ProfLycee

Quelques petites commandes pour La (au lycée)

Cédric Pierquet c pierquet -- at -- outlook . fr

Version 2.5.1 – 10 Février 2023

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

- ⋄ résoudre, de manière approchée, des équations
- ⋄ tracer facilement des repères/grilles/courbes
- tracer des courbes lisses avec gestion des extrema et des dérivées
- simuler une fenêtre de logiciel de calcul formel, une console d'exécution
- présenter du code python ou pseudocode
- 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)
- ⋄ 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
- **«»** ...

Merci à Anne pour ses retours et sa relecture! Merci à Christophe et Denis pour leurs retours et éclairages! Merci aux membres du groupe 😝 du « Coin ŁTEX » pour leur aide et leurs idées!

MEX

pdflETEX

LualATEX

Ti*k*7

T_EXLive

MiKTEX

Table des matières

I	Introduction	6
1	Le package ProfLycee1.1 «Philosophie» du package1.2 Chargement du package1.3 Librairies, option du package	6 6 6 7
2	Compléments 2.1 Le système de « clés/options »	8 8 8
II	Liste des commandes, par thème	10
III	Outils pour l'analyse	14
3		14 14 14
4	Présentation d'une solution d'équation par balayage 4.1 Idée 4.2 Clés et arguments 4.3 Interaction avec la commande de résolution approchée	16 16 16 17
5	Suites récurrentes simples 5.1 Idées	18 18 18 19
IV	Outils graphiques, via TikZ	20
6	Repérage et tracé de courbes, en TikZ 6.1 Idée 6.2 Commandes, clés et options 6.3 Commandes annexes 6.4 Repère non centré en O	20 20 21 23 24
7	L'outil « SplineTikz »7.1 Courbe d'interpolation7.2 Code, clés et options7.3 Compléments sur les coefficients de « compensation »7.4 Exemples7.5 Avec une gestion plus fine des « coefficients »7.6 Conclusion	26 26 26 26 27 28 28
8	L'outil « TangenteTikz » 8.1 Définitions 8.2 Exemple et illustration 8.3 Exemple avec les deux outils, et « personnalisation »	29 29 29 30

9	Petits schémas pour le signe d'une fonction affine ou d'un trinôme	31
	9.1 Idée	31
	9.2 Commandes	
	9.3 Intégration avec tkz-tab	33
10	Suites récurrentes et « toile »	34
	10.1 Idée	34
	10.2 Commandes	34
	10.3 Exemples	34
	10.4 Influence des paramètres	36
V	Présentation de codes	37
		0.7
11	L'outil « Calcul Formel »	37 37
	11.1 Introduction	
	11.3 La commande « CalculFormelLigne »	
	11.4 Visualisation des paramètres	
	•	
12	Code Python « simple » via le package listings	39
	12.1 Introduction	39
	12.2 Commande et options	39
	12.3 Insertion via un fichier « externe »	39 40
	12.4 Exemples	40
13	Code Python via le package piton	42
	13.1 Introduction	42
	13.2 Présentation de code Python	42
14	Code & Console Python, via les packages Pythontex ou Minted	44
	14.1 Librairies	44
	14.2 Introduction	44
	14.3 Présentation de code Python grâce au package pythontex	44
	14.4 Présentation de code Python via le package minted	45
	14.5 Console d'exécution Python	46
15	Pseudo-Code	48
	15.1 Introduction	48
	15.2 Présentation de Pseudo-Code	48
	15.3 Compléments	49
16	Terminal Windows/UNiX/OSX	50
-0	16.1 Introduction	50
	16.2 Commandes	50
17	Cartoucho Canutalo	52
17	Cartouche Capytale 17.1 Introduction	52
	17.1 Introduction	
18	Présentation de code MEX	53
	18.1 Introduction	53 53
	18.2 Commandes	53
.		
VI	Outils pour la géométrie	54

19	Pavé droit « simple » 19.1 Introduction	54 54
	19.2 Commandes	54 55
20	Tétraèdre « simple »20.1 Introduction	56 56 56 57
21	Cercle trigo 21.1 Idée	58 58 58 59
22	Style « main levée » en TikZ 22.1 Idée	61 61
VII	I Outils pour les statistiques	62
	Paramètres d'une régression linéaire par la méthode des moindres carrés 23.1 Idée	62 62 64
24	Statistiques à deux variables24.1 Idées24.2 Commandes, clés et options24.3 Commandes annexes24.4 Interactions avec CalculsRegLin	67 68 71 72
25	Boîtes à moustaches 25.1 Introduction	76 76 76 77
VII	II Outils pour les probabilités	80
26	Calculs de probabilités26.1 Introduction26.2 Calculs « simples »26.3 Complément avec sortie « formatée »	80 80 80 82
	Arbres de probabilités « classiques » 27.1 Introduction	85 85 85 86
28	Petits schémas pour des probabilités continues 28.1 Idée	89 89 89 90

29	Nombres aléatoires 29.1 Idée	91 91 93
30	Combinatoire 30.1 Idée	94 94 94
IX	Outils pour l'arithmétique	95
	Conversions binaire/hexadécimal/décimal 31.1 Idée	95 95 95 96 97
	32.1 Idée	
33	Algorithme d'Euclide pour le PGCD 33.1 Idée	100
X	Écritures, simplifications	103
34	Simplification sous forme d'une fractions 34.1 Idée	
35	Ensembles 35.1 Idée	105 105 105
36	Écriture d'un trinôme, trinôme aléatoire36.1 Idée	
37	Simplification de racines 37.1 Idée 37.2 Exemples	
38	Mesure principale d'un angle 38.1 Idée 38.2 Exemples	
ΧI	Jeux et récréations	110
39	SudoMaths, en TikZ 39.1 Introduction	
ΧI	I Historiane	113

Première partie

Introduction

1 Le package ProfLycee

1.1 « Philosophie » du package

♀ Idée(s)

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, 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 \[\underset \text{amssymb} \] 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!

information(s)

Le package ProfLycee charge les packages :

- <a>§xcolor avec l' option [table,svgnames];
- [tikz, pgf], pgffor, nicefrac;
- \(\begin{aligned}
 \text{tcolorbox} \text{ avec l'option [most];}\end{aligned}\)
- 📱 xparse, 🖺 xstring, 🖺 simplekv, 🖺 xinttools;
- [listofitems, xintexpr , xintbinhex, xintgcd];
- [tabularray, [fontawesome5, [randomlist, [fancyvrb].

♀ Idée(s)

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 MTEX avec listofitems, randomlist, simplekv ou encore xstring!

1.2 Chargement du package

⟨⟩ Code LATEX

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

1.3 Librairies, option du package

Attention

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

⟨→ Code LATEX

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

information(s)

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

- piton (page 42);
- ∰ minted (page 45);
- pythontex (page 44).

information(s)

En compilant (notamment avec les librairies 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}
\useproflyclib{...}
...
```

information(s)

L'option (build) charge certains packages (librairies minted et pythontex) avec les options :

- [a]\setpythontexoutputdir{./build/pythontex-files-\jobname}
- RequirePackage[outputdir=build]{minted}

2 Compléments

2.1 Le système de « clés/options »

♀ Idée(s)

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.

information(s)

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.

Les codes seront présentés dans des boîtes **\(\)** Code **\(\)** Code **\(\)** Sortie **\(\)** Sort

Les clés ou options seront présentées dans des boîtes @ Clés.

information(s)

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

2.2 Compilateur(s)

information(s)

Le package ProfLycee est compatible avec les compilateurs classiques : latex, pdflatex ou encore lualatex. 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 Lual ETEX et shell-escape (ou write18).

2.3 Problèmes éventuels...

information(s)

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

Certains packages ont une fâcheuse tendance à être tatillons sur leurs options (les *fameux* option clash for ...) ou leur *position* dans le chargement, donc attention notamment au chargement de <code>scolor</code> et de <code>massymbl</code>!

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

Bonne(s) découverte(s) m

•

Deuxième partie

Liste des commandes, par thème

information(s)

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

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

```
% Code Language en tikz et courbe

\[ \GrilleTikz[options][options grille ppale][options grille second.]
\[ \AxesTikz[options] \AxesTikz[options]{valeurs} \AxeyTikz[options]{valeurs}
\[ \FenetreSimpleTikz[options](opt axes) < opt axe 0x > {liste valx} < opt axe 0y > {liste valy}
\[ \CourbeTikz[options] {fonction} {valxmin:valxmax}
\]
\[ \% courbe d'interpolation, tangente, dans un environnement tikz
\[ \SplineTikz[options] {liste}
\]
\[ \TangenteTikz[options] {liste}
\]
\[ \% schémas pour le signe affine/trinôme, dans un environnement tikz
\]
\[ \MiniSchemaSignes(*)[clés] < options tikz >
\]
\[ \MiniSchemaSignesTkzTab[options] {numligne}[échelle][décalage horizontal]
\]
```

```
%présentation type calcul formel, dans un environnement tikz
\CalculFormelParametres[options]
\CalculFormelLigne[options] {commande} {résultat}

%présentation de code Python
\begin{CodePythonLst}(*) [largeur] {commandes tcbox}...\end{CodePythonLst}
%:=librairie piton
\begin{CodePiton} [options]...\end{CodePiton}
%:=librairie pythontex
\begin{CodePythontex} [options]...\end{CodePythontex}
\begin{ConsolePythontex} [options]...\end{ConsolePythontex}
%:=librairie minted
\begin{CodePythonMinted}(*) [largeur] [options]...\end{CodePythonMinted}
%présentation de pseudocode
\begin{PseudoCode}(*) [largeur] [options]...\end{PseudoCode}
```

```
% Code Large
%terminal OS
\begin{TerminalWin} [largeur] {clés} [options] ... \end{TerminalWin}
\begin{TerminalUnix} [largeur] {clés} [options] ... \end{TerminalUnix}
\begin{TerminalOSX} [largeur] {clés} [options] ... \end{TerminalOSX}

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

```
% Code LaTeX

%pavé et tétraèdre, dans un environnement tikz

\PaveTikz[options]

\TetraedreTikz[options]

%cercle trigo, dans un environnement tikz

\CercleTrigo[clés]
```

```
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 0x>{liste valx}<opt axe 0y>{liste valy}
\NuagePointsTikz[options]{listeX}{listeY}
\PointMoyenTikz[options]
\CourbeTikz[options]{formule}{domaine}
%boîte à moustaches, dans un environnement tikz
\BoiteMoustaches[options]
\BoiteMoustachesAxe[options]
```

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

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

```
% 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}
```

```
% 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}
\ConversionVersDec[clés]{nombre}
\ConversionBaseDix[clés]{nombre}{base de départ}
\ConversionDepuisBaseDix[options]{nombre en base 10}{base d'arrivée}

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

% Code Land 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 \MesurePrincipale[options]{angle}

Code LATEX

% 200 % s

\SudoMaths[options]{liste}

 $\verb|\begin{EnvSudoMaths}| [options] {grille}... \\| end {EnvSudoMaths}| \\| end {EnvSudoMaths$

Troisième partie

Outils pour l'analyse

3 Résolution approchée d'une équation

3.1 Idée

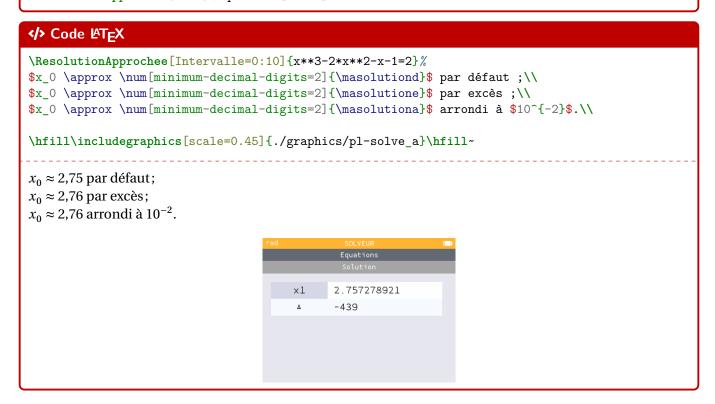
♀ Idée(s)

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]



3.2 Clés et options

© Clés et options

Quelques explications sur les (clés) et sur les arguments :

- la clé (Précision) qui permet de spécifier 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(...) = k (ou f(...) pour f(...) = 0);
- l'argument *optionnel* est la base de la *<macro>* qui sert à stocker les valeurs : défaut *(masolution)*
 - [\macro>d pour la valeur approchée par défaut;
 - [\macro>e pour la valeur approchée par excès;
 - [se \macro>a] pour la valeur approchée.

```
Code 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 $\beta \approx \num[minimum-decimal-digits=4] {\masolutiond}$ par défaut ;
  \item $\beta \approx \num[minimum-decimal-digits=4] {\masolutione}$ par excès;
  \item $\beta \approx \num[minimum-decimal-digits=4]{\masolutiona}$.
\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\,\rm{e}^{-0}, t+1}=4 est t_1
→ avec :
\begin{itemize}
  \item $t 1 \approx \num[minimum-decimal-digits=2] {\SolAd}$ par défaut ;
  \item $t_1 \approx \num[minimum-decimal-digits=2]{\SolAe}$ par excès ;
  \item $t_1 \approx \num[minimum-decimal-digits=2]{\SolAa}$.
\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 31\, \text{ext}e^{-0,5t+1}=4 est t_2
→ avec :
\begin{itemize}
  \item $t_2 \approx \num[minimum-decimal-digits=2]{\SolBd}$ par défaut ;
  \item $t_2 \approx \num[minimum-decimal-digits=2]{\SolBe}$ par excès ;
  \item $t_2 \approx \num[minimum-decimal-digits=2]{\SolBa}$.
\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 \beta avec :
   - β ≈ 1,4068 par défaut;
   - β ≈ 1,4069 par excès;
   - β ≈ 1.4069.
Une valeur approchée, à 10^{-2} près d'une solution de 3t e^{-0.5t+1} = 4 est t_1 avec :
   — t_1 ≈ 0,69 par défaut;
   — t_1 \approx 0.70 par excès;
   — t_1 ≈ 0,69.
Une valeur approchée, à 10^{-2} près d'une solution de 3t e^{-0.5t+1} = 4 est t_2 avec :
   — t_2 ≈ 4,377 par défaut;
   — t_2 \approx 4,378 par excès;
   — t_2 ≈ 4,378.
                                                 Graphique
         x=1.406879
                          f(x) = 0
                                        x=0.6939632
                                                         f(x)=4
                                                                       x=4.377668
```

4 Présentation d'une solution d'équation par balayage

4.1 Idée

♀ Idée(s)

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]*.

Attention

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}

4.2 Clés et arguments

Clés et options

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

— la clé (**NomFct**) qui permet de spécifier le nom de la fonction;

défaut **⟨f⟩**

— la clé (**NomSol**) qui permet de spécifier le nom de la fonction;

défaut (\alpha)

- les clés (va) et (vb) qui sont les bornes inférieure et supérieure de l'encadrement;
- la clé (**Precision**) qui est la précision des calculs pour les images;

défaut (2)

— la clé (**Stretch**) qui permet d'espacer les lignes;

- défaut (1.15)
- les booléens (Balayage) ou (Calculatrice) pour afficher un texte en amont;

défaut **(false)**

— le booléen (Majuscule) qui affiche le texte avant, avec une majuscule au début.

défaut **(true)**

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

Code LATEX

Pour f(x)=0 avec $f(x)=x^2-2$. On obtient

→ \SolutionTVI[va=1.414,vb=1.415,Precision=3] {x**2-2}{0}.

Pour
$$f(x) = 0$$
 avec $f(x) = x^2 - 2$. On obtient $\begin{cases} f(1,414) \approx -0.001 < 0 \\ f(1,415) \approx 0.002 > 0 \end{cases} \Rightarrow 1,414 < \alpha < 1,415$.

</> ✓ Code LATEX

Avec $\frac{1}{-0,5t+1}=5$,

 $\label{lem:control} $$ \SolutionTVI [Majuscule=false, Calculatrice, va=1.02, vb=1.03, NomFct=\varphi] $$ \{3*x*exp(-0.5*x+1)\} $$ \{5\} $$ $$ \Calculatrice, va=1.02, vb=1.03, NomFct=\varphi] $$ \{3*x*exp(-0.5*x+1)\} $$ \{5\} $$ \Calculatrice, va=1.02, vb=1.03, NomFct=\varphi] $$ \{3*x*exp(-0.5*x+1)\} $$ \{5\} $$ \Calculatrice, va=1.02, vb=1.03, NomFct=\varphi] $$ \{3*x*exp(-0.5*x+1)\} $$ \{5\} $$ \Calculatrice, va=1.02, vb=1.03, NomFct=\varphi] $$ \{3*x*exp(-0.5*x+1)\} $$ \{5\} $$ \Calculatrice, va=1.02, vb=1.03, NomFct=\varphi] $$ \{6\} $$ \Calculatrice, va=1.03, NomFct=\varphi] $$ \{6\} $$ \Calcu$

Avec
$$\varphi(t) = 3t e^{-0.5t+1} = 5$$
, par calculatrice, on obtient
$$\begin{cases} \varphi(1,02) \approx 4.99 < 5 \\ \varphi(1,03) \approx 5.02 > 5 \end{cases} \Rightarrow 1.02 < \alpha < 1.03$$

Code Lagrange Approximately Code Lagrange C

4.3 Interaction avec la commande de résolution approchée

♀ Idée(s)

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

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

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

Par balayage, on obtient \begin{cases} g(4,48) \approx 1,4996 < 1,5 \\ g(4,49) \approx 1,5019 > 1,5 \end{cases} \Rightarrow 4,48 < x_0 < 4,49.
```

information(s)

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

5 Suites récurrentes simples

5.1 Idées

♀ Idée(s)

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 $u_n < S$.

