

ProfLycee

Quelques *petites* commandes pour L^AT_EX (au lycée)

Cédric Pierquet

c pierquet -- at -- outlook . fr

Version 2.0.5 – 20 Novembre 2022

Résumé : Quelques commandes pour faciliter l'utilisation de L^AT_EX pour les enseignants de mathématiques en lycée.

Quelques commandes pour des courbes *lisses* avec gestion des extrema et des dérivées.

Quelques commandes pour simuler une fenêtre de logiciel de calcul formel, en TikZ.

Quelques environnements (tcbox) pour présenter du code python ou pseudocode.

Quelques environnements (tcbox) pour présenter des commandes dans un terminal (win ou mac ou linux).

Un cartouche (tcbox) pour présenter des codes de partage capytale.

Une commande pour tracer un pavé en droit, en TikZ, avec création des nœuds liés aux sommets.

Une commande pour simplifier des calculs sous forme fractionnaire.

Une commande pour simplifier l'écriture d'un ensemble, avec espaces « automatiques ».

Des commandes pour effectuer des calculs avec des suites récurrentes.

Une commande pour créer, en TikZ, la *toile* pour une suite récurrente.

Une commande pour créer, en TikZ, un cercle trigo avec options.

Une commande pour afficher un petit schéma, en TikZ, sur le signe d'une fonction affine ou d'un trinôme.

Deux commandes pour, en TikZ, créer des petits schémas « de signe ».

Une commande pour travailler sur les statistiques à deux variables (algébriques et graphiques).

Quelques commandes pour convertir bin/dec/hex avec certains détails.

Une commande pour, en TikZ, créer un pixelart avec correction éventuelle.

Une commande pour, en TikZ, créer un SudoMaths non forcément 9×9 .

Des commandes pour effectuer des calculs de probas (lois binomiale, exponentielle, de Poisson, normale).

Une commande pour, en TikZ, créer des arbres de probas « classiques ».

Merci à Anne pour ses retours et sa relecture!

Merci à Christophe pour ses retours et ses éclairages!

Merci aux membres du groupe  du « Coin L^AT_EX » pour leur aide et leurs idées!

L^AT_EX

pdfL^AT_EX

LuaL^AT_EX

TikZ

T_EXLive

MiK_TE_X

Table des matières

I	Introduction	6
1	Le package ProfLycee	6
1.1	« Philosophie » du package	6
1.2	Chargement du package	6
1.3	Options du package	7
2	Compléments	7
2.1	Changements à partir de cette version!!	7
2.2	Le système de « clés/options »	8
2.3	Compilateur(s)	8
2.4	Problèmes éventuels...	8
II	Liste des commandes, par thème	9
III	Outils pour l'analyse	12
3	L'outil « SplineTikz »	12
3.1	Courbe d'interpolation	12
3.2	Code, clés et options	12
3.3	Compléments sur les coefficients de « compensation »	12
3.4	Exemples	13
3.5	Avec une gestion plus fine des « coefficients »	14
3.6	Conclusion	14
4	L'outil « TangenteTikz »	15
4.1	Définitions	15
4.2	Exemple et illustration	15
4.3	Exemple avec les deux outils, et « personnalisation »	16
5	Présentation d'une solution d'équation par balayage	17
5.1	Idée	17
5.2	Clés et arguments	17
6	Suites récurrentes simples	18
6.1	Idées	18
6.2	Clés et arguments	18
6.3	Exemple d'utilisation	19
7	Suites récurrentes et « toile »	20
7.1	Idée	20
7.2	Commandes	20
7.3	Exemples	20
7.4	Influence des paramètres	22
IV	Présentation de codes	23
8	L'outil « Calcul Formel »	23
8.1	Introduction	23
8.2	La commande « CalculFormelParametres »	23
8.3	La commande « CalculFormelLigne »	23
8.4	Visualisation des paramètres	24

9	Code Python « simple » via le package listings	25
9.1	Introduction	25
9.2	Commande et options	25
9.3	Insertion via un fichier « externe »	25
9.4	Exemples	26
10	Code Python via le package piton	28
10.1	Introduction	28
10.2	Présentation de code Python	28
11	Code & Console Python, via les packages Pythontex ou Minted	30
11.1	Introduction	30
11.2	Présentation de code Python grâce au package pythontex	30
11.3	Présentation de code Python via le package minted	31
11.4	Console d'exécution Python	32
12	Pseudo-Code	34
12.1	Introduction	34
12.2	Présentation de Pseudo-Code	34
12.3	Compléments	35
13	Terminal Windows/UNIX/OSX	36
13.1	Introduction	36
13.2	Commandes	36
14	Cartouche Capytale	38
14.1	Introduction	38
14.2	Commandes	38
V	Outils pour la géométrie	39
15	Pavé droit « simple »	39
15.1	Introduction	39
15.2	Commandes	39
15.3	Influence des paramètres	40
16	Tétraèdre « simple »	41
16.1	Introduction	41
16.2	Commandes	41
16.3	Influence des paramètres	42
17	Cercle trigo	43
17.1	Idée	43
17.2	Commandes	43
17.3	Équations trigos	44
VI	Outils pour les statistiques	46
18	Paramètres d'une régression linéaire par la méthode des moindres carrés	46
18.1	Idée	46
18.2	Commandes	46
18.3	Intégration dans un environnement TikZ	48
19	Statistiques à deux variables	50
19.1	Idées	50
19.2	Commandes, clés et options	51
19.3	Commandes annexes	54
19.4	Interactions avec PLreglin	54

20 Boîtes à moustaches	58
20.1 Introduction	58
20.2 Clés et options	58
20.3 Commande pour placer un axe horizontal	59
VII Outils pour les probabilités	61
21 Calculs de probabilités	61
21.1 Introduction	61
21.2 Calculs « simples »	61
21.3 Complément avec sortie « formatée »	63
22 Arbres de probabilités « classiques »	65
22.1 Introduction	65
22.2 Options et arguments	65
22.3 Exemples complémentaires	66
23 Petits schémas pour des probabilités continues	68
23.1 Idée	68
23.2 Commandes et options	68
23.3 Remarques et compléments	69
VIII Outils pour l'arithmétique	70
24 Conversions binaire/hexadécimal/décimal	70
24.1 Idée	70
24.2 Conversion décimal vers binaire	70
24.3 Conversion binaire vers hexadécimal	71
24.4 Conversion binaire ou hexadécimal en décimal	72
25 Conversion « présentée » d'un nombre en décimal	73
25.1 Idée	73
25.2 Code et clés	73
26 Algorithme d'Euclide pour le PGCD	75
26.1 Idée	75
26.2 Options et clés	75
26.3 Compléments	76
IX Outils divers et variés	77
27 Fractions, ensembles	77
27.1 Fractions	77
27.2 Ensembles	78
28 Petits schémas pour le signe d'une fonction affine ou d'un trinôme	79
28.1 Idée	79
28.2 Commandes	79
28.3 Intégration avec tkz-tab	81
29 Style « main levée » en TikZ	82
29.1 Idée	82
29.2 Utilisation basique	82
30 Écriture d'un trinôme, trinôme aléatoire	83
30.1 Idée	83
30.2 Clés et options	83

X	Jeux et récréations	84
31	PixelART via un fichier csv, en TikZ	84
31.1	Introduction	84
31.2	Package csvsimple et option	84
31.3	Exemple simple, clés et options	85
31.4	Exemples complémentaires	86
32	SudoMaths, en TikZ	89
32.1	Introduction	89
32.2	Clés et options	90
XI	Historique	92

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 quelques packages utiles, mais j'ai fait le choix de laisser l'utilisateur gérer ses autres packages, comme notamment `amssymb` qui peut poser souci en fonction de la *position* de son chargement.

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

📄 Information(s)

Le package `\ProfLycee` charge les packages :

- `xcolor` avec les options `[table,svgnames]`;
- `tikz`, `pgf`, `xfp`;
- `xparse`, `xkeyval`, `xstring`, `simplekv`;
- `listofitems`, `xintexpr`, `xintbinhex` et `xintgcd`;
- `tabularray`, `fontawesome5`, `tcolorbox`;
- `piton` (uniquement si compilation en Lua \LaTeX !) et `pythontex`.

💡 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 \LaTeX avec `listofitems`, `randomlist`, `simplekv` ou encore `xstring`!

1.2 Chargement du package

📄 Code \LaTeX

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

📄 Code \LaTeX

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

1.3 Options du package

Attention

2.0.0 Une compilation shell-escape est recommandée pour l'utilisation de `ProfLycee`, notamment pour le package `minted` et la coloration syntaxique de code. Cependant, si vous ne souhaitez pas charger (et utiliser) `minted` vous pouvez charger le package `ProfLycee` avec l'option `(nonshellescape)`.

Code L^AT_EX

```
...
\usepackage[nonshellescape]{ProfLycee}
%package <minted> et librairie <minted> non chargés
...
```

Information(s)

En compilant (notamment avec les packages `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 vaut mieux créer au préalable le répertoire `build` avant de compiler un fichier!

Code L^AT_EX

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

Information(s)

Les options précédentes sont cumulables, et, pour info, elles conditionnent le chargement des packages avec les options :

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

2 Compléments

2.1 Changements à partir de cette version!!

Attention

2.0.0 Pour des raisons pratiques, les commandes et environnements disponibles dans `ProfLycee` ont été renommés, pour utiliser des noms plus génériques et explicites.

La nomenclature des **(clés)** a été également revue, avec – dans la grande majorité des cas – une majuscule en début.

Je préfère faire ces ajustements tant que le package est assez *jeune*!

Attention

2.0.0 L'ancienne version du package `ProfLycee` est toutefois encore disponible, pour une migration en douceur, elle est désormais accessible sous le nom `ProfLycee-old`, et la documentation est également encore disponible.

Code L^AT_EX

```
...
\usepackage[<options>]{ProfLycee-old} %nouveauités non disponibles par contre...
...
```

2.2 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 (entre [...]) ou mandataires (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 ou arguments.

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

Les codes seront présentés dans des boîtes **📄 Code L^AT_EX**, si possible avec la sortie dans la même boîte, et sinon la sortie sera visible dans des boîtes **📄 Sortie L^AT_EX**.

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!

2.3 Compilateur(s)

📌 Information(s)

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

En ce qui concerne les codes python et/ou pseudocode, il faudra :

- compiler en chaîne pdflatex + pythontex + pdflatex pour les environnements avec `pythontex`;
- compiler avec shell-escape (ou write18) pour les environnements avec `minted`.

⚠ Attention

Certaines commandes ou environnements nécessitent une compilation spécifique, qui sera indiquée clairement dans la documentation!

2.4 Problèmes éventuels...

📌 Information(s)

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

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 `xcolor` et de `amsmath`.

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

↔ Bonne(s) découverte(s) ↔

Deuxième partie

Liste des commandes, par thème

Information(s)

 2.0.0 Cette section contient un *résumé* des différentes commandes et environnements disponibles dans  ProfLycee. Elles sont présentées de manière *succincte*, mais elles sont présentées de manière *détaillée* dans la suite de la documentation.

Code L^AT_EX

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

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

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

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

Code L^AT_EX

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

%présentation de code Python
\begin{CodePythonLst}(*)[<largeur>]{<commandes tbox>}...\end{CodePythonLst}
\begin{CodePiton}[<options>]...\end{CodePiton}
\begin{CodePythontex}[<options>]...\end{CodePythontex}
\begin{CodePythonMinted}(*)[<largeur>][<options>]...\end{CodePythonMinted}

%console Python
\begin{ConsolePythontex}[<options>]...\end{ConsolePythontex}

%présentation de pseudocode
\begin{PseudoCode}(*)[<largeur>][<options>]...\end{PseudoCode}

%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 L^AT_EX

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

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

</> Code L^AT_EX

```
%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 axe Ox>{liste abscisses}<options axe Oy>{liste ordonnées}
\NuagePointsTikz[<options>]{<listeX>}{<listeY>}
\PointMoyenTikz[<options>]
\CourbeTikz[<options>]{<formule>}{<domaine>}

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

</> Code L^AT_EX

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

%loi de Poisson P (l)
\CalcPoisP{l}{k}
\CalcPoisC{l}{a}{b}
\PoissonP[*][prec]{l}{k}
\PoissonC[*][prec]{l}{a}{b}

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

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

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

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

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

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

</> Code L^AT_EX

```
%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 L^AT_EX

```
%conversion en fraction
\ConversionFraction[<option>]{<argument>}

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

%schémas pour le signe affine/trinôme, dans un environnement tikz
\MiniSchemaSignes[<clés>]
\MiniSchemaSignesTikzTab[<options>]{<numligne>}{<echelle>}{<décalage horizontal>}

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

</> Code L^AT_EX

```
%pixelart, dans un environnement tikz
\PixelArtTikz[<clés>]{<fichier>.csv}

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

Troisième partie

Outils pour l'analyse

3 L'outil « SplineTikz »

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

Pour déclarer les paramètres :

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

3.2 Code, clés et options

Code L^AT_EX

```
\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 <i>compensation</i> 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> |

3.3 Compléments sur les coefficients de « compensation »

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.

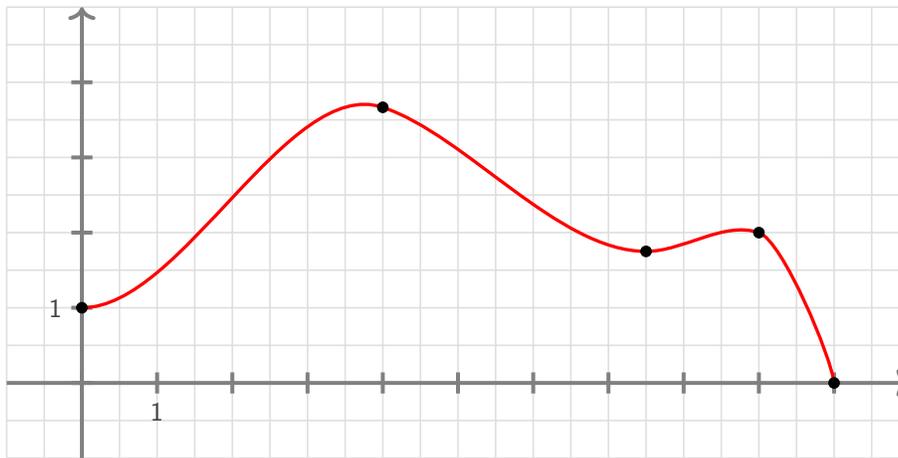
3.4 Exemples

</> Code L^AT_EX

```

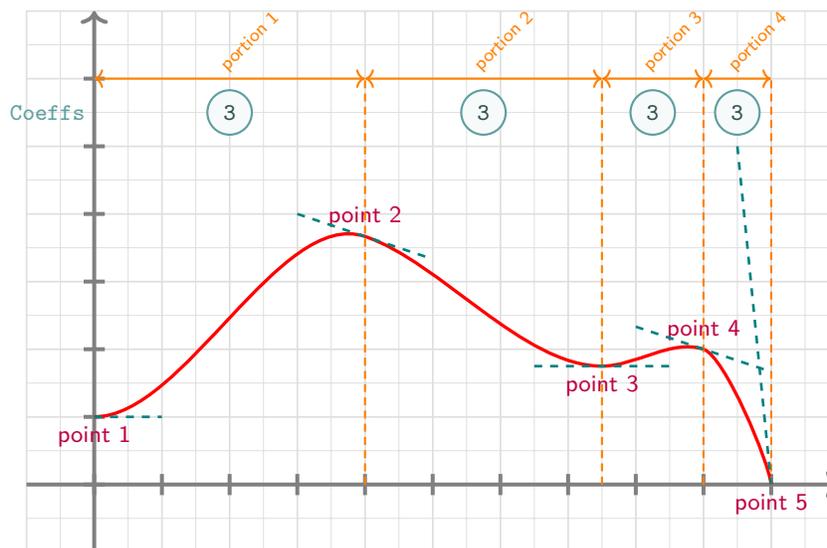
%code tikz
\def\x{0.9cm}\def\y{0.9cm}
\def\xmin{-1}\def\xmax{11}\def\xgrille{1}\def\xgrilles{0.5}
\def\ymin{-1}\def\ymax{5}\def\ygrille{1}\def\ygrilles{0.5}
%axes et grilles
\draw[xstep=\xgrilles,ystep=\ygrilles,line width=0.6pt,lightgray!50] (\xmin,\ymin) grid (\xmax,\ymax);
\draw[line width=1.5pt,->,gray]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

```



Information(s)

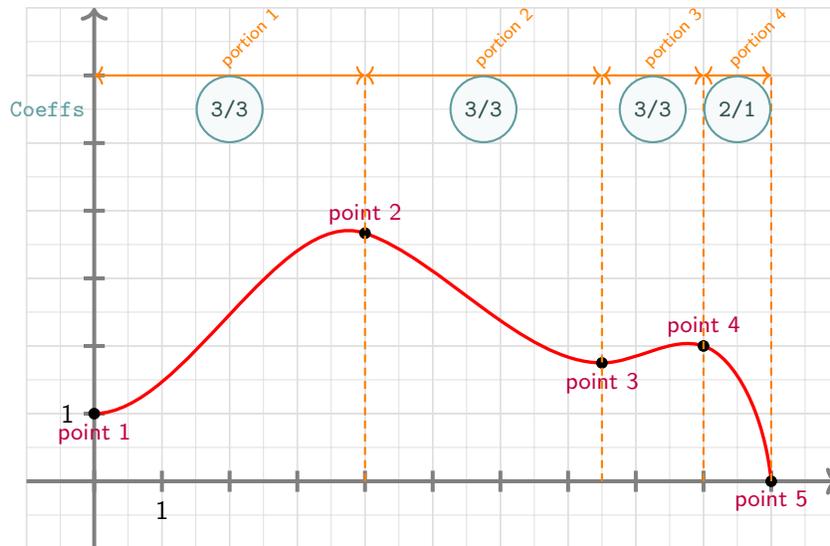
Avec des explications utiles à la compréhension :



3.5 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 L^AT_EX

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

Sortie L^AT_EX



3.6 Conclusion

Information(s)

Le plus « simple » est donc :

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

4 L'outil « TangenteTikz »

4.1 Définitions

💡 Idée(s)

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

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

</> Code L^AT_EX

```
\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 x_i/y_i avec la pente d_i ;
- avec une épaisseur de **<Épaisseur>**, une couleur **<Couleur>** et un style additionnel **<Style>**;
- en la traçant à partir de **<xl>** avant x_i et jusqu'à **<xr>** après x_i .

