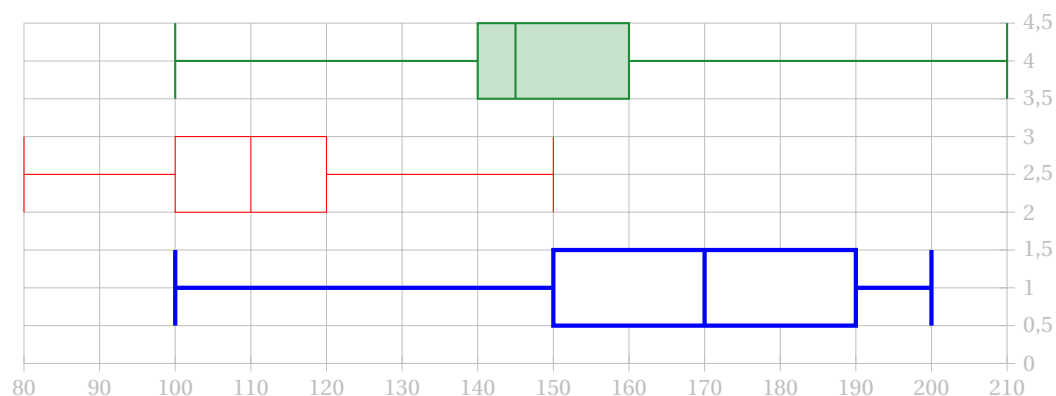


Code \LaTeX

```
%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}
```



Sortie \LaTeX



40.3 Commande pour placer un axe horizontal

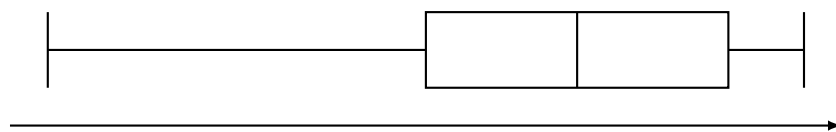


L'idée est de proposer, en parallèle de la commande précédente, une commande pour tracer un axe horizontal « sous » les éventuelles boîtes à moustaches.



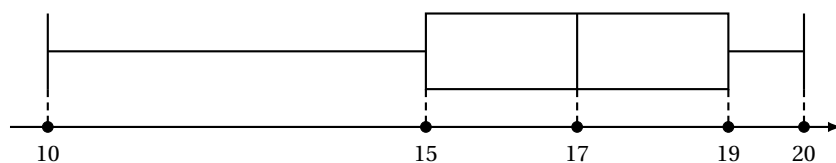
Code \LaTeX et sortie \LaTeX

```
\begin{tikzpicture}
  \BoiteMoustachesAxe[Min=10,Max=20]
  \BoiteMoustaches{10/15/17/19/20}
\end{tikzpicture}
```



Code \LaTeX et sortie \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 :

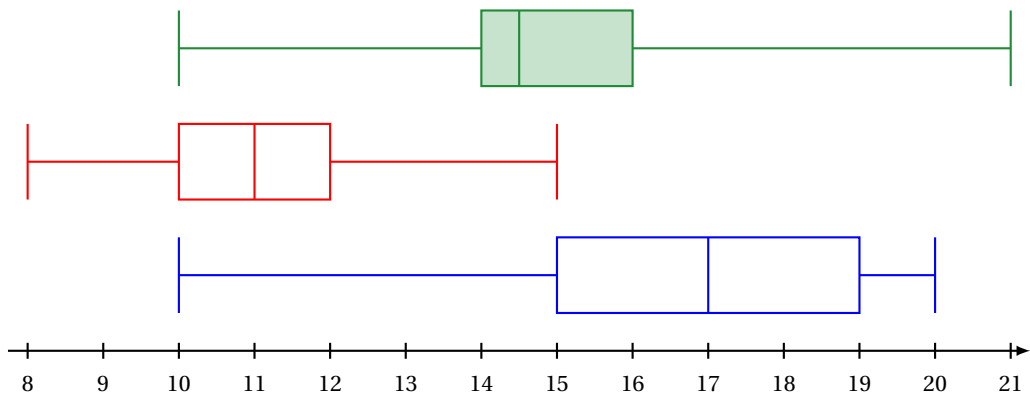
- la clé **Min** qui est la valeur minimale de l'axe horizontal;
- la clé **Max** qui est la valeur maximale de l'axe horizontal;
- la clé **Elargir** qui est le pourcentage d'élargissement de l'axe; défaut **0.1**
- la clé **Epaisseur** qui est l'épaisseur des traits de la boîte; défaut **thick**
- la clé **Valeurs** qui est la liste (compréhensible en TikZ) des valeurs à afficher.



Code \LaTeX et sortie \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`.



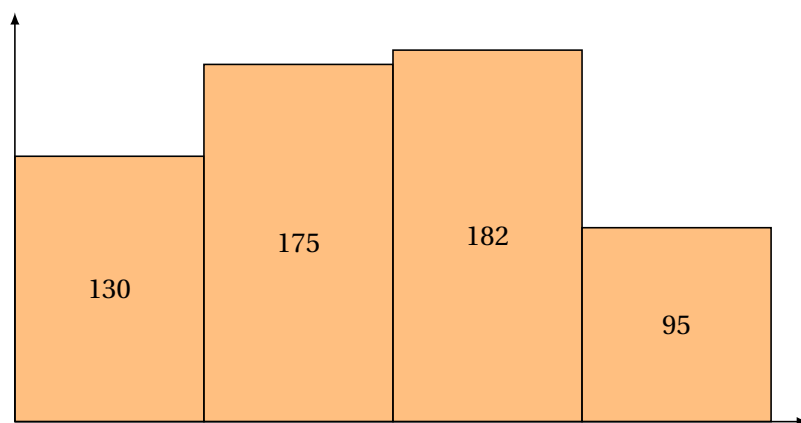
Code \LaTeX

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



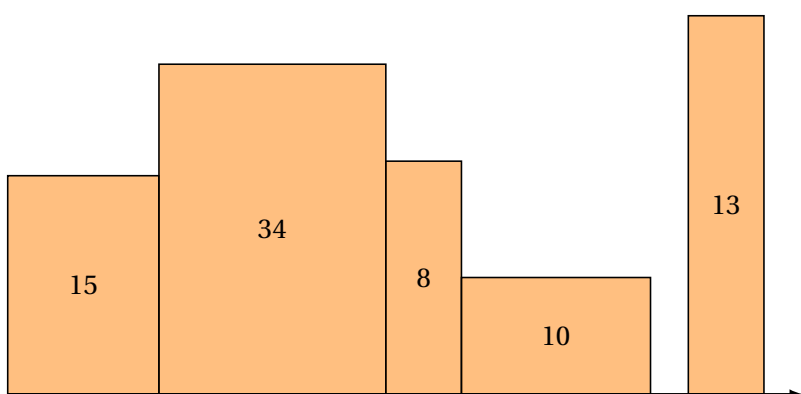
Code \LaTeX et sortie \LaTeX

```
%classes régulières  
\Histogramme{7/9/130 9/11/175 11/13/182 13/15/95}
```



Code \LaTeX et sortie \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 tableur);
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>**
- **<PosLégende>** : 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 ...`.

41.3 Exemple avec des classes régulières



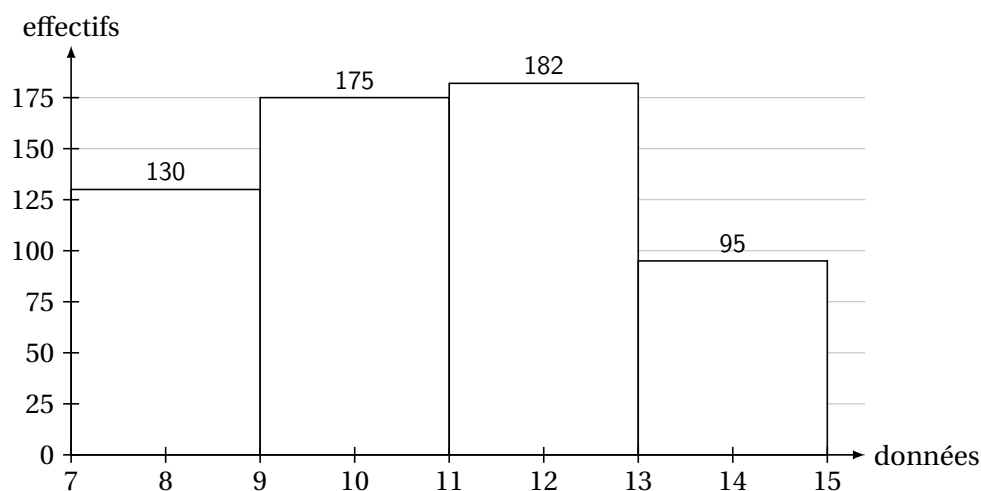
Avec la série suivante :

Classes	[7;9[[9;11[[11;13[[13;15]
Effectifs	130	175	182	95



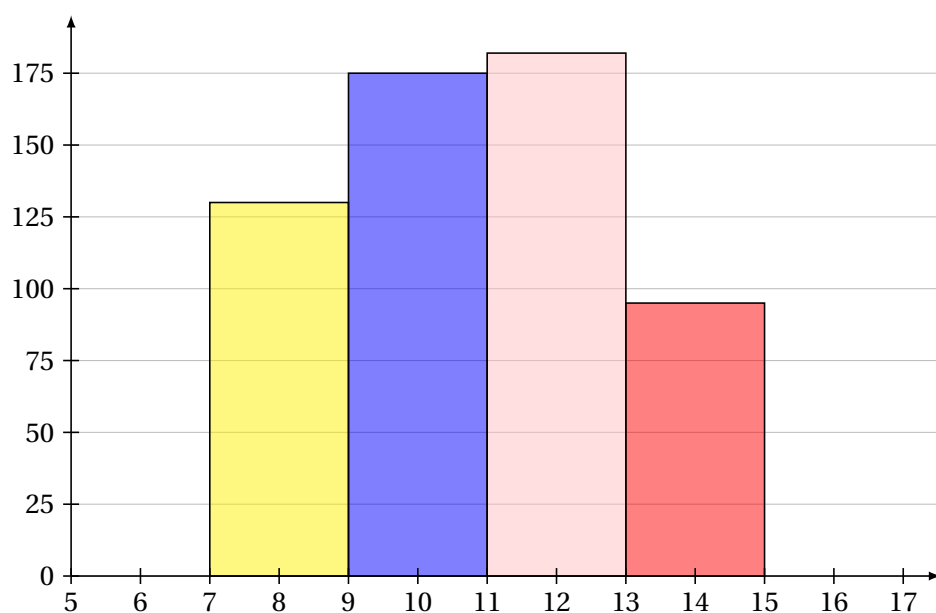
Code \LaTeX et sortie \LaTeX

```
\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 \LaTeX et sortie \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



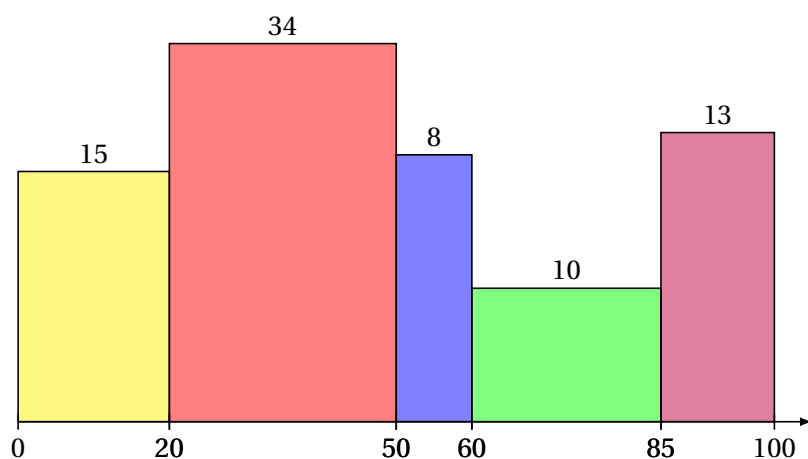
Avec la série suivante :

Classes	[0;20[[20;50[[50;60[[60;85[[85;100]
Effectifs	15	34	8	10	13



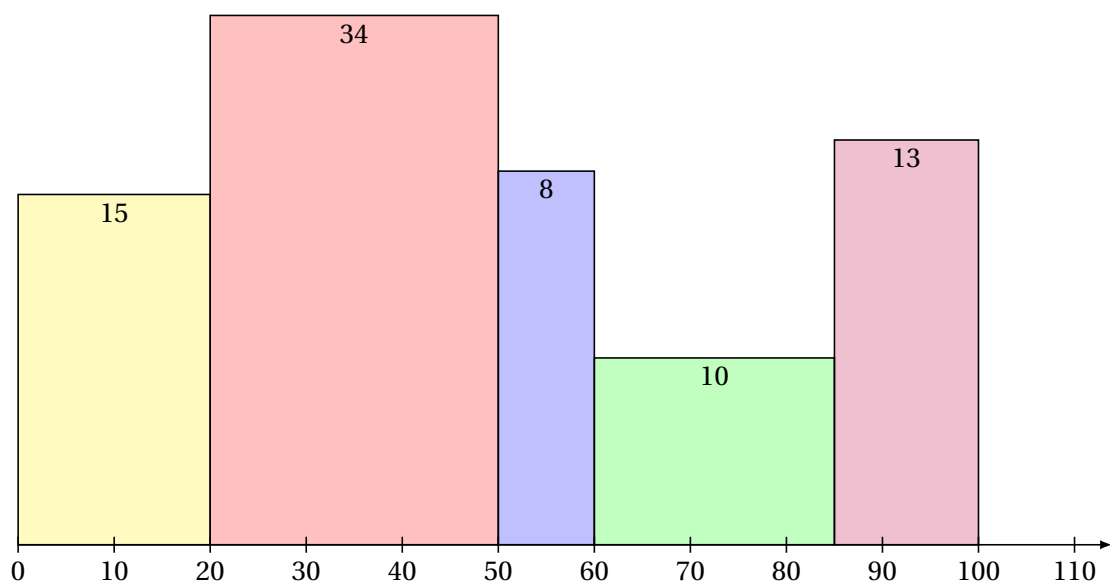
Code \LaTeX et sortie \LaTeX

```
\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 et sortie \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 :

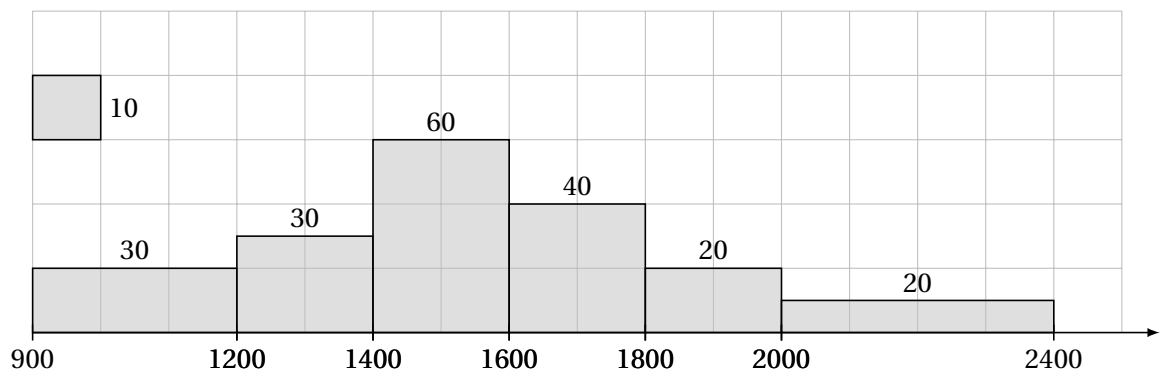
Classes	[900;1200[[1200;1400[[1400;1600[[1600;1800[[1800;2000[[2000;2400]
Effectifs	30	30	60	40	20	20