Attention

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

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

⟨/> Code LATEX

%commande pour calculer et formater

\CalculTermeRecurrence[options]{fonction associée}

%mise en forme de la conclusion d'un seuil

\SolutionSeuil[options]{fonction associée}{seuil}

5.2 Clés et arguments

Clés et options

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;

— la clé (**N**) qui est l'indice du terme à calculer.

L'argument *obligatoire* est la fonction associée à la suite, en syntaxe $\frac{1}{2}$ xint et avec comme variable x.

⟨/> Code LATEX

On obtient $u_{10} \rightarrow CalculTermeRecurrence[No=0,UNo=50,N=10] {1/(x+2)}.$

On obtient $u_{15} \rightarrow CalculTermeRecurrence[Precision=4,No=0,UNo=50,N=15]{1/(x+2)}$.

On obtient $u_{20} \rightarrow CalculTermeRecurrence[Precision=6,No=0,UNo=50,N=20]{1/(x+2)}$.$

Sortie LATEX

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

On obtient $u_{10} \approx 0.414$

On obtient $u_{15} \approx 0.4142$

On obtient $u_{20} \approx 0.414214$

sortie par défaut.

défaut (3)

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

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

```
Clés et options
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 [=] au lieu de [=] \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 Fxint et avec comme variable
x, et le second est le seuil à dépasser.
```

5.3 Exemple d'utilisation

```
Code LATEX
Avec \ \begin{dcases} u_1 = 2 \\ u_{n+1}=1+\dfrac{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 \\
  \xintFor* #1 in {\xintSeq{2}{7}} \do {#1 &
   \leftarrow \CalculTermeRecurrence[No=1,UNo=2,N=#1]{1+(1+x**2)/(1+x)} \\}
\end{tabular}\\

    de $n=\the\CompteurSeuil$)

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

Quatrième partie

Outils graphiques, via TikZ

6 Repérage et tracé de courbes, en TikZ

6.1 Idée

♀ Idée(s)

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

- axes et graduations, grille;
- courbe.

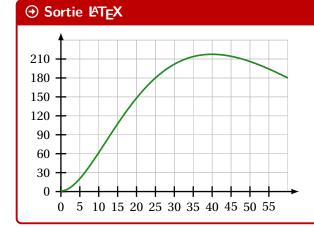
information(s)

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

⟨→ Code LATEX



6.2 Commandes, clés et options

information(s)

Les **(paramètres)** nécessaires à la bonne utilisation des commandes suivantes sont à déclarer directement dans l'environnement **[stikzpicture]**, seules les versions «x» sont présentées ici:

- ⟨xmin⟩, stockée dans [\xmin];
 ⟨xmax⟩, stockée dans [\xmax];
 défaut ⟨-3⟩
 défaut ⟨3⟩
- $-\langle \mathbf{O} \mathbf{x} \rangle$, stockée dans $\langle \mathbf{O} \mathbf{x} \rangle$, origine de l'axe $\langle \mathbf{O} \mathbf{x} \rangle$; défaut $\langle \mathbf{O} \mathbf{x} \rangle$
- ⟨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 $\langle clés \rangle$ pour des réglages fins, mais dans la majorité des cas elles ne sont pas forcément *utiles*.

Code LATEX

```
\label{likz} $$ \cline{Code tikz} $$ \cline{Condern} [options grille ppale] [options grille second.]
```

@ Clés et options

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]

Clés et options

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

- **(Epaisseur)** qui est l'épaisseur des axes; défaut (1pt)
- (**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)
- $\langle Labelx \rangle$ qui est le label de l'axe (Ox);
- défaut (\$x\$) — $\langle Labely \rangle$ qui est le label de l'axe (Oy); défaut (\$y\$)
- $\langle AffLabel \rangle$ qui est le code pour préciser quels labels afficher, entre $\langle x \rangle$, $\langle y \rangle$ ou $\langle xy \rangle$; défaut (vide)
- $\langle PosLabelx \rangle$ 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) défaut (latex)
- **(TypeFleche)** qui est le type de la flèche des axes.

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,Elargir0x=0,Elargir0y=0]

Sortie LATEX

Nombre d'individus infectés, en centaines

Nombre de jours

Code LATEX

%...code tikz

\AxexTikz[options] {valeurs}

\AxeyTikz[options] {valeurs}

@ Clés et options

Ces commande permet de tracer les graduations des axes, avec des (clés) identiques pour les deux directions :

- **(Epaisseur)** 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 (lgta/lgtb)); défaut (4pt)
- le booléen (AffGrad) pour afficher les valeurs (formatés avec num donc dépendant de sisetup) des graduations;

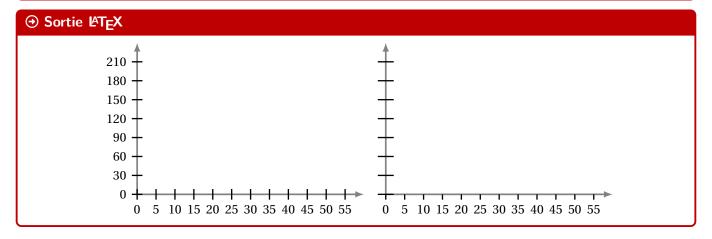
 défaut (true)
- le booléen (**AffOrigine**) pour afficher la graduation de l'origine; défaut (**true**)
- le booléen (Annee) qui permet de ne pas formater les valeurs des graduations (type année). défaut (false)

⟨/> Code LATEX

```
%code tikz
\AxexTikz[Police=\small]{0,5,...,55}
\AxeyTikz[Police=\small]{0,30,...,210}

%code tikz
\AxexTikz[Police=\small,HautGrad=0pt/4pt]{0,5,...,55}
\AxeyTikz[AffGrad=false,HautGrad=6pt]{0,30,...,210}

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



6.3 Commandes annexes

information(s)

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

- 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

♀ Idée(s)

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

Code LATEX

%...code tikz
\CourbeTikz[options]{formule}{domaine}

Clés et options

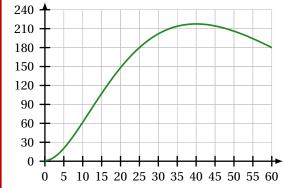
Cette commande permet de rajouter une courbe sur le graphique (sans se soucier de la transformation de son expression) avec les arguments :

- $\langle optionnels \rangle$ qui sont en TikZ les paramètres du tracé;
- le premier *obligatoire*, est en langage TikZ l'expression de la fonction à tracer, donc avec $[x] \times x$ comme variable;
- le second *obligatoire* est le domaine du tracé, sous la forme walxmin: 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,ForestGreen,samples=250]%
   {\x*\x*exp(-0.05*\x)+1}{0:60} %courbe
\end{tikzpicture}
```





6.4 Repère non centré en O

♀ Idée(s)

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 translations à effectuer!

L'outil « SplineTikz »

Courbe d'interpolation

information(s)

On va utiliser les notions suivantes pour paramétrer le tracé « automatique » grâce à 🖺 ...controls :

- il faut rentrer les points de contrôle;
- il faut préciser les pentes 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 : x1/y1/d1§x2/y2/d2§... avec les points (xi;yi) et f'(xi)=di;
- 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§....

Code, clés et options

```
</>
Code LATEX
\begin{tikzpicture}
  \SplineTikz[options] {liste}
\end{tikzpicture}
```

Clés et options

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 géré par la clé **(TaillePoints)**. défaut (2pt)
- Compléments sur les coefficients de « compensation » **7.3**

♀ Idée(s)

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 contrôles seront situés à une distance radiale de $\frac{b-a}{3}$.

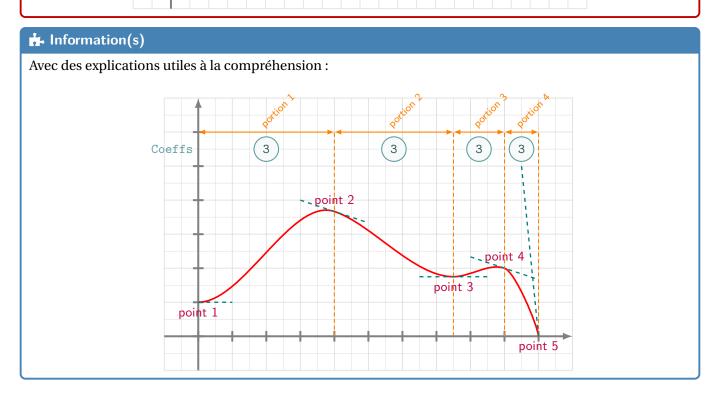
Pour écarter les points de contrôle, on peut du coup réduire le coefficient de compensation!

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.

7.4 Exemples

```
⟨→ Code LATEX
%code tikz
\def \x{0.9cm} \def \y{0.9cm}
%axes et grilles
\draw[xstep=\xgrilles,ystep=\ygrilles,line width=0.6pt,lightgray!50] (\xmin,\ymin) grid
\displaystyle \frac{\operatorname{line width=1.5pt,->,gray,>=latex}}{(xmin,0)--(xmax,0)};
\draw[line width=1.5pt,->,gray,>=latex] (0,\ymin)--(0,\ymax);
\draw[darkgray] (1,-4pt) node[below,font=\sffamily] {1};
\draw[darkgray] (-4pt,1) node[left,font=\sffamily] {1};
%splines
\label{liste} $$ \left(\frac{0}{1}\right)_{4/3.667/-0.333\S7.5/1.75/0\S9/2/-0.333\S10/0/-10} $$
\SplineTikz[AffPoints,Coeffs=3,Couleur=red]{\LISTE}
            1
```

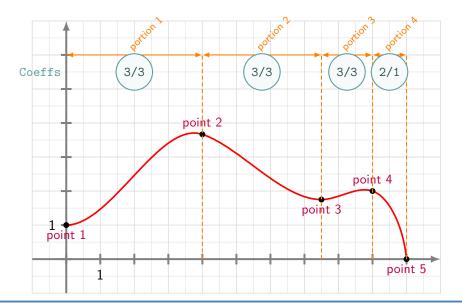


7.5 Avec une gestion plus fine des « coefficients »

information(s)

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

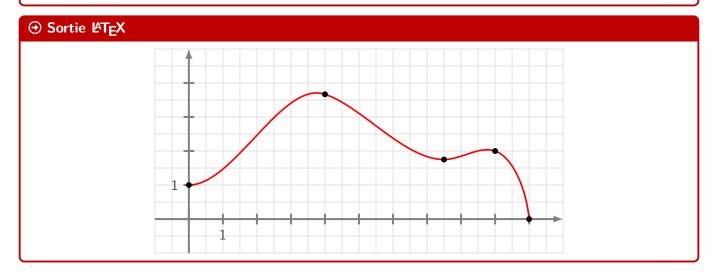
% on 7 in a

%splines

\def\LISTE{0/1/0\subseteq4/3.667/-0.333\subseteq7.5/1.75/0\subseteq9/2/-0.333\subseteq10/-10}

\SplineTikz[AffPoints,Coeffs=3\\$3\\$3\\$2/1]{\LISTE}

. . .



7.6 Conclusion

information(s)

Le plus « simple » est donc :

- de déclarer la liste des points de contrôle, grâce à [\def\LISTE{x1/y1/d1\su2/y2/d2\sum.}];
- de saisir la commande [\SplineTikz[...]{\LISTE}];
- d'ajuster les options et coefficients en fonction du rendu!

8 L'outil « TangenteTikz »

8.1 Définitions

♀ Idée(s)

En parallèle de l'outil FalineTikz, 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 FalineTikz.

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}
```

Clés et options

Cela permet de tracer la tangente :

- au point numéro (**Point**) de la liste (**liste**), de coordonnées xi/yi avec la pente di;
- avec une épaisseur de (**Epaisseur**), une couleur (**Couleur**) et un style additionnel (**Style**);
- en la traçant à partir de (xI) avant xi et jusqu'à (xr) après xi.

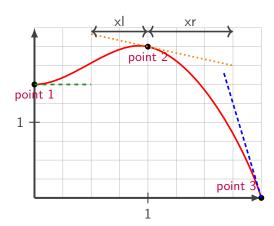
8.2 Exemple et illustration

</> 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=ForestGreen,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}
```

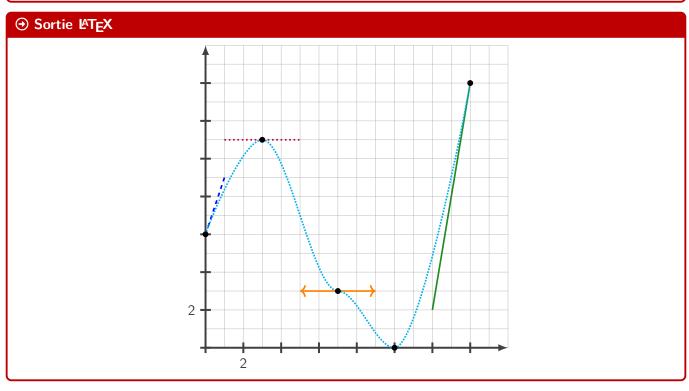
Sortie LATEX

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



8.3 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
  \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) ;\}
 %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=ForestGreen,Point=5]{\liste}
 %la courbe en elle-même
  \SplineTikz[AffPoints, Coeffs=3, Couleur=cyan, Style=densely dotted] {\liste}
\end{tikzpicture}
```



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

9.1 Idée

♀ Idée(s)

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 finsana même si la philosophie est un peu différente.

Comme pour les autres commandes TikZ, l'idée est de laisser la possibilité à l'utilisateur de définir et créer son environnement TikZ, et d'insérer la commande MiniSchemaSignes pour afficher le schéma.

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

⟨→ Code LATEX

\MiniSchemaSignes*



9.2 Commandes

Code LATEX

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

⟨→ Code LATEX

{\tikz[options] \MiniSchemaSignes[clés]}

%ou

\MiniSchemaSignes*[clés]<options tikzpicture>

Clés et options

22.1.9 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 $\langle Clés \rangle$ sont disponibles pour cette commande :

- 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 $\langle Hauteur \rangle$ qui est la hauteur du schéma; défaut $\langle 1 \rangle$
- un booléen **(Cadre)** qui affiche un cadre autour du schéma. défaut **(true)**

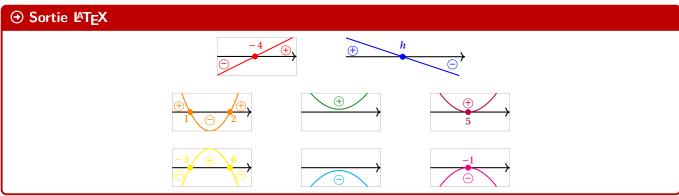
Le second argument, *optionnel* et entre <...>, permet de spécifier (pour la commande *étoilée*), des options à passer à l'environnement <u>l'ikzpicture</u>.