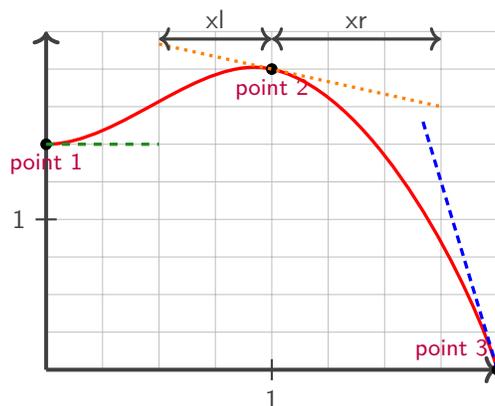
4.2 Exemple et illustration

</> Code L^AT_EX

```
\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 L^AT_EX

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



4.3 Exemple avec les deux outils, et « personnalisation »

</> Code L^AT_EX

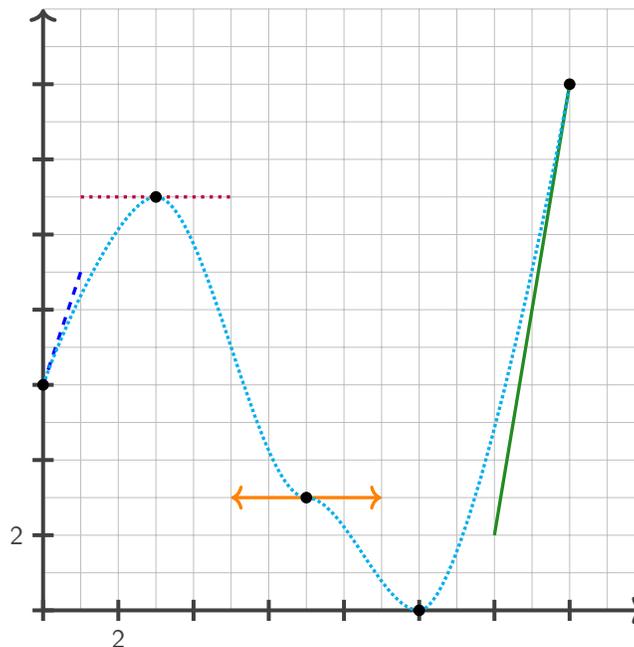
```

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

\begin{tikzpicture}[x=0.5cm,y=0.5cm,xmin=0,xmax=16,xgrilles=1,ymin=0,ymax=16,ygrilles=1]
  \draw[xstep=\xgrilles,ystep=\ygrilles,line width=0.3pt,lightgray] (\xmin,\ymin) grid (\xmax,\ymax) ;
  \draw[line width=1.5pt,->,darkgray] (\xmin,0)--(\xmax,0) ;
  \draw[line width=1.5pt,->,darkgray] (0,\ymin)--(0,\ymax) ;
  \foreach \x in {0,2,...,14} {\draw[darkgray,line width=1.5pt] (\x,4pt) -- (\x,-4pt) ;}
  \foreach \y in {0,2,...,14} {\draw[darkgray,line width=1.5pt] (4pt,\y) -- (-4pt,\y) ;}
  %la liste pour la courbe d'interpolation
  \def\liste{0/6/3$3/11/0$7/3/0$10/0/0$14/14/6}
  %les tangentes "stylis es"
  \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-m eme
  \SplineTikz[AffPoints,Coeffs=3,Couleur=cyan,Style=densely dotted]{\liste}
\end{tikzpicture}

```

⊕ Sortie L^AT_EX



5 Présentation d'une solution d'équation par balayage

5.1 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 L^AT_EX

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

5.2 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 qui affiche un texte avant, en spécifiant la calculatrice; défaut **<false>**
- le booléen **<Majuscule>** qui affiche le texte avant, avec une majuscule au début; défaut **<>true>**

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

Code L^AT_EX

```
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 L^AT_EX

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

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

Code L^AT_EX

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

On s'intéresse à $g(x) = 1,5$ avec $g(x) = \ln(x)$. Par balayage, on obtient
$$\begin{cases} g(4,48) \approx 1,4996 < 1,5 \\ g(4,49) \approx 1,5019 > 1,5 \end{cases} \Rightarrow 4,48 < x_0 < 4,49.$$

6 Suites récurrentes simples

6.1 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

Le code pour le seuil ne trouve pas la solution (...), il met *juste* en forme et effectue quand même les calculs d'images.

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

</> Code L^AT_EX

```
%commande pour calculer et formater
\CalculTermeReccurrence[<options>]{<fonction associée>}

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

6.2 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.

défaut **<3>**

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

</> Code L^AT_EX

```
Avec  $\begin{dcases} u_0 = 50 \\ u_{n+1} = \frac{1}{u_n + 2} \end{dcases}$ .

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

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

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

📄 Sortie L^AT_EX

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

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

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

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

sortie par défaut.

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

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

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 **⟨SolN⟩** qui est la valeur de l'indice cherché;
- 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 qui affiche un texte avant, en spécifiant la calculatrice; défaut **⟨false⟩**
- le booléen **⟨Majuscule⟩** qui affiche le texte avant, avec une majuscule au début; défaut **⟨true⟩**
- la clé **⟨Sens⟩** (parmi **⟨<⟩** ou **⟨>⟩**) pour indiquer le type de seuil. défaut **⟨>⟩**

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

Code L^AT_EX

```
Avec \begin{dcases} u_1 = 2 \\ u_{n+1}=1+\dfrac{1+u_n^2}{1+u_n} \end{dcases} $,
on cherche $n$ tel que $u_n > 5$. \\
\SolutionSeuil[Balayage,No=1,UNo=2,SolN=8]{1+(1+x**2)/(1+x)}{5}. \\
\SolutionSeuil[Calculatrice,Precision=3,No=1,UNo=2,SolN=8]{1+(1+x**2)/(1+x)}{5}. \\
```

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

$$\text{Par balayage, on obtient } \begin{cases} u_7 \approx 4,87 \leq 5 \\ u_8 \approx 5,21 > 5 \end{cases} \Rightarrow n \geq 8.$$

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

6.3 Exemple d'utilisation

Code L^AT_EX

```
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 \\
\end{tabular}
\xintFor* #1 in {\xintSeq{2}{8}} \do {#1 & \CalculTermeRecurrence[No=1,UNo=2,N=#1]{1+(1+x**2)/(1+x)} \\}
\end{tabular}
```

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

n	u_n
1	2
2	2,667
3	3,212
4	3,687
5	4,114
6	4,505
7	4,868
8	5,209

7 Suites récurrentes et « toile »

7.1 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 ».

7.2 Commandes

</> Code L^AT_EX

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

📄 Information(s)

Il est à noter que le code n'est pas autonome, et doit être intégré dans un environnement `\begin{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!

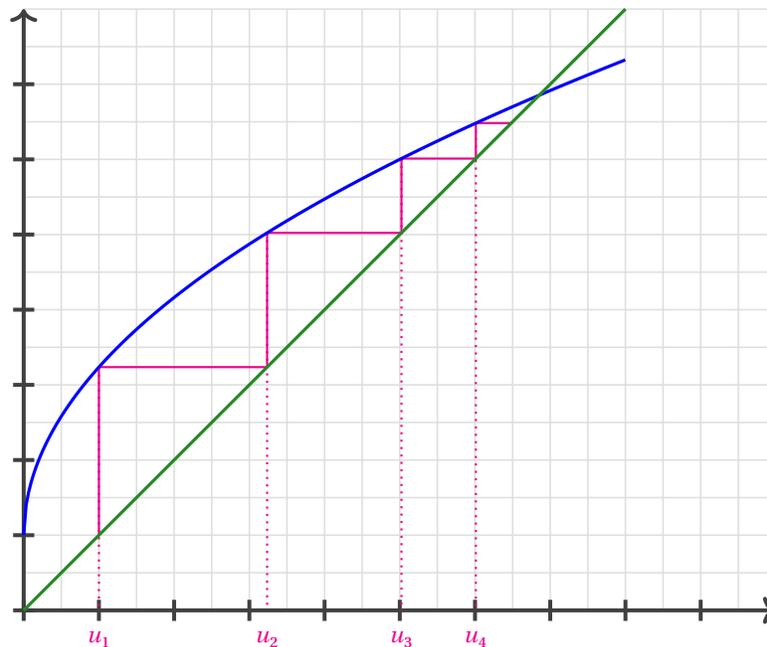
7.3 Exemples

📄 Information(s)

On va tracer la *toile* des 4 premiers termes de la suite récurrente
$$\begin{cases} u_1 = 1 \\ u_{n+1} = \sqrt{5u_n} + 1 \text{ pour tout entier } n \geq 1 \end{cases}$$

</> Code L^AT_EX

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

7.4 Influence des paramètres

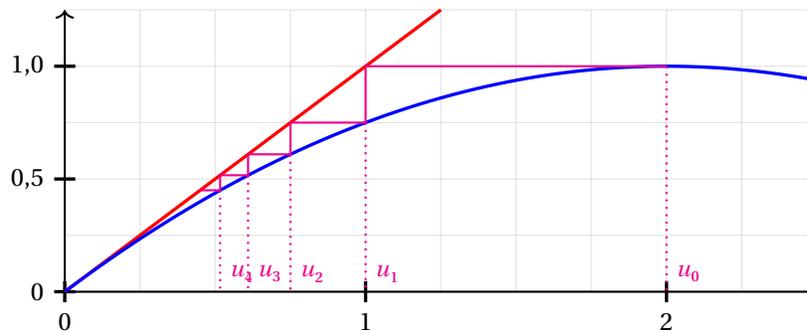
</> Code L^AT_EX

```

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

```

⊖ Sortie L^AT_EX



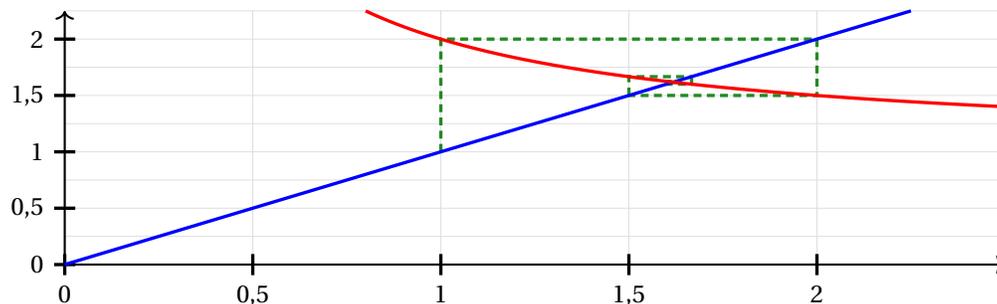
</> Code L^AT_EX

```

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

```

⊖ Sortie L^AT_EX



Quatrième partie

Présentation de codes

8 L'outil « Calcul Formel »

8.1 Introduction

💡 Idée(s)

L'idée des commandes suivantes est de définir, dans un environnement TikZ, 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 `\tikz-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.

8.2 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 L^AT_EX

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

🔗 Clés et options

Les `\Clés` disponibles sont :

- | | |
|--|--|
| — <code>\Largeur</code> : largeur de l'environnement; | défaut <code>\16</code> |
| — <code>\EspaceLg</code> : espacement vertical entre les lignes; | défaut <code>\2pt</code> |
| — <code>\PremCol</code> & <code>\HautPremCol</code> : largeur et hauteur de la case du <i>petit numéro</i> ; | défaut <code>\0.3</code> & <code>\0.4</code> |
| — <code>\Taille</code> : taille du texte; | défaut <code>\normalsize</code> |
| — <code>\Couleur</code> : couleur des traits de l'environnement; | défaut <code>\darkgray</code> |
| — <code>\Titre</code> : booléen pour l'affichage d'un bandeau de titre; | défaut <code>\false</code> |
| — <code>\TailleTitre</code> : taille du titre; | défaut <code>\normalsize</code> |
| — <code>\PosCmd</code> : position horizontale de la commande d'entrée; | défaut <code>\gauche</code> |
| — <code>\PosRes</code> : position horizontale de la commande de sortie; | défaut <code>\centre</code> |
| — <code>\CouleurCmd</code> : couleur de la commande d'entrée; | défaut <code>\red</code> |
| — <code>\CouleurRes</code> : couleur de la commande de sortie; | défaut <code>\blue</code> |
| — <code>\Sep</code> : booléen pour l'affichage du trait de séparation E/S; | défaut <code>\true</code> |
| — <code>\Menu</code> : booléen pour l'affichage du <i>bouton</i> MENU; | défaut <code>\true</code> |
| — <code>\LabelTitre</code> : libellé du titre. | défaut <code>\Résultats obtenus avec un logiciel de Calcul Formel</code> |

8.3 La commande « CalculFormelLigne »

📄 Information(s)

Une fois les paramètres déclarés, il faut créer les différentes lignes, grâce à la commande `\CalculFormelLigne`.

</> Code L^AT_EX

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

🔗 Clés et options

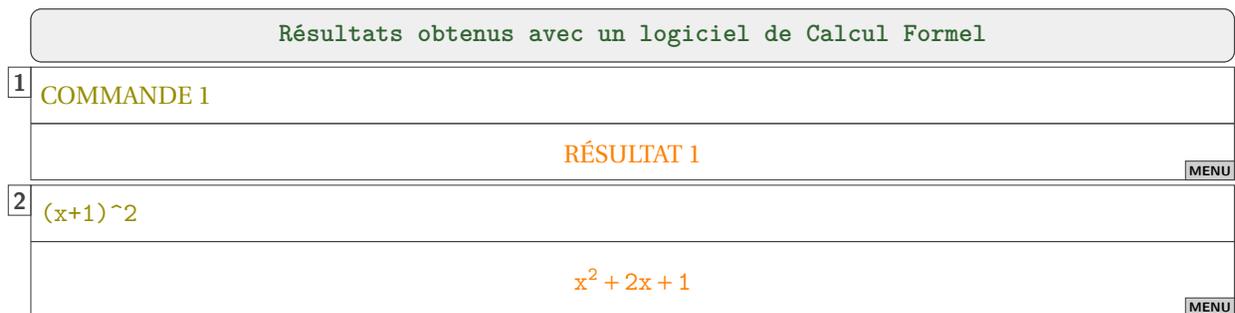
Les (quelques) <Clés> disponibles sont :

- <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.