Code \LaTeX et sortie \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 et sortie \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
```

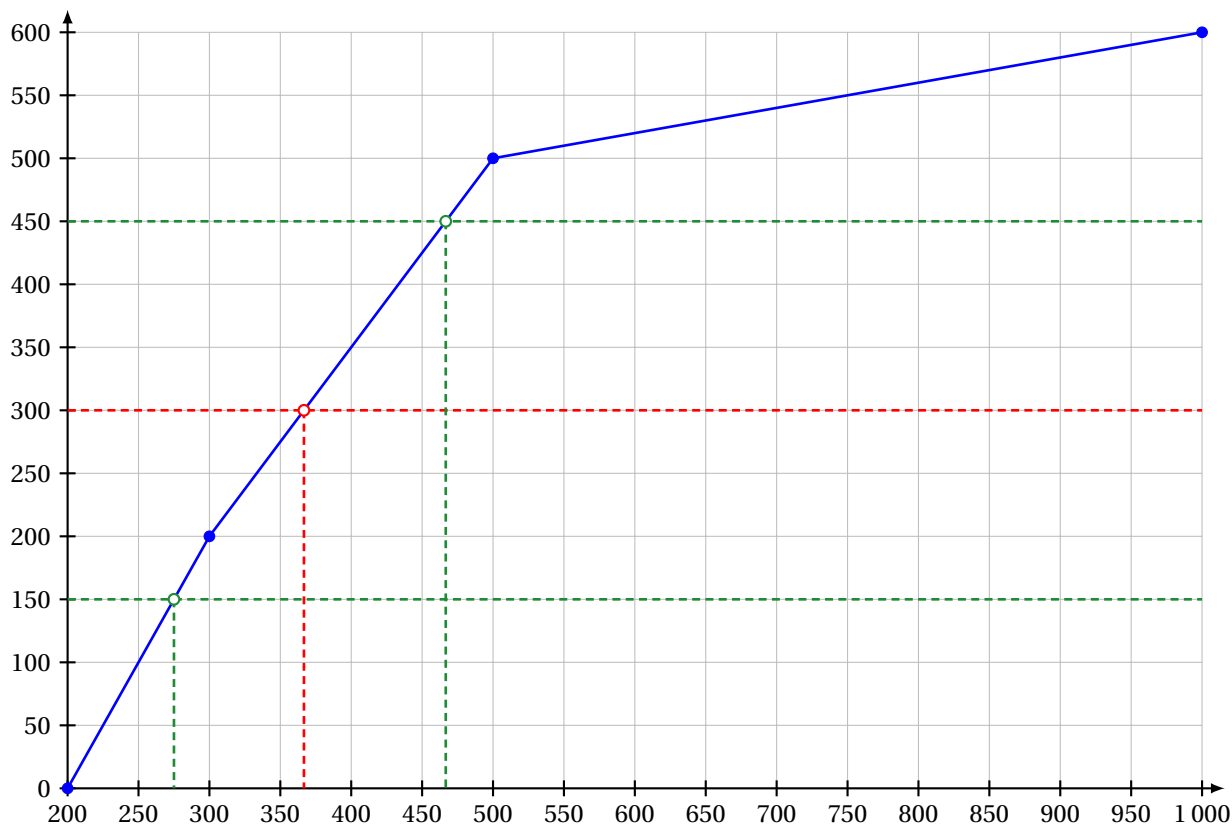


```
Q1 = 275  
Méd = 366.6666666666667  
Q3 = 466.6666666666667
```



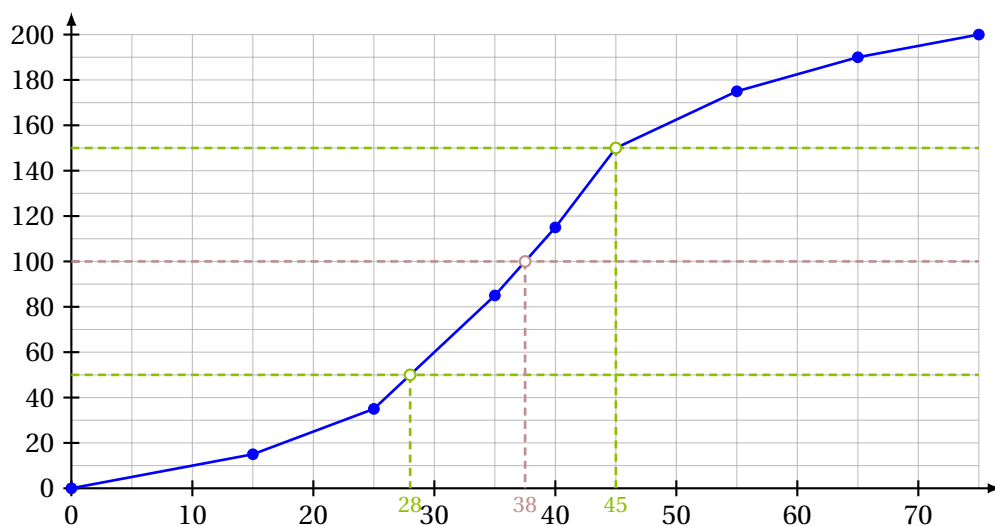
Code \LaTeX et sortie \LaTeX

```
\CourbeECC%
[GraduationsX={200,250,...,1000},GraduationsY={0,50,...,600},PoliceAxes=\small]%
{200,300,500,1000}%borne des classes
{200,300,100}%effectifs
```



Code \LaTeX et sortie \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 `\CourbeECCStylesDefaut` permet de réinitialiser les valeurs par défaut si modification.



Code \LaTeX

```
\tikzset{traitsparamecc/.style={line width=1pt,densely dashed}}
\tikzset{courbeecc/.style={line width=1pt}}
\tikzset{gradsecc/.style={thick}}
\tikzset{axececc/.style={gradsecc,->,>=latex}}
```



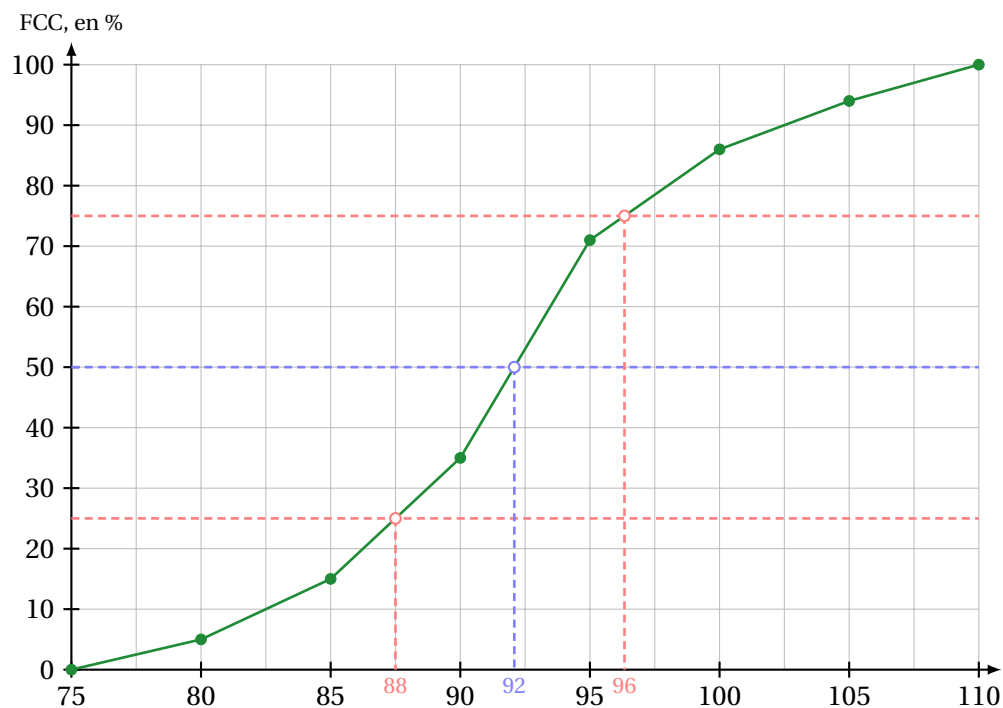
On va utiliser les données suivantes :

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 \LaTeX et sortie \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}
```



Thème

OUTILS POUR LES PROBABILITÉS

Outils pour les probabilités

43 Calculs de probabilités

43.1 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.

43.2 Calculs « simples »



Code L^AT_EX

```
%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{p}{a}{b}          %P(a<=X<=b)

%loi hypergéométrique H(N,n,m)
\CalcHypergeomP{N}{n}{m}{k}   %P(X=k)
\CalcHypergeomP{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 Ponctuelles ($P(X = k)$) et Cumulées ($P(a \leq X \leq b)$).

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)
$P(X=3) \approx \CalcBinomP{5}{0.4}{3}$.
$P(X\leqslant 1) \approx \CalcBinomC{5}{0.4}{*}{1}$.

% X -> B(100, 0.02)
$P(X=10) \approx \CalcBinomP{100}{0.02}{10}$.
$P(15\leqslant X\leqslant 25) \approx \CalcBinomC{100}{0.02}{15}{25}$.

% Y -> P(5)
$P(Y=3) \approx \CalcPoissP{5}{3}$.
$P(Y\geqslant 2) \approx \CalcPoissC{5}{*}{2}$.

% T -> G(0.5)
$P(T=100) \approx \CalcPoissP{0.5}{3}$.
$P(T\leqslant 5) \approx \CalcPoissC{0.5}{*}{5}$.

% W -> H(50, 10, 5)
$P(W=4) \approx \CalcHypergeomP{50}{10}{5}{4}$.
$P(1\leqslant W\leqslant 3) \approx \CalcHypergeomC{50}{10}{5}{1}{3}$.
```



Sortie \LaTeX

- $X \hookrightarrow \mathcal{B}(5; 0, 4)$:
 $P(X = 3) \approx 0.2304$.
 $P(X \leqslant 1) \approx 0.33696$.
- $X \hookrightarrow \mathcal{B}(100; 0, 02)$:
 $P(X = 10) \approx 0.0000287707765846743$.
 $P(15 \leqslant X \leqslant 25) \approx 0.000000001670210428685021$.
- $Y \hookrightarrow \mathcal{P}_5$:
 $P(Y = 3) \approx 0.1403738958142806$.
 $P(Y \geqslant 2) \approx 0.9595723180054873$.
- $T \hookrightarrow \mathcal{G}_{0,5}$:
 $P(T = 3) \approx 0.125$.
 $P(T \leqslant 5) \approx 0.96875$.
- $W \hookrightarrow \mathcal{H}(50; 10; 5)$:
 $P(W = 4) \approx 0.003964583058015065$.
 $P(1 \leqslant W \leqslant 3) \approx 0.6853536974456758$.



Code \LaTeX

```
% X -> N(0, 1)
$P(X\leqslant 1) \approx \CalcNormC{0}{1}{*}{1}$.
$P(-1,96\leqslant Z\leqslant 1,96) \approx \CalcNormC{0}{1}{-1.96}{1.96}$.

% X -> N(550, 30)
$P(Y\geqslant 600) \approx \CalcNormC{550}{30}{600}{*}$.
$P(500\leqslant Y\leqslant 600) \approx \CalcNormC{550}{30}{500}{600}$.

% Z -> E(0.001)
$P(Z\geqslant 400) \approx \CalcExpoC{0.001}{400}{*}$.
$P(300\leqslant Z\leqslant 750) \approx \CalcExpoC{0.001}{300}{750}$.
```



Sortie \LaTeX

- $X \hookrightarrow \mathcal{N}(0; 1)$:
 $P(X \leqslant 1) \approx 0.841344680841397$.
 $P(-1,96 \leqslant Z \leqslant 1,96) \approx 0.9500039553976748$.
- $Y \hookrightarrow \mathcal{N}(550; 30)$:
 $P(Y \geqslant 600) \approx 0.0477903462453939$.
 $P(500 \leqslant Y \leqslant 600) \approx 0.9044193075092122$.
- $Z \hookrightarrow \mathcal{E}_{0,001}$:
 $P(Z \geqslant 400) \approx 0.6703200460356393$.
 $P(300 \leqslant Z \leqslant 750) \approx 0.2684516679407032$.

43.3 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 \LaTeX

```
%avec un formatage manuel
\num[exponent-mode=scientific]{\CalcBinomP{100}{0.02}{10}}
```



Code \LaTeX et sortie \LaTeX

```
$\bullet\sim\sim\sim X \hookrightarrow \mathcal{B}(100\,,\,; \,0,02)$ :

$P(X=10) \approx \num[exponent-mode=scientific]{\CalcBinomP{100}{0.02}{10}}$.
```



- $X \hookrightarrow \mathcal{B}(100; 0,02)$:
 $P(X = 10) \approx 2,877\,077\,765\,846\,743 \times 10^{-5}$.



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 \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.



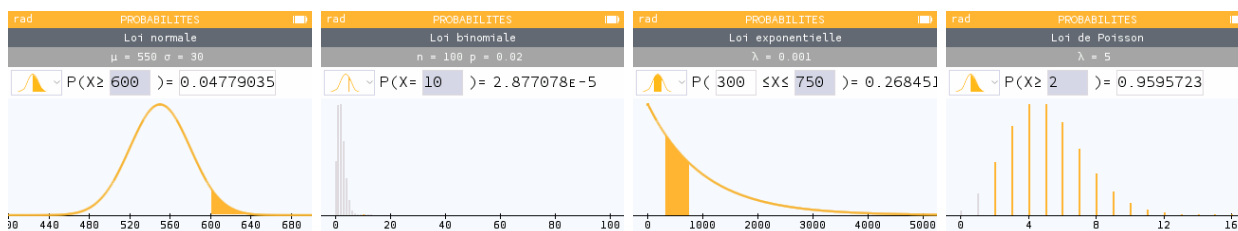
Code \LaTeX

```
% X -> N(550,30)
$P(Y\geqslant 600) \approx \NormalC[4]{550}{30}{600}{*}$.
$P(500\leqslant Y\leqslant 600) \approx \NormalC[4]{550}{30}{500}{600}$.
% X -> B(100,0.02)
$P(X=10) \approx \BinomP[7]{100}{0.02}{10} \approx \BinomP*[7]{100}{0.02}{10}$.
$P(15\leqslant X\leqslant 25) \approx \BinomC[10]{100}{0.02}{15}{25} \approx \approx
\BinomC*[10]{100}{0.02}{15}{25}$.
% H -> H(50,10,5)
$P(W=4) \approx \HypergeomP[5]{50}{10}{5}{4}$.
$P(1\leqslant W\leqslant 3) \approx \HypergeomC[4]{50}{10}{5}{1}{3}$.
% Z -> E(0,001)$ :
$P(Z\geqslant 400) \approx \ExpoC{0.001}{400}{*}$.
$P(300\leqslant Z\leqslant 750) \approx \ExpoC{0.001}{300}{750}$.
% T -> P(5)
$P(T=3) \approx \PoissonP{5}{3}$.
$P(T\geqslant 2) \approx \PoissonC[4]{5}{2}{*}$.
```



Sortie \LaTeX

- $Y \hookrightarrow \mathcal{N}(550; 30)$:
 $P(Y \geq 600) \approx 0,0478$.
 $P(500 \leq Y \leq 600) \approx 0,9044$.
- $X \hookrightarrow \mathcal{B}(100; 0,02)$:
 $P(X = 10) \approx 0,000\,028\,8 \approx 2,88 \times 10^{-5}$.
 $P(15 \leq X \leq 25) \approx 0,000\,000\,001\,7 \approx 1,7 \times 10^{-9}$.
- $W \hookrightarrow \mathcal{H}(50; 10; 5)$:
 $P(W = 4) \approx 0,003\,96$.
 $P(1 \leq W \leq 3) \approx 0,685\,4$.
- $Z \hookrightarrow \mathcal{E}_{0,001}$:
 $P(Z \geq 400) \approx 0,670$.
 $P(300 \leq Z \leq 750) \approx 0,268$.
- $T \hookrightarrow \mathcal{P}_5$:
 $P(T = 3) \approx 0,140$.
 $P(T \geq 2) \approx 0,9596$.