Clés et options

Pour la clé $\langle code \rangle$, 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;
- $\langle Code = pa + d + \rangle := une parabole souriante avec deux racines;$
- etc

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





9.3 Intégration avec tkz-tab

♀ Idée(s)

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 $\[Mathemax]$ in Schema Signes Tkz Tab 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}

@ Clés et options

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 :

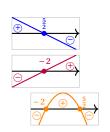
- le deuxième argument, *obligatoire*, est le numéro de la ligne à côté de laquelle placer le schéma;
- le troisième argument, *optionnel* et valant (**0.85**) par défaut, est l'échelle à appliquer sur l'ensemble du schéma (à ajuster en fonction de la hauteur de la ligne);
- le quatrième argument, *optionnel* et valant $\langle 1.5 \rangle$ par défait, 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
```

Sortie LATEX

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



10 Suites récurrentes et « toile »

10.1 Idée

♀ Idée(s)

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 ToileRecurrence pour afficher la « toile ».

10.2 Commandes

```
Code LATEX
\begin{tikzpicture}[options]
  \ToileRecurrence[clés][options du tracé][options supplémentaires des termes]
\end{tikzpicture}
```

Clés et options

Plusieurs (arguments) (optionnels) sont disponibles :

- le premier argument optionnel définit les (**Clés**) de la commande :
 - la clé $\langle \mathbf{Fct} \rangle$ 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;
 - 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 $\langle AffTermes \rangle$ qui permet d'afficher les termes de la suite sur l'axe $\langle Ox \rangle$. défaut $\langle true \rangle$
 - le deuxième argument optionnel concerne les **(options)** du tracé de l'escalier en langage TikZ;

 - le troisième argument optionnel concerne les **(options)** du tracé des termes en *langage TikZ*.

défaut (dotted).

défaut (thick,color=magenta);

défaut (0)

information(s)

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

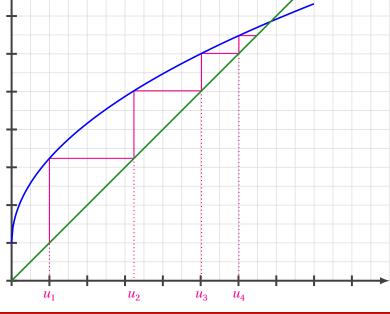
En effet, il y aurait trop d'options pour ces deux éléments, et l'idée est quand même de conserver une commande simple! Donc l'utilisateur se chargera de tracer et de personnaliser sa courbe et sa bissectrice!

10.3 **Exemples**

information(s)

On va tracer la *toile* des 4 premiers termes de la suite récurrente $u_{n+1} = \sqrt{5u_n} + 1$ pour tout entier $n \ge 1$.

Code LATEX %code tikz $\def \x{1.5cm}\def \y{1.5cm}$ $\label{lem:condition} $$ \operatorname{xmin}_{0}\operatorname{xmax}_{10}\operatorname{xgrille}_{1}\operatorname{xgrille}_{0.5}$$ \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 \y in {0,1,...,7} {\draw[darkgray,line width=1.5pt] (4pt,\y) -- (-4pt,\y);} %fonction définie et réutilisable $\left(5*\x)+1 \right)$ %toile \ToileRecurrence[Fct={\f},No=1,Uno=1,Nb=4,DecalLabel=4pt] *%éléments supplémentaires* $\draw[very thick,blue,domain=0:8,samples=250] plot (\x,{\f}) ;$ $\draw[very thick,ForestGreen,domain=0:8,samples=2] plot (\x,\x);$

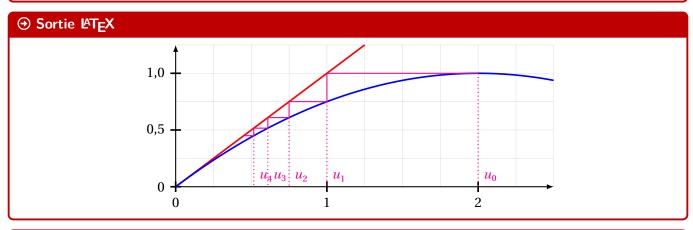


information(s)

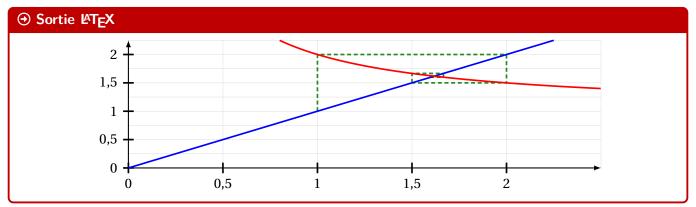
Peut-être que – ultérieurement – des options *booléennes* seront disponibles pour un tracé *générique* de la courbe et de la bissectrice, mais pour le moment la macro ne fait *que* l'escalier.

10.4 Influence des paramètres

```
\begin{center}
\begin{tikzpicture} [x=4cm,y=3cm]
%axes + grilles + graduations
...
%fonction
\def\f{-0.25*\x*\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}
```



```
\rightarrow Code Late:
\text{\text{begin}{center}}
\text{\text{begin}{tikzpicture}[x=5cm,y=1.5cm]}
\text{...}
\text{\text{def}{f1+1/\x}}
\text{\text{ToileRecurrence}%}
\text{\text{[Fct={\f},No=0,Uno=1,Nb=7,PosLabel=above right,DecalLabel=0pt,AffTermes=false]%}
\text{\text{[line width=1.25pt,ForestGreen,densely dashed][]}
\text{\text{\text{draw}[line width=1.25pt,blue,domain=0:2.25,samples=2] plot(\x,{\x});}
\text{\text{\text{\text{draw}[line width=1.25pt,red,domain=0.8:2.5,samples=250] plot(\x,{\f});}
\text{\text{\text{\text{end}{\text{\text{tikzpicture}}}}}
\text{\text{\text{\text{end}{\text{\text{\text{center}}}}}
\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\
```



Cinquième partie

Présentation de codes

11 L'outil « Calcul Formel »

11.1 Introduction

L'idée des commandes suivantes est de définir, dans un environnement $\mathrm{Ti}k\mathrm{Z}$, une présentation proche de celle d'un logiciel de calcul formel comme XCas ou Geogebra.

Les sujets d'examens, depuis quelques années, peuvent comporter des *captures d'écran* de logiciel de calcul formel, l'idée est ici de reproduire, de manière autonome, une telle présentation.

À la manière du package tkz-tab, l'environnement de référence est un environnement TikZ, dans lequel les lignes sont créées petit à petit, à l'aide de nœuds qui peuvent être réutilisés à loisir ultérieurement.

11.2 La commande « CalculFormelParametres »

information(s)

La première chose à définir est l'ensemble des paramètres *globaux* de la fenêtre de calcul formel, à l'aide de **(Clés)**.

⟨/> Code LATEX

```
...
\begin{tikzpicture}[...]
\CalculFormelParametres[options]
...
\end{tikzpicture}
```

— (Largeur) : largeur de l'environnement;

Clés et options

Les (Clés) disponibles sont :

⟨EspaceLg⟩: espacement vertical entre les lignes;
 ⟨PremCol⟩ & ⟨HautPremCol⟩: largeur et hauteur de la case du petit numéro;
 ⟨Taille⟩: taille du texte;
 ⟨Couleur⟩: couleur des traits de l'environnement;
 ⟨Titre⟩: booléen pour l'affichage d'un bandeau de titre;
 ⟨TailleTitre⟩: taille du titre;
 ⟨TailleTitre⟩: taille du titre;

défaut (16)

- ⟨PosCmd⟩: position horizontale de la commande d'entrée; défaut ⟨gauche⟩
- (PosRes) : position horizontale de la commande de sortie; défaut (centre)
- (CouleurCmd) : couleur de la commande d'entrée; défaut (red)
- (CouleurRes) : couleur de la commande de sortie; défaut (blue)
- (Sep): booléen pour l'affichage du trait de séparation E/S;
 (Menu): booléen pour l'affichage du bouton MENU;
 défaut (true)
- (LabelTitre): libellé du titre. défaut (Résultats obtenus avec un logiciel de Calcul Formel)

11.3 La commande « CalculFormelLigne »

information(s)

Une fois les paramètres déclarés, il faut créer les différentes lignes, grâce à la CalculformelLigne.

```
\begin{tikzpicture}[...]
  \CalculFormelParametres[options]
  \CalculFormelLigne[options]{commande}{résultat}
  ...
  \end{tikzpicture}
```

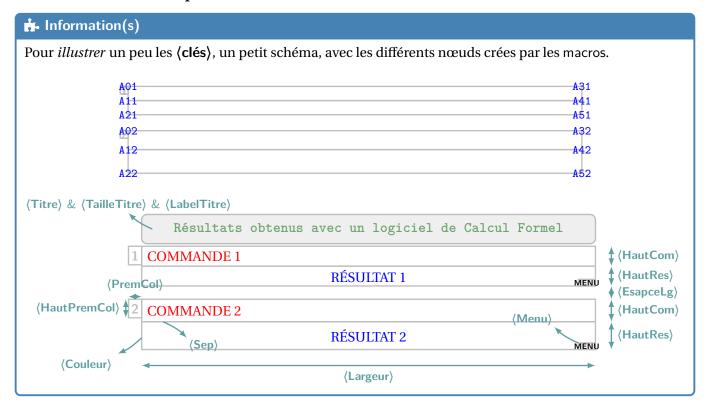
@ Clés et options

Les (quelques) (Clés) disponibles sont :

- (HautCmd) et (HautRes) : hauteur de la ligne de commande d'entrée et de sortie; défaut (0.75)
- deux arguments, celui de la commande d'entrée et celui de la commande de sortie.

Chaque argument COMMANDE & RÉSULTAT peut être formaté (niveau police) de manière indépendante.

11.4 Visualisation des paramètres



12 Code Python « simple » via le package listings

12.1 Introduction

♀ Idée(s)

Le package listings permet d'insérer et de formater du code, notamment du code Python.

En partenariat avec [stcolorbox], on peut donc présenter joliment du code python!

information(s)

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

information(s)

Le style utilisé pour formater le code Python n'est pas modifiable. Il donne un rendu proche de celui des packages comme pythontex ou piton.

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

12.2 Commande et options

♀ Idée(s)

L'environnement CodePythonLst permet de présenter du code python, dans une tcolorbox avec un style particulier.

</r> ⟨/> Code LATEX

\begin{CodePythonLst}(*)[largeur]{commandes tcbox}
...
\end{CodePythonLst}

Clés et options

Plusieurs (arguments) sont disponibles:

- la version étoilée qui permet de ne pas afficher les numéros de lignes;
- le premier argument (optionnel), concerne la ⟨largeur⟩ de la [tcbox]; défaut ⟨\linewidth⟩
- le second argument (*obligatoire*), concerne des **(options)** de la **(eptions)** en *langage tcolorbox*, comme l'alignement.

Attention

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

12.3 Insertion via un fichier « externe »

O Idée(s)

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

Dans ce cas, il n'est pas nécessaire d'aligner le code « à gauche », en utilisant une commande alternative.

Si cette méthode est utilisée, il ne faut oublier de charger le package scontents.

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

12.4 Exemples

```
\login{CodePythonLst}{} %les {}, même vides, sont 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

#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
```

```
\langle CodePythonLst\*[0.5\linewidth]{flush right}
#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
\end{CodePythonLst}
```

```
#largeur de 50%, sans numéro, et aligné à
droite

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

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

```
Code LATEX
\begin{scontents} [overwrite, write-out=testscript.py]
# Calcul de la factorielle en langage Python
def factorielle(x):
  if x < 2:
    return 1
  else:
    return x * factorielle(x-1)
# rapidité de tracé
import matplotlib.pyplot as plt
import time
def trace_parabole_tableaux():
  depart=time.clock()
  X = [] # Initialisation des listes
 Y = \lceil \rceil
  a = -2
  h = 0.001
  while a<2:
    X.append(a) # Ajout des valeurs
    Y.append(a*a) # au "bout" de X et Y
  # 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 LATEX

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

13 Code Python via le package piton

13.1 Introduction

information(s)

2.5.0 Cette section nécessite de charger la librairie piton dans le préambule.

⟨→ Code LATEX

\useproflyclib{piton}

information(s)

[2.0.0] La librairie [piton] (qui charge piton), est compatible uniquement avec Lual [X]) permet d'insérer du code Python avec une coloration syntaxique en utilisant la bibliothèque Lua LPEG.

En partenariat avec [stcolorbox], on peut avoir une présentation de code Python!

💈 2.0.2 Depuis la version 🖺 0.95 de 📳 piton, (left-margin=auto) est disponible et activée dans 🖺 ProfLycee.

2.1.2 Depuis la version 0.99 de piton, (break-lines) est disponible et activée dans ProfLycee.

💈 2.1.3 Depuis la version 🔋 1.0 de 🖺 piton, (tabs-auto-gobble) est disponible et activée dans 🖺 ProfLycee.

Attention

Le package piton nécessite donc obligatoirement l'emploi de Lua MEX!

2.0.1 Ce package n'est chargé que si la compilation détectée est en Lua MEX!

13.2 Présentation de code Python

⟨⟩ Code LATEX

\begin{CodePiton} [options]
...
\end{CodePiton}

@ Clés et options

Plusieurs (clés) sont disponibles :

- la clé booléenne (**Lignes**) pour afficher ou non les numéros de lignes; défaut (**true**)
- 2.1.3 la clé booléenne (**Gobble**) pour activer les options liées au gobble; défaut (**true**)
- la clé (Largeur) qui correspond à la largeur de la ĕtcbox;

défaut (\linewidth)

— la clé (**TaillePolice**) pour la taille des caractères;

défaut (\footnotesize)

— la clé (Alignement) qui paramètre l'alignement de la [tcbox].

défaut (center)

⟨ Code LATEX

```
\begin{CodePiton}
#environnement piton avec numéros de ligne, pleine largeur
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}
```

```
    Sortie LATEX

1 #environnement piton avec numéros de ligne, pleine largeur
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 LATEX
\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=9cm]
    def f(x) :
      return x**2
  \end{CodePiton}
\end{itemize}
```

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

14.1 Librairies

information(s)

🛮 2.5.0 Cette section nécessite de charger les librairies 🖺 minted et/ou 🖺 pythontex dans le préambule.

Code LATEX

\useproflyclib{minted}
\useproflyclib{pythontex}
%ou
\useproflyclib{minted,pythontex}

14.2 Introduction

♀ Idée(s)

🛾 2.5.0 La librairie 🖺 pythontex permet d'insérer et d'exécuter du code Python. On peut :

- présenter du code Python;
- exécuter du code Python dans un environnement type « console »;
- charger du code Python, et éventuellement l'utiliser dans la console.

Attention

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

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

★ Information(s)

Compte tenu de la *relative complexité* pour gérer les options (par paramètres/clés...) des *tcbox* et des *fancy-vrb*, le style est « fixé » tel quel, et seules la taille et la position de la *tcbox* 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 *pré*existante!

14.3 Présentation de code Python grâce au package pythontex

♀ Idée(s)

L'environnement CodePythontex est donc lié à pythontex (chargé par ProfLycee), avec l'option autogobble) permet de présenter du code python, dans une tcolorbox avec un style particulier.

 ⟨ Code LATEX

\begin{CodePythontex} [options]
...

\end{CodePythontex}

Clés et options

Comme précédemment, des (Clés) qui permettent de légèrement modifier le style :

- $\langle Largeur \rangle$: largeur de la tcbox; défaut $\langle linewidth \rangle$
- (**Centre**) : booléen pour centrer ou non la *tcbox*; défaut (**true**)
- ⟨TaillePolice⟩ : taille des caractères; défaut ⟨\footnotesize⟩
- (Espacement Vertical): option (*stretch*) pour l'espacement entre les lignes;
 (Lignes): booléen pour afficher ou non les numéros de ligne.
 défaut (1)
 défaut (true)

```
\rightarrow Code Large Largeur=12cm]
#environnement Python(tex) centré avec numéros de ligne
def f(x):
    return x**2
\end{CodePythontex}
```

Sortie LATEX

```
#environnement Python(tex) centré avec numéros de ligne
def f(x):
return x**2
```

⟨/> Code LATEX

```
\begin{CodePythontex} [Largeur=12cm,Lignes=false,Centre=false]
#environnement Python(tex) non centré sans numéro de ligne
def f(x):
    return x**2
\end{CodePythontex}
```

Sortie LATEX

```
#environnement Python(tex) non centré sans numéro de ligne

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

14.4 Présentation de code Python via le package minted

information(s)

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.

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

Attention

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

⟨/> Code LATEX

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

Clés et options

Plusieurs (arguments) (optionnels) sont disponibles :

- la version étoilée qui permet de ne pas afficher les numéros de lignes;
- le 1^{er} argument *optionnel* concerne la ⟨largeur⟩ de la [tcbox]; défaut ⟨12cm⟩
- le 2nd argument *optionnel* concerne les **(options)** de la **[]** tcbox en *langage tcbox*.

```
\rightarrow Code LATEX

\begin{CodePythonMinted} [12cm] [center]
    #environnement Python(minted) centré avec numéros, de largeur 12cm
    def f(x):
        return x**2
\end{CodePythonMinted}
```

Sortie LATEX

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

⟨→ Code LATEX

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

Sortie L⁴TEX

```
#environnement Python(minted) sans numéro, de largeur 0.8\linewidth

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

14.5 Console d'exécution Python

♀ Idée(s)

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}[options]
...
\end{ConsolePythontex}
```

Clés et options

Les **(Clés)** disponibles sont :

- **(Largeur)**: largeur de la *console*;
- défaut **(\linewidth)**

— **(Centre)**: booléen pour centrer ou non la *console*;

- défaut **(true)**
- (TaillePolice): taille des caractères;
 (EspacementVertical): option (stretch) pour l'espacement entre les lignes;
- défaut ⟨\footnotesize⟩

— **(Label)**: booléen pour afficher ou non le titre.

défaut $\langle 1 \rangle$

défaut (**true**)

\begin{ConsolePythontex} [Largeur=14cm,Centre=false] #console Python(tex) non centrée avec label from math import sqrt 1+1 sqrt(12) \end{ConsolePythontex}

```
    Sortie LATEX
```

```
Début de la console python

>>> #console Python(tex) non centrée avec label

>>> from math import sqrt

>>> 1+1
2

>>> sqrt(12)
3.4641016151377544

Fin de la console python
```

 ⟨ Code LATEX

```
begin{ConsolePythontex} [Largeur=14cm,Label=false]
  #console Python(tex) centrée sans label
  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
>>> 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
[[3, 9, 10, 5, 17, 12], [11, 5, 13, 10, 12, 13], [7, 2, 10, 13, 15, 15]]
>>> len(tableau), len(tableau[0])
(3, 6)
>>> tableau[1][4]
12
```

information(s)

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*.

15 Pseudo-Code

15.1 Introduction

information(s)

Le package listings permet d'insérer et de présenter du code, et avec toolorbox 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)**, car l'environnement toblisting est un peu différent...

15.2 Présentation de Pseudo-Code

♀ Idée(s)

L'environnement PseudoCode permet de présenter du (pseudo-code) dans une tcolorbox.

Attention

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

Clés et options

Plusieurs (arguments) (optionnels) sont disponibles :

- la version étoilée qui permet de ne pas afficher les numéros de lignes;
- le premier argument optionnel concerne la (largeur) de la [tcbox]; défaut (12cm)
- le second argument optionnel concerne les **(options**) de la **[tcbox]** en *langage tcolorbox*. défaut **(vide)**

Code LATEX

```
\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
```

O Sortie La List ← [...] # à déclarer au préalable n ← longueur(List) Pour i allant de 0 à n-1 Faire Afficher(List[i]) FinPour

15.3 Compléments

information(s)

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 définir des mots clés à mettre en valeur car cela reviendrait à *imposer* des choix! Donc ici, pas de coloration syntaxique ou de mise en évidence de mots clés, uniquement un formatage basique de pseudo-code.

♀ Idée(s)

Évidemment, le code source est récupérable et adaptable à volonté, en utilisant les possibilités du package

Celles et ceux qui sont déjà à l'aise avec les packages listings ou minted doivent déjà avoir leur environnement personnel prêt!