</> Code L^AT_EX

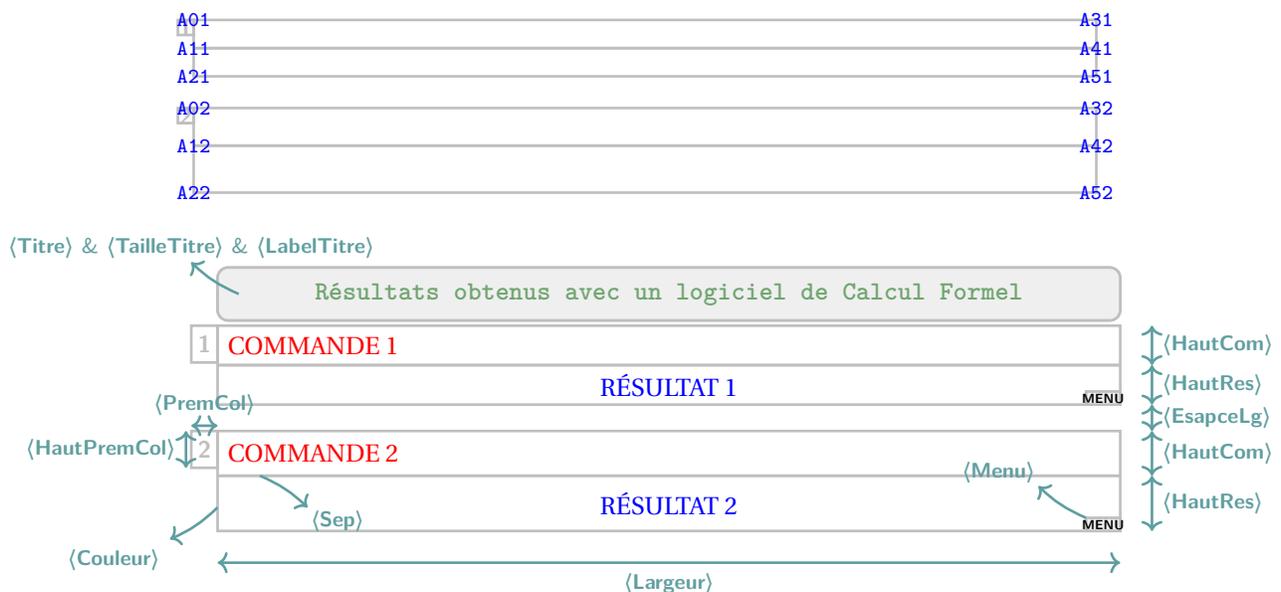
```
%code tikz
\CalculFormelParametres[Titre=true,CouleurCmd=olive,CouleurRes=orange]
\CalculFormelLigne{COMMANDE 1}{RÉSULTAT 1}
\CalculFormelLigne[HautCmd=0.75,HautRes=1]{\texttt{(x+1)\CFchap2}}{\mathhtt{x^2+2x+1}}
%\CFchap := ^ en mathtt
```



8.4 Visualisation des paramètres

📄 Information(s)

Pour *illustrer* un peu les <clés>, un petit schéma, avec les différents nœuds créés par les macros.



9 Code Python « simple » via le package listings

9.1 Introduction

💡 Idée(s)

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

📌 Information(s)

Le package `\usepackage{listings}` ne nécessite pas de compilation particulière, au contraire d'autres (comme `\usepackage{pythontex}` ou `\usepackage{minted}` ou `\usepackage{pilon}`) 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 `\usepackage{pythontex}` ou `\usepackage{minted}` ou `\usepackage{pilon}`.

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

9.2 Commande et options

💡 Idée(s)

L'environnement `\usepackage{CodePythonLst}` permet de présenter du code python, dans une `\usepackage{tcolorbox}` avec un style particulier.

</> Code L^AT_EX

```
\begin{CodePythonLst}(*)[<largeur>]{<commandes tcolorbox>}  
...  
\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 `\usepackage{tcolorbox}`; défaut (`\linewidth`)
- le second argument (mandataire), concerne des **<options>** de la `\usepackage{tcolorbox}` en *langage tcolorbox*, comme l'alignement.

⚠ Attention

Les environnements `\usepackage{DeclareTCBListing}` créés par `\usepackage{tcolorbox}` et `\usepackage{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!

9.3 Insertion via un fichier « externe »

💡 Idée(s)

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

</> Code L^AT_EX

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

9.4 Exemples

</> Code L^AT_EX

```
\begin{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 L^AT_EX

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

Code Python



</> Code L^AT_EX

```
\begin{CodePythonLst}*[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

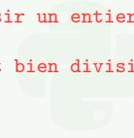
def f(x) :
    return x**2
\end{CodePythonLst}
```

⊖ Sortie L^AT_EX

```
#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 Python



</> Code L^AT_EX

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

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

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

⊖ Sortie L^AT_EX

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

10 Code Python via le package piton

10.1 Introduction

Information(s)

`2.0.0` Le package `piton` (compatible uniquement avec une compilation en Lua \TeX !) permet d'insérer du code Python avec une coloration syntaxique en utilisant la bibliothèque Lua LPEG.

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

`2.0.2` Depuis la version `0.95` du package `piton`, `\left-margin=auto` est disponible et activée dans ProfLycee.

Attention

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

`2.0.1` Ce package n'est chargé que si la compilation détectée est en Lua \TeX !

10.2 Présentation de code Python

Code \TeX

```
\begin{CodePiton}[<options>]
...
...
\end{CodePiton}
```

Attention

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

Clés et options

Plusieurs **clés** sont disponibles :

- la clé booléenne **Lignes** pour afficher ou non les numéros de lignes; défaut `\true`
- la clé **Largeur** qui correspond à la largeur de la `tcbbox`; défaut `\linewidth`
- la clé **Alignement** qui paramètre l'alignement de la `tcbbox`. défaut `\center`

Code \TeX

```
\begin{CodePiton}
#environnement piton avec numéros de ligne, pleine largeur
def f(x) :
    """fonction qui renvoie le carré d'un réel"""
    return x**2
\end{CodePiton}
```

Sortie \TeX

```
1 #environnement piton avec numéros de ligne, pleine largeur
2 def f(x) :
3     """fonction qui renvoie le carré d'un réel"""
4     return x**2
```

Code Python

</> Code L^AT_EX

```
\begin{CodePiton}[Lignes=false,Largeur=15cm]
#sans numéro, de largeur 15cm
def f(x) :
    """fonction qui renvoie le carré d'un réel"""
    return x**2
\end{CodePiton}

\begin{CodePiton}[Alignement=flush right,Largeur=13cm]
#avec numéros, de largeur 13cm, aligné à droite
def f(x) :
    """fonction qui renvoie le carré d'un réel"""
    return x**2
\end{CodePiton}

\begin{CodePiton}[Alignement=flush left,Largeur=11cm]
#avec numéros, de largeur 11cm, aligné à gauche
def f(x) :
    """fonction qui renvoie le carré d'un réel"""
    return x**2
\end{CodePiton}
```

⊖ Sortie L^AT_EX

```
#sans numéro, de largeur 15cm
def f(x) :
    """fonction qui renvoie le carré d'un réel"""
    return x**2
```

```
1 #avec numéros, de largeur 13cm, aligné à droite
2 def f(x) :
3     """fonction qui renvoie le carré d'un réel"""
4     return x**2
```

```
1 #avec numéros, de largeur 11cm, aligné à gauche
2 def f(x) :
3     """fonction qui renvoie le carré d'un réel"""
4     return x**2
```

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

11.1 Introduction

💡 Idée(s)

Le package `\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 `tcbbox` et des `fancyvrb`, le style est « fixé » tel quel, et seules la taille et la position de la `tcbbox` sont modifiables. Si toutefois vous souhaitez personnaliser davantage, il faudra prendre le code correspondant et appliquer vos modifications!

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

11.2 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 L^AT_EX

```
\begin{CodePythontex}[<options>]
...
\end{CodePythontex}
```

🔑 Clés et options

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

- **<Largeur>** : largeur de la `tcbbox`; défaut `\linewidth`
- **<Centre>** : booléen pour centrer ou non la `tcbbox`; défaut `<true>`
- **<Lignes>** : booléen pour afficher ou non les numéros de ligne. défaut `<true>`

🔗 Code L^AT_EX

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

📄 Sortie L^AT_EX

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

📄 Code Python

</> Code L^AT_EX

```
\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 L^AT_EX

```
#environnement Python(tex) non centré sans numéro de ligne
def f(x) :
    return x**2
```

Code Python

11.3 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.

⚠ Attention

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

</> Code L^AT_EX

```
\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 premier argument optionnel concerne la **largeur** de la `tcbbox`; défaut **12cm**
- le second argument optionnel concerne les **options** de la `tcbbox` en langage *tcolorbox*. défaut **vide**

</> Code L^AT_EX

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

☉ Sortie L^AT_EX

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

Code Python

</> Code L^AT_EX

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

☉ Sortie L^AT_EX

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

Code Python

11.4 Console d'exécution Python

💡 Idée(s)

`\usepackage{pythontex}` permet également de *simuler* (en exécutant également!) du code python dans une *console*. C'est l'environnement `\usepackage{ConsolePythontex}` qui permet de le faire.

</> Code L^AT_EX

```
\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>
- <Label> : booléen pour afficher ou non le titre. défaut <true>

</> Code L^AT_EX

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

☉ Sortie L^AT_EX

```
----- 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 L^AT_EX

```
\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 L^AT_EX

```
>>> #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
[[1, 19, 20, 5, 7, 1], [13, 7, 20, 16, 10, 12], [6, 13, 10, 1, 1, 6]]
>>> len(tableau), len(tableau[0])
(3, 6)
>>> tableau[1][4]
10
```

+ Information(s)

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

12 Pseudo-Code

12.1 Introduction

Information(s)

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

12.2 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 L^AT_EX

```
\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 `tcbbox`; défaut **(12cm)**
- le second argument optionnel concerne les **(options)** de la `tcbbox` en langage *tcolorbox*. défaut **(vide)**

Code L^AT_EX

```
\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 L^AT_EX

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

Pseudo-Code

</> Code L^AT_EX

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

⊖ Sortie L^AT_EX

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

Pseudo-Code

12.3 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 `listings`. 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 **(mathescape)** activée, et accessible grâce aux délimiteurs `((...*))`. Cela permet d'insérer du code L^AT_EX dans l'environnement `PseudoCode` (attention au fontes de rendu par contre!).

</> Code L^AT_EX

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

⊖ Sortie L^AT_EX

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

Pseudo-Code

13 Terminal Windows/UNIX/OSX

13.1 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!

13.2 Commandes

🔗 Code L^AT_EX

```
\begin{TerminalWin}[<largeur>]{<titre=...>}[<options>]
...
\end{TerminalWin}

\begin{TerminalUnix}[<largeur>]{<titre=...>}[<options>]
...
\end{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 `tcbox`; défaut `\linewidth`
- le deuxième, mandataire, permet de spécifier le titre par la clé **(titre)**. défaut **(Terminal Windows/UNIX/OSX)**
- le troisième, optionnel, concerne les **(options)** de la `tcbox` en langage `tcbox`. défaut **(vide)**

📄 Information(s)

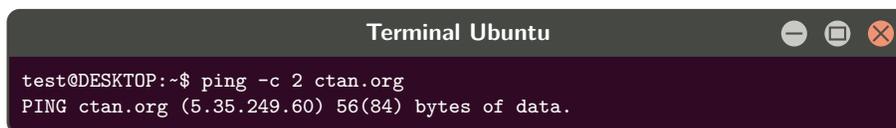
Le code n'est pas formaté, ni mis en coloration syntaxique.

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 L^AT_EX

```
\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 L^AT_EX



</> Code L^AT_EX

```
\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}
```

☉ Sortie L^AT_EX

> Terminal Windows

```
Microsoft Windows [version 10.0.22000.493]
(c) Microsoft Corporation. Tous droits réservés.
C:\Users\test>ping ctan.org

Envoi 'dune 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
```

Terminal Ubuntu

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

Terminal MacOSX

```
[test@server]$ ping -c 2 ctan.org
PING ctan.org (5.35.249.60) 56(84) bytes of data.
```

14 Cartouche Capytale

14.1 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.

14.2 Commandes

</> Code L^AT_EX

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

🔑 Clés et options

Peu d'options pour ces commandes :

- la version *étoilée* qui permet de passer de la police (**sf**family) à la police (**tt**family), 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, mandataire, est le code capytale à afficher.

</> Code L^AT_EX

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

🕒 Sortie L^AT_EX

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

📄 Information(s)

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

</> Code L^AT_EX

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

🕒 Sortie L^AT_EX

```
abcd-12345 🔗
```

Cinquième partie

Outils pour la géométrie

15 Pavé droit « simple »

15.1 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.

15.2 Commandes

</> Code L^AT_EX

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

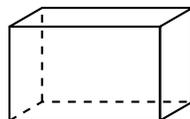
🔑 Clés et options

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> |
| — <Epaisseur> : épaisseur des arêtes (en <i>langage simplifié TikZ</i>); | défaut <thick> |
| — <Aff> : booléen pour afficher les noms des sommets; | défaut <false> |
| — <Plein> : booléen pour ne pas afficher les arêtes <i>invisibles</i> ; | défaut <false> |
| — <Cube> : booléen pour préciser qu'il s'agit d'un cube (seule la valeur <Largeur> est util(isé)e). | défaut <false> |

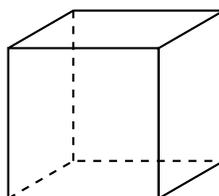
</> Code L^AT_EX

```
%code tikz
\PaveTikz
```



</> Code L^AT_EX

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



Information(s)

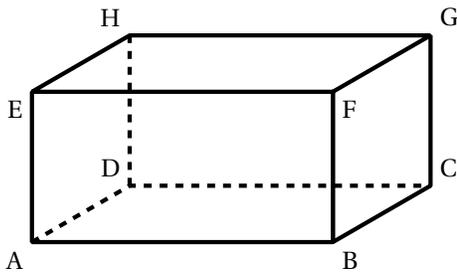
La ligne est de ce fait à insérer dans un environnement `TikZ`, avec les options au choix pour cet environnement. Le code crée les nœuds relatifs aux sommets, et les nomme comme les sommets, ce qui permet de les réutiliser pour éventuellement compléter la figure!