44 Arbres de probabilités « classiques »

44.1 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 \LaTeX

```
%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}
```

44.2 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).



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>**
- 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 \LaTeX

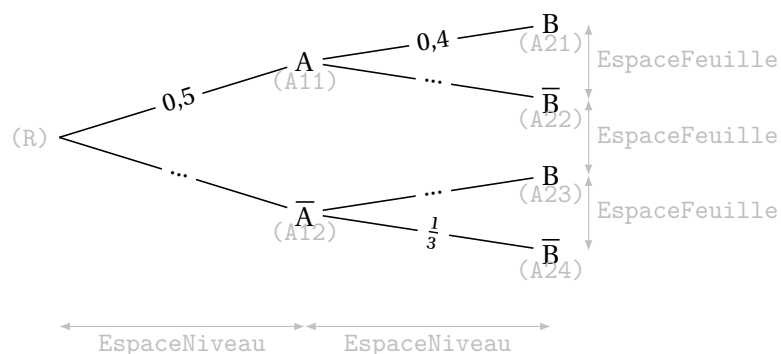
```
\def\ArbreDeuxDeux{
  $A$/\num{0.5}/,
  $B$/\num{0.4}/,
  $\overline{B}$/\dots/,
  $\overline{A}$/\dots/,
  $B$/\dots/,
  $\overline{B}$/$\frac{1}{3}$/$/
}
```

```
\ArbreProbabTikz{\ArbreDeuxDeux}
```

%des éléments, en gris, ont été rajoutés pour illustrer certaines options



Sortie \LaTeX



Les nœuds créés par les commandes sont :

- \LaTeX `R` pour la racine;
- \LaTeX `A1x` pour les nœuds du 1^{er} niveau (de haut en bas);
- \LaTeX `A2x` pour les nœuds du 2^d niveau (de haut en bas).

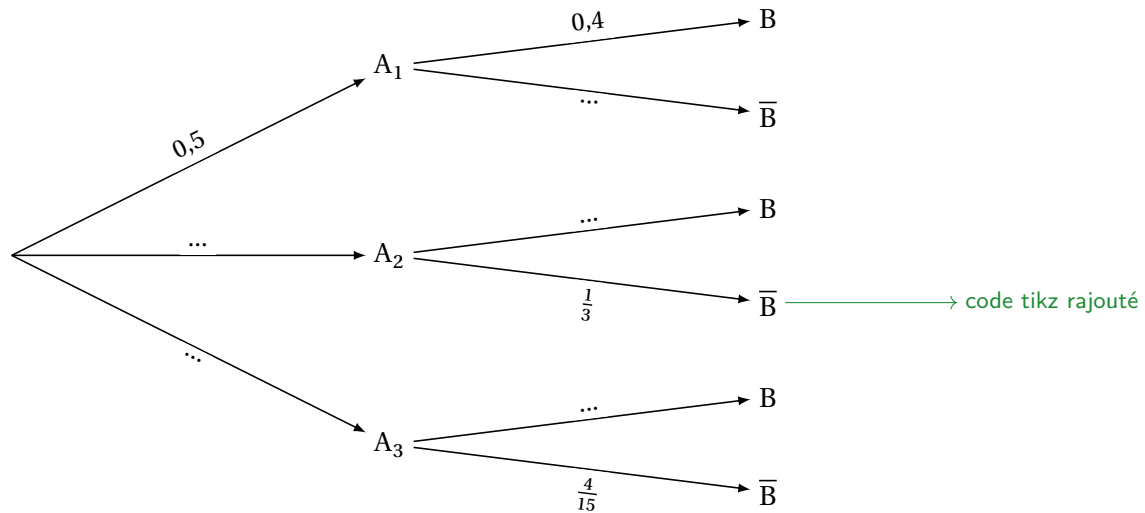
44.3 Exemples complémentaires



Code \LaTeX

```
\def\ArbreTroisDeux{
  $A_1$/\num{0.5}/above,
  $B$/\num{0.4}/above,
  $\overline{B}$/\dots/below,
  $A_2$/\dots/above,
  $B$/\dots/above,
  $\overline{B}$/$\frac{1}{3}$/$below,
  $A_3$/\dots/below,
  $B$/\dots/above,
  $\overline{B}$/$\frac{4}{15}$/$below
}

\begin{EnvArbreProbabTikz}[Type=3x2,Fleche,EspaceNiveau=5,EspaceFeuille=1.25]%
{\ArbreTroisDeux}
\draw[CouleurVertForet,->] (A24)--($(A24)+(2.5,0)$) node[right,font=\sffamily] {code tikz
rajouté} ;
\end{EnvArbreProbabTikz}
```

Sortie \LaTeX Code \LaTeX

```

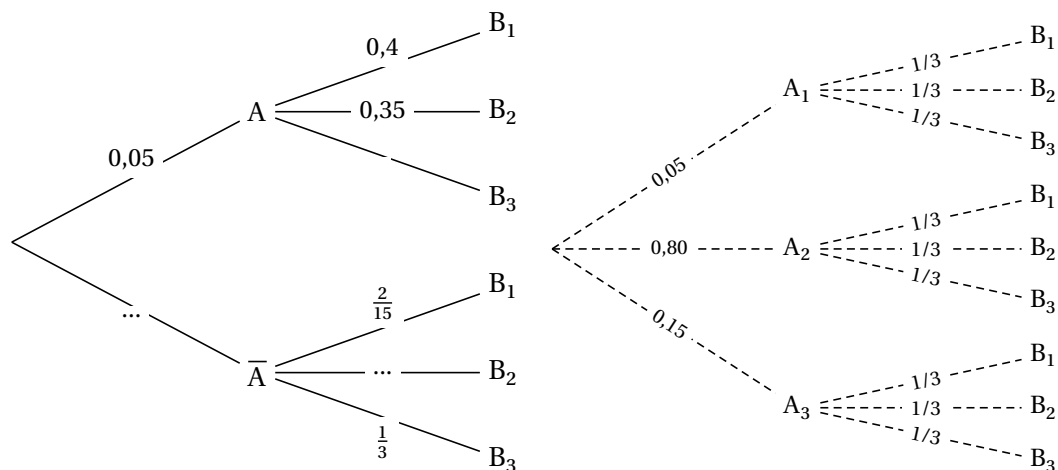
\def\ArbreDeuxTrois{
  $A$/\num{0.05}/above,
  $B_1$/\num{0.4}/above,$B_2$/\num{0.35}/,$B_3$//below,
  $\overline{A}$/\dots/below,
  $B_1$/$\frac{2}{15}$//above,$B_2$/\dots/,$B_3$/$\frac{1}{3}$//below
}

\ArbreProbasTikz[Type=2x3,InclineProbas=false,EspaceFeuille=1.15]{\ArbreDeuxTrois}

\def\ArbreTroisTrois{
  $A_1$/\num{0.05}/,$B_1$/{1/3}/,$B_2$/{1/3}/,$B_3$/{1/3}/,
  $A_2$/\num{0.80}/,$B_1$/{1/3}/,$B_2$/{1/3}/,$B_3$/{1/3}/,
  $A_3$/\num{0.15}/,$B_1$/{1/3}/,$B_2$/{1/3}/,$B_3$/{1/3}/
}

\ArbreProbasTikz[Type=3x3,StyleTrait={densely
dashed},EspaceFeuille=0.7,PoliceProbas=\scriptsize,Police=\small]{\ArbreTroisTrois}

```

Sortie \LaTeX 

45 Petits schémas pour des probabilités continues

45.1 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!

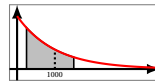
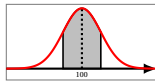


Code \LaTeX

```
\LoiNormaleGraphe[options]<options tikz>\{m\}{s\}{a\}{b\}  
\LoiExpoGraphe[options]<options tikz>\{l\}{a\}{b\}
```



Sortie \LaTeX



Les probabilités *illustrables* sont donc des probabilités Cumulées ($P(a \leq X \leq b)$).

On peut utiliser `\mu` comme borne (a ou b), pour les probabilités du type $P(X \leq b)$ et $P(X \geq a)$.

45.2 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 et sortie \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 :





Code \LaTeX et sortie \LaTeX

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 :

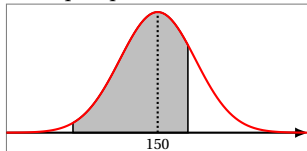
```
\LoiExpoGraphe
[AfficheM=false,CouleurCourbe=blue,CouleurAire=cyan]<baseline=Opt>{0.05}{*}{32}
```

```
\medskip
```

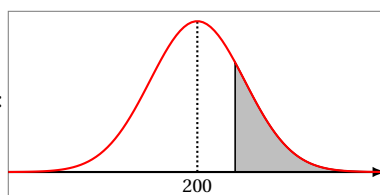
```
\LoiExpoGraphe[Largeur=4,Hauteur=2]{0.00025}{5000}{*}
```



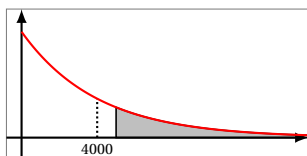
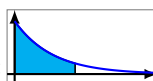
Avec quelques modifications :



Avec centrage vertical :



Avec centrage vertical sur l'axe des abscisses :



45.3 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...

46 Nombres aléatoires

46.1 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 \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 \LaTeX et sortie \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 : 49
Entier à partir du précédent : 52
Décimal entre 0 et 10,999... : 9.886
Liste par défaut (6 entre 1 et 50) : 6,5,26,23,45,15



Les listes créées sont exploitables, *a posteriori*, par le package `listofitems` par exemple!



Code \LaTeX et sortie \LaTeX

```
Liste générée : \TirageAleatoireEntiers{\TestListeA}\TestListeA

Liste traitée : \readlist*\LISTEa{\TestListeA}\showitems{\LISTEa}
```



Liste générée : 40,35,43,4,20,16
Liste traitée :

40	35	43	4	20	16
----	----	----	---	----	----

46.2 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/décroissant>` pour trier les valeurs; défaut `<non>`
- le booléen `<Repetition>` pour autoriser la répétition d'éléments. défaut `<false>`



Code \LaTeX et sortie \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=décroissant]%
{\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 : 52,67,17,91,74,66,46,27,64,15,20,56,13,76,49

Une liste de 12 valeurs (différentes), entre 1 et 50, ordre croissant :
Liste : 1,11,12,15,18,29,31,35,37,41,45,50

Une liste de 12 valeurs (différentes), entre 1 et 50, ordre décroissant :
Liste : 45,44,38,37,34,33,31,25,18,8,3,1

15 tirages de dé à 6 faces :
5,1,4,2,1,1,2,1,1,4,2,6,6,6,1



Code \LaTeX et sortie \LaTeX

```
Une liste (10) pour le Keno\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©, ordonnée, et séparée par des - :

7 – 11 – 12 – 29 – 48 – 53 – 59 – 60 – 68 – 69

7	11	12	29	48	53	59	60	68	69
---	----	----	----	----	----	----	----	----	----

47 Combinatoire

47.1 Idée



L'idée est de proposer une commande pour calculer un arrangement ou une combinaison, en utilisant les capacités de calcul du package `\xint` (`\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{...}
```

47.2 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 et sortie \LaTeX

On a $A_{20}^3 = \text{\texttt{\textbackslash Arrangement}\{3\}\{20\}}$ en non formaté,
et $\text{\texttt{\textbackslash Arrangement}\{3\}\{20\}}$ en formaté avec la notation au début.



On a $A_{20}^3 = 6840$ en non formaté, et $A_{20}^3 = 6840$ en formaté avec la notation au début.



Code \LaTeX et sortie \LaTeX

On a $\text{\texttt{\textbackslash displaystyle}\binom{20}{3}} = \text{\texttt{\textbackslash Combinaison}\{3\}\{20\}}$ en non formaté,~
et $\text{\texttt{\textbackslash displaystyle}\binom{20}{3}}$ 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 $\binom{20}{3} = 1140$ en formaté avec la notation au début.
Et $\binom{20}{3} + \binom{20}{4} = 5985$.



Code \LaTeX et sortie \LaTeX

On a $\text{\texttt{\textbackslash displaystyle}\Arrangement}\{3\}\{20\}$.



On a $A_{20}^3 = \frac{20!}{17!} = 6840$.



Code \LaTeX et sortie \LaTeX

On a $\text{\texttt{\textbackslash displaystyle}\Combinaison}\{3\}\{20\}$. *%ancienne notation FR*



On a $C_{20}^3 = \frac{20!}{3! \times 17!} = 1140$.