Il s'agit ici de présenter une version « clé en main ».

★ Information(s)

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 LEX dans l'environnement Pseudocode (attention au fontes de rendu par contre!).

 ⟨/> Code LATEX

```
\begin{PseudoCode}[12cm][]
#Utilisation du mode mathescape
Afficher (*\og*) ......(*\fg*)
m = (*$\tfrac{\texttt{1}}{\texttt{2}}$*)
\end{PseudoCode}
```

Sortie LATEX

16 Terminal Windows/UNiX/OSX

16.1 Introduction

♀ Idée(s)

L'idée des commandes suivantes est de permettre de simuler des fenêtres de Terminal, que ce soit pour Windows, Ubuntu ou OSX.

L'idée de base vient du package termsim, mais ici la gestion du code et des fenêtres est légèrement différente. Le contenu est géré par le package listings, sans langage particulier, et donc sans coloration syntaxique particulière.

Attention

Comme pour le pseudo-code, pas d'autogobble, donc commandes à aligner à gauche!

16.2 Commandes

```
\hook Code Largeur] \titre=... \ [options]
...
\end{TerminalWin}
\begin{TerminalUnix} [largeur] \titre=... \ [options]
...
\end{TerminalUnix}
\begin{TerminalUnix}
\begin{TerminalOSX} [largeur] \titre=... \ [options]
...
\end{TerminalOSX}
```

Clés et options

Peu d'options pour ces commandes :

- le premier, *optionnel*, est la **(largeur)** de la [stcbox];
- défaut (\linewidth)
- le deuxième, *obligatoire*, permet de spécifier le titre par la clé **\(\frac{\titre}\)**. défaut **\(\Terminal\)**Windows/UNiX/OSX
- le troisième, *optionnel*, concerne les **(options)** de la [stcbox] en langage tcolorbox. défaut **(vide)**

information(s)

Le code n'est pas formaté, ni mis en coloration syntaxique.

De ce fait tous les caractères sont autorisés : même si l'éditeur pourra détecter le % comme le début d'un commentaire, tout sera intégré dans le code mis en forme!

⟨/> Code LATEX

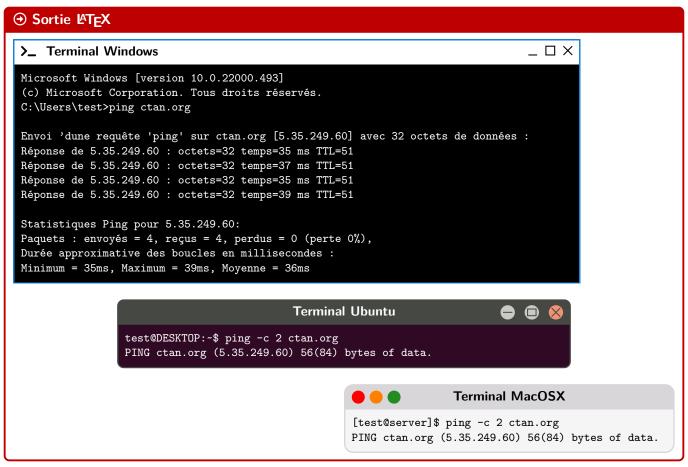
```
\begin{TerminalUnix}[12cm]{Titre=Terminal Ubuntu}[center] %12cm, avec titre modifié et centré test@DESKTOP:~$ ping -c 2 ctan.org
PING ctan.org (5.35.249.60) 56(84) bytes of data.
\end{TerminalUnix}
```

Sortie LATEX

```
Terminal Ubuntu 

test@DESKTOP:~$ ping -c 2 ctan.org
PING ctan.org (5.35.249.60) 56(84) bytes of data.
```

```
Code LATEX
\begin{TerminalWin}[15cm]{} %largeur 15cm avec titre par défaut
Microsoft Windows [version 10.0.22000.493]
(c) Microsoft Corporation. Tous droits réservés.
C:\Users\test>ping ctan.org
Envoi d'une requête 'ping' sur ctan.org [5.35.249.60] avec 32 octets de données :
Réponse de 5.35.249.60 : octets=32 temps=35 ms TTL=51
Réponse de 5.35.249.60 : octets=32 temps=37 ms TTL=51
Réponse de 5.35.249.60 : octets=32 temps=35 ms TTL=51
Réponse de 5.35.249.60 : octets=32 temps=39 ms TTL=51
Statistiques Ping pour 5.35.249.60:
Paquets: envoyés = 4, reçus = 4, perdus = 0 (perte 0%),
Durée approximative des boucles en millisecondes :
Minimum = 35ms, Maximum = 39ms, Moyenne = 36ms
\end{TerminalWin}
\begin{TerminalOSX}[0.5\linewidth]{Titre=Terminal MacOSX}[flush right] %1/2-largeur et titre
→ modifié et droite
[test@server]$ ping -c 2 ctan.org
PING ctan.org (5.35.249.60) 56(84) bytes of data.
\end{TerminalOSX}
```



17 Cartouche Capytale

17.1 Introduction

♀ Idée(s)

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

17.2 Commandes

Code LATEX

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

@ Clés et options

Peu d'options pour ces commandes :

- la version *étoilée* qui permet de passer de la police **(sffamily)** à la police **(ttfamily)**, et donc dépendante des fontes du document;
- le deuxième, *optionnel*, permet de rajouter des caractères après le code (comme un espace); défaut (vide)
- 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

 $\colonerge{ \colonerge{ \colone \colonerge{ \colone \colonerge{ \colone \colonerge{ \colone \colonerge{ \colone \colonerge{ \colonerge{$

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

Sortie LATEX

abcd-12345 *𝚱*

abcd-12345 &

abcd-12345 🔗

abcd-12345 🔗

information(s)

Le cartouche peut être « cliquable » grâce à href.

⟨→ Code LATEX

```
\usepackage{hyperref}
```

\urlstyle{same}

. . .

\href{https://capytale2.ac-paris.fr/web/c/abcd-12345}{\CartoucheCapytale{abcd-12345}}

Sortie LATEX

abcd-12345 8

18 Présentation de code **ETEX**

18.1 Introduction

♀ Idée(s)

<u>2.0.6</u> L'idée est de proposer un environnement pour présenter du code MEX. Ce n'est pas forcément lié à l'enseignement en Lycée mais pourquoi pas!

Il s'agir d'un environnement créé en [stcolorbox], et utilisant la présentation basique de code via [slistings].

18.2 Commandes

```
</> Code Later

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

défaut

Clés et options

Peu de personnalisations pour ces commandes :

- le premier argument, optionnel, permet de préciser la couleur de la présentation;
 (ForestGreen)
- le second, obligatoire, correspond aux éventuelles options liées à la [stcolorbox].

information(s)

Il est à noter que, même dans le cas d'option vide pour la [[tcolorbox], les [[t]]] sont nécessaires.

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

```
⟨/> Code LATEX
\begin{PresentationCode}{}
\xdef\ValAleaA{\fpeval{randint(1,100)}}
\xdef\ValAleaB{\fpeval{randint(1,100)}}
Avec A=ValAleaA et B=ValAleaB, on a A\times B=ValAleaA * ValAleaB
\end{PresentationCode}
\begin{PresentationCode} [DarkBlue] {}
On peut faire beaucoup de choses avec \LaTeX{} !
\end{PresentationCode}
                                                                                    Code ATEX
   \xdef\ValAleaA{\fpeval{randint(1,100)}}
   \xdef\ValAleaB{\fpeval{randint(1,100)}}
  Avec A=ValAleaA et B=ValAleaB, on a A\times B=ValAleaA * ValAleaB
   Avec A = 75 et B = 49, on a A \times B = 3675
                                                                                    Code ATEX
   On peut faire beaucoup de choses avec \LaTeX{} !
   On peut faire beaucoup de choses avec ETEX!
```

Sixième partie

Outils pour la géométrie

19 Pavé droit « simple »

19.1 Introduction

☐ Idée(s)

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

19.2 Commandes

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

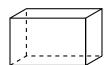
Clés et options

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)**
- (Light Seur): epaisseur des arctes (en unigage simplifie 11kZ);
- (Aff): booléen pour afficher les noms des sommets;
 (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

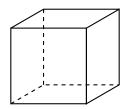
%code tikz
\PaveTikz



Code LATEX

%code tikz

\PaveTikz[Cube,Largeur=2]



information(s)

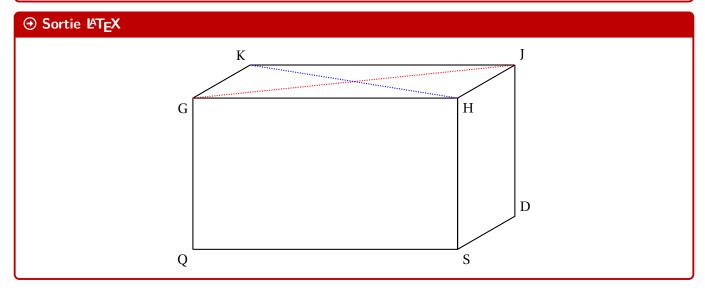
La ligne est de ce fait à insérer dans un environnement TikZ, avec les options au choix pour cet environnement

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!

19.3 Influence des paramètres



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



20 Tétraèdre « simple »

20.1 Introduction

♀ Idée(s)

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

20.2 Commandes

⟨⟩ Code LATEX

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

Clés et options

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)

— $\langle \mathbf{Beta} \rangle$: 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);

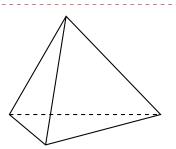
défaut (thick)

— (Aff): booléen pour afficher les noms des sommets; défaut (false)

— (**Plein**) : booléen pour ne pas afficher l'arête *invisible* . défaut (false)

⟨→ Code LATEX

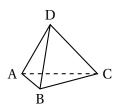
%code tikz \TetraedreTikz

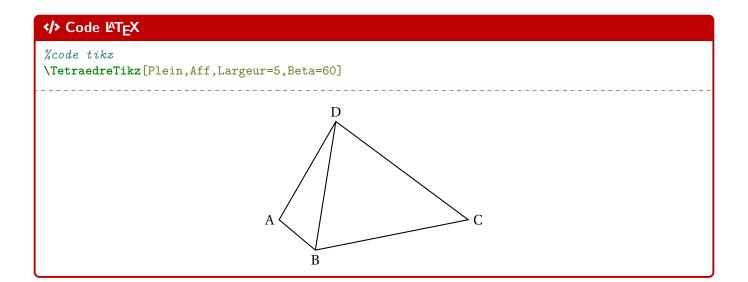


⟨/> Code LATEX

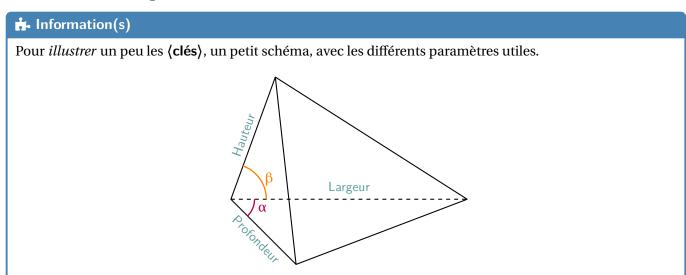
%code tikz

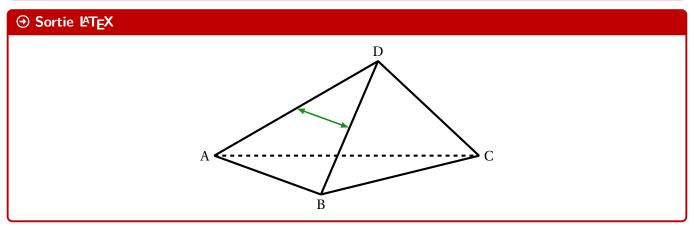
\TetraedreTikz[Aff,Largeur=2,Profondeur=0.625,Hauteur=1.5]





20.3 Influence des paramètres





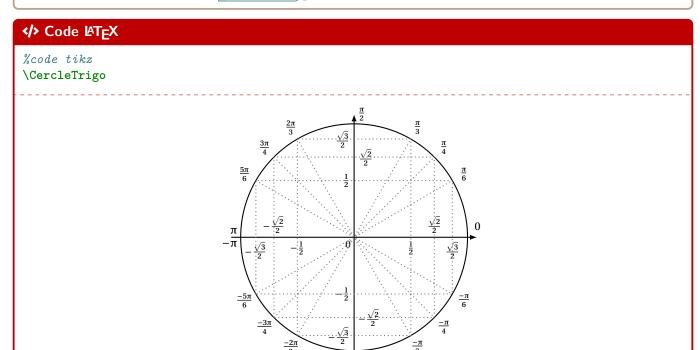
21 Cercle trigo

21.1 Idée

♀ Idée(s)

L'idée est d'obtenir une commande pour tracer (en $\mathrm{Ti}k\mathrm{Z}$) 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.



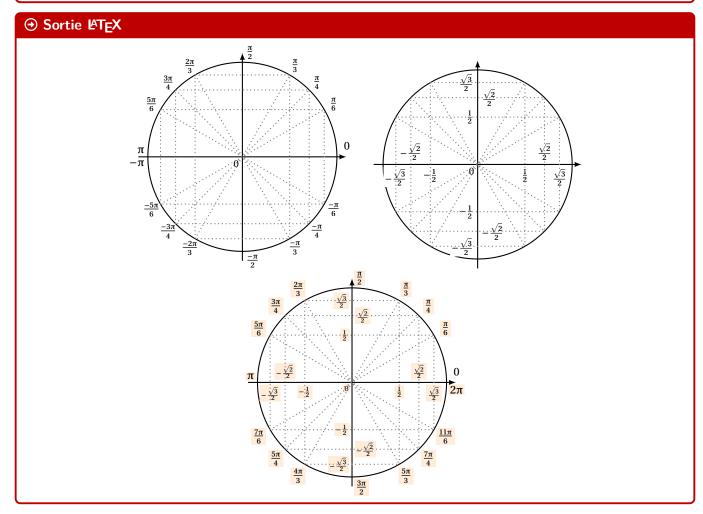
21.2 Commandes

Clés et options

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)

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



21.3 Équations trigos

information(s)

En plus des $\langle Cl\acute{e}s \rangle$ précédentes, il existe un complément pour *visualiser* des solutions d'équations simples du type $\cos(x) = \dots$ ou $\sin(x) = \dots$

Clés et options

Les (Clés) pour cette possibilité sont :

- un booléen (**Equationcos**) pour *activer* « cos = »; défaut (**false**)
- un booléen (**Equationsin**) pour *activer* « sin = »;
- la clé (sin) qui est la valeur de l'angle (en degrés) du sin;
- la clé (cos) qui est la valeur de l'angle (en degrés) cos;
- la clé **(CouleurSol)** qui est la couleur des *solutions*.

défaut **(45)** défaut (**blue**)

défaut (false)

défaut (30)

Code LATEX

\begin{center}

\begin{tikzpicture}

\CercleTrigo[%

AffAngles=false, AffValeurs=false, AffTraits=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}

\medskip

\begin{tikzpicture}

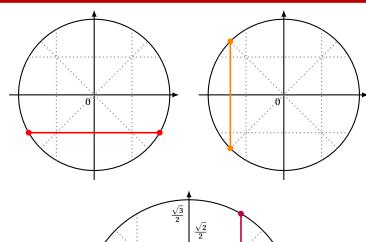
\CercleTrigo[%

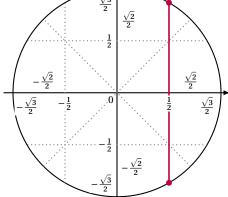
AffTraits=false,AffAngles=false,Rayon=2.75,Equationcos,cos=60,CouleurSol=purple,TailleVvaleurs=\tiny

\end{tikzpicture}

\end{center}

### **⊙** Sortie LATEX





- 60 -

### 22 Style « main levée » en TikZ

### 22.1 Idée

### ♀ Idée(s)

L'idée est de *proposer* un style *tout prêt* pour simuler un tracé, en TikZ, à « main levée ». Il s'agit d'un style *basique* utilisant la librairie decorations avec random steps.

### 22.2 Utilisation basique

### information(s)

Il s'agit ni plus ni moins d'un style TikZ à intégrer dans les tracés et constructions TikZ!

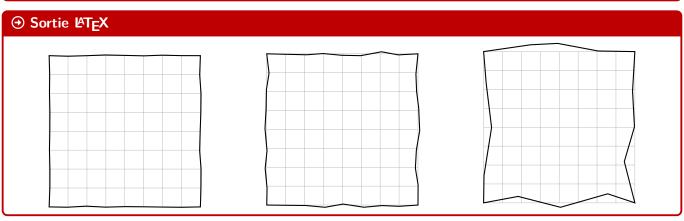
### Clés et options

Concernant le style en lui-même, deux paramètres peuvent être précisés via  $\langle mainlevee = \#1 \ et \ \#2 \rangle$ :

- $-\langle \#1 \rangle$  correspond à l'option segment length (longueur des segments *types*); défaut  $\langle 5mm \rangle$
- $\langle \#2 \rangle$  correspond à l'option amplitude (amplitude maximale de la *déformation*). défaut  $\langle 0.6pt \rangle$

Les valeurs (mainlevee=5mm et 0.6pt) donnent des résultats – à mon sens – satisfaisants, mais l'utilisateur pourra modifier à loisir ces paramètres!