15.3 Influence des paramètres

</> Code \LaTeX

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

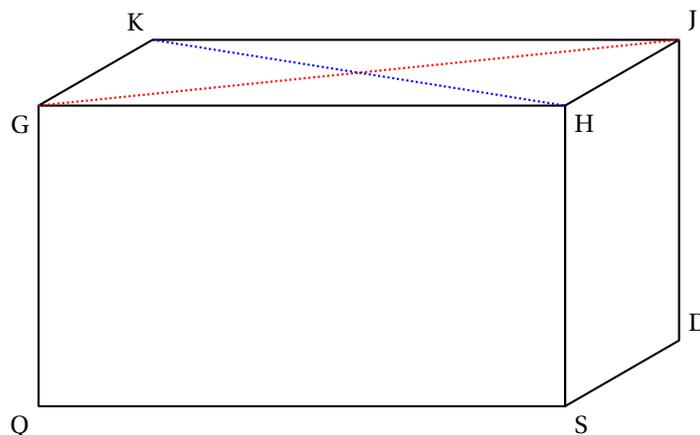
Sortie \LaTeX



</> Code \LaTeX

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

Sortie \LaTeX



16 Tétraèdre « simple »

16.1 Introduction

💡 Idée(s)

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

16.2 Commandes

🔗 Code L^AT_EX

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

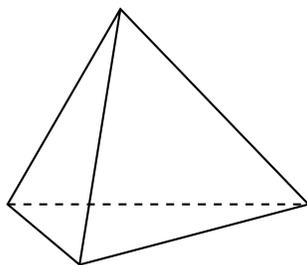
🔑 Clés et options

Quelques <clés> sont disponibles pour cette commande :

- | | |
|---|---------------------|
| — <Largeur> : <i>largeur</i> du tétraèdre; | défaut <4> |
| — <Profondeur> : <i>profondeur</i> du tétraèdre; | défaut <1.25> |
| — <Hauteur> : <i>hauteur</i> du tétraèdre; | défaut <3> |
| — <Alpha> : angle <i>du sommet de devant</i> ; | défaut <40> |
| — <Beta> : angle <i>du sommet du haut</i> ; | défaut <60> |
| — <Sommets> : liste des sommets (avec délimiteur \$!); | défaut <A\$B\$C\$D> |
| — <Epaisseur> : épaisseur des arêtes (en <i>langage simplifié TikZ</i>); | défaut <thick> |
| — <Aff> : booléen pour afficher les noms des sommets; | défaut <false> |
| — <Plein> : booléen pour ne pas afficher l'arête <i>invisible</i> . | défaut <false> |

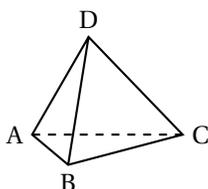
🔗 Code L^AT_EX

```
%code tikz  
\TetraedreTikz
```



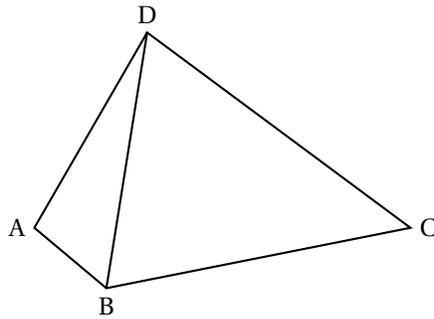
🔗 Code L^AT_EX

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



</> Code L^AT_EX

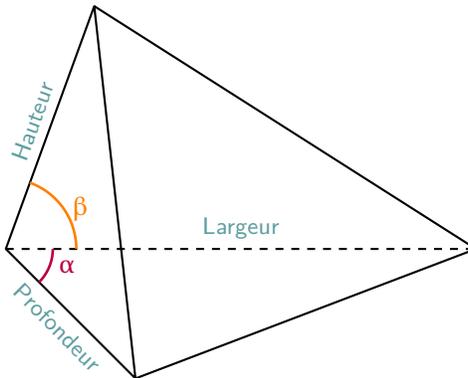
```
%code tikz
\TetraedreTikz[Plein,Aff,Largeur=5,Beta=60]
```



16.3 Influence des paramètres

Information(s)

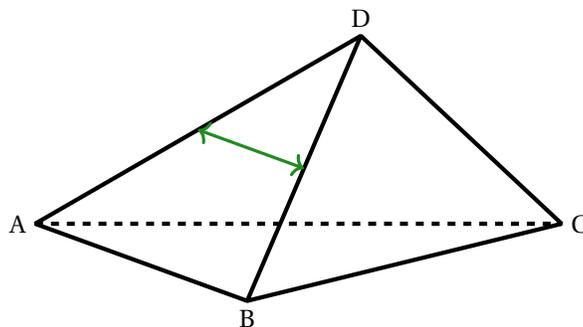
Pour *illustrer* un peu les <clés>, un petit schéma, avec les différents paramètres utiles.



</> Code L^AT_EX

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

Sortie L^AT_EX



17 Cercle trigo

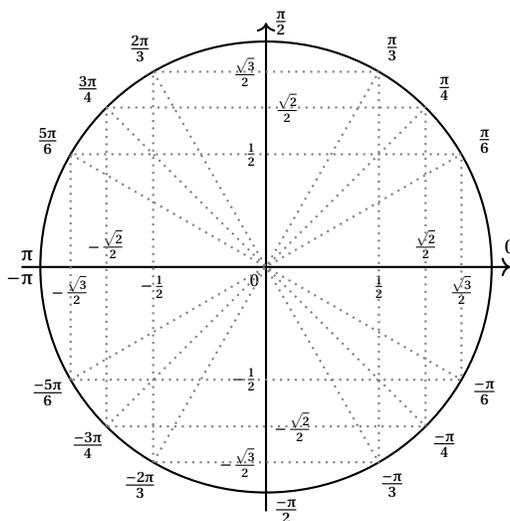
17.1 Idée

💡 Idée(s)

L'idée est d'obtenir une commande pour tracer (en TikZ) un cercle trigonométrique, avec personnalisation des affichages. Comme pour les autres commandes TikZ, l'idée est de laisser l'utilisateur définir et créer son environnement TikZ, et d'insérer la commande `\CercleTrigo` pour afficher le cercle.

</> Code L^AT_EX

```
%code tikz  
\CercleTrigo
```



17.2 Commandes

</> Code L^AT_EX

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

🔑 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>**

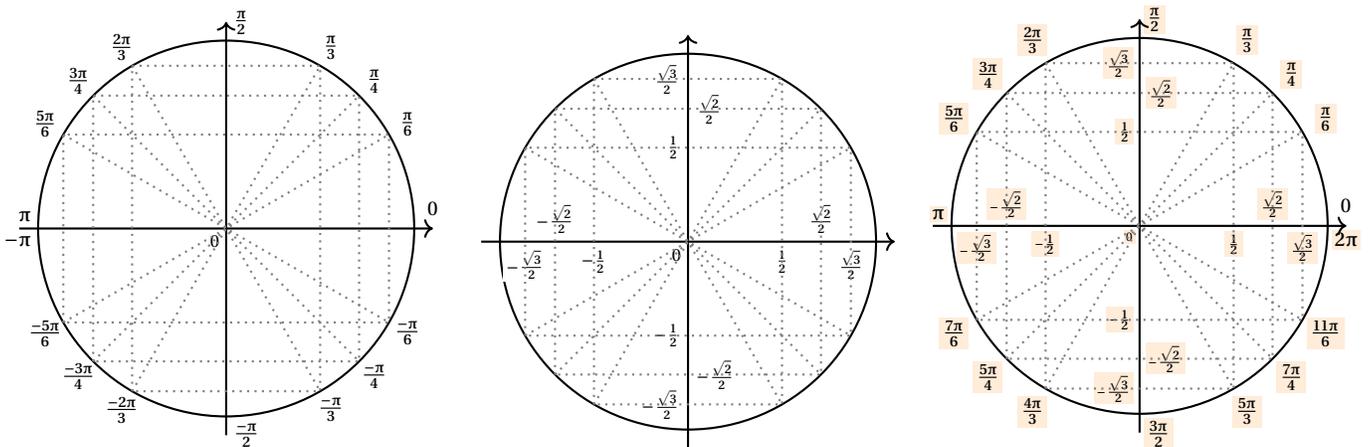
</> Code L^AT_EX

```

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

```

⊖ Sortie L^AT_EX



17.3 Équations trigos

📌 Information(s)

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

🔗 Clés et options

Les **<Clés>** pour cette possibilité sont :

- un booléen **<Equationcos>** pour *activer* « cos = »;
- 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 **<false>**
défaut **<false>**
défaut **<30>**
défaut **<45>**
défaut **<blue>**

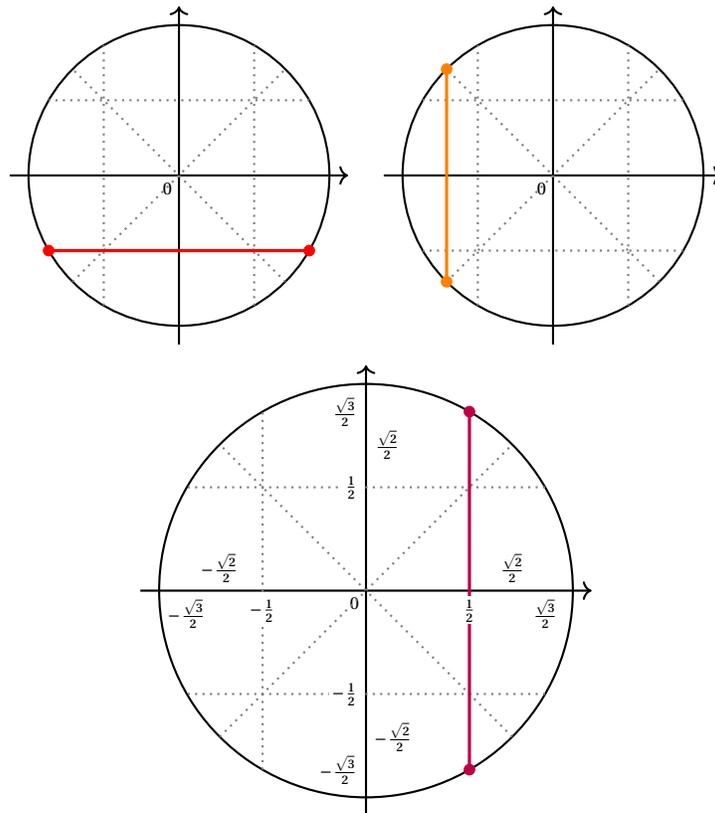
</> Code L^AT_EX

```
\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 L^AT_EX



Sixième partie

Outils pour les statistiques

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

18.1 Idée

💡 Idée(s)

L'idée est d'utiliser une commande qui va permettre de calculer les paramètres principaux d'une régression linéaire par la méthode des moindres carrés.

Le package `pgfpots` permet de le faire nativement, mais le moteur de calculs de `pgf` n'est pas des plus performants avec de grandes valeurs, donc ici cela passe par `xfp` qui permet de gagner en précision!

L'idée est que cette macro calcule et stocke les paramètres dans des variables (le nom peut être personnalisé!) pour exploitation ultérieure :

- en calculs *purs*;
- dans un environnement TikZ via `pgfplots` ou bien en *natif*;
- dans un environnement PSTricks;
- dans un environnement METAPOST (à vérifier quand même);
- ...

</> Code L^AT_EX

```
...  
\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!

18.2 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> |
| — la clé <NomCoeffb> qui permet de définir la variable qui contiendra b ; | défaut <COEFFb> |
| — la clé <NomCoeffr> qui permet de définir la variable qui contiendra r ; | défaut <COEFFr> |
| — la clé <NomCoeffrd> qui permet de définir la variable qui contiendra r^2 ; | défaut <COEFFrd> |
| — la clé <NomXmin> qui permet de définir la variable qui contiendra x_{\min} ; | défaut <LXmin> |
| — la clé <NomXmax> qui permet de définir la variable qui contiendra x_{\max} . | défaut <LXmax> |

</> Code L^AT_EX

```
%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}
```


Sortie L^AT_EX

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 $r^2 = 0,7954$.

Information(s)

rad REGRESSIONS			rad REGRESSIONS		
Données	Graphique	Stats	Données	Graphique	Stats
X1	Y1	X2	Covariance		-3.133333
0	-35		Σxy		-742.7
1	-37.4		Régression	$y = a \cdot x + b$	
3	-37.7		a		-0.7006211
4	-39.9		b		-35.88137
5	-39		r		-0.891847
6	-39.6		r ²		0.7953911

18.3 Intégration dans un environnement TikZ

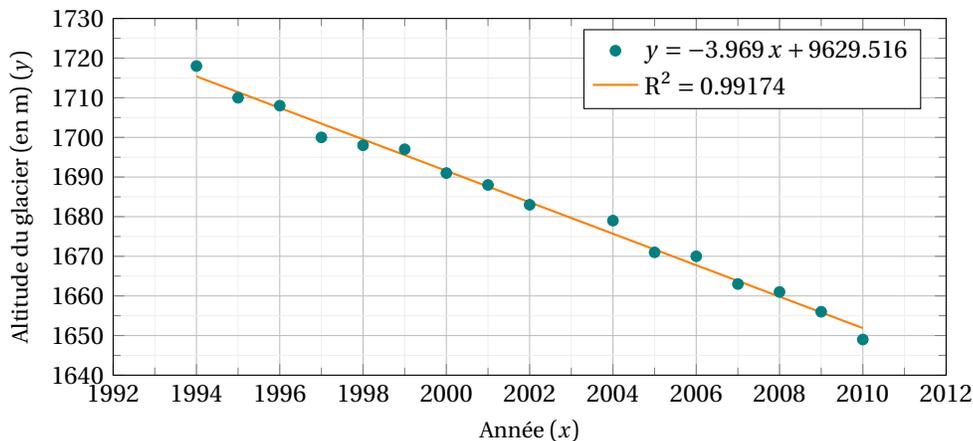
Information(s)

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

Code L^AT_EX

```
\begin{tikzpicture}
  \begin{axis}[<options des axes, non présentées ici...>]
    \addplot[teal, only marks] table{
      X Y
      1994 1718 1995 1710 1996 1708 1997 1700 1998 1698 1999 1697 2000 1691 2001 1688
      2002 1683 2004 1679 2005 1671 2006 1670 2007 1663 2008 1661 2009 1656 2010 1649
    };
    \def\LLX{1994,1995,1996,1997,1998,1999,2000,2001,2002,2004,2005,2006,2007,2008,2009,2010}
    \def\LLY{1718,1710,1708,1700,1698,1697,1691,1688,1683,1679,1671,1670,1663,1661,1656,1649}
    \CalculsRegLin{\LLX}{\LLY}
    \addplot [thick,orange,domain=\LXmin:\LXmax,samples=2]{\COEFFa*x+\COEFFb};
    \addlegendentry{\$y = \fpeval{round(\COEFFa,3)}\,x + \fpeval{round(\COEFFb,3)}\$};
    \addlegendentry{\$R^2=\fpeval{round(\COEFFrd,5)}\$};
  \end{axis}
\end{tikzpicture}
```

Sortie L^AT_EX



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 L^AT_EX

```
...
\begin{tikzpicture}[<options>]
  ...
  \PointsRegLin[<clés>]{<listeX>}{<listeY>}
  ...
\end{tikzpicture}
```

Clés et options

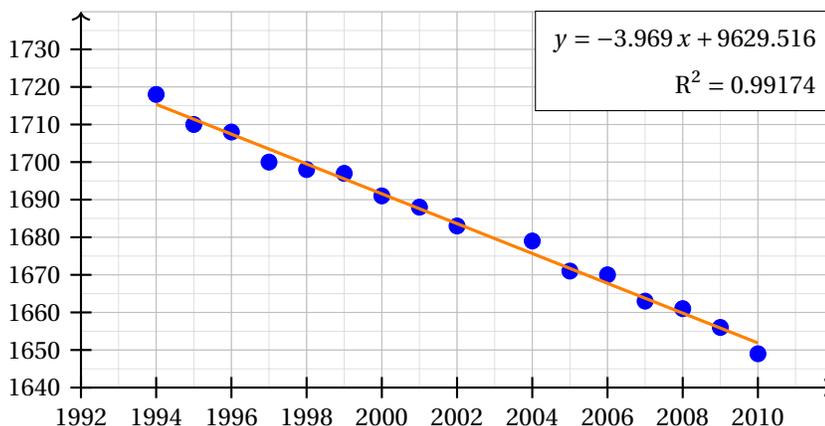
Quelques **Clés** sont disponibles pour cette commande, essentiellement pour la mise en forme du nuage :

- 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 **⟨Ox⟩** pour spécifier la valeur initiale Ox (si changement d'origine); défaut **⟨0⟩**
- la clé **⟨Oy⟩** pour spécifier la valeur initiale Oy (si changement d'origine). défaut **⟨0⟩**

Code L^AT_EX

```
\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,->] (0,0)--(20,0) ;
  \draw[thick,->] (0,0)--(0,100) ;
  \foreach \x in {1992,1994,...,2010} \draw[thick] ({\x-1992},4pt)--({\x-1992},-4pt) node[below] {$\x$} ;
  \foreach \y in {1640,1650,...,1730} \draw[thick] (4pt,{\y-1640})--(-4pt,{\y-1640}) node[left] {$\y$} ;
  \def\LLX{1994,1995,1996,1997,1998,1999,2000,2001,2002,2004,2005,2006,2007,2008,2009,2010}
  \def\LLY{1718,1710,1708,1700,1698,1697,1691,1688,1683,1679,1671,1670,1663,1661,1656,1649}
  \def\Ox{1992}\def\Oy{1640}
  \CalculsRegLin{\LLX}{\LLY}
  \PointsRegLin[Ox=1992,Oy=1640,Couleur=blue,Taille=3pt]{\LLX}{\LLY}
  \draw[orange,very thick,samples=2,domain=\LXmin:\LXmax] plot ({\x-\Ox},{\COEFFa*{\x}+\COEFFb-\Oy}) ;
  \matrix [draw,fill=white,below left] at (current bounding box.north east) {
    \node {$y = \fpeval{\round{\COEFFa,3}}\x + \fpeval{\round{\COEFFb,3}}$} ; \\
    \node {$R^2=\fpeval{\round{\COEFFrd,5}}$} ; \\
  };
\end{tikzpicture}
```

Sortie L^AT_EX



19 Statistiques à deux variables

19.1 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 `tikzpicture`, va donc :

- préciser les informations utiles comme `xmin`, `xmax`, `Ox`, `xgrille`, etc
- proposer des commandes (sans se soucier des *translations*!) pour :
 - tracer une grille (principale et/ou secondaire);
 - tracer les axes (avec légendes éventuelles) et éventuellement les graduer;

En utilisant les commandes de régression linéaire du paragraphe précédent, il sera de plus possible (sans calculs!) de :

- représenter le nuage de points;
- placer le point moyen;
- tracer la droite d'ajustement (obtenue par `ProfLycee`) ou une autre courbe.