48 Fonction de répartition

48.1 Idée



2.7.0 L'idée est de proposer une commande (en accord avec les commandes de repérage, page 40) 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}
```

48.2 Utilisation



Le premier argument, optionnel et entre $\text{\LaTeX} [\dots]$ propose les clés suivantes :

- la clé $\langle \text{Couleur} \rangle$ pour la couleur du tracé; défaut $\langle \text{red} \rangle$
- la clé $\langle \text{Epaisseur} \rangle$ pour gérer l'épaisseur des tracés (en raccourci TikZ); défaut $\langle \text{thick} \rangle$
- le booléen $\langle \text{Pointillés} \rangle$ pour afficher les pointillés horizontaux; défaut $\langle \text{true} \rangle$
- la clé $\langle \text{Extremite} \rangle$ parmi $\langle \text{crochet}/\text{point} \rangle$ pour gérer les extrémités des segments. défaut $\langle \text{crochet} \rangle$

L'argument obligatoire et entre $\text{\LaTeX} \{ \dots \}$ permet de spécifier la liste des probas-intervalles :

- avec $\text{\LaTeX} *$ pour remplacer ∞ ;
- sous la forme $\text{\LaTeX} \text{proba,borneinf,bornesup} / \text{proba,borneinf,bornesup} / \dots$.

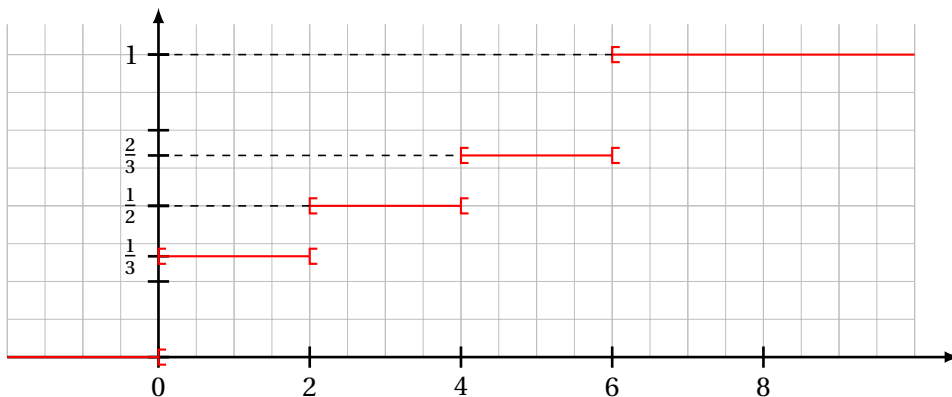


Le code remplace $\text{\LaTeX} *$ par les valeurs stockées dans $\text{\LaTeX} \text{\xmin}$ ou $\text{\LaTeX} \text{\xmax}$, d'où l'intérêt d'utiliser la commande en partenariat des commandes de repérage de $\text{\LaTeX} \text{Proflycee}$.



Code \LaTeX et sortie \LaTeX

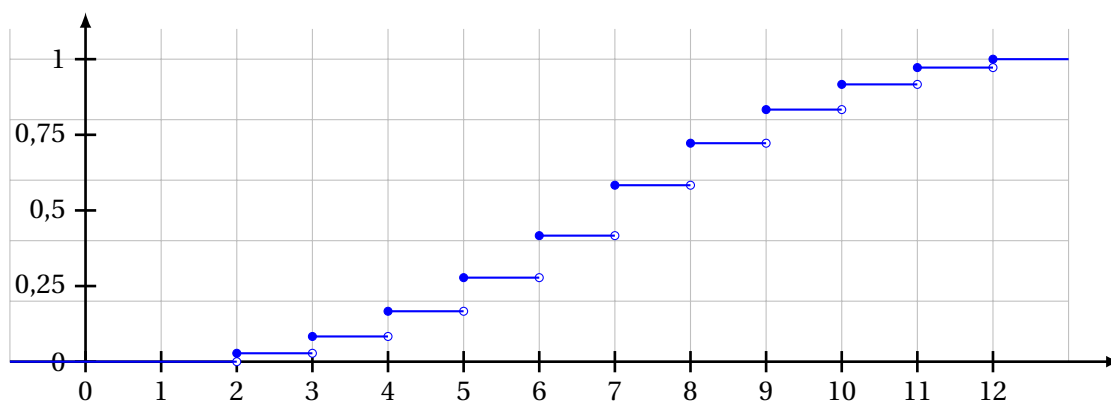
```
\begin{tikzpicture}[y=4cm,xmin=-2,xmax=10,ymin=0,ymax=1.1,
  xgrille=1,xgrilles=0.5,ygrille=0.5,ygrilles=0.125]
  \GrilleTikz %grille
  \AxesTikz %axes
  \AxexTikz{0,2,4,6,8} %graduations de (Ox)
  \AxeYtikz[AffGrad=false]{0,0.25,\dots,1} %graduations de (Oy) sans valeurs
  \AxeYtikz[Frac]{1/3,1/2,2/3,1} %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}
```





Code \LaTeX et sortie \LaTeX

```
\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}
```



Thème

OUTILS POUR L'ARITHMÉTIQUE

Outils pour l'arithmétique

49 Division euclidienne


49.1 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.



Le package  `xinttools` est la base de cette macro. C'est lui qui s'occupe de la partie *calculs*.



 Code \LaTeX

```
\DivEucl(*)[clés]{a}{b}
```



 Code \LaTeX et sortie \LaTeX

```
\DivEucl{145}{7}\par
\DivEucl*{145}{7}\par
\DivEucl*[Quotient=false]{-145}{7}\par
\DivEucl[Reste=false]{145}{-7}\par
\DivEucl[Vide]{-145}{-7}
```




```
145 = 7 × 20 + 5
145 = 7 × 20 + 5 avec 0 ≤ 5 < 7
-145 = 7 × ... + 2 avec 0 ≤ 2 < 7
145 = -7 × (-20) + ...
-145 = -7 × ... + ...
```

49.2 Clés et options



En ce qui concerne la commande pour la division euclidienne :

- 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  `...` ne sont pas nécessaires.



 Code \LaTeX et sortie \LaTeX

```
\DivEucl*{123456789}{8547}\par
\DivEucl[Vide,Pointilles={\makebox[1cm]{\dotfill}}]{123456789}{8547}
```



```
123456789 = 8547 × 14444 + 3921 avec 0 ≤ 3921 < 8547
123456789 = 8547 × ..... + .....
```

50 Conversions binaire/hexadécimal/décimal

50.1 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 \LaTeX

```
\xintDecToHex{100}  
\xintDecToBin{51}  
\xintHexToDec{A4C}  
\xintBinToDec{110011}  
\xintBinToHex{11111111}  
\xintHexToBin{ACDC}  
\xintCHexToBin{3F}
```



Sortie \LaTeX

```
64  
110011  
2636  
51  
FF  
1010110011011100  
00111111
```

50.2 Conversion décimal vers binaire



Code \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 \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  
$\ConversionDecBin*[AffBase=false]{415}$
```



Sortie \LaTeX

```

41510 = 0001 1001 11112
41510 = 1 1001 11112
415 = 0001 1001 1111
415 = 1 1001 1111

```

50.3 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
 $\backslash\text{ConversionBinHex}\{110011111\}\$$ 
%conversion sans détails et affichage de la base
 $\backslash\text{ConversionBinHex}[\text{Details}=\text{false}]\{110011111\}\$$ 
%conversion sans détails et sans affichage de la base
 $\backslash\text{ConversionBinHex}[\text{AffBase}=\text{false},\text{Details}=\text{false}]\{110011111\}\$$ 

```



Sortie \LaTeX

```

1 1001 11112 = 0001 1001 1111 =  $\frac{0001}{1}\frac{1001}{9}\frac{1111}{F}$  = 19F16
1 1001 11112 = 19F16
1 1001 1111 = 19F

```


50.4 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 \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 \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
$\ConversionHexBin[AffBase=false,Details=false]{ACDC}$
```



Sortie \LaTeX

```
ACDC16 =  $\overline{1010}\overline{1100}\overline{1101}\overline{1100}_2$ 
          A      C      D      C
ACDC16 = 1010 1100 1101 11002
ACDC = 1010 1100 1101 1100
```

50.5 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 \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 L^AT_EX

```
%conversion 16->10 avec détails et affichage de la base et zéros
$\ConversionVersDec[BaseDep=16]{19F}$
%conversion 2->10 avec détails et affichage de la base et zéros
$\ConversionVersDec{110011}$
%conversion 2->10 avec détails et affichage de la base et sans zéros
$\ConversionVersDec[Zeros=false]{110011}$
%conversion 16->10 sans détails et affichage de la base et avec zéros
$\ConversionVersDec[BaseDep=16,Details=false]{AC0DC}$
%conversion 16->10 avec détails et sans affichage de la base et sans zéros
$\ConversionVersDec[Zeros=false,BaseDep=16]{AC0DC}$
```



Sortie L^AT_EX

$19F_{16} = 1 \times 16^2 + 9 \times 16^1 + 15 \times 16^0 = 415_{10}$
 $110011_2 = 1 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 51_{10}$
 $110011_2 = 1 \times 2^5 + 1 \times 2^4 + 1 \times 2^1 + 1 \times 2^0 = 51_{10}$
 $AC0DC_{16} = 704732_{10}$
 $AC0DC_{16} = 10 \times 16^4 + 12 \times 16^3 + 13 \times 16^1 + 12 \times 16^0 = 704732_{10}$

51 Conversion « présentée » d'un nombre en base décimale

51.1 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.

La commande utilise – par défaut – du code TikZ en mode `\tikzoverlay`, donc on pourra déclarer – si ce n'est pas fait – dans le préambule, la commande qui suit.



</> Code \LaTeX

```
...
\tikzstyle{every picture}+=[remember picture]
...
```

51.2 Code et clés



Code \LaTeX et sortie \LaTeX

```
%conversion basique
\ConversionDepuisBaseDix{78}{2}
```



$$\left\{ \begin{array}{lcl} 78 = 2 \times 39 + 0 \\ 39 = 2 \times 19 + 1 \\ 19 = 2 \times 9 + 1 \\ 9 = 2 \times 4 + 1 \\ 4 = 2 \times 2 + 0 \\ 2 = 2 \times 1 + 0 \\ 1 = 2 \times 0 + 1 \end{array} \right. \Rightarrow 78_{10} = 1001110_2$$



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 L^AT_EX

```

%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}

```



Sortie L^AT_EX

$$\left\{ \begin{array}{l} 45 = 2 \times 22 + 1 \\ 22 = 2 \times 11 + 0 \\ 11 = 2 \times 5 + 1 \\ 5 = 2 \times 2 + 1 \\ 2 = 2 \times 1 + 0 \\ 1 = 2 \times 0 + 1 \end{array} \right. \Rightarrow 45_{10} = 101101_2$$

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

52 Algorithme d'Euclide pour le PGCD

52.1 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 et sortie \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 géré par du code TikZ, en mode `overlay`, donc il faut bien penser à déclarer dans le préambule : `\tikzstyle{every picture}+=[remember picture]`

52.2 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 et sortie \LaTeX

```
\PresentationPGCD[CouleurResultat]{150}{27}
```



$$\left\{ \begin{array}{l} 150 = 27 \times 5 + 15 \\ 27 = 15 \times 1 + 12 \\ 15 = 12 \times 1 + 3 \\ 12 = 3 \times 4 + 0 \end{array} \right. \Rightarrow \text{PGCD}(150; 27) = 3$$



Code \LaTeX et sortie \LaTeX

```
\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,AfficheDelimitateurs=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}
```



$$\left\{ \begin{array}{l} 1250 = 450 \times 2 + 350 \\ 450 = 350 \times 1 + 100 \\ 350 = 100 \times 3 + 50 \\ 100 = 50 \times 2 + 0 \end{array} \right. \Rightarrow \text{PGCD}(1250; 450) = 50.$$

$$\left\{ \begin{array}{l} 13500 = 2500 \times 5 + 1000 \\ 2500 = 1000 \times 2 + 500 \\ 1000 = 500 \times 2 + 0 \end{array} \right. \Rightarrow \text{PGCD}(13500; 2500) = 500.$$

$$\left\{ \begin{array}{l} 420 = 540 \times 0 + 420 \\ 540 = 420 \times 1 + 120 \\ 420 = 120 \times 3 + 60 \\ 120 = 60 \times 2 + 0 \end{array} \right. \Rightarrow \text{PGCD}(420; 540) = 60.$$

D'après l'algorithme d'Euclide, on a

$$\begin{array}{rcl} 123456789 & = & 9876 \times 12500 + 6789 \\ 9876 & = & 6789 \times 1 + 3087 \\ 6789 & = & 3087 \times 2 + 615 \\ 3087 & = & 615 \times 5 + 12 \\ 615 & = & 12 \times 51 + 3 \\ 12 & = & 3 \times 4 + 0 \end{array}$$

dernier reste non nul

52.3 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 `\AfficheDelimitateurs=false` permet de choisir des délimiteurs différents.



Code \LaTeX et sortie \LaTeX

```
$\left[ \right] \PresentationPGCD[AfficheConclusion=false,AfficheDelimitateurs=false]{1234}{5} \right]$
```



$$\left[\begin{array}{l} 1234 = 5 \times 246 + 4 \\ 5 = 4 \times 1 + 1 \\ 4 = 1 \times 4 + 0 \end{array} \right]$$

53 Résolution d'une équation diophantienne

53.1 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}
```

53.2 Options et clés



Concernant les Clés disponibles pour cette commande, à donner entre `[...]` :

- | | |
|--|-----------------------------|
| — la clé <Lettre> pour spécifier le <i>nom</i> 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 `{...}` est quant à lui l'équation, en langage « naturel » du type `ax+by=c` (le *code* se charge d'extraire les coefficients, donc pas besoin des signes *).



Code \LaTeX et sortie \LaTeX

```
\EquationDiophantienne{48x+18y=3}
```



On cherche à résoudre l'équation diophantienne :

$$48x + 18y = 3 \quad (\text{E})$$

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 ne divise pas 3, donc l'équation (E) n'admet aucune solution.



Code \LaTeX et sortie \LaTeX

```
\EquationDiophantienne[PresPGCD=false]{48x+18y=-5}
```



On cherche à résoudre l'équation diophantienne :

$$48x + 18y = -5 \quad (\text{E})$$

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.


$$3x + 4y = 1 \quad (\text{E})$$

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) :

$$3 \times (-1) + 4 \times 1 = 1 \quad (E_0)$$

$$\begin{array}{r} 3 \times x + 4 \times y = 1 \\ - \quad 3 \times (-1) + 4 \times 1 = 1 \\ \hline 3 \times (x+1) + 4 \times (y-1) = 0 \end{array}$$

En remplaçant, on obtient :

$$\begin{aligned} 3 \times (x+1) &= -4 \times (y-1) \implies 3 \times (x+1) = -4 \times (\underbrace{1+3k-1}_y) \\ &\implies 3 \times (x+1) = -4 \times (3k) \\ &\implies x+1 = -4k \\ &\implies x = -1-4k \end{aligned}$$

Réciproquement, soit k un entier quelconque :

$$\begin{aligned} 3 \times (-1 - 4k) + 4 \times (1 + 3k) &= 3 \times (-1) + \cancel{3 \times (-4)k} + 4 \times 1 + \cancel{4 \times 3k} \\ &= \underbrace{3 \times (-1) + 4 \times 1}_{=1 \text{ d'après } (E_0)} \\ &= 1 \end{aligned}$$

En conclusion, les solutions de (E) sont donc les couples $(-1 - 4k; 1 + 3k)$, avec k un entier relatif.