```
% Code LTEX
% la grille a été rajoutée pour la sortie
\text{begin{tikzpicture}}
\draw[thick,mainlevee] (0,0) --++ (4,0) --++ (-4,0) --cycle;
\end{tikzpicture}
\text{begin{tikzpicture}}
\draw[thick,mainlevee=5mm et 2pt] (0,0) --++ (4,0) --++ (-4,0) --cycle;
\end{tikzpicture}
\text{begin{tikzpicture}}
\draw[thick,mainlevee=10mm et 3mm] (0,0) --++ (4,0) --++ (-4,0) --cycle;
\end{tikzpicture}
```



### Septième partie

### Outils pour les statistiques

### Paramètres d'une régression linéaire par la méthode des moindres carrés 23

### 23.1 Idée

### ☐ Idée(s)

L'idée est d'utiliser une commande qui va permettre de calculer les paramètres principaux d'un 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 grafp 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 ,

### information(s)

La commande CalculsRegLin va définir également des macros pour chaque coefficient, qui de ce fait seront réutilisables après!

### 23.2 **Commandes**

### Clés et options

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)
- défaut (COEFFb) — la clé (**NomCoeffb**) qui permet de définir la variable qui contiendra *b*;
- 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  $x_{\text{max}}$ . 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}
```

# %vérif des calculs (noms non modifiables...) Liste des X := \showitems\LX. Liste des Y := \showitems\LY. Somme des X := \LXSomme{} et somme des Y := \LYSomme. Moyenne des X := \LXmoy{} et moyenne des Y := \LYmoy. Variance des X := \LXvar{} et variance des Y := \LYvar{} Covariance des X/Y := \LXYvar. %les coefficients, avec des noms modifiables ! Min des X := \LXmin{} et Max des X := \LXmax. Coefficient \$a=\COEFFa\$. Coefficient \$r=\COEFFr\$. Coefficient \$r^2=\COEFFr\$.

### Sortie LATEX

 $\text{Liste des X} := \boxed{1994} \ \boxed{1995} \ \boxed{1996} \ \boxed{1997} \ \boxed{1998} \ \boxed{1999} \ \boxed{2000} \ \boxed{2001} \ \boxed{2002} \ \boxed{2004} \ \boxed{2005} \ \boxed{2006} \ \boxed{2007} \ \boxed{2008} \ \boxed{2009} \ \boxed{2010} \ .$   $\text{Liste des Y} := \boxed{1718} \ \boxed{1710} \ \boxed{1708} \ \boxed{1700} \ \boxed{1698} \ \boxed{1697} \ \boxed{1691} \ \boxed{1688} \ \boxed{1683} \ \boxed{1679} \ \boxed{1671} \ \boxed{1670} \ \boxed{1663} \ \boxed{1661} \ \boxed{1656} \ \boxed{1649} \ .$ 

Somme des X := 32031 et somme des Y := 26942.

Moyenne des X := 2001.9375 et moyenne des Y := 1683.875.

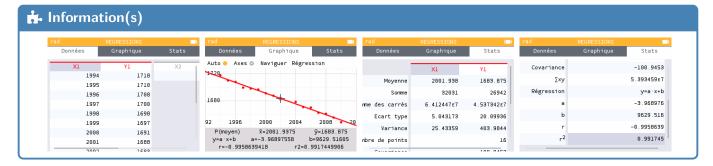
Variance des X := 25.43359375 et variance des Y := 403.984375

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  $r^2 = 0.9917449906486436$ .



### information(s)

Les macros qui contiennent les paramètres de la régression sont donc réutilisables, en tant que nombres réels, donc exploitables par siunitz et signitz et refp 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 0{3} m }{% * pour afficher signe / opt = précision /
  argument = 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=\operatorname{1}{4}{\text{TESTrd}}.
```

### Sortie L⁴TEX

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.

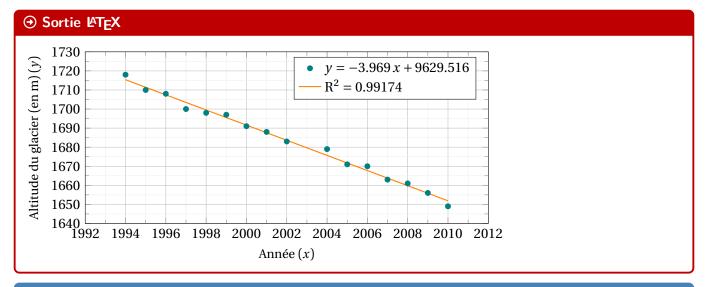
Le coefficient de corrélation linéaire est r = -0.8918, et son carré est  $r^2 = 0.7954$ .



### 23.3 Intégration dans un environnement TikZ

### information(s)

La commande étant « autonome », elle va pouvoir être intégrée dans des environnements graphiques pour permettre un tracé *facile* de la droite de régression.



### information(s)

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}
```

### Clés et options

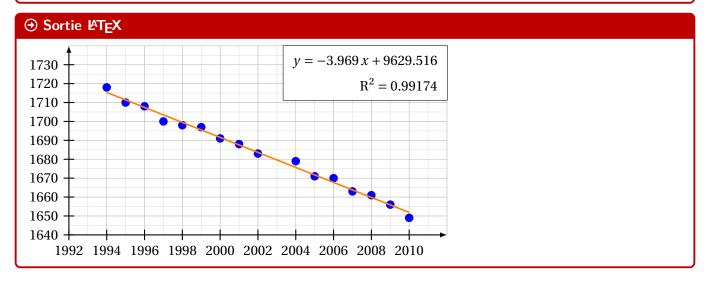
Quelques (Clés) sont disponibles pour cette commande, essentiellement pour la mise en forme du nuage :

- 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 ⟨**O**x⟩ 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) ;
  \frac{\text{draw}[\text{thick}, ->, >= \text{latex}] (0,0) -- (0,100)}{};
  foreach \ in \{1992,1994,...,2010\} \ draw[thick] (\{x-1992\},4pt)--(\{x-1992\},-4pt)
  \rightarrow node[below] \{\$\x\$\};
  \foreach \y in \{1640,1650,...,1730\} \draw[thick] (4pt,\{\y-1640\})--(-4pt,\{\y-1640\})
  \rightarrow 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 \0x{1992}\def \0y{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-\0x\},\{\0EFFa*(\x)+\0EFFb-\0y\});
  \matrix [draw,fill=white,below left] at (current bounding box.north east) {
    \label{eq:cond} $$ \ = \frac{\text{y} = \frac{\text{cound}(\COEFFa,3)}\,x + \frac{\text{cound}(\COEFFb,3)}$}; \ \ $$
    \node {R^2=\left( \COEFFrd, 5)}; \\
  };
\end{tikzpicture}
```



### 24 Statistiques à deux variables

### 24.1 Idées

### ♀ Idée(s)

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 li

- préciser les informations utiles comme [xmin], [xmax], [0x], [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.

### information(s)

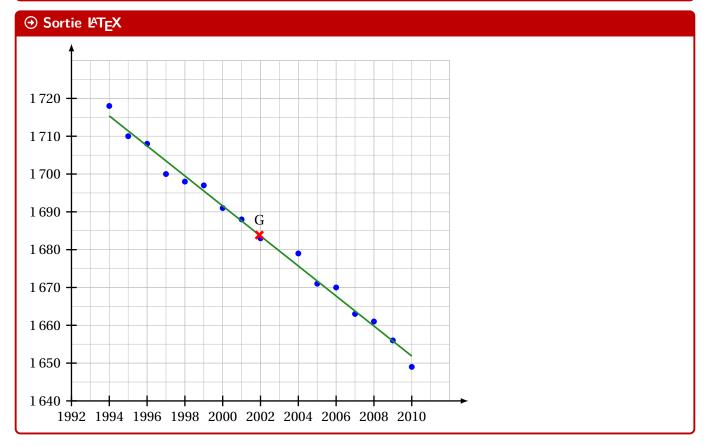
Le package pgfplots peut être utilisé pour traiter ce genre de situation, mais ne l'utilisant pas, j'ai préféré préparer des macros permettant de s'affranchir de ce package (est-ce pertinent, ça c'est une autre question...).

### 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,ForestGreen,samples=2] %
    {\COEFFa*\x+\COEFFb}{\LXmin:\LXmax}
                                                                    %droite de régression
 \PointMoyenTikz
                                                                      %point moyen
\end{tikzpicture}
```



### 24.2 Commandes, clés et options

### information(s)

Les **(paramètres)** nécessaires à la bonne utilisation des commandes suivantes sont à déclarer directement dans l'environnement **[stikzpicture]**, seules les versions «x» sont présentées ici :

- ⟨xmin⟩, stockée dans [⟨xmin⟩; défaut ⟨-3⟩
- ⟨xmax⟩, stockée dans ░ \xmax⟩; défaut ⟨3⟩
- $\langle \mathbf{O} \mathbf{x} \rangle$ , stockée dans  $\langle \mathbf{O} \mathbf{x} \rangle$ , origine de l'axe  $\langle \mathbf{O} \mathbf{x} \rangle$ ; défaut  $\langle \mathbf{O} \mathbf{x} \rangle$
- ⟨**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  $\langle clés \rangle$  pour des réglages fins, mais dans la majorité des cas elles ne sont pas forcément *utiles*.

### information(s)

Pour illustrer les commandes et options de ce paragraphe, la base sera le graphique présenté précédemment.

```
% Code Language
%...code tikz
\GrilleTikz[options][options grille ppale][options grille second.]
```

### Clés et options

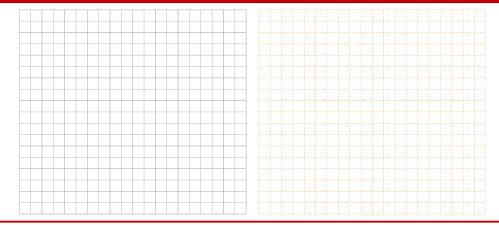
Cette commande permet de tracer une grille principale et/ou une grille secondaire :

- 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

### Sortie LATEX



### </> Code LATEX

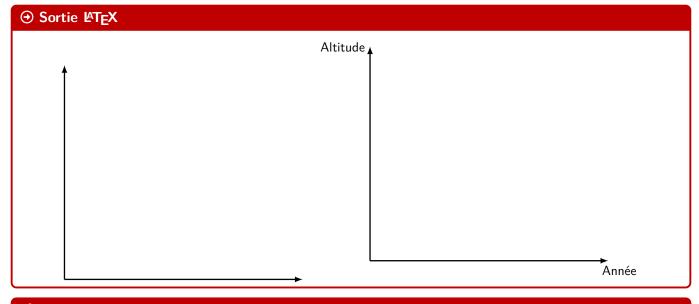
```
%...code tikz
\AxesTikz[options]
```

```
Clés et options
Cette commande permet de tracer les axes, avec des (clés):
   — (Epaisseur) 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)
    — \langle Labelx \rangle qui est le label de l'axe (Ox);
                                                                                                         défaut ($x$)
   — \langle Labely \rangle qui est le label de l'axe (Oy);
                                                                                                         défaut ($y$)
    — \langle AffLabel \rangle qui est le code pour préciser quels labels afficher, entre \langle x \rangle, \langle y \rangle ou \langle xy \rangle;
                                                                                                         défaut (vide)
   — \langle PosLabelx \rangle pour la position du label de (Ox) en bout d'axe;
                                                                                                        défaut (right)
    - (PosLabely) pour la position du label de (O<sub>V</sub>) 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 Lativa
%code tikz
%code tikz
%code tikz
%code tikz
\AxesTikz%
   [AffLabel=xy,Labelx={Année},Labely={Altitude},%
   PosLabelx={below right},PosLabely={above left},%
   Police=\small\sffamily]
```



```
</> Code Late

%...code tikz

\AxexTikz[options] {valeurs}

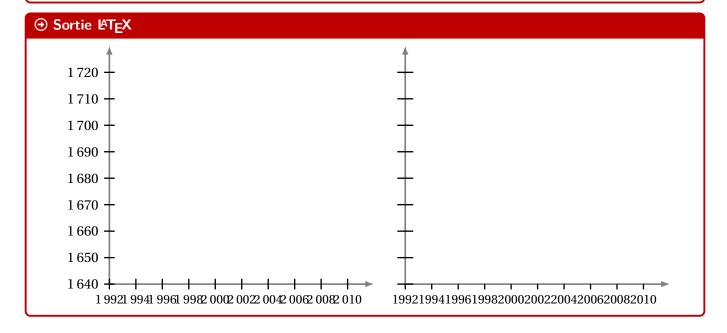
\AxeyTikz[options] {valeurs}
```

### Clés et options

Ces commande permet de tracer les graduations des axes, avec des (clés) identiques pour les deux directions :

- (**Epaisseur**) 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 (lgta/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 Late %code tikz \AxexTikz[Police=\small]{1992,1994,...,2010} \AxexTikz{1640,1650,...,1720} %code tikz \AxeyTikz[Police=\small,Annee,HautGrad=0pt/4pt]{1992,1994,...,2010} \AxeyTikz[AffGrad=false,HautGrad=6pt]{1640,1650,...,1720} %des axes fictifs (en gris) sont rajoutés pour la lisibilité du code de sortie



### 24.3 Commandes annexes

### information(s)

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 LATEX

%code tikz

\FenetreTikz %on restreint les tracés

\FenetreSimpleTikz[opt](opt axes)<opt axe 0x>{liste valx}<opt axe 0y>{liste valy}

### 24.4 Interactions avec CalculsRegLin

### ⟨/> Code LATEX

%...code tikz

\NuagePointsTikz[options]{listeX}{listeY}

### Clés et options

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)

—  $\langle Style \rangle$  parmi  $\langle o \rangle$  (rond) ou  $\langle x \rangle$  (croix) ou  $\langle + \rangle$  (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}

\begin{tikzpicture}[...]

\NuagePointsTikz[Couleur=blue/red]{\LLX}{\LLY}

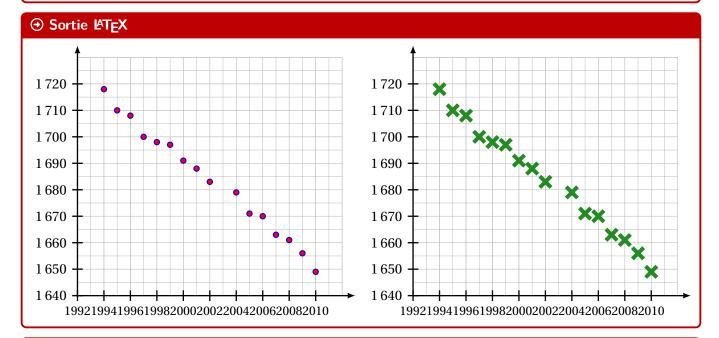
\end{tikzpicture}

~ ~

\begin{tikzpicture}[...]

\NuagePointsTikz[Couleur=ForestGreen,Style=x,Taille=6pt]{\LLX}{\LLY}

\end{tikzpicture}



### Code LATEX

%...code tikz

\PointMoyenTikz[options]

### Clés et 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)

-  $\langle Style \rangle$  parmi  $\langle o \rangle$  (rond) ou  $\langle x \rangle$  (croix) ou  $\langle + \rangle$  (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

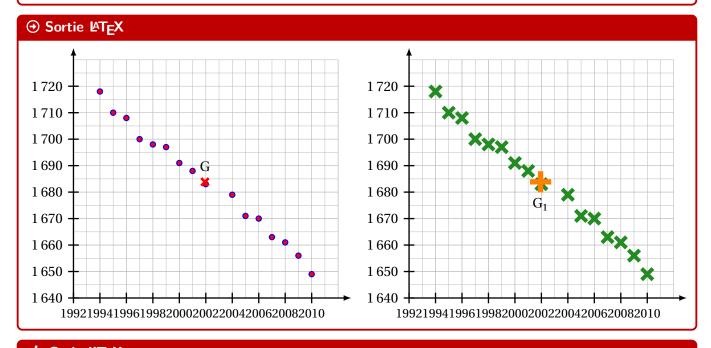
\end{tikzpicture}

\begin{tikzpicture}[...]

\NuagePointsTikz[Couleur=ForestGreen,Style=x,Taille=6pt]{\LLX}{\LLY}

\PointMoyenTikz[Couleur=orange, Taille=8pt, Style=+, Nom={\$G\_1\$}, Pos=below]

\end{tikzpicture}



### Code LATEX

%...code tikz

\CourbeTikz[options] {formule} {domaine}

### Clés et options

Cette commande permet de rajouter une courbe sur le graphique (sans se soucier de la transformation de son expression) avec les arguments :

- $\langle optionnels \rangle$  qui sont en TikZ les paramètres du tracé;
- le premier *obligatoire*, est en langage TikZ l'expression de la fonction à tracer, donc avec  $\sqrt[n]{x}$  comme variable;
- le second *obligatoire* est le domaine du tracé, sous la forme [valxmin:valxmax].

### information(s)

L'idée principale est de récupérer les variables de la régression linéaire pour tracer la droite d'ajustement à *moindres frais*!

#### information(s)

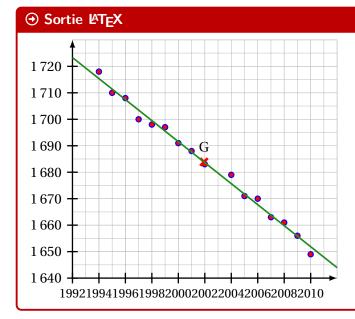
Toute courbe peut être tracée sur ce principe, par contre il faudra saisir la fonction à la main.