📄 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 L^AT_EX

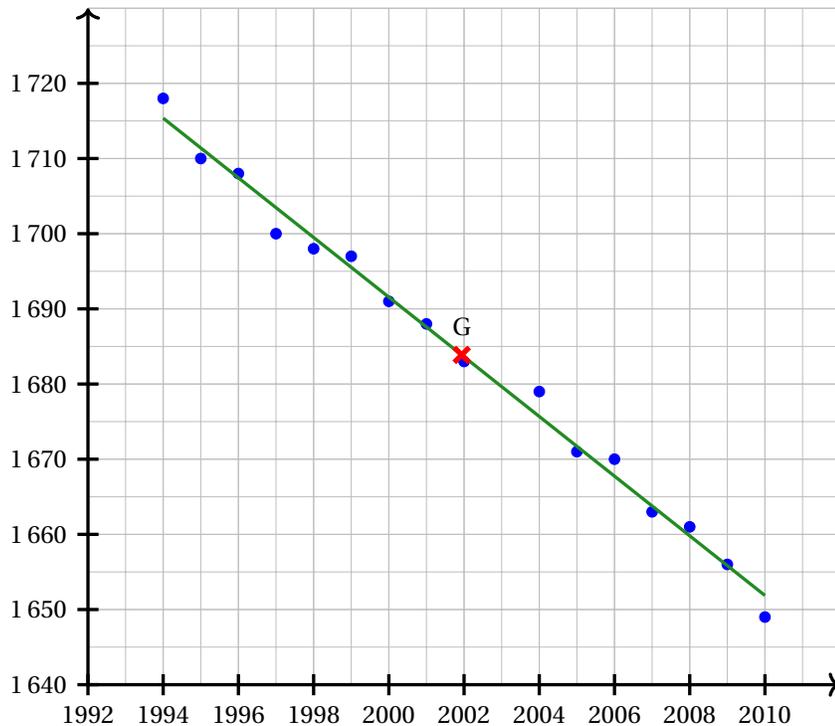
```
%Listes et calculs
\def\LLX{1994,1995,1996,1997,1998,1999,2000,2001,2002,2004,2005,2006,2007,2008,2009,2010}
\def\LLY{1718,1710,1708,1700,1698,1697,1691,1688,1683,1679,1671,1670,1663,1661,1656,1649}
\CalculsRegLin{\LLX}{\LLY}
```

📄 Code L^AT_EX

```
%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}
```

📄 Code L^AT_EX

```
%tracé avec options fenêtre par défaut
\begin{tikzpicture}%
  [...]                                               %paramètres
  \FenetreSimpleTikz<Annee>{1992,1994,...,2010}{1640,1650,...,1720} %fenêtre "simple"
  \NuagePointsTikz{\LLX}{\LLY}                       %nuage
  \CourbeTikz[line width=1.25pt,ForestGreen,samples=2] %
    {\COEFFa*\x+\COEFFb}{\LXmin:\LXmax}              %droite de régression
  \PLnuageptmoy                                       %point moyen
\end{tikzpicture}
```



19.2 Commandes, clés et options

Information(s)

Les **paramètres** nécessaires à la bonne utilisation des commandes suivantes sont à déclarer directement dans l'environnement `\tikzpicture`, seules versions « x » sont présentées ici :

- `\xmin`, stockée dans `\xmin`; défaut `{-3}`
- `\xmax`, stockée dans `\xmax`; défaut `{3}`
- `\Ox`, stockée dans `\axeOx`, origine de l'axe (Ox); défaut `{0}`
- `\xgrille`, stockée dans `\xgrille`, graduation principale; défaut `{1}`
- `\xgrilles`, stockée dans `\xgrilles`, graduation secondaire. défaut `{0.5}`

La fenêtre d'affichage (de sortie) sera donc *portée* par le rectangle de coins $(xmin; ymin)$ et $(xmax; ymax)$; ce qui correspond en fait à la fenêtre TikZ *portée* par le rectangle de coins $(xmin - Ox; ymin - Oy)$ et $(xmax - Ox; ymax - Oy)$.

Les commandes ont – pour certaines – pas mal de **clés** pour des réglages fins, mais dans la majorité des cas elles ne sont pas forcément *utiles*.

Information(s)

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

Code L^AT_EX

```
%...code tikz
\GrilleTikz[<options>][<options grille ppale>][<options grille second.>]
```

Clés et options

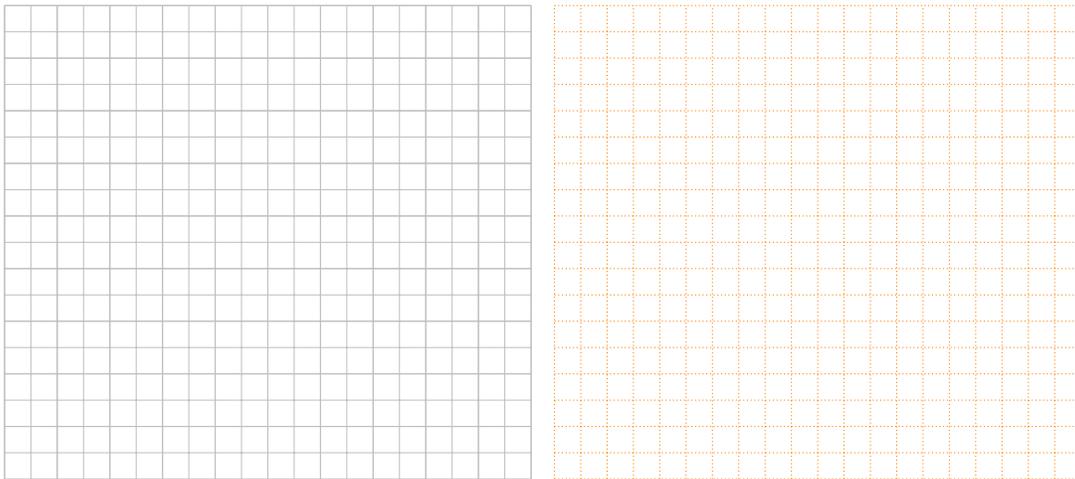
Cette commande permet de tracer une grille principale et/ou une grille secondaire :

- 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 L^AT_EX

```
\begin{tikzpicture}%
[x=0.35cm,y=0.07cm,%
Ox=1992,xmin=1992,xmax=2012,xgrille=2,xgrilles=1,%
Oy=1640,ymin=1640,ymax=1730,ygrille=10,ygrilles=5]
\GrilleTikz
\end{tikzpicture}
~~
\begin{tikzpicture}%
[x=0.35cm,y=0.07cm,%
Ox=1992,xmin=1992,xmax=2012,xgrille=2,xgrilles=1,%
Oy=1640,ymin=1640,ymax=1730,ygrille=10,ygrilles=5]
\GrilleTikz[Affp=false][orange,densely dotted]
\end{tikzpicture}
```

⊖ Sortie L^AT_EX



</> Code L^AT_EX

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

📎 Clés et options

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

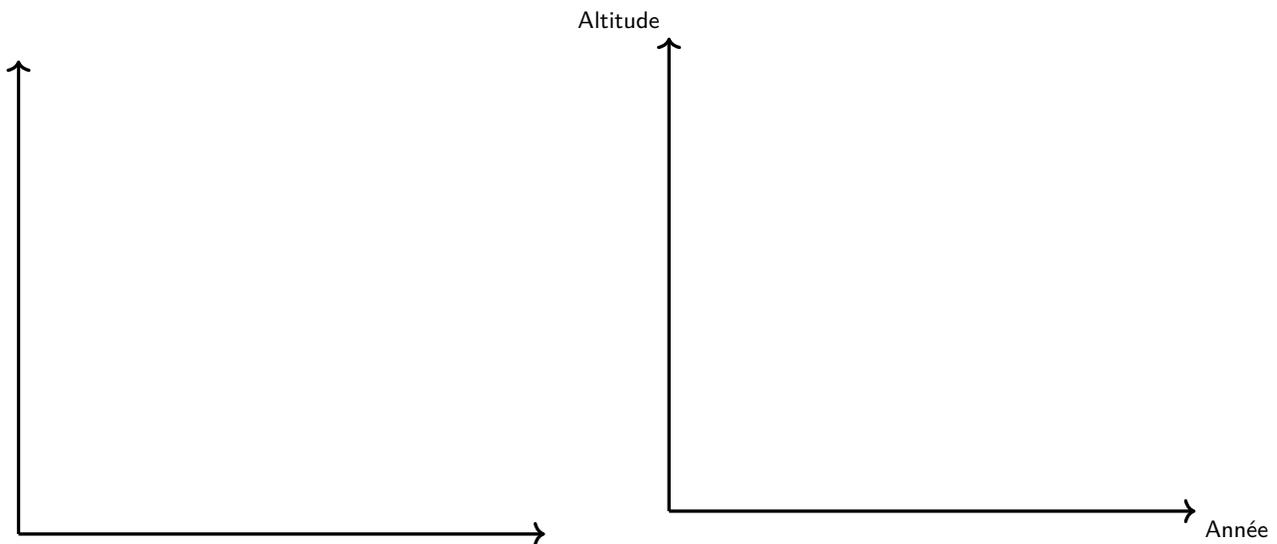
- **Épaisseur** qui est l'épaisseur des traits; défaut **1.25pt**
- **Police** qui est le style des labels des axes; défaut **(\normalsize\normalfont)**
- **Labelx** qui est le label de l'axe (Ox); défaut **(x)**
- **Labely** qui est le label de l'axe (Oy); défaut **(y)**
- **AffLabel** qui est le code pour préciser quels labels afficher, entre **(x)**, **(y)** ou **(xy)**; défaut **(vide)**
- **PosLabelx** pour la position du label de (Ox) en bout d'axe; défaut **(right)**
- **PosLabely** pour la position du label de (Oy) en bout d'axe; défaut **(above)**
- **EchelleFleche** qui est l'échelle de la flèche des axes; défaut **(1)**
- **TypeFleche** qui est le type de la flèche des axes. défaut **(>)**

</> Code L^AT_EX

```
%code tikz
\AxesTikz

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

⊖ Sortie L^AT_EX



</> Code L^AT_EX

```
%...code tikz
\AxexTikz[<options>]{valeurs}
\AxexTikz[<options>]{valeurs}
```

🔗 Clés et options

Ces commande permet de tracer les graduations des axes, avec des **<clés>** identiques pour les deux directions :

- **<Épaisseur>** qui est l'épaisseur des graduations; défaut **<1.25pt>**
- **<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 **<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>**

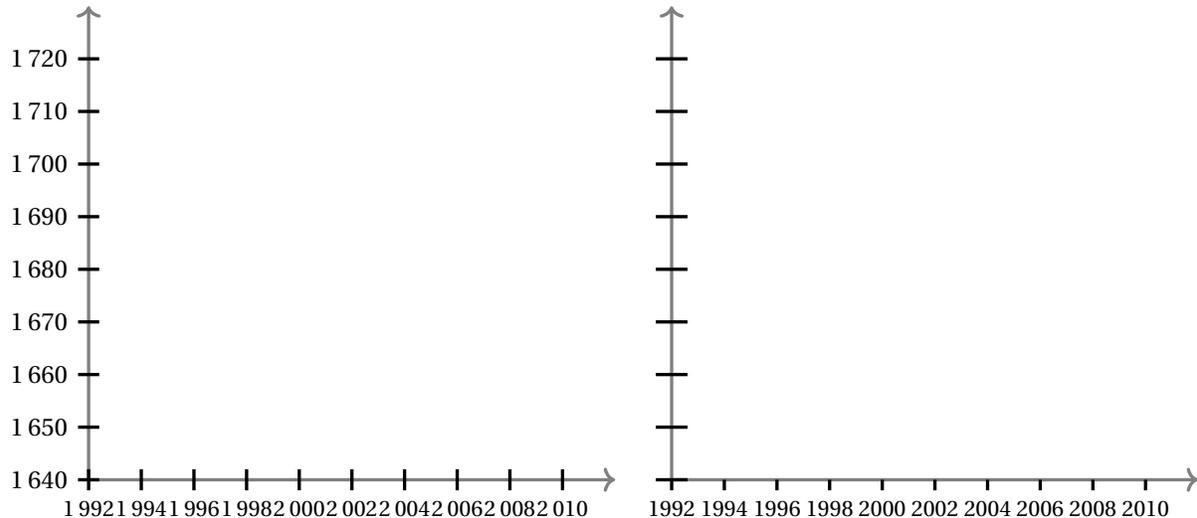
</> Code L^AT_EX

```
%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

Sortie L^AT_EX



19.3 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 L^AT_EX

```
%code tikz
\FenetreTikz           %on restreint les tracés
\FenetreSimpleTikz<options axe Ox>{liste abscisses}<options axe Oy>{liste ordonnées}
```

19.4 Interactions avec PLreglin

Code L^AT_EX

```
%.code tikz
\NuagePointsTikz[<options>]{listeX}{listeY}
```

Clés et options

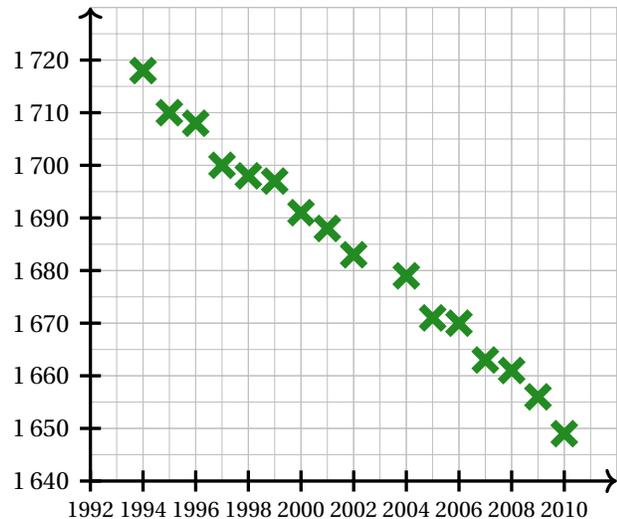
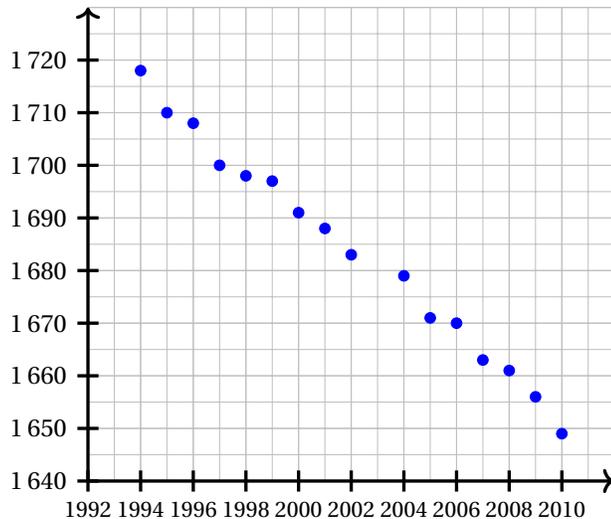
Cette commande, liée à la commande `\CalculsRegLin` permet de représenter le nuage de points associé aux deux listes, avec les **(clés)** suivantes :

- **(Taille)** qui est la taille des points du nuage; défaut **(2pt)**
- **(Style)** parmi **(o)** (rond) ou **(x)** (croix) ou **(+)** (plus); défaut **(o)**
- **(Couleur)** qui est la couleur (éventuellement **(couleurA/couleurB)** pour les ronds). défaut **(blue)**

</> Code L^AT_EX

```
\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}
```