```
\EquationDiophantienne[Cadres,Inconnues=u/v,Entier=1]{48u+18v=12}
```



On cherche à résoudre l'équation diophantienne :

$$48u + 18v = 12$$

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 divise 12, donc on peut simplifier l'équation diophantienne par 6.

$$48u + 18v = 12 \xrightarrow{+6} 8u + 3v = 2 \quad (\text{E})$$

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) :

$$8 \times (-1) + 3 \times 3 = 1 \xrightarrow{\times 2} 8 \times (-2) + 3 \times 6 = 2 \quad (\text{E}_0)$$

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 $\boxed{v = 6 + 8l}$.

En remplaçant, on obtient :

$$\begin{aligned} 8 \times (u+2) &= -3 \times (v-6) \Rightarrow 8 \times (u+2) = -3 \times \underbrace{(6+8l-6)}_v \\ &\Rightarrow 8 \times (u+2) = -3 \times (8l) \\ &\Rightarrow u+2 = -3l \\ &\Rightarrow \boxed{u = -2 - 3l} \end{aligned}$$

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 :

$$\begin{aligned} 8 \times (-2 - 3l) + 3 \times (6 + 8l) &= 8 \times (-2) + \cancel{8 \times (-3)l} + 3 \times 6 + \cancel{3 \times 8l} \\ &= \underbrace{8 \times (-2) + 3 \times 6}_{= 2 \text{ d'après } (\text{E}_0)} \\ &= 2 \end{aligned}$$

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 :

$$47x + (-18)y = 1 \quad (\text{E})$$

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) :

$$47 \times 5 + (-18) \times 13 = 1 \quad (\text{E}_0)$$

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 $y = 13 + 47k$.

En remplaçant, on obtient :

$$\begin{aligned} 47 \times (x-5) &= 18 \times (y-13) \Rightarrow 47 \times (x-5) = 18 \times (\underbrace{13+47k}_{y}-13) \\ &\Rightarrow 47 \times (x-5) = 18 \times (47k) \\ &\Rightarrow x-5 = 18k \\ &\Rightarrow x = 5 + 18k \end{aligned}$$

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 :

$$\begin{aligned} 47 \times (5 + 18k) + (-18) \times (13 + 47k) &= 47 \times 5 + \cancel{47 \times 18k} + (-18) \times 13 + \cancel{(-18) \times 47k} \\ &= \underbrace{47 \times 5 + (-18) \times 13}_{= 1 \text{ d'après } (\text{E}_0)} \\ &= 1 \end{aligned}$$

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.

54 Diviseurs

54.1 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 \LaTeX

```
\ListeDiviseurs(*) [Clé] {nombre}
```



Code \LaTeX

```
\ArbreDiviseurs [Clés] {nombre}
```

54.2 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 $\text{\texttt{\mathscr}}$, $\text{\texttt{\mathcal}}$ sinon;
- le booléen $\langle \text{\texttt{AffNom}} \rangle$ permet d'afficher le nom de l'ensemble; défaut $\langle \text{\texttt{true}} \rangle$
- l'argument obligatoire, et entre $\text{\texttt{\{...}}}$, est quant lui le nombre, y compris en langage $\text{\texttt{xint}}$.

À noter que la commande est incluse dans un bloc $\text{\texttt{\ensuremath}}$, donc les $\text{\texttt{\{...}}}$ ne sont pas nécessaires.



Code \LaTeX et sortie \LaTeX

```
%sortie par défaut
\ListeDiviseurs{150}

%sortie avec \mathscr (si chargé)
\ListeDiviseurs*{300}

%sortie sans libellé
\ListeDiviseurs[AffNom=false]{60}
```



```
 $\mathcal{D}_{150} = \{1; 2; 3; 5; 6; 10; 15; 25; 30; 50; 75; 150\}$ 
 $\mathcal{D}_{300} = \{1; 2; 3; 4; 5; 6; 10; 12; 15; 20; 25; 30; 50; 60; 75; 100; 150; 300\}$ 
 $\{1; 2; 3; 4; 5; 6; 10; 12; 15; 20; 30; 60\}$ 
```



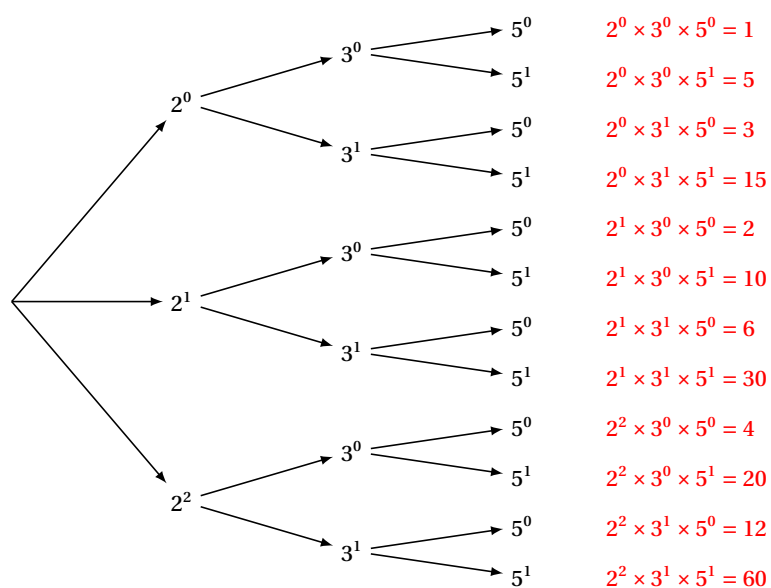
En ce qui concerne la commande pour l'arbre (créé en TikZ) permettant d'obtenir la liste des diviseurs (je remercie Christophe Poulain pour un *bout de code* que j'ai adapté et pour son aide précieuse) :

- les clés disponible sont :
 - la clé $\langle \text{\texttt{EspaceNiveau}} \rangle$ qui est l'espace horizontal (en cm) entre les étages; défaut $\langle \text{\texttt{2.25}} \rangle$
 - la clé $\langle \text{\texttt{EspaceFeuille}} \rangle$ qui est l'espace vertical entre les feuilles; défaut $\langle \text{\texttt{0.66}} \rangle$
 - le booléen $\langle \text{\texttt{Details}} \rangle$ pour afficher les calculs des diviseurs; défaut $\langle \text{\texttt{true}} \rangle$
 - la clé $\langle \text{\texttt{CouleurDetails}} \rangle$ pour la couleur des détails; défaut $\langle \text{\texttt{red}} \rangle$
 - la clé $\langle \text{\texttt{Echelle}} \rangle$ pour spécifier une échelle globale (y compris le texte) de la figure; défaut $\langle \text{\texttt{1}} \rangle$
 - le booléen $\langle \text{\texttt{Fleches}} \rangle$ pour afficher une flèche sur les branches. défaut $\langle \text{\texttt{true}} \rangle$
- l'argument obligatoire, et entre $\text{\texttt{\{...}}}$, est quant lui le nombre, y compris en langage $\text{\texttt{xint}}$.



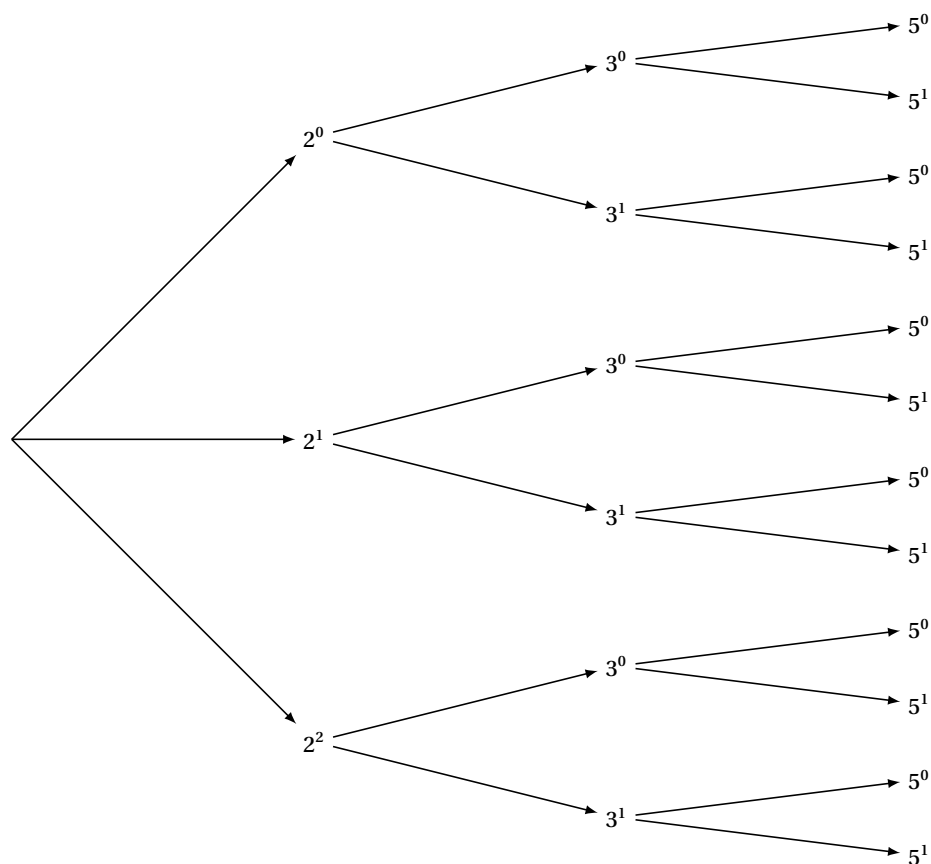
Code \LaTeX et sortie \LaTeX

```
%sortie par défaut  
\ArbreDiviseurs{60}
```



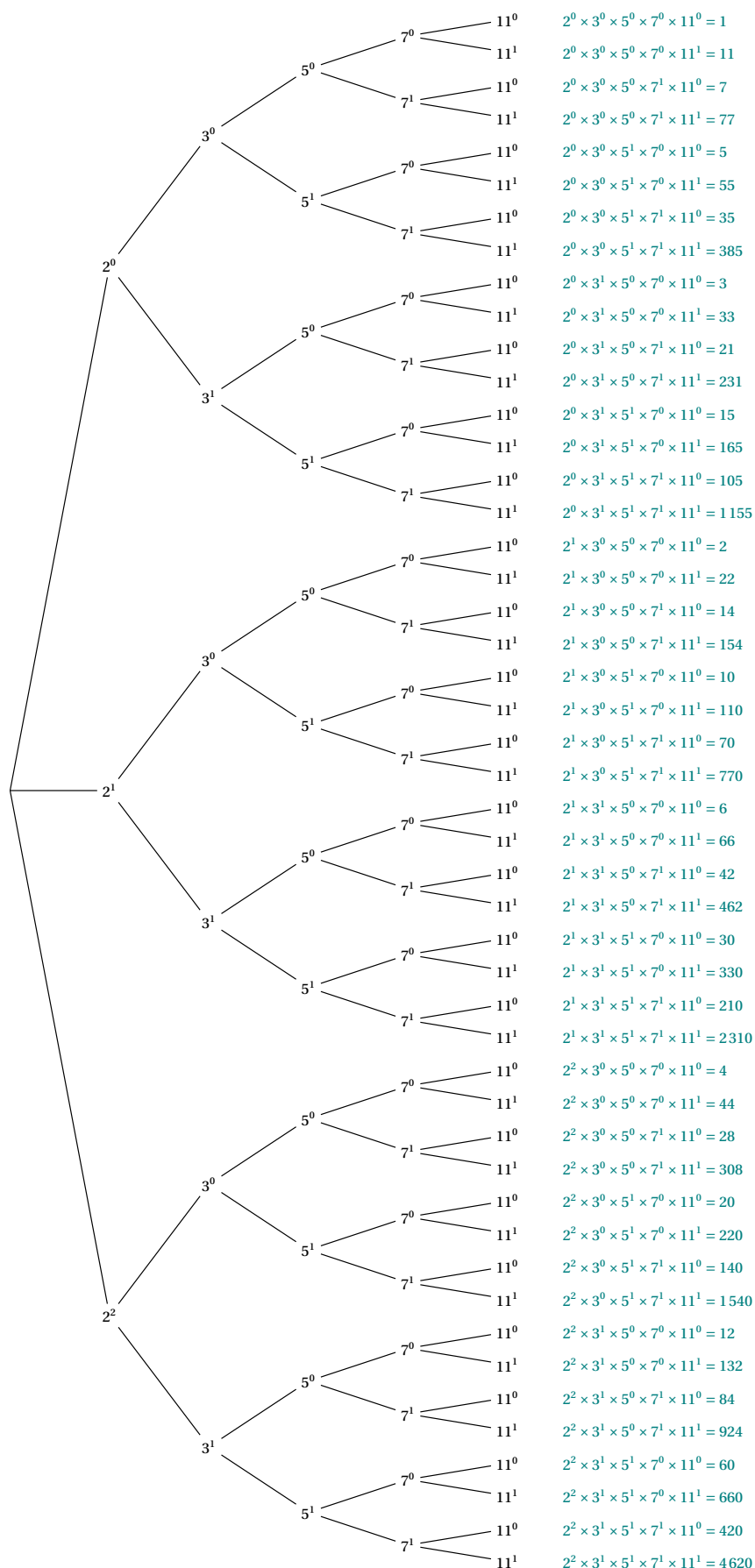
Code \LaTeX et sortie \LaTeX

```
%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}
```



55 Chiffrements

55.1 Idées



 L'idée est de proposer des commandes pour travailler sur des chiffrements/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}
```

55.2 Chiffrement de César



 Code \LaTeX et sortie \LaTeX

```
%Chiffrement de César
\ChiffrementCesar[Decal=4]{ABCDEFGHIJKLMNOPQRSTUVWXYZ}
```




EFGHIJKLMNOPQRSTUVWXYZABCD



Les clés disponibles sont :

- la clé **<Decal>** qui spécifie le décalage à appliquer; défaut **<5>**
- le booléen **<Dechiffr>** pour déchiffrer. défaut **<false>**

L'argument obligatoire, entre  `{...}` est le message, avec espace et/ou majuscules et/ou minuscules.



 Code \LaTeX et sortie \LaTeX

```
%Chiffrement de César, décalage de 5
\ChiffrementCesar{TEXTE A CHIFFRER}
```



YJCYJ F HMNKKWJW



 Code \LaTeX et sortie \LaTeX

```
%Chiffrement de César, décalage de 5
\ChiffrementCesar[Decal=7]{TEXTE A CHIFFRER}
```



ALEAL H JOPMMYLY



 Code \LaTeX et sortie \LaTeX

```
%Déchiffrement de César, décalage de 5
\ChiffrementCesar[Dechiffr]{IJHTIJW HJXFW}
```



DECODER CESAR

55.3 Inverse modulo



Code \LaTeX et sortie \LaTeX

```
%Inverse modulo, version non étoilée, résultat stocké dans \resinvmod  
\InverseModulo{3}{26}\resinvmod
```



9



Code \LaTeX et sortie \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 et sortie \LaTeX

```
\InverseModulo{2}{26}\resinvmod
```



2 n'est pas inversible modulo 26.



Code \LaTeX et sortie \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.

55.4 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 et sortie \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 `` 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 et sortie \LaTeX $\backslash\text{ChiffrementAffine}[a=5,b=17]\{\text{BTSSIO}\}$ 

WIDDFJ

Code \LaTeX et sortie \LaTeX $\backslash\text{ChiffrementAffine}[\text{Dechiffr},a=5,b=17]\{\text{widdfj}\}$ 

btssio

Code \LaTeX et sortie \LaTeX $\backslash\text{ChiffrementAffine}[\text{Dechiffr},a=2,b=13]\{\text{WIDDFJ}\}$ Le message ne peut pas être déchiffré car $\text{PGCD}(2;26) \neq 1!$

55.5 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 et sortie \LaTeX $\backslash\text{ChiffrementHill}[\text{Matrice}=\{3,5,1,2\}]\{\text{ELECTION}\}$ 

PAWITJDO



Les clés disponibles sont :

- la clé $\langle \text{Matrice} \rangle$ qui spécifie les coefficients de la matrice 2×2 défaut $\langle 1,2,3,5 \rangle$
- le booléen $\langle \text{Dechiffr} \rangle$ qui propose le déchiffrement. défaut $\langle \text{false} \rangle$

L'argument obligatoire, entre $\{\dots\}$ est le message, avec espace et/ou majuscules et/ou minuscules.