#### ⟨→ 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}[...]

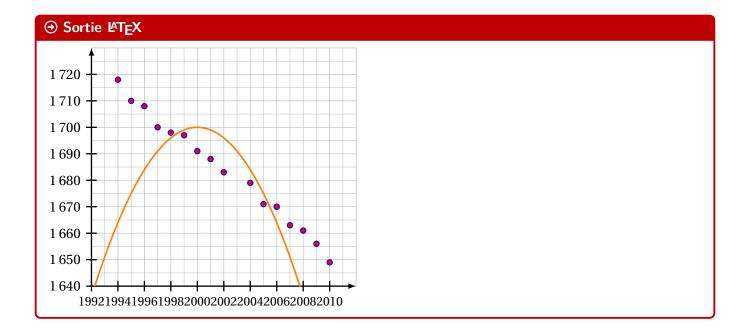
\NuagePointsTikz[Couleur=blue/red]{\LLX}{\LLY} \PointMoyenTikz \CourbeTikz[line width=1.25pt,ForestGreen,samples=2]{\COEFFa\*\x+\COEFFb}{\xmin:\xmax} \end{tikzpicture}



### </> ⟨ 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}

\begin{tikzpicture}[...]



### 25 Boîtes à moustaches

#### 25.1 Introduction

### ♀ Idée(s)

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                                       |         |  |  |
|------------------------------------------------------|---------|--|--|
| \begin{tikzpicture}<br>10/15/17<br>\end{tikzpicture} | /19/20} |  |  |
|                                                      |         |  |  |

### information(s)

Étant donnée que la commande est intégrée dans un environnement Ti*k*Z, les unités peuvent/doivent donc être précisées, *comme d'habitude*, si besoin.

### 25.2 Clés et options

### Clés et options

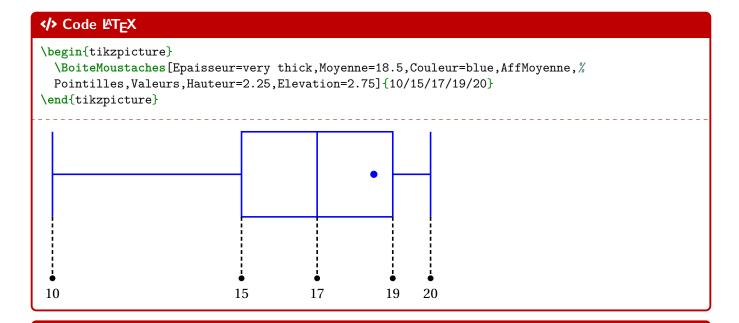
Quelques (clés) sont disponibles pour cette commande :

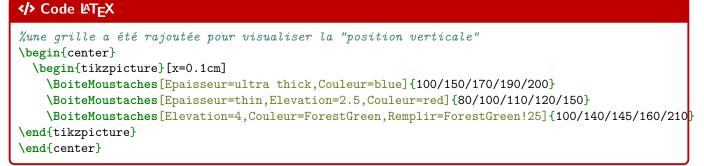
- la clé (**Couleur**) qui est la couleur de la boîte; défaut (**black**)
- la clé (**Elevation**) qui est la position verticale (ordonnée des moustaches) de la boîte; défaut (1.5)
- la clé (**Hauteur**) qui est la hauteur de la boîte;

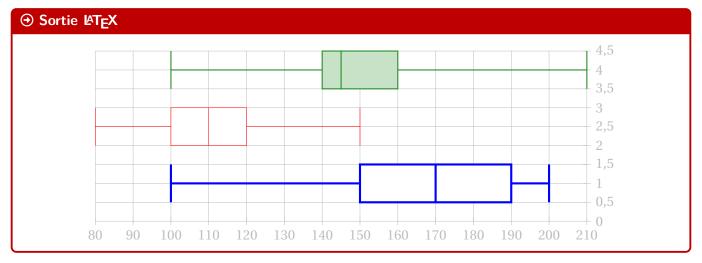
défaut (1)

- 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) défaut (white)
- la clé (**Remplir**) qui est la couleur de remplissage de la boîte;
- 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)



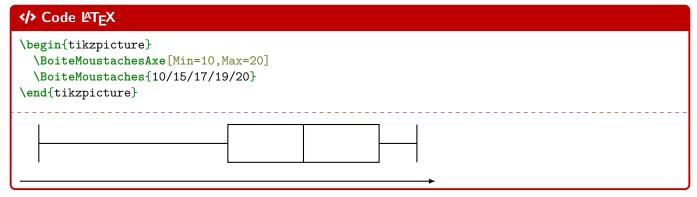


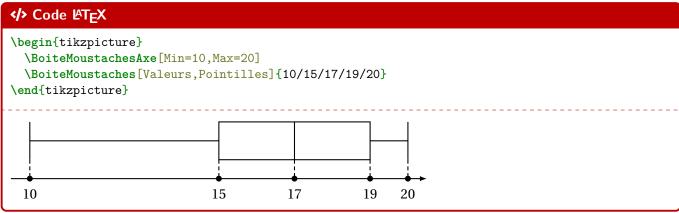


### 25.3 Commande pour placer un axe horizontal

### ♀ Idée(s)

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





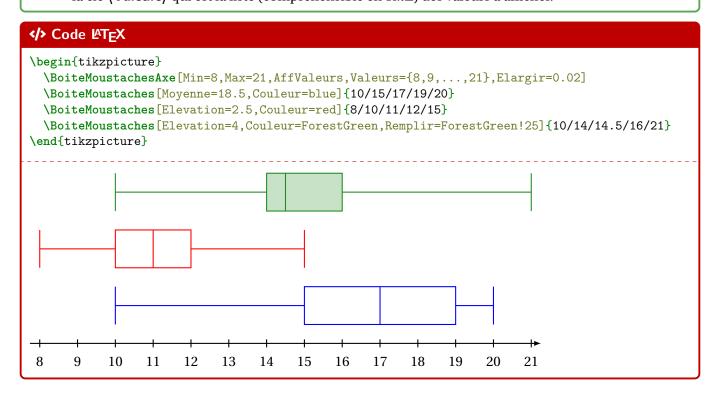
### Clés et options

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 minimale de l'axe horizontal;
- la clé **(Elargir)** qui est le pourcentage l'élargissement de l'axe;

défaut (0.1) défaut (thick)

- la clé **(Epaisseur)** qui est l'épaisseur des traits de la boîte;
- la clé (**Valeurs**) qui est la liste (compréhensible en TikZ) des valeurs à afficher.



## ★ Information(s)

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  $\langle$  élévations $\rangle$  de 1 puis 2,5 puis 4 etc

### Huitième partie

# Outils pour les probabilités

## 26 Calculs de probabilités

#### 26.1 Introduction

### ♀ Idée(s)

L'idée est de proposer des commandes permettant de calculer des probabilités avec des lois classiques :

- binomiale;
- normale;
- exponentielle;
- de Poisson;
- géométrique;
- hypergéométrique.

### information(s)

Les commandes sont de deux natures :

- des commandes pour calculer, grâce au package [xintexpr];
- des commandes pour formater le résultat de printexpr, grâce à siunitx.

De ce fait, les options de siunitx de l'utilisateur affecterons les formatages du résultat, la commande va « forcer » les arrondis et l'écriture scientifique.

### 26.2 Calculs « simples »

```
</i>
✓ Code LATEX
%loi\ binomiale\ B(n,p)
\CalcBinomP{n}{p}{k}
                                     %P(X=k)
\CalcBinomC{n}{p}{a}{b}
                                     %P(a \le X \le b)
%loi de Poisson P (l)
\CalcPoissP{1}{k}
                                     %P(X=k)
\CalcPoissC{1}{a}{b}
                                     %P(a \le X \le b)
%loi géométrique G (p)
\CalcGeomP{p}{k}
                                     %P(X=k)
\CalcGeomC{1}{a}{b}
                                     %P(a \le X \le b)
%loi hypergéométrique H (N,n,m)
\CalcHypergeomP{N}{n}{m}{k}
                                    %P(X=k)
\CalcHypergeomP{N}{n}{m}{a}{b}
                                     %P(a \le X \le b)
%loi\ normale\ N(m,s)
\CalcNormC{m}{s}{a}{b}
                                     %P(a \le X \le b)
%loi\ exponentielle\ E(l)
\CalcExpoC{1}{a}{b}
                                     %P(a \le X \le b)
```

#### Clés et options

Les probabilités calculables sont donc – comme pour beaucoup de modèles de calculatrices – les probabilités **P**onctuelles (P(X = k)) et **C**umulées ( $P(a \le X \le b)$ ).

Pour les probabilités cumulées, on peut utiliser le caractère [\*]\* comme borne (a ou b), pour les probabilités du type  $P(X \le b)$  et  $P(X \ge a)$ .

```
Code LATEX
% X \rightarrow B(5,0.4)
$P(X=3) \approx \CalcBinomP{5}{0.4}{3}$.
$P(X\leqslant1) \approx \CalcBinomC{5}{0.4}{*}{1}$.
% X \rightarrow B(100, 0.02)
$P(X=10) \approx \CalcBinomP{100}{0.02}{10}$.
P(15\leq X\leq X) \simeq CalcBinomC(100)(0.02)(15)(25)
% Y -> P(5)
$P(Y=3) \approx \CalcPoissP{5}{3}$.
$P(Y\geqslant2) \approx \CalcPoissC{5}{2}{*}$.
% T \rightarrow G(0.5)
$P(T=100) \approx \CalcPoissP{0.5}{3}$.
$P(T\leqslant5) \approx \CalcPoissC{0.5}{*}{5}$.
% W \rightarrow H(50, 10, 5)
$P(W=4) \approx \CalcHypergeomP{50}{10}{5}{4}$.
$P(1\leqslant W\leqslant3) \approx \CalcHypergeomC{50}{10}{5}{1}{3}$.
```

```
    Sortie LATEX

• X \hookrightarrow \mathcal{B}(5;0,4):
P(X = 3) \approx 0.2304.
P(X \le 1) \approx 0.33696.
• X \hookrightarrow \mathcal{B}(100;0,02):
P(X = 10) \approx 0.00002877077765846743.
P(15 \le X \le 25) \approx 0.000000001670210428685021.
• Y \hookrightarrow \mathscr{P}_5:
P(Y = 3) \approx 0.1403738958142806.
P(Y \ge 2) \approx 0.9595723180054873.
• T \hookrightarrow \mathcal{G}_{0.5}:
P(T = 3) \approx 0.125.
P(T \le 5) \approx 0.96875.
• W \hookrightarrow \mathcal{H}(50;10;5):
P(W = 4) \approx 0.003964583058015065.
P(1 \le W \le 3) \approx 0.6853536974456758.
```

```
% Code Lamber Code Lamber
```

## Sortie L⁴TEX

•  $X \hookrightarrow \mathcal{N}(0;1)$ :

 $P(X \le 1) \approx 0.841344680841397.$ 

 $P(-1,96 \le Z \le 1,96) \approx 0.9500039553976748.$ 

•  $Y \hookrightarrow \mathcal{N}(550;30)$ :

 $P(Y \ge 600) \approx 0.0477903462453939.$ 

 $P(500 \le Y \le 600) \approx 0.9044193075092122.$ 

•  $Z \hookrightarrow \mathcal{E}_{0.001}$ :

 $P(Z \ge 400) \approx 0.6703200460356393.$ 

 $P(300 \le Z \le 750) \approx 0.2684516679407032.$ 

### 26.3 Complément avec sortie « formatée »

### ♀ Idée(s)

L'idée est ensuite de formater le résultat obtenu par sintexpr, 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}}

### **⊙** Sortie LATEX

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

 $P(X = 10) \approx 2,877077765846743 \times 10^{-5}.$ 

### ♀ Idée(s)

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 :  $n, ... \times 10^{...}$ .

```
Code LATEX
%loi\ binomiale\ B(n,p)
\mathbb{P}(*)[prec]{n}{p}{k}
                                     %P(X=k)
\BinomC(*)[prec]{n}{p}{a}{b}
                                     %P(a \le X \le b)
%loi de Poisson P (l)
\PoissonP(*)[prec]{1}{k}
                                       %P(X=k)
\PoissonC(*)[prec]{1}{a}{b}
                                       %P(a \le X \le b)
%loi géométrique G (p)
\mathbb{Q}_{p}_{k}
                                     %P(X=k)
\GeomC{1}{a}{b}
                                     %P(a \le X \le b)
%loi hypergéométrique H (N,n,m)
                                     %P(X=k)
\HypergeomP{N}{n}{m}{k}
\HypergeomC{N}{n}{m}{a}{b}
                                     %P(a \le X \le b)
%loi normale N(m,s)
\NormaleC(*)[prec]{m}{s}{a}{b}
                                         %P(a \le X \le b)
%loi\ exponentielle\ E(l)
\ExpoC(*)[prec]{1}{a}{b}
                                     %P(a \le X \le b)
```

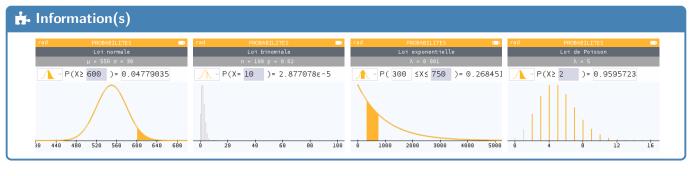
#### Clés et options

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 \rightarrow N(550,30)
$P(Y\geqslant600) \approx \NormaleC[4]{550}{30}{600}{*}.
$P(500\leqslant Y\leqslant600) \approx \NormaleC[4]{550}{30}{500}{600}$.
% X \rightarrow B(100, 0.02)
$P(X=10) \approx \BinomP[7]{100}{0.02}{10} \approx \BinomP*[7]{100}{0.02}{10}$.
$P(15\leqslant X\leqslant25) \approx \BinomC[10]{100}{0.02}{15}{25} \approx
\rightarrow \BinomC*[10]{100}{0.02}{15}{25}$.
% H \rightarrow H(50, 10, 5)
$P(W=4) \approx \HypergeomP[5]{50}{10}{5}{4}$.
$P(1\leqslant W\leqslant3) \approx \HypergeomC[4]{50}{10}{5}{1}}{3}<b>$.
% Z \rightarrow E(0,001)$:
$P(Z\geqslant400) \approx \ExpoC{0.001}{400}{*}$.
$P(300\leqslant Z\leqslant750) \approx \ExpoC{0.001}{300}{750}$.
% T \rightarrow P(5)
P(T=3) \sim \Pr(5){3}
P(T\geq 1) \simeq PoissonC[4]{5}{2}{*}
```

## **⊙** Sortie LATEX • $Y \hookrightarrow \mathcal{N}(550;30)$ : $P(Y \ge 600) \approx 0.0478$ . $P(500 \le Y \le 600) \approx 0,9044.$ • $X \hookrightarrow \mathcal{B}(100;0,02)$ : $P(X = 10) \approx 0,0000288 \approx 2,88 \times 10^{-5}$ . $P(15 \le X \le 25) \approx 0,000000017 \approx 1,7 \times 10^{-9}.$ • $W \hookrightarrow \mathcal{H}(50;10;5)$ : $P(W = 4) \approx 0.00396$ . $P(1 \le W \le 3) \approx 0.6854.$ • $Z \hookrightarrow \mathcal{E}_{0,001}$ : $P(Z \ge 400) \approx 0.670$ . $P(300 \le Z \le 750) \approx 0.268.$ • $T \hookrightarrow \mathscr{P}_5$ : $P(T = 3) \approx 0.140.$ $P(T \ge 2) \approx 0.9596$ .



#### **27** Arbres de probabilités « classiques »

#### 27.1 Introduction

### ♀ Idée(s)

L'idée est de proposer des commandes pour créer des arbres de probabilités classiques (et homogènes), en TikZ, de format :

- $-2 \times 2$  ou  $2 \times 3$ ;
- $-3 \times 2$  ou  $3 \times 3$ .

Les (deux) commandes sont donc liées à un environnement \(\begin{align\*}\) 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}
```

#### 27.2 **Options et arguments**

#### information(s)

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:

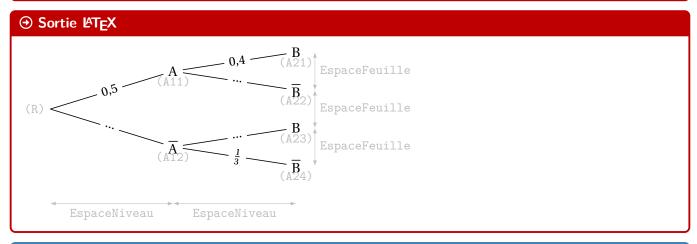
- une donnée (proba) peut être laissée vide;
- une donnée (position) peut valoir (above) (au-dessus), (below) (en-dessous) ou être laissée (vide) (sur).