⊙ Sortie L^AT_EX



</> Code L^AT_EX

```
%...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)** :

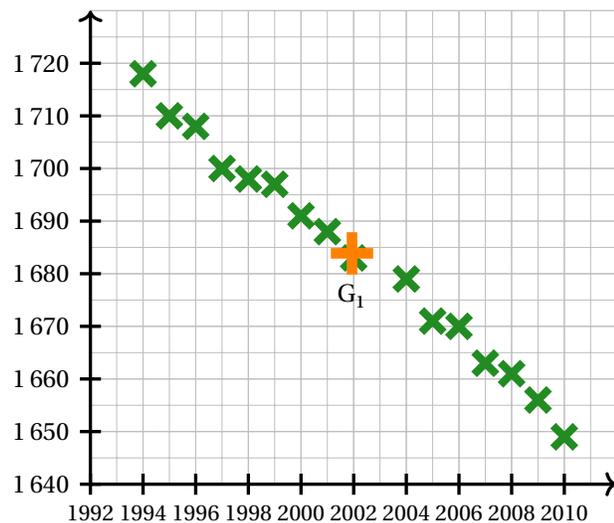
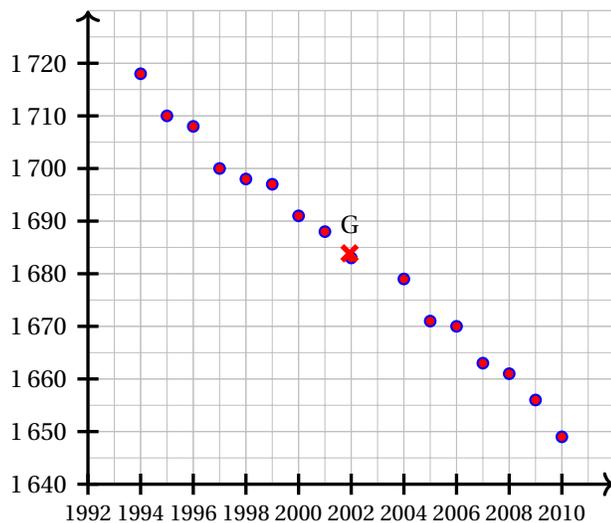
- **<Police>**, comme précédemment; défaut `(\normalsize\normalfont)`;
- **<Taille>**, taille du point moyen; défaut `(4pt)`
- **<Couleur>**, couleur du point moyen; défaut `(red)`
- **<Style>** parmi `(o)` (rond) ou `(x)` (croix) ou `(+)` (plus); défaut `(o)`
- **<xg>**, abscisse du point moyen, récupérable via `\PLRegLin`; défaut `(\LXmoy)`
- **<yg>**, ordonnée du point moyen, récupérable via `\PLRegLin`; 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 L^AT_EX

```
\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}
```

⊖ Sortie L^AT_EX



</> Code L^AT_EX

```
%. . . 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 :

- **(optionnels)** qui sont - en TikZ - les paramètres du tracé ;
- le premier mandataire, est - en langage TikZ - l'expression de la fonction à tracer, donc avec \x comme variable ;
- le second mandataire 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!*

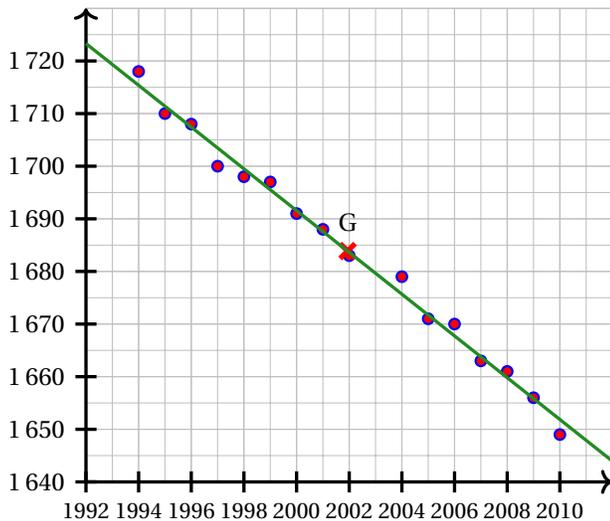
🔗 Information(s)

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

</> Code L^AT_EX

```
\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} \NuagePointsTikz  
  \CourbeTikz[line width=1.25pt,ForestGreen,samples=2]{\COEFFa*\x+\COEFFb}{\xmin:\xmax}  
\end{tikzpicture}
```

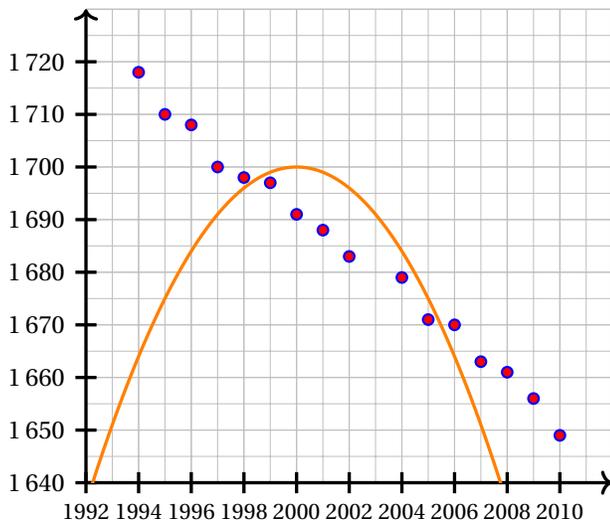
⊖ Sortie L^AT_EX



</> Code L^AT_EX

```
\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} \FenetreTikz %on fixe la fenetre  
  \CourbeTikz[line width=1.25pt,orange,samples=500]{-(\x-2000)*(\x-2000)+1700}{\xmin:\xmax}  
\end{tikzpicture}
```

⊖ Sortie L^AT_EX



20 Boîtes à moustaches

20.1 Introduction

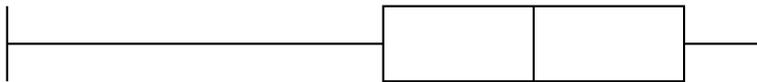
💡 Idée(s)

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 L^AT_EX

```
\begin{tikzpicture}
  \BoiteMoustaches{10/15/17/19/20}
\end{tikzpicture}
```



📄 Information(s)

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

20.2 Clés et options

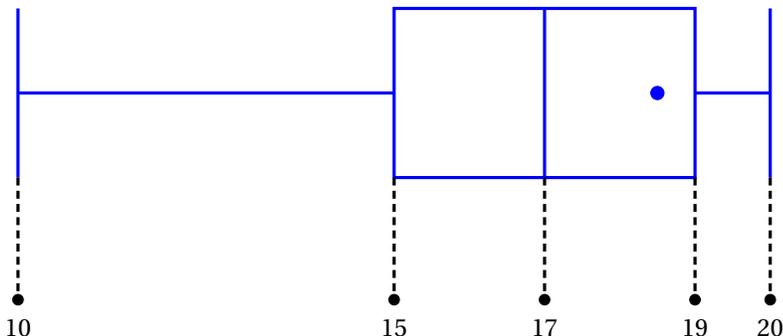
🔑 Clés et options

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 **⟨Épaisseur⟩** qui est l'épaisseur des traits de la boîte; défaut **⟨thick⟩**
- la clé **⟨Remplir⟩** qui est la couleur de remplissage de la boîte; défaut **⟨white⟩**
- le booléen **⟨AffMoyenne⟩** qui permet d'afficher ou non la moyenne (sous forme d'un point); défaut **⟨false⟩**
- le booléen **⟨Pointilles⟩** qui permet d'afficher des pointillés au niveau des paramètres; défaut **⟨false⟩**
- le booléen **⟨Valeurs⟩** qui permet d'afficher les valeurs des paramètres au niveau des abscisses. défaut **⟨false⟩**

</> Code L^AT_EX

```
\begin{tikzpicture}
  \BoiteMoustaches[Epaisseur=very thick,Moyenne=18.5,Couleur=blue,AffMoyenne,%
  Pointilles,Valeurs,Hauteur=2.25,Elevation=2.75]{10/15/17/19/20}
\end{tikzpicture}
```

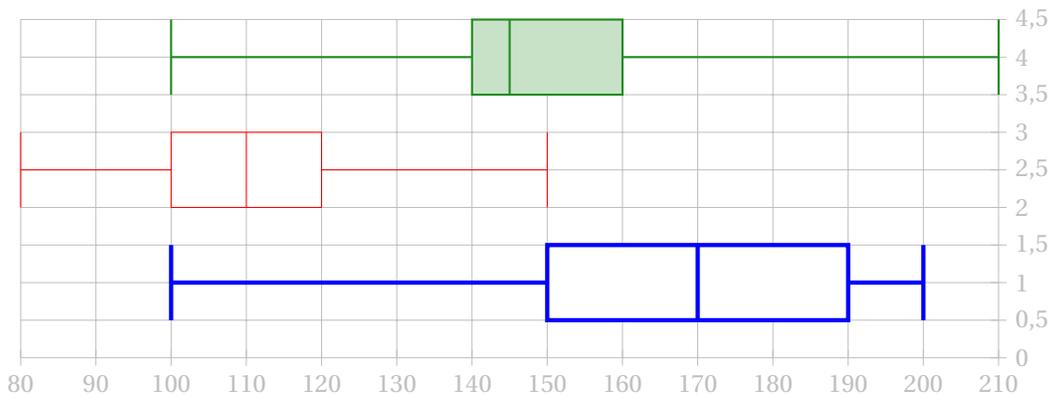


</> Code L^AT_EX

%une grille a été rajoutée pour visualiser la "position verticale"

```
\begin{center}
  \begin{tikzpicture}[x=0.1cm]
    \BoiteMoustaches[Epaisseur=ultra thick,Couleur=blue]{100/150/170/190/200}
    \BoiteMoustaches[Epaisseur=thin,Elevation=2.5,Couleur=red]{80/100/110/120/150}
    \BoiteMoustaches[Elevation=4,Couleur=ForestGreen,Remplir=ForestGreen!25]{100/140/145/160/210}
  \end{tikzpicture}
\end{center}
```

⊖ Sortie L^AT_EX



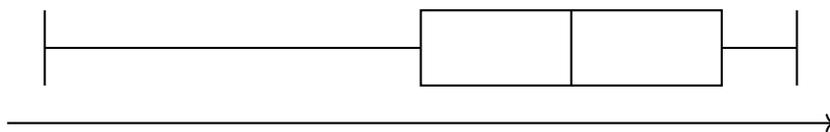
20.3 Commande pour placer un axe horizontal

💡 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.

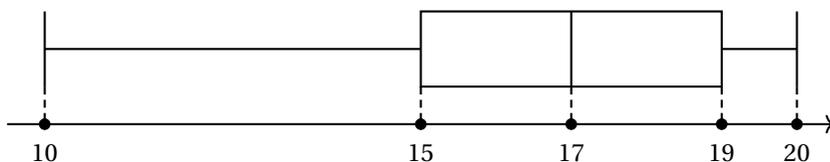
</> Code L^AT_EX

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



</> Code L^AT_EX

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



🔗 Clés et options

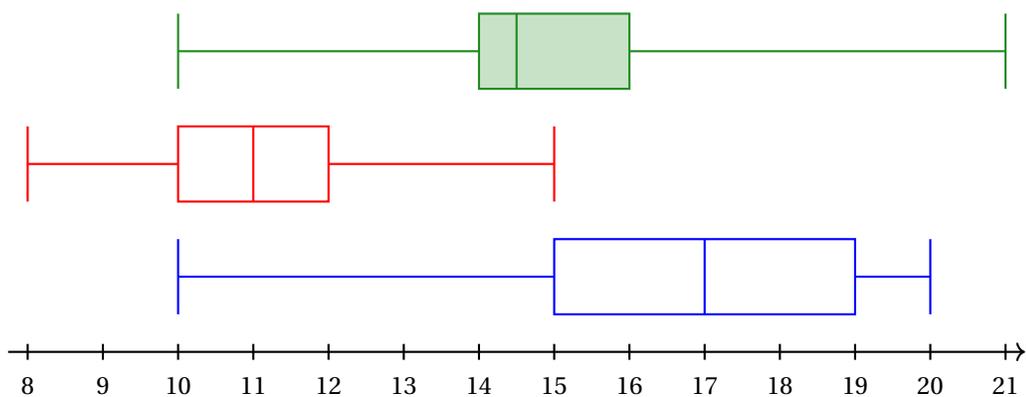
Quelques **clés** sont disponibles pour cette commande :

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

défaut **⟨0.1⟩**
défaut **⟨thick⟩**

🔗 Code L^AT_EX

```
\begin{tikzpicture}
  \BoiteMoustachesAxe[Min=8,Max=21,AffValeurs,Valeurs={8,9,...,21},Elargir=0.02]
  \BoiteMoustaches[Moyenne=18.5,Couleur=blue]{10/15/17/19/20}
  \BoiteMoustaches[Elevation=2.5,Couleur=red]{8/10/11/12/15}
  \BoiteMoustaches[Elevation=4,Couleur=ForestGreen,Remplir=ForestGreen!25]{10/14/14.5/16/21}
\end{tikzpicture}
```



🔗 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 **⟨élevations⟩** de 1 puis 2,5 puis 4 etc

Septième partie

Outils pour les probabilités

21 Calculs de probabilités

21.1 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 `xintexpr`, grâce à `siunitx`.

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

21.2 Calculs « simples »

🔗 Code L^AT_EX

```
%loi binomiale B(n,p)
\CalcBinomP{n}{p}{k}           %P(X=k)
\CalcBinomC{n}{p}{a}{b}       %P(a<=X<=b)

%loi de Poisson P(l)
\CalcPoissP{l}{k}             %P(X=k)
\CalcPoissC{l}{a}{b}         %P(a<=X<=b)

%loi géométrique G(p)
\CalcGeomP{p}{k}             %P(X=k)
\CalcGeomC{p}{a}{b}         %P(a<=X<=b)

%loi hypergéométrique H(N,n,m)
\CalcHypergeomP{N}{n}{m}{k}  %P(X=k)
\CalcHypergeomP{N}{n}{m}{a}{b} %P(a<=X<=b)

%loi normale N(m,s)
\CalcNormC{m}{s}{a}{b}       %P(a<=X<=b)

%loi exponentielle E(l)
\CalcExpoC{l}{a}{b}         %P(a<=X<=b)
```

🔑 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 ($P(a \leq X \leq b)$).

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

</> Code L^AT_EX

```
% X -> B(5,0.4)
$P(X=3) \approx \CalcBinomP{5}{0.4}{3}$.
$P(X\leqslant 1) \approx \CalcBinomC{5}{0.4}{*}{1}$.

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

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

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

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

⊖ Sortie L^AT_EX

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

</> Code L^AT_EX

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

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

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

Sortie L^AT_EX

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

21.3 Complément avec sortie « formatée »

Idée(s)

L'idée est ensuite de formater le résultat obtenu par `\xintexpr`, pour un affichage homogène. L'utilisateur peut donc utiliser « sa » méthode pour formater les résultats obtenus par `\xintexpr`!

Code L^AT_EX

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

Sortie L^AT_EX

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

Idée(s)

Le package `ProfLycee` propose – en complément – des commandes pour formater, grâce à `siunitx`, le résultat. Les commandes sont dans ce cas préfixées par `\num` au lieu de `\calc` :

- formatage sous forme décimale *pure* : 0,00... ;
- formatage sous forme scientifique : $n, \dots \times 10^{\dots}$.

Code L^AT_EX

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

🔑 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 L^AT_EX

```
% X -> 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 -> B(100,0.02)
$P(X=10) \approx \BinomP[7]{100}{0.02}{10} \approx \BinomP*[7]{100}{0.02}{10}$ .
$P(15\leqslant X\leqslant25) \approx \BinomC[10]{100}{0.02}{15}{25} \approx
- \BinomC*[10]{100}{0.02}{15}{25}$ .
% H -> H(50,10,5)
$P(W=4) \approx \HypergeomP[5]{50}{10}{5}{4}$ .
$P(1\leqslant W\leqslant3) \approx \HypergeomC[4]{50}{10}{5}{1}{3}$ .
% Z -> E(0,001)$ :
$P(Z\geqslant400) \approx \ExpoC{0.001}{400}{*}$ .
$P(300\leqslant Z\leqslant750) \approx \ExpoC{0.001}{300}{750}$ .
% T -> P(5)
$P(T=3) \approx \PoissonP{5}{3}$ .
$P(T\geqslant2) \approx \PoissonC[4]{5}{2}{*}$ .
```

📄 Sortie L^AT_EX

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

$P(Y \geq 600) \approx 0,0478$.

$P(500 \leq Y \leq 600) \approx 0,9044$.

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

$P(X = 10) \approx 0,0000288 \approx 2,88 \times 10^{-5}$.

$P(15 \leq X \leq 25) \approx 0,000000017 \approx 1,7 \times 10^{-9}$.

- $W \hookrightarrow \mathcal{H}(50; 10; 5)$:

$P(W = 4) \approx 0,00396$.

$P(1 \leq W \leq 3) \approx 0,6854$.

- $Z \hookrightarrow \mathcal{E}_{0,001}$:

$P(Z \geq 400) \approx 0,670$.

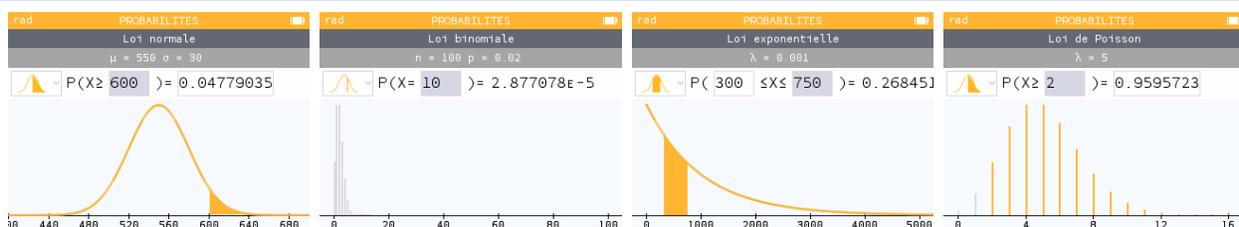
$P(300 \leq Z \leq 750) \approx 0,268$.

- $T \hookrightarrow \mathcal{P}_5$:

$P(T = 3) \approx 0,140$.

$P(T \geq 2) \approx 0,9596$.

📄 Information(s)



22 Arbres de probabilités « classiques »

22.1 Introduction

💡 Idée(s)

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

- 2×2 ou 2×3 ;
- 3×2 ou 3×3 .

Les (deux) commandes sont donc liées à un environnement `\tikzpicture`, et elles créent les nœuds de l'arbre, pour exploitation ultérieure éventuelle.

</> Code L^AT_EX