Code \LaTeX et sortie \LaTeX $\backslash\text{ChiffrementHill}[\text{Matrice}=\{3,5,6,17\}]\{\text{TEXTEACRYPTER}\}$ 

ZAITMYNPRJZAZY

Code \LaTeX et sortie \LaTeX $\backslash\text{ChiffrementHill}[\text{Matrice}=\{3,5,1,2\},\text{Dechiffr}]\{\text{pawitjdo}\}$ 

election

Code \LaTeX et sortie \LaTeX $\backslash\text{ChiffrementHill}[\text{Matrice}=\{1,2,3,4\},\text{Dechiffr}]\{\text{PAWITJDO}\}$ 

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 et sortie \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!

Thème

ÉCRITURES, SIMPLIFICATIONS

Écritures, simplifications

56 Simplification sous forme d'une fractions

56.1 Idée



L'idée est d'obtenir une commande pour *simplifier* un calcul sous forme de fraction irréductible.



Code \LaTeX

```
\ConversionFraction(*)[option de formatage]{calcul}
```

56.2 Commande et options



Quelques explications sur cette commande :

- **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 $\texttt{\ensuremath}$ donc les $\texttt{\$...\$}$ 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}
```



Sortie \LaTeX

```
- 485
 48
-485
 48
- 485
 48
-485/48
-10,1042
15
 37
 715
37
```



Code \LaTeX et sortie \LaTeX

```

 $\frac{111}{2145}=\frac{111}{2145}$  \\

 $\frac{3}{15}=\frac{3}{15}$  \\

 $\tfrac{3}{15}=\frac{3}{15}$  \\

 $\dfrac{3}{15}=\frac{3}{15}$  \\

 $\dfrac{0,42}{0,015}=\frac{0.42}{0.015}$  \\

 $\dfrac{0,41}{0,015}=\frac{0.41}{0.015}$  \\

 $\dfrac{1}{7}-\dfrac{3}{8}=\frac{1}{7}-\frac{3}{8}$  \\

 $\frac{1+1/2}{2}$  \\

 $\frac{0.1}{0.7+30/80}$ 

```



$$\frac{111}{2145} = \frac{37}{715}$$

$$\frac{3}{15} = \frac{1}{5}$$

$$\frac{3}{15} = \frac{1}{5}$$

$$\frac{3}{15} = \frac{1}{5}$$

$$\frac{0,42}{0,015} = 28$$

$$\frac{0,41}{0,015} = \frac{82}{3}$$

$$\frac{1}{7} - \frac{3}{8} = -\frac{13}{56}$$

$$\frac{3}{2}$$

$$\frac{29}{56}$$



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* !

57 Écriture d'un trinôme, trinôme aléatoire

57.1 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 et sortie \LaTeX

```
\EcritureTrinome{1}{7}{0}\  
\EcritureTrinome{1.5}{7.3}{2.56}\  
\EcritureTrinome{-1}{0}{12}\  
\EcritureTrinome{-1}{-5}{0}
```



```
 $x^2 + 7x$   
 $1,5x^2 + 7,3x + 2,56$   
 $-x^2 + 12$   
 $-x^2 - 5x$ 
```

57.2 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 et sortie \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 :  
 $f(x) = 3x^2 + 5x - 5$   
 $g(x) = -4x^2 + x - 5$   
 $h(x) = -5x^2 - 3x - 9$   
Avec  $a$  entre 1 et 10 (forcément positif) puis  $b$  entre  $-2$  et 2 puis  $c$  entre 0 et 4 :  
 $8x^2 - x + 2$   
 $5x^2 + x$   
 $3x^2 + x + 1$ 
```

58 Simplification de racines

58.1 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 et sortie \LaTeX

```
\SimplificationRacine{48} \ \ \SimplificationRacine{100/34} \ \ \
\SimplificationRacine{99999} \ \ \SimplificationRacine{1500*0.31*(1-0.31)} \ \ \
```



$$4\sqrt{3}$$

$$\frac{5\sqrt{34}}{17}$$

$$3\sqrt{\frac{11111}{10}}$$

$$\frac{3\sqrt{3565}}{10}$$


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!).

58.2 Exemples



Code \LaTeX et sortie \LaTeX

```
%Simplification d'un module de complexe
$\left| 4+6i\right| \text{\texttt{i}}\right| = \sqrt{4^2+6^2} =
\sqrt{\xinteval{4**2+6**2}}=\SimplificationRacine{4**2+6**2}$

%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
$\sqrt{\num{150}\times\num{0.35}\times(1-\num{0.35})} =
\displaystyle\SimplificationRacine{150*0.35*(1-0.35)}$
```



$$|4 + 6i| = \sqrt{4^2 + 6^2} = \sqrt{52} = 2\sqrt{13}$$

$$\frac{1}{\sqrt{6}} = \left(\sqrt{\frac{1}{6}}\right) = \frac{\sqrt{6}}{6}$$

$$\frac{42}{\sqrt{5}} = \left(\sqrt{\frac{42^2}{5}}\right) = \frac{42\sqrt{5}}{5}$$

$$\sqrt{150 \times 0,35 \times (1 - 0,35)} = \frac{\sqrt{546}}{4}$$

59 Mesure principale d'un angle

59.1 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 \LaTeX

```
\MesurePrincipale[booléens]{angle} %dans un mode mathématique
```



La commande est à insérer dans un environnement mathématique, via $\text{\$...\$}$ ou \[...\] .
L'angle est donné sous forme *explicite* avec la chaîne \pi .

59.2 Exemples



Pour cette commande :

- le booléen $\langle \text{d} \rangle$ permet de forcer l'affichage en \displaystyle ; défaut $\langle \text{false} \rangle$
- le booléen $\langle \text{Crochets} \rangle$ permet d'afficher le *modulo* entre crochets (sinon parenthèses); défaut $\langle \text{false} \rangle$
- **2.6.0** le booléen $\langle \text{Brut} \rangle$ pour afficher uniquement la mesure principale; défaut $\langle \text{false} \rangle$
- l'argument *obligatoire* est en écriture *en ligne*.



</> Code \LaTeX

```
\$ \MesurePrincipale[d]{54pi/7}$  
\$ \MesurePrincipale[d]{-128pi/15}$  
\$ \MesurePrincipale{3pi/2}$  
\$ \MesurePrincipale[Crochets]{5pi/2}$  
\$ \MesurePrincipale{-13pi}$  
\$ \MesurePrincipale{28pi}$  
\$ \MesurePrincipale[d]{14pi/4}$  
\$ \MesurePrincipale[Crochets]{14pi/7}$  
\$ \dfrac{121\pi}{12} = \MesurePrincipale[Brut]{121pi/12}$ à $2\pi$ près
```



Sortie \LaTeX

```
54π  = -2π  (2π)  
 7    =  7  
-128π  = -8π  (2π)  
 15    = 15  
3π    = -π  (2π)  
 2    =  2  
5π    = π  [2π]  
 2    =  2  
-13π = π (2π)  
28π = 0 (2π)  
14π  = -π  (2π)  
 4    =  2  
14π  = 0 [2π]  
 7  
121π  = π  à 2π près  
 12    = 12
```

60 Lignes trigonométriques

60.1 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)
```

60.2 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 et sortie \LaTeX

```
\$ \LigneTrigo{cos}(56\pi/3)$ et \$ \LigneTrigo{sin}(56\pi/3)$ et \$ \LigneTrigo{tan}(56\pi/3)$
```



$-\frac{1}{2}$ et $\frac{\sqrt{3}}{2}$ et $-\sqrt{3}$



Code \LaTeX et sortie \LaTeX

```
\$ \LigneTrigo[d,Etapes]{cos}(56\pi/3)$ et \$ \LigneTrigo[d,Etapes]{sin}(56\pi/3)$
```



$\cos\left(\frac{56\pi}{3}\right) = \cos\left(\frac{2\pi}{3}\right) = -\frac{1}{2}$ et $\sin\left(\frac{56\pi}{3}\right) = \sin\left(\frac{2\pi}{3}\right) = \frac{\sqrt{3}}{2}$



Code \LaTeX et sortie \LaTeX

```
\$ \LigneTrigo*[d,Etapes]{cos}(2\pi/3)$ et \$ \LigneTrigo*[d,Etapes]{sin}(2\pi/3)$
```



$\cos\left(\frac{2\pi}{3}\right) = -\frac{1}{2}$ et $\sin\left(\frac{2\pi}{3}\right) = \frac{\sqrt{3}}{2}$



Code \LaTeX et sortie \LaTeX

```
\$ \LigneTrigo[d,Etapes]{cos}(146\pi)$ et \$ \LigneTrigo[d,Etapes]{sin}(146\pi)$
```



$\cos(146\pi) = \cos(0) = 1$ et $\sin(146\pi) = \sin(0) = 0$



Code \LaTeX et sortie \LaTeX

```
\$ \LigneTrigo[d,Etapes]{cos}(-551\pi/12)$ et \$ \LigneTrigo[d,Etapes]{sin}(-551\pi/12)$
```



$\cos\left(\frac{-551\pi}{12}\right) = \cos\left(\frac{\pi}{12}\right) = \frac{\sqrt{6} + \sqrt{2}}{4}$ et $\sin\left(\frac{-551\pi}{12}\right) = \sin\left(\frac{\pi}{12}\right) = \frac{\sqrt{6} - \sqrt{2}}{4}$



Code \LaTeX et sortie \LaTeX

$\backslash\text{LigneTrigo}[d,\text{Etapes}]\{\text{cos}\}(447\pi/8)\$$ et $\backslash\text{LigneTrigo}[d,\text{Etapes}]\{\text{sin}\}(447\pi/8)\$$



$$\cos\left(\frac{447\pi}{8}\right) = \cos\left(\frac{-\pi}{8}\right) = \frac{\sqrt{2+\sqrt{2}}}{2} \text{ et } \sin\left(\frac{447\pi}{8}\right) = \sin\left(\frac{-\pi}{8}\right) = -\frac{\sqrt{2-\sqrt{2}}}{2}$$



Code \LaTeX et sortie \LaTeX

$\backslash\text{LigneTrigo}[d,\text{Etapes}]\{\text{cos}\}(-\pi/8)\$$ et $\backslash\text{LigneTrigo}[d,\text{Etapes}]\{\text{sin}\}(-\pi/8)\$$



$$\cos\left(\frac{-\pi}{8}\right) = \frac{\sqrt{2+\sqrt{2}}}{2} \text{ et } \sin\left(\frac{-\pi}{8}\right) = -\frac{\sqrt{2-\sqrt{2}}}{2}$$



Code \LaTeX et sortie \LaTeX

$\backslash\text{LigneTrigo}[d,\text{Etapes}]\{\text{cos}\}(-595\pi/12)\$$ et $\backslash\text{LigneTrigo}[d,\text{Etapes}]\{\text{sin}\}(-595\pi/12)\$$ et $\backslash\text{LigneTrigo}[d,\text{Etapes}]\{\text{tan}\}(-595\pi/12)\$$



$$\cos\left(\frac{-595\pi}{12}\right) = \cos\left(\frac{5\pi}{12}\right) = \frac{\sqrt{6}-\sqrt{2}}{4} \text{ et } \sin\left(\frac{-595\pi}{12}\right) = \sin\left(\frac{5\pi}{12}\right) = \frac{\sqrt{6}+\sqrt{2}}{4} \text{ et } \tan\left(\frac{-595\pi}{12}\right) = \tan\left(\frac{5\pi}{12}\right) = 2+\sqrt{3}$$



Code \LaTeX et sortie \LaTeX

$\backslash\text{LigneTrigo}[d,\text{Etapes}]\{\text{cos}\}(33\pi/10)\$$ et $\backslash\text{LigneTrigo}[d,\text{Etapes}]\{\text{sin}\}(33\pi/10)\$$ et $\backslash\text{LigneTrigo}[d,\text{Etapes}]\{\text{tan}\}(33\pi/10)\$$



$$\cos\left(\frac{33\pi}{10}\right) = \cos\left(\frac{-7\pi}{10}\right) = -\frac{\sqrt{10-2\sqrt{5}}}{4} \text{ et } \sin\left(\frac{33\pi}{10}\right) = \sin\left(\frac{-7\pi}{10}\right) = -\frac{1+\sqrt{5}}{4}$$

$$\tan\left(\frac{33\pi}{10}\right) = \tan\left(\frac{-7\pi}{10}\right) = \frac{\sqrt{25+10\sqrt{5}}}{5}$$



Code \LaTeX et sortie \LaTeX

$\backslash\text{LigneTrigo}[d,\text{Etapes}]\{\text{cos}\}(-14\pi/5)\$$ et $\backslash\text{LigneTrigo}[d,\text{Etapes}]\{\text{sin}\}(-14\pi/5)\$$ et $\backslash\text{LigneTrigo}[d,\text{Etapes}]\{\text{tan}\}(-14\pi/5)\$$



$$\cos\left(\frac{-14\pi}{5}\right) = \cos\left(\frac{-4\pi}{5}\right) = \frac{-1-\sqrt{5}}{4} \text{ et } \sin\left(\frac{-14\pi}{5}\right) = \sin\left(\frac{-4\pi}{5}\right) = -\frac{\sqrt{10-2\sqrt{5}}}{4}$$

$$\tan\left(\frac{-14\pi}{5}\right) = \tan\left(\frac{-4\pi}{5}\right) = \sqrt{5-2\sqrt{5}}$$

60.3 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	$\frac{\sqrt{25-10\sqrt{5}}}{5}$	$\frac{\sqrt{25+10\sqrt{5}}}{5}$	$-\frac{\sqrt{25+10\sqrt{5}}}{5}$	$-\frac{\sqrt{25-10\sqrt{5}}}{5}$	$\frac{\sqrt{25-10\sqrt{5}}}{5}$	$\frac{\sqrt{25+10\sqrt{5}}}{5}$	$-\frac{\sqrt{25+10\sqrt{5}}}{5}$	$-\frac{\sqrt{25-10\sqrt{5}}}{5}$

61 Écriture sous forme de fraction irréductible d'un décimal périodique

61.1 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.BC̄», 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}
```

61.2 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`; défaut **<true>**
- la clé **<Inconnue>** qui gère l'inconnue dans la rédaction; défaut **<x>**
- le booléen **<Solution>** qui affiche la résolution *complète*; défaut **<false>**
- le booléen **<Simple>** qui affiche uniquement la *conclusion*; défaut **<false>**
- le booléen **<Enonce>** (en mode **<Simple=true>**) qui affiche le nombre de départ (formaté). défaut **<false>**

À noter que la commande gère automatiquement le mode mathématique éventuel.