#### Clés et options

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)

**①** [ProfLycee] - 85 -



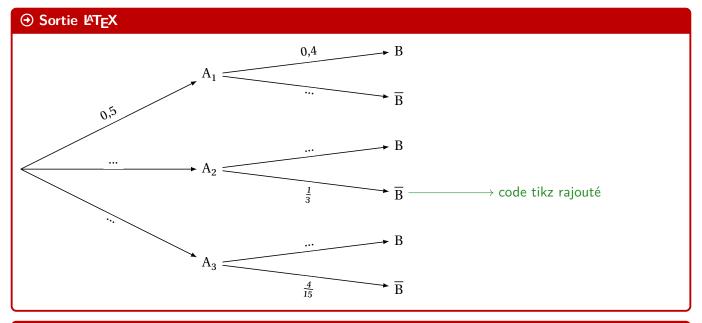
#### information(s)

Les nœuds crées par les commandes sont :

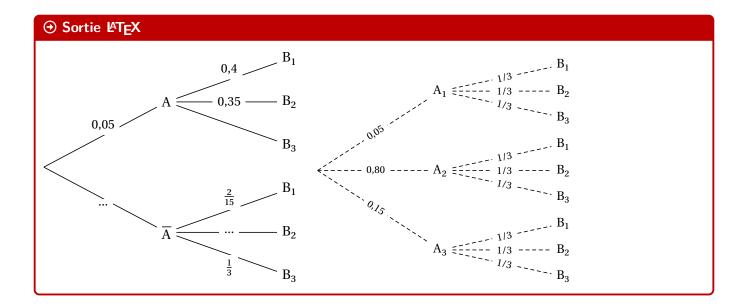
- FR pour la racine;
- FAIR pour les nœuds du 1er niveau (de haut en bas);
- PA2x pour les nœuds du 2<sup>d</sup> niveau (de haut en bas).

### 27.3 Exemples complémentaires

```
\Code LTEX
\def\ArbreTroisDeux{
    $A_1$\/\num{0.5}\/\above,
    $B$\/\num{0.4}\/\above,
    $\overline{B}$\/\...\/\below,
    $A_2$\/\...\/\above,
    $\overline{B}$\/\...\/\above,
    $\overline{B}$\/\s\frac{1}{3}$\/\below,
    $A_3$\/\...\/\above,
    $\overline{B}$\/\s\frac{1}{3}$\/\below,
    $A_3$\/\...\/\below,
    $\B$\/\...\/\above,
    $\overline{B}$\/\s\frac{4}{15}$\/\below
}
\begin{EnvArbreProbasTikz} [Type=3x2,Fleche,EspaceNiveau=5,EspaceFeuille=1.25]{\ArbreTroisDeux}\\draw[ForestGreen,->] (A24)--($(A24)+(2.5,0)$) node[right,font=\sffamily] {code tikz
    \_\rangle rajouté};
\end{EnvArbreProbasTikz}
```



```
⟨→ Code LATEX
\def\ArbreDeuxTrois{
 A^{0.05}/above
   B_1^{\infty}
   B_2\/ num\{0.35\}/
   $B_3$//below,
 $\overline{A}$/.../below,
   $B_1$/$\frac{2}{15}$/above,
   $B_2$/.../,
   $B_3$/$\frac{1}{3}$/below
\ArbreProbasTikz[Type=2x3,InclineProbas=false,EspaceFeuille=1.15]{\ArbreDeuxTrois}
\def\ArbreTroisTrois{
 A_1/\num{0.05}/,
   $B_1$/{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}
```



#### **28** Petits schémas pour des probabilités continues

#### 28.1 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 > {1}{a}{b}

### → Sortie LATEX





### Clés et options

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

On peut utiliser  $\ast$  comme borne (a ou b), pour les probabilités du type  $P(X \le b)$  et  $P(X \ge a)$ .

#### 28.2 Commandes et options

#### Clés 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)** 

### information(s)

Les commandes sont donc des environnements TikZ, sans possibilité de « rajouter » des éléments. Ces petis schémas sont donc vraiment dédiés à montrer rapidement une probabilité continue, sans fioriture.

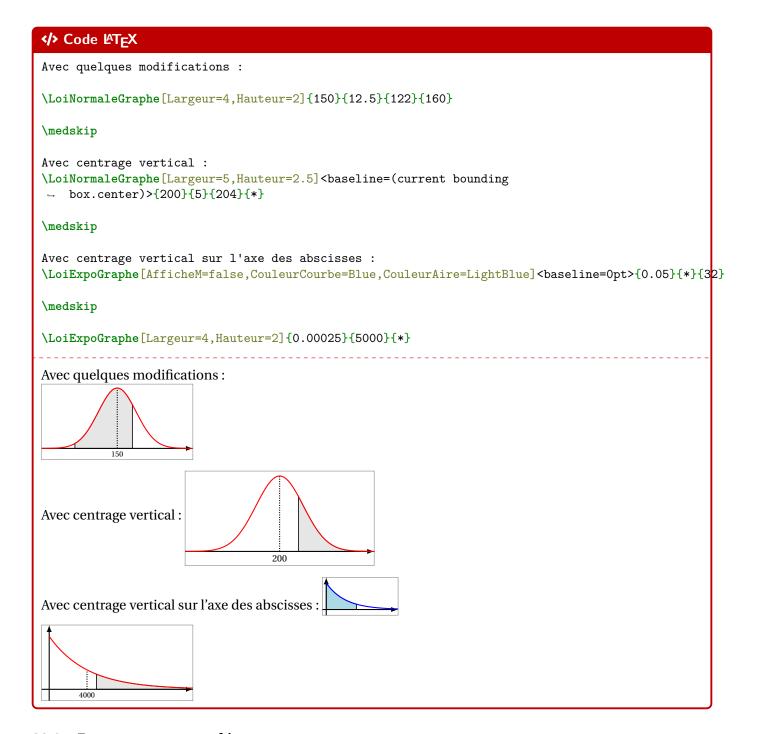
#### ⟨/> Code LATEX

Avec centrage vertical sur l'axe des abscisses :

\LoiNormaleGraphe [AfficheM=false, CouleurCourbe=Blue, CouleurAire=LightBlue] <br/> | CouleurAire=Det | CouleurAire | CouleurAir

Avec centrage vertical sur l'axe des abscisses :





### 28.3 Remarques et compléments

#### information(s)

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...

#### 29 Nombres aléatoires

#### 29.1 Idée

#### ♀ Idée(s)

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.

### information(s)

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

#### information(s)

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

# </b> ✓ Code LATEX

Liste générée : 45,31,24,21,42,18 Liste traitée : 45 31 24 21 42 18

#### 29.2 Clés et options

### 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;
- la clé **(Tri)** parmi **(non/croissant/decroissant)** pour trier les valeurs; défaut (non)
- le booléen (**Repetition**) pour autoriser la répétition d'éléments.

défaut (false)

défaut (,)

### Code 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 : \\
    \TirageAleatoireEntiers[ValMin=1, ValMax=50, NbVal=12, Tri=decroissant] {\MaListeC}\MaListeC
   - / /
15 tirages de dé à 6 faces : \\
   \TirageAleatoireEntiers[ValMin=1, ValMax=6, NbVal=15, Repetition] {\TestDes}\TestDes
Une liste de 15 valeurs (différentes), entre 10 et 100, stockée dans la macro MaListeA:
Liste: 73,68,24,32,22,15,65,23,62,13,69,26,10,75,18
Une liste de 12 valeurs (différentes), entre 1 et 50, ordre croissant :
Liste: 2,6,11,13,19,23,24,28,31,36,41,49
Une liste de 12 valeurs (différentes), entre 1 et 50, ordre décroissant :
Liste: 48,44,41,37,36,35,30,29,27,20,12,4
15 tirages de dé à 6 faces :
```

#### ⟨/> Code LATEX

5,2,5,6,5,2,3,4,5,5,3,1,6,4,4

```
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 - :
5 - 16 - 33 - 39 - 47 - 52 - 55 - 57 - 62 - 63
5 16 33 39 47 52 55 57 62 63
```

### 30 Combinatoire

#### 30.1 Idée

### ♀ Idée(s)

L'idée est de proposer une commande pour calculer un arrangement ou une combinaison, en utilisant les capacités de calcul du package **xfp**.

### Code LATEX

\Arrangement(\*)[option]{p}{n} \Combinaison(\*)[option]{p}{n}

#### 30.2 Utilisation

### Clés et options

Peu de paramétrage pour ces commandes qui permettent de calculer  $A_n^p$  et  $C_n^p$ :

- les versions étoilées ne formatent pas le résultat grâce à \[
  \text{\text{\left} \num de } \text{\text{\sinuitx}};
  \]
- le booléen (**Notation**) pour avoir la notation au début;

défaut (false)

— le booléen (**Formule**) permet de présenter la formule avant le résultat;

défaut (false)

- le premier argument, *obligatoire*, est la valeur de p;
- le second argument, *obligatoire*, est la valeur de *n*.

### ⟨/> Code LATEX

On a  $A_{20}^3=\Lambda \times {3}_{20}$  en non formaté, et  $\Lambda \times {3}_{20}$  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

On a  $\$  en non formaté,~ et  $\$  en formaté avec la notation au début.

On a  $\binom{20}{3}$  = 1140 en non formaté, et  $\binom{20}{3}$  = 1140 en formaté avec la notation au début.

### Code LATEX

On a \$\displaystyle\Arrangement[Notation,Formule]{3}{20}\$.

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

### ⟨/> Code LATEX

On a \$\displaystyle\Combinaison[NotationAncien,Formule]{3}{20}\$. %ancienne notation FR

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

### Neuvième partie

# Outils pour l'arithmétique

### 31 Conversions binaire/hexadécimal/décimal

#### 31.1 Idée

### □ Idée(s)

L'idée est de *compléter* les possibilités offertes par le package <u>mintbinhex</u>, en mettant en forme quelques conversions :

- décimal en binaire avec blocs de 4 chiffres en sortie;
- conversion binaire ou hexadécimal en décimal avec écriture polynomiale.

### information(s)

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

#### 31.2 Conversion décimal vers binaire

### Code LATEX

\ConversionDecBin(\*)[clés]{nombre}

#### Clés et options

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.

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

```
Sortie LATEX
415<sub>10</sub> = 0001 1001 1111<sub>2</sub>
415<sub>10</sub> = 11001 1111<sub>2</sub>
415 = 0001 1001 1111
```

#### 31.3 Conversion binaire vers hexadécimal

### information(s)

415 = 1100111111

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}

#### Clés et options

Quelques (clés) sont disponibles pour cette commande :

- le booléen (AffBase) qui permet d'afficher ou non la base des nombres; défaut (true)
- le booléen (**Details**) qui permet d'afficher ou le détail par bloc de 4. défaut (**true**)

Le formatage est géré par le package sinuitx, le mieux est de positionner la commande dans un environnement mathématique.

#### ⟨ Code LATEX

```
%conversion avec détails et affichage de la base
$\ConversionBinHex{110011111}$

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

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

```
⊙ Sortie LATEX
```

```
110011111_{2} = 000110011111 = \underbrace{000110011111}_{1} = 19F_{16}
110011111_{2} = 19F_{16}
110011111 = 19F
```

#### 31.4 Conversion binaire ou hexadécimal en décimal

### information(s)

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}

#### Clés et options

Quelques (clés) sont disponibles pour cette commande :

- la clé (BaseDep) qui est la base de départ (2 ou 16!); défaut (2)
- le booléen (AffBase) qui permet d'afficher ou non la base des nombres; défaut (true)
- le booléen (**Details**) qui permet d'afficher ou le détail par bloc de 4; défaut (**true**)
- le booléen (**Zeros**) qui affiche les chiffres 0 dans la somme. défaut (**true**)

Le formatage est toujours géré par le package sinuitx, le mieux est de positionner la commande dans un environnement mathématique.

### </> Code LATEX

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

### **⊙** Sortie LATEX

```
\begin{split} &19F_{16} = 1\times16^2 + 9\times16^1 + 15\times16^0 = 415_{10} \\ &11\,0011_2 = 1\times2^5 + 1\times2^4 + 0\times2^3 + 0\times2^2 + 1\times2^1 + 1\times2^0 = 51_{10} \\ &11\,0011_2 = 1\times2^5 + 1\times2^4 + 1\times2^1 + 1\times2^0 = 51_{10} \\ &AC0DC_{16} = 704\,732_{10} \\ &AC0DC_{16} = 10\times16^4 + 12\times16^3 + 13\times16^1 + 12\times16^0 = 704\,732_{10} \end{split}
```

### 32 Conversion « présentée » d'un nombre en base décimale

#### 32.1 Idée

### ♀ Idée(s)

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 overlay, donc on pourra déclarer – si ce n'est pas fait – dans le préambule, la commande qui suit.

```
</> Code LATEX

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

#### 32.2 Code et clés

#### information(s)

La «tableau», qui est géré par paray est inséré dans un persurement, donc les structures.

#### ⟨/> Code LATEX

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

#### Clés et options

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  $\langle Noeud \rangle$  pour le préfixe du nœud du premier et du dernier reste (pour utilisation en TikZ); défaut  $\langle EEE \rangle$
- 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 ETEX
%conversion avec changement de couleur
\ConversionDepuisBaseDix[Couleur=DarkBlue] {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=Goldenrod,DecalH=6pt/2pt] {1012} {16}

%conversion avec nœud personnalisé et réutilisation
\ConversionDepuisBaseDix[Couleur=ForestGreen,CouleurRes,Noeud=TEST] {100} {9}.

\begin{tikzpicture}
\draw[overlay,ForestGreen,thick,->] (TEST2.south east) to[bend right] ++ (3cm,-1cm)
\cup node[right] {test};
\end{tikzpicture}
```

→ test

- 99 -

## Sortie LATEX

$$\begin{cases} 45 = 2 \times 22 + 1 \\ 22 = 2 \times 11 + 0 \\ 11 = 2 \times 5 + 1 \\ 5 = 2 \times 2 + 1 \\ 2 = 2 \times 1 + 0 \\ 1 = 2 \times 0 + 1 \end{cases} \Rightarrow 45_{10} = 101101_{2}$$

Par divisions euclidiennes successives,  $\begin{cases} 54 = 3 \times 18 + 0 \\ 18 = 3 \times 6 + 0 \\ 6 = 3 \times 2 + 0 \\ 2 = 3 \times 0 + 2 \end{cases} \Rightarrow 54_{10} = 2000_3.$ 

$$\begin{cases} 1012 = 16 \times 63 + 4 \\ 63 = 16 \times 3 + 15 \\ 3 = 16 \times 0 + 3 \end{cases} \Rightarrow 1012_{10} = 3F4_{16}$$

On obtient donc 
$$\begin{cases} 100 = 9 \times 11 + 1 \\ 11 = 9 \times 1 + 2 \\ 1 = 9 \times 0 + 1 \end{cases} \Rightarrow 100_{10} = 121_9.$$

**①** 

### 33 Algorithme d'Euclide pour le PGCD

#### 33.1 Idée

### ♀ Idée(s)

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* avec alignement des restes.

### Code LATEX

\PresentationPGCD[options]{a}{b}

### Code LATEX

```
...
\tikzstyle{every picture}+=[remember picture]
...
\PresentationPGCD{150}{27}
```

### Sortie LATEX

```
\begin{cases}
150 = 27 \times 5 + 15 \\
27 = 15 \times 1 + 12 \\
15 = 12 \times 1 + 3 \\
12 = 3 \times 4 + 0
\end{cases} \Rightarrow PGCD(150; 27) = 3
```

#### Attention

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]

#### 33.2 Options et clés

### Clés et options

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 ne 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 [st...s] ne sont pas indispensables.

# </b> ✓ Code LATEX

\PresentationPGCD[CouleurResultat]{150}{27}

$$\begin{cases} 150 = 27 \times 5 + 15 \\ 27 = 15 \times 1 + 12 \\ 15 = 12 \times 1 + 3 \\ 12 = 3 \times 4 + 0 \end{cases} \Rightarrow PGCD(150; 27) = 3$$

```
Code LATEX
\PresentationPGCD[CouleurResultat,Couleur=ForestGreen] {1250} {450}.
\PresentationPGCD[CouleurResultat,Couleur=DarkBlue] {13500}{2500}.
\PresentationPGCD[Rectangle=false]{420}{540}.
\medskip
D'après l'algorithme d'Euclide, on a $\left|
- \PresentationPGCD[Couleur=LightSkyBlue, AfficheConclusion=false, AfficheDelimiteurs=false] {123456789}{

    \right.$
\begin{tikzpicture}
  \draw[overlay,LightSkyBlue,thick,<-] (FFF1.east) to[bend right] ++ (2cm,0.75cm) node[right]
  \end{tikzpicture}
  1250 = 450 \times 2 + 350
   450 = 350 \times 1 + 100
                        \Rightarrow PGCD (1250; 450) = 50.
   350 = 100 \times 3 + (50)
  100 = 50 \times 2 + 0
 13500 = 2500 \times 5 + 1000
   2500 = 1000 \times 2 + (500) \Rightarrow PGCD(13500; 2500) = 500.
  1000 = 500 \times 2 +
 420 = 540 \times 0 + 420
 540 = 420 \times 1 + 120
                      \Rightarrow PGCD (420; 540) = 60.
  420 = 120 \times 3 + 60
120 = 60 \times 2 + 0
                                    123456789 = 9876 \times 12500 + 6789
                                          9876 = 6789 \times 1
                                                               +3087
                                          6789 = 3087 \times 2
                                                               + 615
D'après l'algorithme d'Euclide, on a
                                          3087 = 615 \times 5
                                                              + 12
                                                                    3
                                           615 = 12 \times 51
                                             12 = 3 \times 4
                                                                     0
```

### 33.3 Compléments

#### information(s)

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 **(Affi-cheDelimiteurs=false)** permet de choisir des délimiteurs différents.

#### ⟨⟩ Code LATEX

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

$$\begin{bmatrix} 1234 = 5 \times 246 + 4 \\ 5 = 4 \times 1 & + 1 \\ 4 = 1 \times 4 & + 0 \end{bmatrix}$$

## Dixième partie

# Écritures, simplifications

## 34 Simplification sous forme d'une fractions

#### 34.1 Idée

### 

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

### ⟨→ Code LATEX

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

#### 34.2 Commande et options

### Clés 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 :
  - $\langle \mathbf{t} \rangle$  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!);
  - $\langle \mathbf{dec} = \mathbf{k} \rangle$  pour l'affichage du résultat en mode décimal arrondi à  $10^{-k}$ ;
- le second argument, *obligatoire*, est quant à lui, le calcul en syntaxe xint.