```
%commande simple pour tracé de l'arbre
\ArbreProbasTikz[<options>]{<donnees>}

%environnement pour tracé et exploitation éventuelle
\begin{EnvArbreProbasTikz}[<options>]{<donnees>}
  <code tikz supplémentaire>
\end{EnvArbreProbasTikz}
```

22.2 Options et arguments

📌 Information(s)

Les **<donnees>** seront à préciser sous forme `<sommet1>/<proba1>/<position1>,<sommet2>/<proba2>/<position2>,...` avec comme « sens de lecture » de la gauche vers la droite puis du haut vers le bas (on balaye les *sous-arbres*), avec comme possibilités :

- 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>**

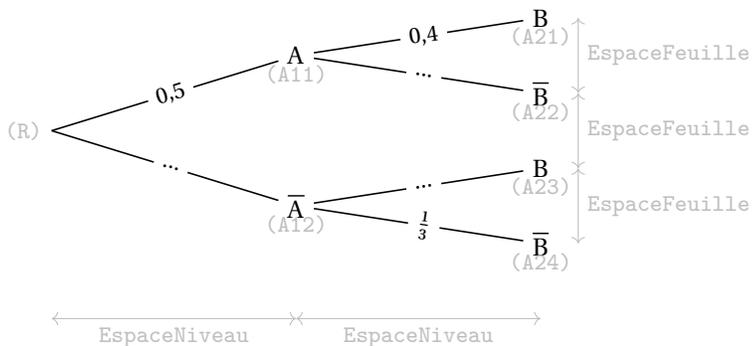
</> Code L^AT_EX

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

```
\ArbreProbasTikz{\ArbreDeuxDeux}
```

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

Sortie L^AT_EX



Information(s)

Les nœuds créés par les commandes sont :

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

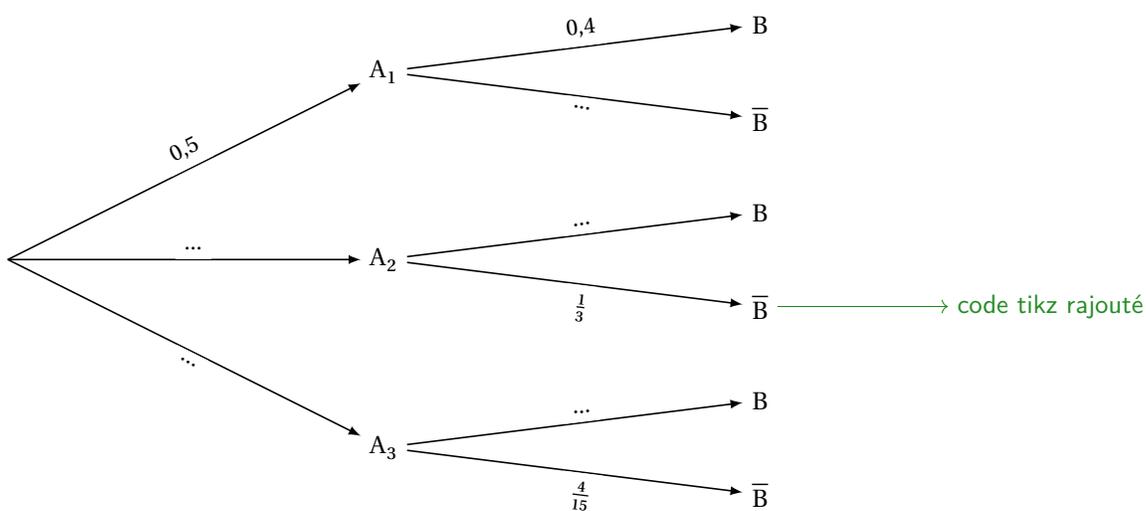
22.3 Exemples complémentaires

Code L^AT_EX

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

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

Sortie L^AT_EX



</> Code L^AT_EX

```

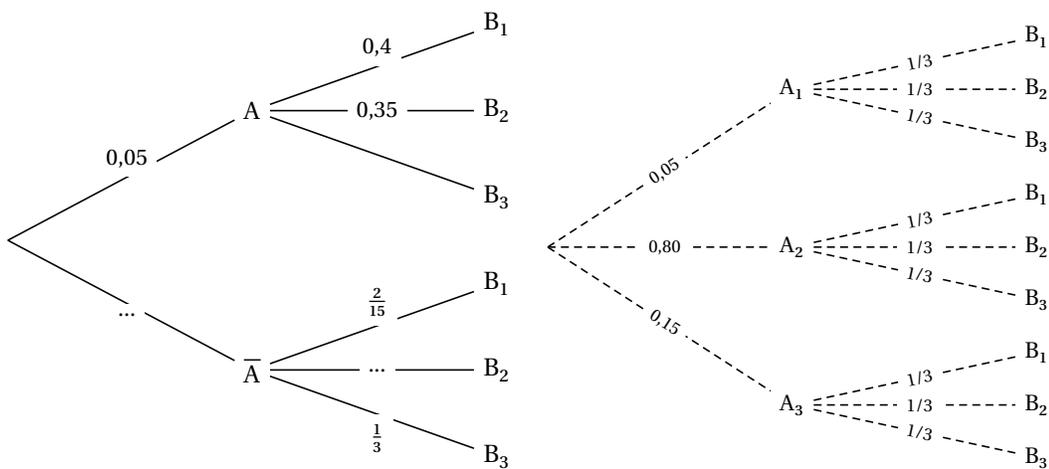
\def\ArbreDeuxTrois{
  $A$/\num{0.05}/above,
  $B_1$/\num{0.4}/above,
  $B_2$/\num{0.35}/,
  $B_3$//below,
  $\overline{A}$//.../below,
  $B_1$/\frac{2}{15}/above,
  $B_2$//.../,
  $B_3$/\frac{1}{3}/below
}
\ArbreProbasTikz[Type=2x3,InclineProbas=false,EspaceFeuille=1.15]{\ArbreDeuxTrois}

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

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

```

⊖ Sortie L^AT_EX



23 Petits schémas pour des probabilités continues

23.1 Idée

💡 Idée(s)

L'idée est de proposer des commandes pour illustrer, sous forme de schémas en TikZ, des probabilités avec des lois continues (normales et exponentielles).

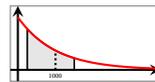
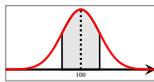
Ces « schémas » peuvent être insérés en tant que graphique explicatif, ou bien en tant que petite illustration rapide!

</> Code L^AT_EX

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

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

🕒 Sortie L^AT_EX



🔑 Clés et options

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

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

23.2 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)

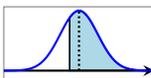
Les commandes sont donc des environnements TikZ, sans possibilité de « rajouter » des éléments. Ces petits *schémas* sont donc vraiment dédiés à *montrer* rapidement une probabilité continue, sans fioriture.

</> Code L^AT_EX

Avec centrage vertical sur l'axe des abscisses :

```
\LoiNormaleGraphe [AfficheM=false,CouleurCourbe=Blue,CouleurAire=LightBlue] <baseline=0pt> {1000} {100} {950} {*}
```

Avec centrage vertical sur l'axe des abscisses :



</> Code L^AT_EX

Avec quelques modifications :

```
\LoiNormaleGraphe [Largeur=4,Hauteur=2]{150}{12.5}{122}{160}
```

```
\medskip
```

Avec centrage vertical :

```
\LoiNormaleGraphe [Largeur=5,Hauteur=2.5]<baseline=(current bounding box.center)>{200}{5}{204}{*}
```

```
\medskip
```

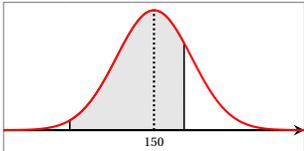
Avec centrage vertical sur l'axe des abscisses :

```
\LoiExpoGraphe [AfficheM=false,CouleurCourbe=Blue,CouleurAire=LightBlue]<baseline=0pt>{0.05}{*}{32}
```

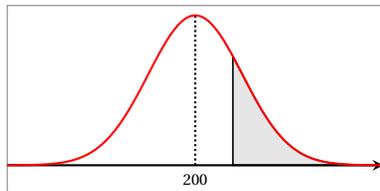
```
\medskip
```

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

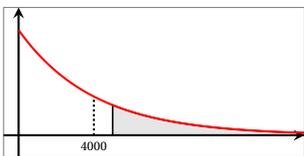
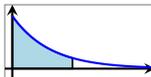
Avec quelques modifications :



Avec centrage vertical :



Avec centrage vertical sur l'axe des abscisses :



23.3 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...

Huitième partie

Outils pour l'arithmétique

24 Conversions binaire/hexadécimal/décimal

24.1 Idée

💡 Idée(s)

L'idée est de *compléter* les possibilités offertes par le package `xintbinhex`, en mettant en forme quelques conversions :

- décimal en binaire avec blocs de 4 chiffres en sortie;
- 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 L^AT_EX

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

🔗 Sortie L^AT_EX

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

24.2 Conversion décimal vers binaire

🔗 Code L^AT_EX

```
\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, mandataire, est le nombre entier à convertir.

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

Les nombres écrits en binaire sont, par défaut, présentés en bloc(s) de 4 chiffres.

</> Code L^AT_EX

```
% Conversion avec affichage de la base et par bloc de 4
 $\backslash$ ConversionDecBin{415}$
% Conversion avec affichage de la base et sans forçément des blocs de 4
 $\backslash$ ConversionDecBin*{415}$
% Conversion sans affichage de la base et par bloc de 4
 $\backslash$ ConversionDecBin[AffBase=false]{415}$
% Conversion sans affichage de la base et sans forçément des blocs de 4
 $\backslash$ ConversionDecBin*[AffBase=false]{415}$
```

⊖ Sortie L^AT_EX

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

24.3 Conversion binaire vers hexadécimal

📄 Information(s)

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 L^AT_EX

```
 $\backslash$ 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 `lATEX sinuitx`, le mieux est de positionner la commande dans un environnement mathématique.

</> Code L^AT_EX

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

⊖ Sortie L^AT_EX

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

24.4 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 L^AT_EX

```
\ConversionVersDec[<clés>]{<nombre>}
```

Clés et options

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

- la clé **BaseDp** 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 `sinitx`, le mieux est de positionner la commande dans un environnement mathématique.

Code L^AT_EX

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

Sortie L^AT_EX

```
19F16 = 1 × 162 + 9 × 161 + 15 × 160 = 41510  
1100112 = 1 × 25 + 1 × 24 + 0 × 23 + 0 × 22 + 1 × 21 + 1 × 20 = 5110  
1100112 = 1 × 25 + 1 × 24 + 1 × 21 + 1 × 20 = 5110  
AC0DC16 = 70473210  
AC0DC16 = 10 × 164 + 12 × 163 + 13 × 161 + 12 × 160 = 70473210
```

25 Conversion « présentée » d'un nombre en décimal

25.1 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 `\tikzset{overlay}`, donc on pourra déclarer – si ce n'est pas fait – dans le préambule, la commande qui suit.

</> Code L^AT_EX

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

25.2 Code et clés

</> Code L^AT_EX

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

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

📌 Information(s)

La « table » , qui est géré par `\array` est inséré dans un `\ensuremath`, donc les `\$...$` ne sont pas utiles.

</> Code L^AT_EX

```
\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 **<Noeud>** pour le préfixe du nœud du premier et du dernier reste (pour utilisation en TikZ); défaut **<EEE>**
- le booléen **<Rect>** pour afficher ou non le « rectangle » des restes; défaut **<>true>**
- le booléen **<CouleurRes>** pour afficher ou non la conversion en couleur (identique au rectangle). défaut **<>false>**

</> Code L^AT_EX

```
%conversion avec changement de couleur
\ConversionDepuisBaseDix[Couleur=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) node[right] {test } ;
\end{tikzpicture}
```

⊖ Sortie L^AT_EX

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

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

$$\left\{ \begin{array}{l} 1012 = 16 \times 63 + 4 \\ 63 = 16 \times 3 + 15 \\ 3 = 16 \times 0 + 3 \end{array} \right. \Rightarrow 1012_{10} = 3F4_{16}$$

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

→ test

26 Algorithme d'Euclide pour le PGCD

26.1 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 L^AT_EX

```
\PresentationPGCD[<options>]{a}{b}
```

</> Code L^AT_EX

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

⊖ Sortie L^AT_EX

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

🚨 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]
```

26.2 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 en valeur; défaut **<true>**
- la clé **<Noeud>** qui gère le préfixe du nom du nœud TikZ du rectangle (pour exploitation ultérieure); défaut **<FFF>**
- le booléen **<CouleurResultat>** pour mettre ou non en couleur de PGCD; défaut **<false>**
- le booléen **<AfficheConclusion>** pour afficher ou non la conclusion; défaut **<true>**
- le booléen **<AfficheDelimitateurs>** pour afficher ou non les délimiteurs (accolade gauche et trait droit). défaut **<true>**

Le rectangle de mise en valeur est donc un nœud TikZ qui sera nommé, par défaut `FFF1`.

La présentation est dans un environnement `ensuremath` donc les `$. . . $` ne sont pas indispensables.

</> Code L^AT_EX

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

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

</> Code L^AT_EX

```
\PresentationPGCD[CouleurResultat,Couleur=ForestGreen]{1250}{450}.

\PresentationPGCD[CouleurResultat,Couleur=DarkBlue]{13500}{2500}.

\PresentationPGCD[Rectangle=false]{420}{540}.