⚙️ Code \LaTeX et sortie \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  $\$ \text{\num{45.1}} \overline{\text{\num{23}}} = \dfrac{\text{\num{\FracPerNum}}}{\text{\num{\FracPerDenom}}} =$ 
 $\dfrac{\text{\num{\FracPerNumSimpl}}}{\text{\num{\FracPerDenomSimpl}}} =$ 
 $\text{\nicefrac{\num{\FracPerNumSimpl}}{\num{\FracPerDenomSimpl}}\$.$ 
```



Numérateur : 44672 et 22336
Dénominateur = 990 et 495

On a $45,1\overline{23} = \frac{44\,672}{990} = \frac{22\,336}{495} = 22\,336/495.$

Code \LaTeX et sortie \LaTeX $\backslash\text{FractionPeriode}[\text{Simple}]{45.1}{23}$ 

$$45,1\overline{23} = \frac{44\,672}{990} = \frac{22\,336}{495}$$

Code \LaTeX et sortie \LaTeX $\backslash\text{FractionPeriode}[\text{Solution}]{45.1}{23}$ On note $x = 45,1\overline{23}$.On ramène la période près de la virgule en multipliant par 10^1 :

$$10^1 \times x = 451,2\overline{3} \quad (1)$$

On décale la période avant la virgule, en multipliant l'égalité (1) par 10^2 :

$$10^2 \times 10^1 \times x = 10^2 \times 451,2\overline{3} \Rightarrow 10^3 \times x = 45\,123,2\overline{3} \quad (2)$$

On soustrait les deux égalités, (2) – (1), ce qui permet d'enlever la partie décimale :

$$\begin{aligned} 10^3 \times x - 10^1 \times x &= 45\,123,2\overline{3} - 451,2\overline{3} \Rightarrow (10^3 - 10^1) \times x = 45\,123 - 451 \\ &\Rightarrow 990 \times x = 44\,672 \\ &\Rightarrow x = \frac{44\,672}{990} \end{aligned}$$

$$\text{Ainsi on a } 45,1\overline{23} = \frac{44\,672}{990} = \frac{22\,336}{495}.$$

Code \LaTeX et sortie \LaTeX $\backslash\text{FractionPeriode}[\text{Simple}]{0.}{4}$ 

$$0,\overline{4} = \frac{4}{9}$$

Code \LaTeX et sortie \LaTeX $\backslash\text{FractionPeriode}[\text{Solution}, \text{Inconnue}=n]{154.99}{174}$ On note $n = 154,99\overline{174}$.On ramène la période près de la virgule en multipliant par 10^2 :

$$10^2 \times n = 15\,499,1\overline{74} \quad (1)$$

On décale la période avant la virgule, en multipliant l'égalité (1) par 10^3 :

$$10^3 \times 10^2 \times n = 10^3 \times 15\,499,1\overline{74} \Rightarrow 10^5 \times n = 15\,499\,174,1\overline{74} \quad (2)$$

On soustrait les deux égalités, (2) – (1), ce qui permet d'enlever la partie décimale :

$$\begin{aligned} 10^5 \times n - 10^2 \times n &= 15\,499\,174,1\overline{74} - 15\,499,1\overline{74} \Rightarrow (10^5 - 10^2) \times n = 15\,499\,174 - 15\,499 \\ &\Rightarrow 99\,900 \times n = 15\,483\,675 \\ &\Rightarrow n = \frac{15\,483\,675}{99\,900} \end{aligned}$$

$$\text{Ainsi on a } 154,99\overline{174} = \frac{15\,483\,675}{99\,900} = \frac{206\,449}{1\,332}.$$

Thème

JEUX ET RÉCRÉATIONS

Treizième partie

Jeux et récréations

62 SudoMaths, en TikZ

62.1 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 \LaTeX

```
%grille classique non remplie, avec légendes H/V, {} nécessaires pour préciser que les cases  
seront "vides"  
\SudoMaths{}
```



Code \LaTeX et sortie \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 \LaTeX

```
%grille "toute seule"  
\SudoMaths[clés]{liste}  
  
%grille avec ajout de code  
\begin{EnvSudoMaths}[clés]{grille}  
  %commandes tikz  
\end{EnvSudoMaths}
```

62.2 Clés et options



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

- la clé **Épaisseur** pour gérer l'épaisseur des traits épais; défaut **1.5pt**
- la clé **Épaisseur** 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 \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}
```



Sortie \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).



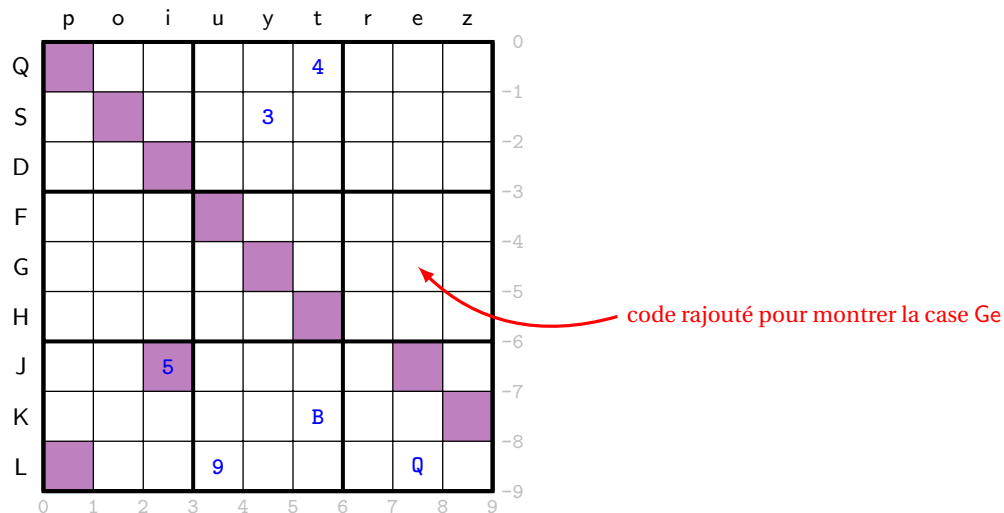
</> Code \LaTeX

```
%grille classique avec coloration de cases et commande tikz
%graduations rajoutées pour la lecture des coordonnées
\def\grilleSuMaB{%
  *////4///S%
  /*///3///S%
  //*/////S%
  ///*/////S%
  ////*/////S%
  /////*///S%
  //5*/////S%
  ////B///S%
  *///9///Q/S%
}

\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}
```



Sortie \LaTeX



63 Quelques fractales, en TikZ

63.1 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*.

63.2 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 \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 \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}
```



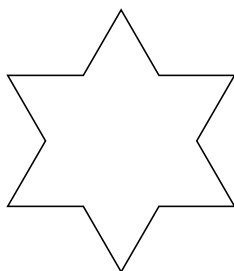
Les **<clés>** disponibles pour cette commande sont :

- la clé **<Epaisseur>** 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 **<Etape>** (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 **<Depart>** pour fixer le point de départ; défaut **<(0,0)>**
- le booléen **<AlignV>** (pour **<Type=Koch>**) pour forcer l'alignement de la *base*. défaut **<false>**



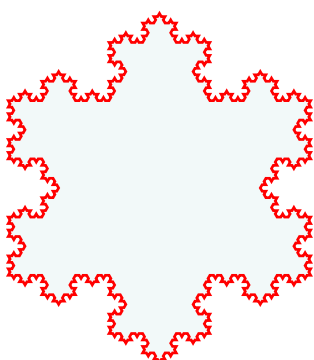
</> Code \LaTeX

```
%Koch par défaut
\FractaleTikz
```



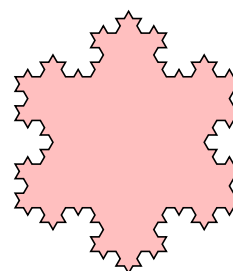
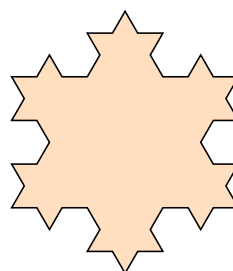
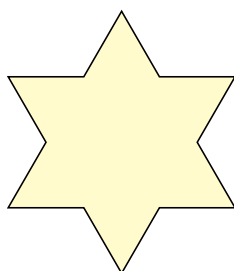
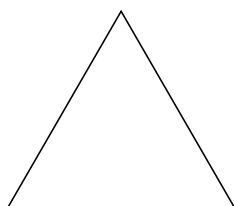
</> Code \LaTeX

```
%Koch par défaut
\FractaleTikz[Etape=4,LongueurCote=4,Remplir,Remplissage=teal!5,Couleur=red,Epaisseur=1pt]
```



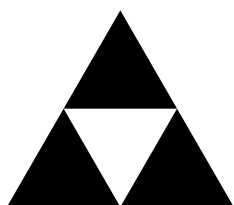
</> Code \LaTeX

```
%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

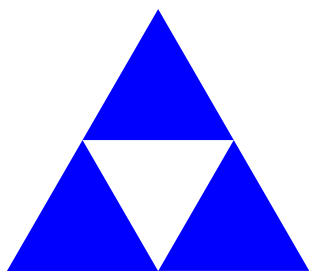
```
%Sierpinski par défaut
\FractaleTikz[Type=Sierp]
```





</> Code \LaTeX

```
%Sierpinski par défaut
\FractaleTikz[Type=Sierp,LongueurCote=4,Couleur=blue]
```



63.3 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**

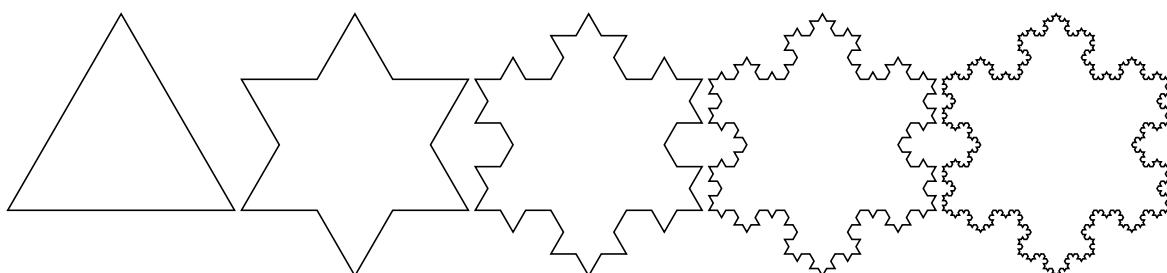
L'argument obligatoire, et entre `{...}`, permet de spécifier les étapes à afficher, sous la forme TikZ :

- `n1,n2,n3` pour spécifier une liste d'étapes;
- `n1,...,n2` pour spécifier une plage d'étapes.



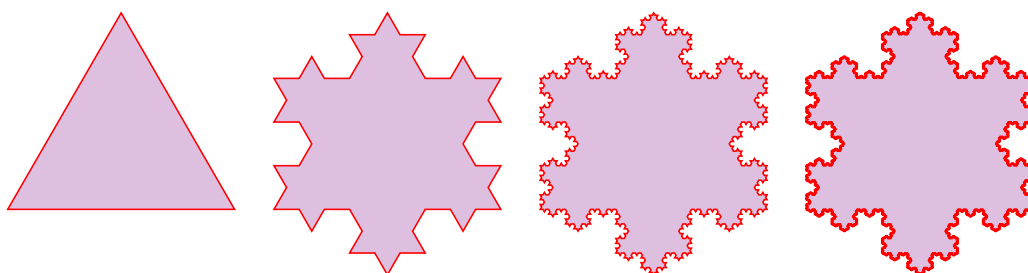
</> Code \LaTeX

```
\EtapesFloconKoch{0,...,4}
```



</> Code \LaTeX

```
\EtapesFloconKoch[Offset=5mm,Couleur=red,Remplir,Remplissage=violet!25]{0,2,4,6}
```



63.4 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 :

- `\TeX n1,n2,n3` pour spécifier une liste d'étapes ;
- `\TeX n1,...,n2` pour spécifier une plage d'étapes.



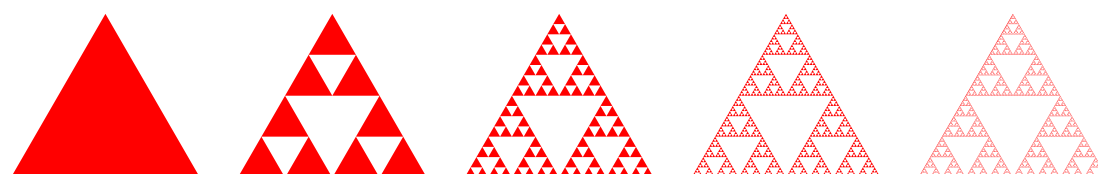
</> Code \LaTeX

```
\EtapesTapisSierpinski{0,...,4}
```



</> Code \LaTeX

```
\EtapesTapisSierpinski[LongueurCote=2.5,Offset=5mm,Couleur=red]{0,2,4,6,8}
```



64 Châteaux de cartes

64.1 Introduction



`3.00d` L'idée est de présenter des empilements type *châteaux de cartes* pour illustrer un travail sur des suites, par exemple.



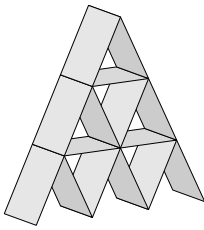
</> Code \LaTeX

```
\ChateauCartes[clés]{niveau}<options tikz>
```



</> Code \LaTeX

```
%clés par défaut  
\ChateauCartes{3}
```



64.2 Clés et options



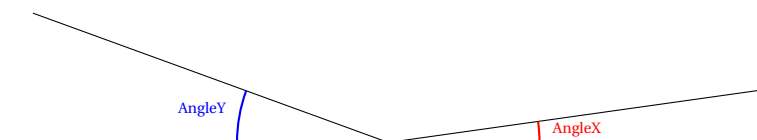
Les **<clés>** disponibles pour cette commande sont :

- 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.



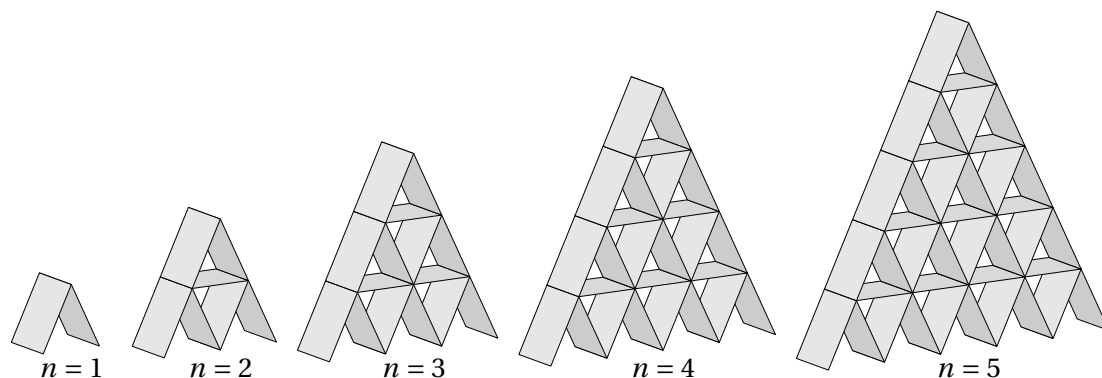
Concernant les clés **<AngleX>** et **<AngleY>**, elles correspondent à l'*inclinaison* des axes pour la vue 3D :





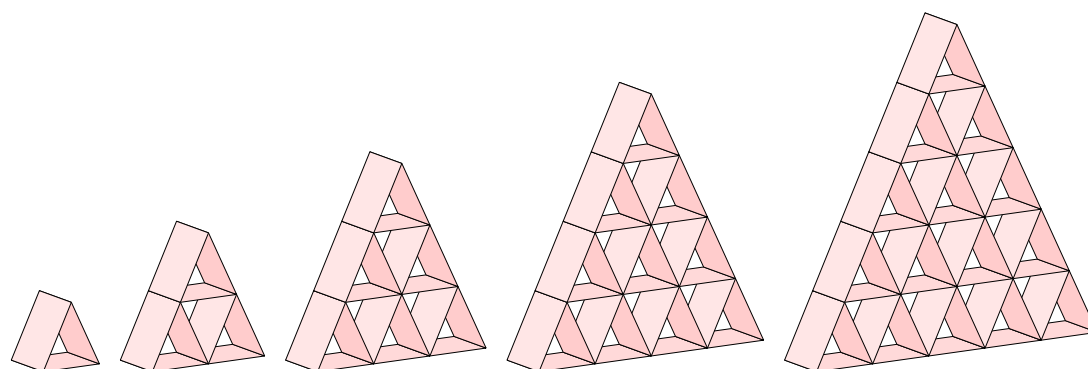
Code \LaTeX

```
\xintFor* #1 in {\xintSeq{1}{5}}\do{\ChateauCartes[Legende]{#1}}
```



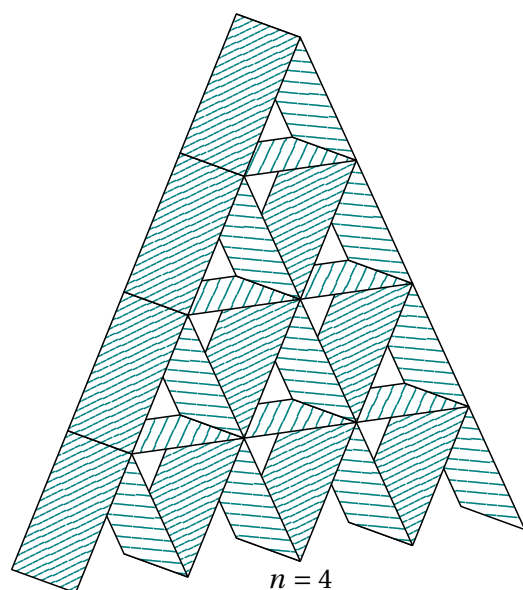
Code \LaTeX

```
\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}
```



65 Allumettes

65.1 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.