À noter que la macro est dans un bloc ensuremath donc les st...s ne sont pas nécessaires.

### ⟨/> Code LATEX

```
\label{lem:conversionFraction} $$ \conversionFraction {-10+1/3*(-5/16)} $$ %sortie par défaut $$ \conversionFraction {-10+1/3*(-5/16)} $$ %sortie fraction avec - sur numérateur $$ \conversionFraction [d] {-10+1/3*(-5/16)} $$ %sortie en displaystyle $$ \conversionFraction [dec=4] {-10+1/3*(-5/16)} $$ %sortie en nicefrac $$ \conversionFraction {-10+1/3*(-5/16)} $$ %sortie en décimal arrondi à 0,0001 $$ \conversionFraction {2+91/7} $$ %entier formaté $$ \conversionFraction {111/2145} $$ \conversionFraction {111/3}$
```

#### **⊙** Sortie LATEX

```
\begin{array}{r}
-\frac{485}{48} \\
-485 \\
48 \\
-\frac{485}{48} \\
-\frac{485}{48} \\
-10,1042 \\
15 \\
37 \\
-\frac{37}{48}
\end{array}
```

37

```
Code LATEX
\frac{111}{2145}=\ ConversionFraction{111/2145}
\smallskip
\frac{3}{15}=\ConversionFraction[]{3/15}
\smallskip
\frac{3}{15}=\conversionFraction[t]{3/15}
\smallskip
$\dfrac{3}{15}=\ConversionFraction[d]{3/15}$
\smallskip
\frac{0,42}{0,015}=\conversionFraction[d] {0.42/0.015}}
\smallskip
\frac{0,41}{0,015}=\conversionFraction[d] {0.41/0.015}
\smallskip
\frac{1}{7}-\frac{3}{8}=\ ConversionFraction[d] {1/7-3/8}
\smallskip
$\ConversionFraction[d]{1+1/2}$
\smallskip
$\ConversionFraction{0.1/0.7+30/80}$
\frac{111}{2145} = \frac{37}{715}
\frac{3}{15} = \frac{1}{5}
\frac{3}{15} = \frac{1}{5}
0,42
\frac{-}{0,015} = 28
0,41
\frac{0,41}{0,015} = \frac{82}{3}
\overline{7}
3
\frac{-}{2}
```

### information(s)

A priori le package <code>sint</code> permet de s'en sortir pour des calculs « simples », je ne garantis pas que tout calcul ou toute division donne un résultat *satisfaisant*!

### 35 Ensembles

#### 35.1 Idée

#### ♀ Idée(s)

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}

### 35.2 Commande et options

### Clés 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

$\EcritureEnsemble{a/b/c/d/e}$

$\EcritureEnsemble[Mathpunct=false]{a/b/c/d/e}$

$\EcritureEnsemble[Sep={,}]{a/b/c/d/e}$

$\EcritureEnsemble[Option={\strut}]{a/b/c/d/e}$

$\text{trut pour "augmenter"}$
```

 $\label{lem:lambda} $$ un peu la hauteur des {} $$ \ CritureEnsemble{ \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\} 

  \{\frac{1}{1+\frac{1}{3}};b;c;d;\frac{1}{2}\}
```

#### information(s)

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...

## 36 Écriture d'un trinôme, trinôme aléatoire

#### 36.1 Idée

#### ♀ Idée(s)

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  $\pm 1$ .

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 ensurement.

### </> Code LATEX

\EcritureTrinome[options]{a}{b}{c}

### ⟨/> Code LATEX

$$x^{2} + 7x$$

$$1,5x^{2} + 7,3x + 2,56$$

$$-x^{2} + 12$$

$$-x^{2} - 5x$$

### 36.2 Clés et options

### Clés et options

Quelques clés et options sont disponibles :

- la clé booléenne (Alea) pour autoriser les coefficients aléatoires (voir plus bas pour la syntaxe); défaut (false)
- la clé booléenne (**Anegatif**) pour autoriser *a* à être négatif.

défaut (true)

### information(s)

La clé (Alea) va modifier la manière de saisir les coefficients, il suffira dans ce cas de préciser les bornes, sous la forme valmin, valmax, de chacun des coefficients. C'est ensuite le package xfp qui va se charger de générer les coefficients.

```
Code LATEX
Avec $a$ entre 1 et 5 (et signe aléatoire)
→ puis $b$ entre $-2$ et 7 puis $c$ entre
                                          Avec a entre 1 et 5 (et signe aléatoire) puis b entre
\rightarrow $-10$ et 20 :
                                          -2 et 7 puis c entre -10 et 20:
f(x) = \text{EcritureTrinome}[Alea] \{1,5\} \{-5,5\} \{-10,10\} \{x\} = x^2 + 2x + 3
Avec $a$ entre 1 et 10 (forcément positif)
                                          Avec a entre 1 et 10 (forcément positif) puis b entre
→ puis $b$ entre $-2$ et 2 puis $c$ entre
                                          -2 et 2 puis c entre 0 et 4 :
\rightarrow 0 et 4 :
\EcritureTrinome[Alea,Anegatif=false]\{1,10\}\{-2,2\}\{\{5,x_4^2\}\\\x\ + 1
\EcritureTrinome[Alea,Anegatif=false]{1,10}{-2,2}{0,4}
```

### 37 Simplification de racines

#### 37.1 Idée

#### ♀ Idée(s)

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}

## 

### information(s)

 $3\sqrt{3565}$ 

C'est – comme souvent – le package <u>sint</u> 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!).

#### 37.2 Exemples

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

%Simplification n°1
$\frac{1}{\sqrt{6}}=\left(\sqrt{\frac{1}{6}}\right)=\SimplificationRacine{1/6}$

%Simplification n°2
$\frac{42}{\sqrt{5}}=\left(\sqrt{\frac{42^2}{5}}\right)=\SimplificationRacine{(42*42)/5}$

%\frac{42}{\sqrt{5}}=\left(\sqrt{\frac{42^2}{5}}\right)=\SimplificationRacine{(42*42)/5}$

%\frac{42}{\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}} = \sqrt{\sqrt{6}} \\ \frac{6}{6} \\ \frac{42}{\sqrt{5}} = \frac{42\sqrt{5}}{5}}{\sqrt{5}} = \sqrt{\sqrt{42\sqrt{5}}}

\sqrt{150 \times 0.35 \times (1-0.35)} = \frac{\sqrt{546}}{4}}
```

## 38 Mesure principale d'un angle

#### 38.1 Idée

### ♀ Idée(s)

**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

### information(s)

La commande est à insérer dans un environnement mathématique, via \( \) ou \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \( \) \(

### 38.2 Exemples

### Clés et options

Pour cette commande:

- le booléen ⟨d⟩ permet de forcer l'affichage en ∰displaystyle; défaut ⟨false⟩
- le booléen (Crochets) permet d'afficher le modulo entre crochets plutôt qu'entre parenthèses; défaut (false)
- l'argument obligatoire est en écriture en ligne.

### ⟨→ Code LATEX

```
$\MesurePrincipale[d]{54pi/7}$
$\MesurePrincipale[d]{-128pi/15}$
$\MesurePrincipale{3pi/2}$
$\MesurePrincipale[Crochets]{5pi/2}$
$\MesurePrincipale{-13pi}$
$\MesurePrincipale{28pi}$
$\MesurePrincipale[d]{14pi/4}$
$\MesurePrincipale[Crochets]{14pi/7}$
```

#### Sortie LATEX

$$\frac{54\pi}{7} = \frac{-2\pi}{7} (2\pi)$$
$$\frac{-128\pi}{15} = \frac{-8\pi}{15} (2\pi)$$

$$\frac{3\pi}{2} = \frac{-\pi}{2} \left( 2\pi \right)$$

$$\frac{5\pi}{2} = \frac{\pi}{2} \left[ 2\pi \right]$$

$$-13\pi = \pi \left( 2\pi \right)$$

$$28\pi = 0(2\pi)$$

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

$$\frac{14\pi}{7} = 0 [2\pi]$$

## Onzième partie

# Jeux et récréations

### 39 SudoMaths, en TikZ

#### 39.1 Introduction

### ♀ Idée(s)

L'idée est de proposer un environnement TikZ, une commande permettant de tracer des grilles de Sudo-Maths

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  $\neg$  seront "vides"  $\SudoMaths\{\}$ 

### Sortie LATEX

|   | а | b | С | d | е | f | g | h | i |
|---|---|---|---|---|---|---|---|---|---|
| Α |   |   |   |   |   |   |   |   |   |
| В |   |   |   |   |   |   |   |   |   |
| С |   |   |   |   |   |   |   |   |   |
| D |   |   |   |   |   |   |   |   |   |
| Ε |   |   |   |   |   |   |   |   |   |
| F |   |   |   |   |   |   |   |   |   |
| G |   |   |   |   |   |   |   |   |   |
| Н |   |   |   |   |   |   |   |   |   |
| I |   |   |   |   |   |   |   |   |   |

### information(s)

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

#### 39.2 Clés et options

#### Clés et options

Quelques (clés) sont disponibles pour cette commande :

- la clé (**Epaisseurg**) pour gérer l'épaisseur des traits épais; défaut (**1.5pt**)
- la clé (Epaisseur) pour gérer l'épaisseur des traits fins;
   défaut (0.5pt)
- la clé (**Unite**) qui est l'unité graphique de la figure; défaut (**1cm**)
- la clé (CouleurCase) pour la couleur (éventuelles) des cases; défaut (LightBlue!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 formatte le label des cases; défaut (\normalfont\normalsize)
- le booléen **(Legendes)** qui affiche ou non les légendes (H et V) des cases; défaut **(true)**
- le booleen (Legendes) qui aniche ou non les legendes (11 et v) des cases, defaut (the
- la clé (PoliceLeg) qui formatte 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**)

### information(s)

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!

### information(s)

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{%
                 / (c)* / (d)* §%
 (a)* / (b)* /
           /
                 / (f)* / (g)* / (h)* §%
 (e)* /
                     / / (j)* §%
            / (i)* /
                       / (1)* / (m)* §%
            / (k)* /
                     / / (p)* §%
            / (o)* /
 (n)* /
                  / (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

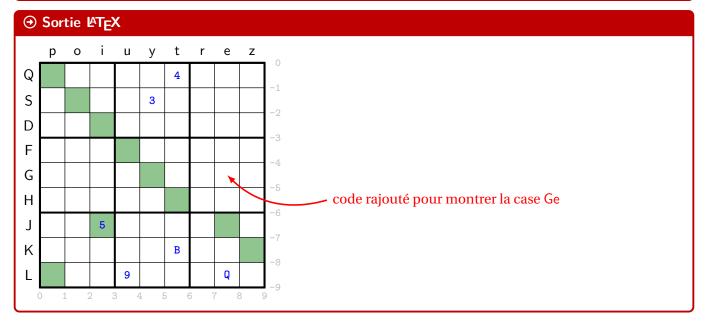
| (a) | (b) |     |     | (c) | (d) |
|-----|-----|-----|-----|-----|-----|
| (e) |     |     | (f) | (g) | (h) |
|     |     | (i) |     |     | (j) |
|     |     | (k) |     | (1) | (m) |
| (n) |     | (o) |     |     | (p) |
|     |     |     | (q) |     |     |

### information(s)

La grille, créée en TikZ, est portée par le rectangle de « coins » (0; 0) et (nbcol; –nblig), de sorte que les labels des cases sont situés au nœuds de coordonnées (x,5; –y,5).

```
⟨→ Code LATEX
%grille classique avec coloration de cases et commande tikz
%graduations rajoutées pour la lecture des coordonnées
\def\grilleSuMaB{%
  *////4///§%
  /*///3///\§%
  //*/////$%
  ///*////§%
  ////*////§%
  /////*///§%
  //5*////*/§%
  ////B///*§%
  *///9////Q/§%
\begin{EnvSudoMaths}[%
   Unite=0.66cm, Police=\footnotesize\bfseries\ttfamily, CouleurCase=ForestGreen!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}
```



### Douzième partie

# Historique

```
v 2.5.1: Ajout d'une version étoilée pour la conversion en fraction (page 103)
v 2.5.0: Système de librairies pour certains packages/commandes (page 7)
v 2.2.0: Ajout d'une clé (Notation) pour les arrangements et combinaisons (page 94)
v 2.1.9: Correction d'un bug (et ajout d'une version étoilée) pour les petits schémas « de signe » (page 31)
v 2.1.8: Suppression des commandes de PixelArt, désormais dans le package PixelArtTikz
v 2.1.7: Ajout d'une clé Math pour les sommets des figures de l'espace (pages 54 et 56)
v 2.1.6: Correction d'un bug lié au chargement de hylogos, remplacé par hologo
v 2.1.5: Combinatoire avec arrangements et combinaisons (page 94)
v 2.1.4: Résolution approchée d'équations f(x) = k (page 14)
v 2.1.3: Améliorations dans les présentations Piton (page 42)
v 2.1.2: Ajout d'une commande pour la mesure principale d'un angle (page 109)
v 2.1.1: Ajout d'une section pour des repères en TikZ (page 20)
v 2.1.0: Calcul du seuil, en interne désormais (page 18)
        : Commande pour simplifier une racine carrée (page 108)
        : Option [pythontex] pour charger le nécessaire pour pythontex
v 2.0.9: Nombres aléatoires, tirages aléatoires d'entiers (page 91)
v 2.0.8: Ajout d'un environnement pour présenter du code ETFX (page 53)
v 2.0.7: Ajout d'options pour stretch et fonte env python(s) (pas tous...)
v 2.0.6: Changement de taille de la police des codes Python (page 39)
v 2.0.5: Correction d'un bug avec les calculs de suites récurrentes (page 18)
v 2.0.4: Ajout d'une commande pour une présentation de solution par TVI (page 16)
v 2.0.3: Commandes pour des suites récurrentes simples (page 18)
v 2.0.2: Option left-margin=auto pour le package piton (page 42)
v 2.0.1: Chargement du package piton uniquement si compilation en Lua MFX (page 42)
v 2.0.0: Refonte du code source avec modification des commandes, et de la documentation
v 1.3.7: Commandes pour du code python via piton, en compilation Lua ETFX (page 42)
        : Corrections et modifications mineures de la documentation
v 1.3.6: Présentation de l'algorithme d'Euclide pour le PGCD (page 100)
        : Affichage d'un trinôme par coefficients, aléatoires ou non (page 106)
v 1.3.5: Correction d'un bug avec la loi géométrique (page 80)
v 1.3.4: Ajout de petits schémas, en TikZ, de lois normales et exponentielles (page 89)
        : Calculs de probas avec les lois géométriques et hypergéométriques (page 80)
v 1.3.3: Ajout d'un environnement pour des arbres de probas classiques, en TikZ (page 85)
v 1.3.2: Correction d'un bug sur les conversions bintohex avec lualatex (page 95)
v 1.3.1: Ajout d'une option pour ne pas afficher les bordures des corrections de pixelart
v 1.3.0: Commande pour présenter une conversion depuis la base 10 (page 98)
v 1.2.9: Correction des commandes avec simplekv
v 1.2.7: Ajout de commandes pour des calculs de probabilités (page 80)
v 1.2.6: Ajout d'un environnement pour des SudoMaths (page 110)
v 1.2.5: Ajout de commandes pour des boîtes à moustaches (page 76)
v 1.2.4: Correction de quelques bugs mineurs, et mise à jour de la doc
v 1.2.3: Commandes pour du code python "simple", sans compilation particulière (page 39)
  1.2.2: Commandes pour travailler sur des stats à 2 variables (page 67)
v 1.2.1: Amélioration de la gestion du csv pour Pixelart
v 1.1.9: Pixelart en TikZ
v 1.1.8: Style "Mainlevée" basique pour TikZ(page 61)
v 1.1.7: Conversions bin/hex/dec (basées sur xintbinhex) avec quelques détails (page 95)
v 1.1.6: Ajout d'une commande pour déterminer les paramètres d'une régression linéaire par moindres carrés (page
62)
v 1.1.5: Ajout de deux commandes pour, en TikZ, créer des petits schémas « de signe » (page 31)
v 1.1.4: Ajout d'une commande pour, en TikZ, créer facilement un cercle trigo avec options (page 58)
v 1.1.3: Ajout des commandes pour fractions, ensembles et récurrence (pages 103, 105 et 34)
v 1.1.1: Modification mineure de l'environnement calcul formel, avec prise de charge de la taille du texte
v 1.1.0: Ajout d'une commande pour créer des tétraèdres (avec nœuds) en TikZ (page 56)
v 1.0.9: Ajout d'une commande pour créer des pavés droits (avec nœuds) en TikZ (page 54)
v 1.0.8: Ajout d'une commande pour créer des cartouches de lien "comme capytale" (page 52)
v 1.0.7: Ajout d'une option build pour placer certains fichiers auxiliaires dans un répertoire externe
v 1.0.6: Ajout d'une option nominted pour ne pas charger (pas besoin de compiler avec shell-escape)
```

- v 1.0.5: Ajout d'un environnement pour Python (minted) (page 45)
- v 1.0.4: Ajout des environnements pour Terminal (win, osx, unix) (page 50)
- v 1.0.3: Ajout des environnements pour PseudoCode (page 48)
- v 1.0.2: Ajout des environnements pour Python (pythontex) (page 44)
- v 1.0 : Version initiale