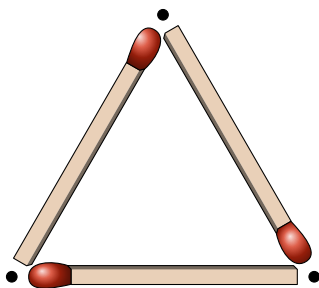
</> Code \LaTeX

```
%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=...`.

65.2 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**).

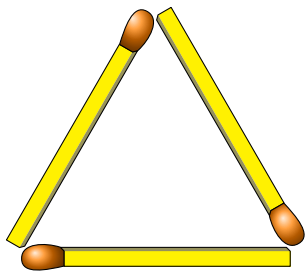
L'argument obligatoire, entre `{...}`, correspond à la liste des chemins, sous la forme

```
A1>B1 A2>B2 ...
```



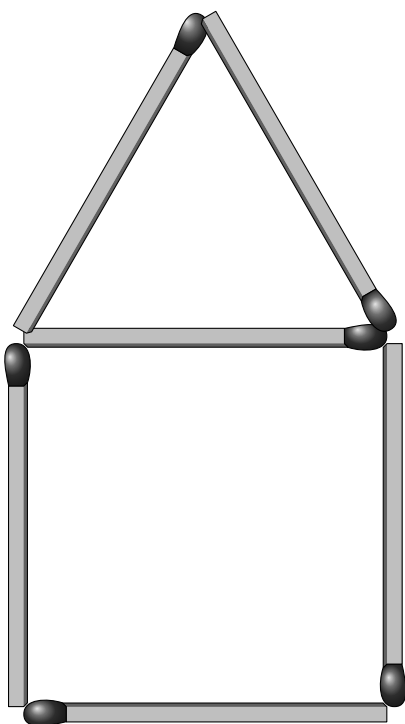

</> Code \LaTeX

```
\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}
```



66 Machines à transformer, en TikZ

66.1 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.

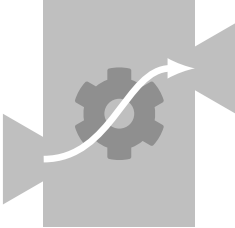


Code \LaTeX

```
\MachineTransformer[clés]{e/s}<options tikz>
```

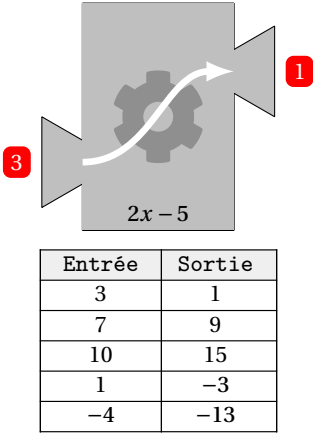


Code \LaTeX



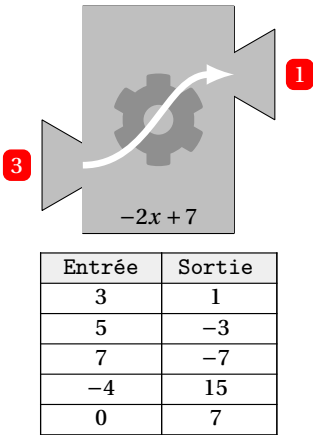
Code \LaTeX

```
\MachineTransformer[Bordure,Tableau,Fct=$2x-5$] %  
{3/1,7/9,10/15,1/$-3$, $-4$/$-13$}
```



Code \LaTeX

```
\MachineTransformer[Auto,Formule={-2*X+7},Bordure,Tableau,Fct=$-2x+7$] %  
{3,5,7,-4,0}
```



66.2 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.



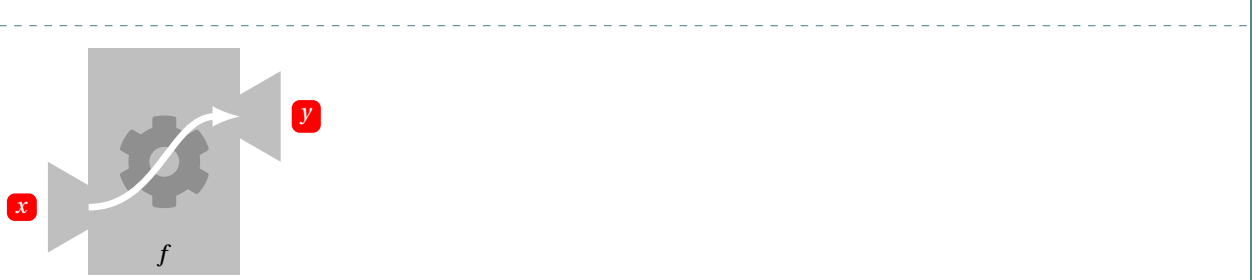
Dans le cas où les calculs seraient faits en automatique :

- 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 $xint$, avec x comme variable!



Code \LaTeX

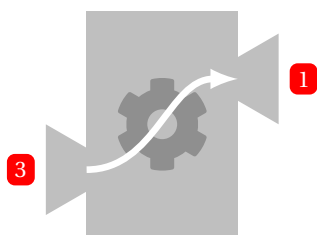
```
\MachineTransformer[Fct=$f$]{$x$/$y$}
```





Code \LaTeX

```
\MachineTransformer[Tableau]{3/1,7/9,10/15,1/$-3$, $-4$/$-13$}
```

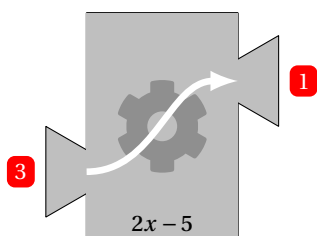


Entrée	Sortie
3	1
7	9
10	15
1	-3
-4	-13



Code \LaTeX

```
\MachineTransformer[Bordure,Tableau,Fct=$2x-5$] %  
{3/1,7/9,10/15,1/$-3$, $-4$/$-13$}
```

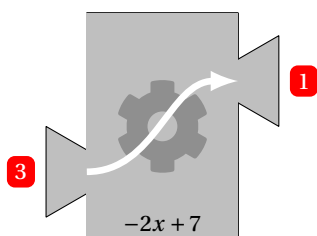


Entrée	Sortie
3	1
7	9
10	15
1	-3
-4	-13



Code \LaTeX

```
\MachineTransformer[Auto,Formule={-2*X+7},Bordure,Tableau,Fct=$-2x+7$] %  
{3,5,7,-4,0}
```

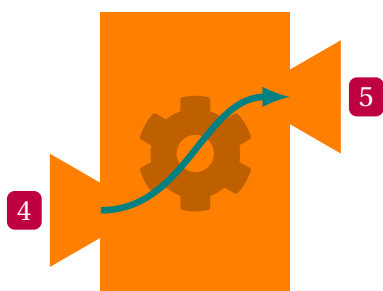


Entrée	Sortie
3	1
5	-3
7	-7
-4	15
0	7



</> Code \LaTeX

```
\MachineTransformer[%  
  Echelle=1.25,Tableau,Couleur=orange,CouleurFct=teal,%  
  CouleurBloc=purple,Offset=1mm  
1%  
{4/5}
```



Entrée	Sortie
4	5

Thème

HISTORIQUE

Quatorzième partie

Historique

- v3.01f : PseudoCode via le package piton (page 80)
 - : Suppression des 'Terminal' (doubleton avec sim-os-menus)
 - v3.01e : Mise en conformité avec piton et pyluatex
 - v3.01d : Ajout d'un style (gris italique) pour les commentaires en PseudoCode (page 78)
 - v3.01c : Ajout d'une commande pour générer des splines cubiques (page 50)
 - v3.01b : Division euclidienne (page 150) + Correction de bugs
 - v3.01a : Courbes ECC/FCC pour les paramètres (pages 130)
 - v3.00g : Amélioration de la marge gauche dans les codes listings (pages 78)
 - v3.00f : Conversion d'une écriture décimale périodique en fraction, avec rédaction éventuelle (pages 179)
 - v3.00e : Machines à transformer (pages 194) + Correction du nom pour les allumettes (192)
 - v3.00d : Commandes pour travailler avec des châteaux de cartes et des allumettes (pages 190 et 192)
 - v3.00c : Commandes pour travailler avec des chiffrements classiques (page 166)
 - v3.00b : Amélioration de la commande des intervalles avec gestion d'un label pour les bornes (page 37)
 - v3.00a : Environnement et commande pour travailler avec des intervalles (page 37)
-
- v2.8.0 : Amélioration du tapis de Sierpinski (page 186)
 - v2.7.9 : Ajout de quelques fractales (Koch + Sierpinski) (page 186)
 - v2.7.8 : Conversion présentée hexa/bin (page 153) + Liste et arbres de diviseurs (page 163)
 - v2.7.7 : Points de discontinuité pour des splines (page 55)
 - v2.7.6 : Amélioration dans la console d'exécution avec piton
 - v2.7.5 : Possibilités de numérotter les lignes des codes à une autre valeur que 1 (page 67)
 - v2.7.4 : Ajout de macros pour des écritures mathématiques classiques (page 21)
 - v2.7.3 : Correction de la couleur de bordures vertes pour les codes python
 - v2.7.2 : xcolor n'est plus chargé par défaut (option [xcolor] pour le charger)
 - v2.7.1 : Chargement de tcolorbox par librairies (au lieu de [most])
 - v2.7.0 : Ajout de la clé **<Frac>** pour les axes verticaux (page 40) + Fonction de répartition discrète (147)
 - v2.6.9 : Amélioration de la présentation de code Piton (page 71)
 - v2.6.8 : Ajout d'une grille pour les histogrammes non réguliers (page 125)
 - v2.6.7 : Histogramme à classes régulières ou non (page 125) + Correction de bugs mineurs
 - v2.6.6 : Style mainlevee en TikZ désormais dans le package tikz2d-fr
 - v2.6.5 : Ajout d'une option **<nonamssymb>** pour éviter de charger amssymb (page 11) + Distance entre deux points (page 102)
 - v2.6.4 : Résolution d'une équation diophantienne $ax + by = c$ (page 159) + Correction de bugs mineurs + Ajout de commandes en géométrie analytique (pages 94 et 96 et 98 et 100 et 103)
 - v2.6.3 : Ajout d'une commande pour déterminer une équation réduite (page 105)
 - v2.6.2 : Ajout d'une clé **<AffTraitsEq>** pour les équations trigo (page 90)
 - v2.6.1 : Ajout de commandes pour du calcul intégral (pages 34 et 62)
 - v2.6.0 : Ajout d'une clé **<Brut>** pour les mesures principales + correction d'un bug + Refonte de la doc + Commande calcul ligne trigo (pages 175 et 176)
 - v2.5.9 : Ajout clé **<CouleurNombres>** pour Piton (v1.5 mini) (page 71)
 - v2.5.8 : Ajout d'un style Alt pour les codes (pages 67 et 75) + Modification de la syntaxe des commandes avec Pythontex et PseudoCode (pages 74 et 78)
 - v2.5.7 : Ajout de clés pour les codes Piton + Console via Pyluatex (page 71)
 - v2.5.6 : Ajout d'une clé **<Trigo>** pour l'axe (Ox) (page 40)
 - v2.5.5 : Externalisation de la fenêtre XCas (dans le package FentreCas)
 - v2.5.4 : Modification des calculs (via xint) en combinatoire (page 146)
 - v2.5.3 : Modification du traitement des tests dans les arbres de probas (page 139)
 - v2.5.1 : Ajout d'une version étoilée pour la conversion en fraction (page 171)
 - v2.5.0 : Système de librairies pour certains packages/commandes (page 11)
 - v2.2.0 : Ajout d'une clé **<Notation>** pour les arrangements et combinaisons (page 146)
 - v2.1.9 : Correction d'un bug (et ajout d'une version étoilée) pour les petits schémas « de signe » (page 56)
 - v2.1.8 : Suppression des commandes de PixelArt, désormais dans le package PixelArtTikz
 - v2.1.7 : Ajout d'une clé **<Math>** pour les sommets des figures de l'espace (pages 86 et 88)
 - v2.1.6 : Correction d'un bug lié au chargement de hvlogos, remplacé par hologo
 - v2.1.5 : Combinatoire avec arrangements et combinaisons (page 146)
 - v2.1.4 : Résolution approchée d'équations $f(x) = k$ (page 28)
 - v2.1.3 : Améliorations dans les présentations Piton (page 71)
 - v2.1.2 : Ajout d'une commande pour la mesure principale d'un angle (page 175)

- [v2.1.1](#) : Ajout d'une section pour des repères en *TikZ* (page 40)
 - [v2.1.0](#) : Calcul du seuil, en interne désormais (page 32) + Commande pour simplifier une racine carrée (page 174)
 - [v2.0.9](#) : Nombres aléatoires, tirages aléatoires d'entiers (page 144)
 - [v2.0.8](#) : Ajout d'un environnement pour présenter du code \LaTeX (page 84)
 - [v2.0.6](#) : Changement de taille de la police des codes Python (page 67)
 - [v2.0.5](#) : Correction d'un bug avec les calculs de suites récurrentes (page 32)
 - [v2.0.4](#) : Ajout d'une commande pour une présentation de solution par TVI (page 30)
 - [v2.0.3](#) : Commandes pour des suites récurrentes *simples* (page 32)
 - [v2.0.2](#) : Option `left-margin=auto` pour le package `piton` (page 71)
 - [v2.0.1](#) : Chargement du package `piton` uniquement si compilation en \LaTeX (page 71)
 - [v2.0.0](#) : Refonte du code source avec modification des commandes, et de la documentation
-