\medskip

D'après l'algorithmme d'Euclide, on a  $\left[ \right]$ 
- \PresentationPGCD[Couleur=LightSkyBlue,AfficheConclusion=false,AfficheDelimitteurs=false]{123456789}{9876}
- \right.$
\begin{tikzpicture}
  \draw[overlay,LightSkyBlue,thick,<-] (FFF1.east) to[bend right] ++ (2cm,0.75cm) node[right] {dernier reste
  - non nul} ;
\end{tikzpicture}
```

$$\left\{ \begin{array}{l} 1250 = 450 \times 2 + 350 \\ 450 = 350 \times 1 + 100 \\ 350 = 100 \times 3 + \boxed{50} \\ 100 = 50 \times 2 + 0 \end{array} \right. \Rightarrow \text{PGCD}(1250;450) = 50.$$
$$\left\{ \begin{array}{l} 13500 = 2500 \times 5 + 1000 \\ 2500 = 1000 \times 2 + \boxed{500} \\ 1000 = 500 \times 2 + 0 \end{array} \right. \Rightarrow \text{PGCD}(13500;2500) = 500.$$
$$\left\{ \begin{array}{l} 420 = 540 \times 0 + 420 \\ 540 = 420 \times 1 + 120 \\ 420 = 120 \times 3 + 60 \\ 120 = 60 \times 2 + 0 \end{array} \right. \Rightarrow \text{PGCD}(420;540) = 60.$$

D'après l'algorithmme d'Euclide, on a

$$\left| \begin{array}{r} 123456789 = 9876 \times 12500 + 6789 \\ 9876 = 6789 \times 1 + 3087 \\ 6789 = 3087 \times 2 + 615 \\ 3087 = 615 \times 5 + 12 \\ 615 = 12 \times 51 + \boxed{3} \\ 12 = 3 \times 4 + 0 \end{array} \right.$$

dernier reste non nul

26.3 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 `\AfficheDelimitteurs=false` permet de choisir des délimiteurs différents.

</> Code L^AT_EX

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

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

Neuvième partie

Outils divers et variés

27 Fractions, ensembles

27.1 Fractions

💡 Idée(s)

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

🔗 Code L^AT_EX

```
\ConversionFraction[<option>]{<argument>}
```

🔑 Clés et options

Peu d'options pour ces commandes :

- le premier argument, optionnel, permet de spécifier le mode de sortie de la fraction [t] pour tfrac et [d] pour dfrac ;
- le second, mandataire, est le calcul ou la division à convertir.

À noter que la macro est dans un bloc `\ensuremath` donc les `\$...$` ne sont pas nécessaires.

🔗 Code L^AT_EX

```
\ConversionFraction{111/2145}
\ConversionFraction{111/3}
 $\frac{111}{2145}=\ConversionFraction{111/2145}$ 
 $\frac{3}{15}=\ConversionFraction[]{3/15}$ 
 $\tfrac{3}{15}=\ConversionFraction[t]{3/15}$  %formatage en \tfrac
 $\dfrac{3}{15}=\ConversionFraction[d]{3/15}$  %formatage en \dfrac
 $\dfrac{0,42}{0,015}=\ConversionFraction[d]{0.42/0.015}$ 
 $\dfrac{0,41}{0,015}=\ConversionFraction[d]{0.41/0.015}$ 
 $\dfrac{1}{7}+\dfrac{3}{8}=\ConversionFraction[d]{1/7+3/8}$ 
 $\ConversionFraction[d]{1+1/2}$ 
 $\ConversionFraction{0.1/0.7+30/80}$ 
```

🔗 Sortie L^AT_EX

$\frac{37}{715}$	$\frac{0,42}{0,015} = 28$
37	$\frac{0,41}{0,015} = \frac{82}{3}$
$\frac{111}{2145} = \frac{37}{715}$	$\frac{1}{7} + \frac{3}{8} = \frac{29}{56}$
$\frac{3}{15} = \frac{1}{5}$	$\frac{3}{2}$
$\frac{3}{15} = \frac{1}{5}$	$\frac{29}{56}$
$\frac{3}{15} = \frac{1}{5}$	

📌 Information(s)

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

27.2 Ensembles

💡 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 L^AT_EX

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

🔗 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, mandataire, est la liste des éléments, séparés par /.

</> Code L^AT_EX

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

⊖ Sortie L^AT_EX

```
{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...

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

28.1 Idée

💡 Idée(s)

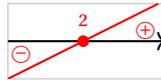
L'idée est d'obtenir une commande pour tracer (en TikZ) un petit schéma pour *visualiser* le signe d'une fonction affine ou d'un trinôme.

Le code est très largement inspiré de celui du package `tnsana` même si la philosophie est légèrement différente.

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

</> Code L^AT_EX

```
%code tikz
\MiniSchemaSignes [<clés>]
```



28.2 Commandes

</> Code L^AT_EX

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

</> Code L^AT_EX

```
... {\tikz [<options>] \MiniSchemaSignes [<clés>]}...
```

🔗 Clés et options

Plusieurs **<Clés>** 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 <Hauteur> qui est la hauteur du schéma; | défaut <1> |
| — un booléen <Cadre> qui affiche un cadre autour du schéma. | défaut <>true> |

🔗 Clés et options

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

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

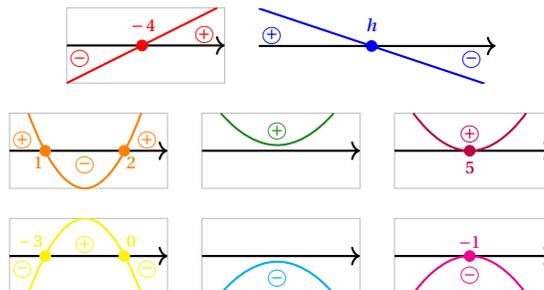
</> Code L^AT_EX

```

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

```

⊖ Sortie L^AT_EX

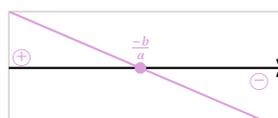


</> Code L^AT_EX

```

%code tikz
\MiniSchemaSignes[Largeur=3.5,Hauteur=1.5,Code=da-,Racines=\tfrac{-b}{a},Couleur=Plum]

```



28.3 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 `\aidesignetkztabPL` 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 L^AT_EX

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

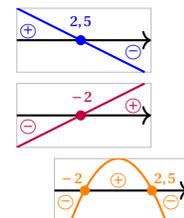
À noter que si l'un des arguments optionnels (le n°3 et/ou le n°4) sont utilisés, il vaut mieux préciser les 2 !

</> Code L^AT_EX

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

🔍 Sortie L^AT_EX

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



29 Style « main levée » en TikZ

29.1 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.

🔗 Code L^AT_EX

```
\tikzset{%
  mainlevee/.style args={#1et#2}{decorate,decoration={random steps, segment length=#1,amplitude=#2}},
  mainlevee/.default={5mm et 0.6pt}
}
```

29.2 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 **⟨mainlevee=#1 et #2⟩** :

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

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

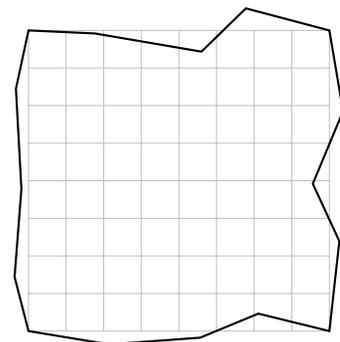
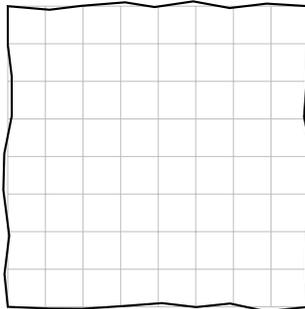
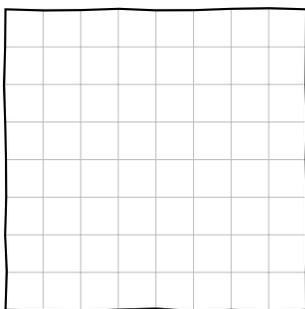
🔗 Code L^AT_EX

```
%la grille a été rajoutée pour la sortie
\begin{tikzpicture}
  \draw[thick,mainlevee] (0,0) ---- (4,0) ---- (0,4) ---- (-4,0) --cycle ;
\end{tikzpicture}

\begin{tikzpicture}
  \draw[thick,mainlevee=5mm et 2pt] (0,0) ---- (4,0) ---- (0,4) ---- (-4,0) --cycle ;
\end{tikzpicture}

\begin{tikzpicture}
  \draw[thick,mainlevee=10mm et 3mm] (0,0) ---- (4,0) ---- (0,4) ---- (-4,0) --cycle ;
\end{tikzpicture}
```

📄 Sortie L^AT_EX



30 Écriture d'un trinôme, trinôme aléatoire

30.1 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 ± 1 .

En combinant avec le package `\xfp` et fonction de générateur d'entiers aléatoires, on peut de ce fait proposer une commande pour générer aléatoirement des trinômes à coefficients entiers (pour des fiches d'exercices par exemple).

L'affichage des monômes est géré par le package `\siunitx` et le tout est dans un environnement `\ensuremath`.

</> Code L^AT_EX

```
\EcritureTrinome[<options>]{a}{b}{c}
```

</> Code L^AT_EX

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

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

30.2 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 L^AT_EX

Avec $\$a\$$ entre 1 et 5 (et signe aléatoire) puis $\$b\$$
— entre $\$-2\$$ et 7 puis $\$c\$$ entre $\$-10\$$ et 20 :

```
\$f(x)=\EcritureTrinome[Alea]{1,5}{-5,5}{-10,10}$\  
\$g(x)=\EcritureTrinome[Alea]{1,5}{-5,5}{-10,10}$\  
\$h(x)=\EcritureTrinome[Alea]{1,5}{-5,5}{-10,10}$
```

Avec $\$a\$$ entre 1 et 10 (forcément positif) puis $\$b\$$
— entre $\$-2\$$ et 2 puis $\$c\$$ entre 0 et 4 :

```
\EcritureTrinome[Alea,Anegatif=false]{1,10}{-  
— 2,2}{0,4}\  
\EcritureTrinome[Alea,Anegatif=false]{1,10}{-  
— 2,2}{0,4}\  
\EcritureTrinome[Alea,Anegatif=false]{1,10}{-  
— 2,2}{0,4}
```

Avec a entre 1 et 5 (et signe aléatoire) puis b entre -2 et 7
puis c entre -10 et 20 :

$f(x) = 3x^2 - 5x + 6$
 $g(x) = -2x^2 - 4x + 7$
 $h(x) = -4x^2 + 2x + 9$

Avec a entre 1 et 10 (forcément positif) puis b entre -2 et 2
puis c entre 0 et 4 :

$3x^2 + 2x + 2$
 $3x^2 - 2x + 3$
 $8x^2 + x + 3$

Dixième partie

Jeux et récréations

31 PixelART via un fichier csv, en TikZ

31.1 Introduction

💡 Idée(s)

L'idée est de *proposer*, dans un environnement TikZ, une commande permettant de générer des grilles PixelART. Les données sont *lues* à partir d'un fichier csv, externe au fichier tex ou déclaré en interne grâce à l'environnement `\filecontents`.

📄 Information(s)

Avant toute chose, quelques petites infos sur les données au format csv, surtout dans l'optique de sa lecture et de son traitement par `\ProfLycee` :

- le fichier de données csv doit être formaté avec le séparateur décimal « , » ;
- des cases vides seront codées par « - ».

Le fichier csv peut être déclaré directement dans le fichier tex, grâce à l'environnement `\filecontents` (intégré en natif sur les dernières versions de \LaTeX) :

```
\begin{filecontents*}{<nomfichier>.csv}
A,B,C,D
A,B,D,C
B,A,C,D
B,A,D,C
\end{filecontents*}
```

À la compilation, le fichier `<nomfichier>.csv` sera créé automatiquement, et l'option `\[overwrite]` permet (logiquement) de propager les modifications au fichier csv.

31.2 Package csvsimple et option

📄 Information(s)

Le package *central* est ici `\csvsimple`, qui permet de lire et traiter le fichier csv.

Il est « disponible » en version $\LaTeX 2_{\epsilon}$ ou en version $\LaTeX 3$. Par défaut, `\ProfLycee` le charge en version $\LaTeX 3$, mais une **option** est disponible pour une *rétro-compatibilité* avec la version $\LaTeX 2_{\epsilon}$.

L'option `\csvii` permet de passer l'appel au package en version $\LaTeX 2_{\epsilon}$.

📄 Code \LaTeX

```
\usepackage{ProfLycee} %chargement du package version 3
%qui charge :
%\RequirePackage{expl3}
%\RequirePackage[l3]{csvsimple}

\usepackage[csvii]{ProfLycee} %chargement du package version 2
%qui charge :
%\RequirePackage[legacy]{csvsimple}
```

31.3 Exemple simple, clés et options

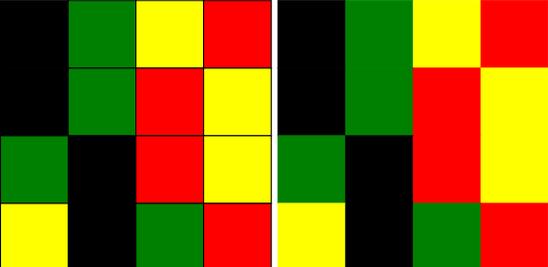
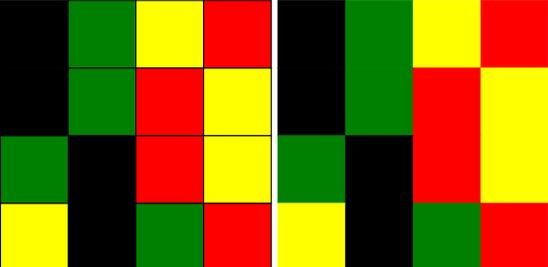
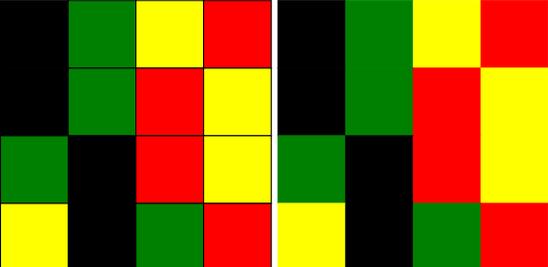
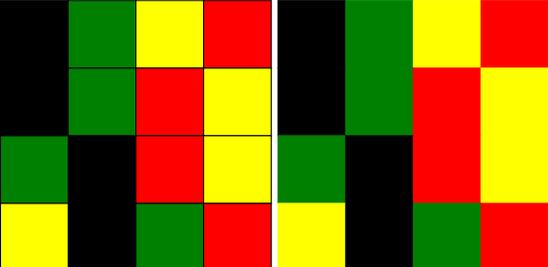
</> Code L^AT_EX

```
%déclaration du fichier csv
\begin{filecontents*}[overwrite]{basique.csv}
A,B,C,D
A,B,D,C
B,A,D,C
C,A,B,D
\end{filecontents*}

\begin{tikzpicture}%avec lettres
  \PixelArtTikz[Codes=ABCD,Style=\large\sffamily]{basique.csv}
\end{tikzpicture}
\begin{tikzpicture}%avec chiffres
  \PixelArtTikz[Codes=ABCD,Symboles={45,22,1,7},Symb,Style=\large\sffamily]{basique.csv}
\end{tikzpicture}
\begin{tikzpicture}%avec correction
  \PixelArtTikz[Codes=ABCD,Couleurs={Black,Green,Yellow,Red},Correction]{basique.csv}
\end{tikzpicture}
\begin{tikzpicture}%avec correction sans bordure
  \PixelArtTikz[Codes=ABCD,Couleurs={Black,Green,Yellow,Red},Correction,BordCases=false]{basique.csv}
\end{tikzpicture}
```

⊖ Sortie L^AT_EX

Notice			
A	B	C	D
45	22	1	7
Noir	Vert	Jaune	Rouge

A	B	C	D	45	22	1	7	
A	B	D	C	45	22	7	1	
B	A	D	C	22	45	7	1	
C	A	B	D	1	45	22	7	

📄 Information(s)

La commande `\PixelArtTikz` nécessite de connaître :

- le fichier csv à traiter;
- la liste (en fait sous forme de chaîne) des codes utilisés dans le fichier csv (comme `234679` ou `ABCDJK...`);
- la liste des symboles (éventuellement!) à afficher dans les cases s'il y a ambiguïté, comme `25,44,12` ou `AA,AB,AC`;
- la liste des couleurs (si la correction est demandée), dans le même ordre que la liste des caractères.

</> Code L^AT_EX

```
%environnement tikz
\PixelArtTikz[<clés>]{<fichier>.csv}
```

🔑 Clés et options

Quelques **⟨Clés⟩** sont nécessaires au bon fonctionnement de la commande :

- la clé **⟨Codes⟩** contient la chaîne des codes *simples* du fichier csv;
- la clé **⟨Couleurs⟩** qui contient la liste des couleurs associées;
- la clé **⟨Symboles⟩** qui contient la liste éventuelles des caractères alternatifs à afficher dans les cases;
- la clé booléenne **⟨Correction⟩** qui permet de colorier le PixelART; défaut **⟨false⟩**
- la clé booléenne **⟨Symb⟩** qui permet d'afficher les caractères *alternatifs*; défaut **⟨false⟩**
- la clé booléenne **⟨BordCases⟩** qui permet d'afficher les bords des cases de la correction; défaut **⟨true⟩**
- la clé **⟨Style⟩** qui permet de spécifier le style des caractères. défaut **⟨scriptsize⟩**

🔗 Code L^AT_EX

```
%codes simples et sans ambiguïté
%une case vide sera codée par -

\begin{filecontents*}[overwrite]{perroquet.csv}
-, -, -, -, -, -, 4, 4, 4, -, -, -, -, -
-, -, -, -, 4, 4, 1, 1, 1, 1, 4, 4, -, -, -, -
-, -, -, 4, 1, 1, 1, 1, 1, 1, 1, 1, 4, -, -, -
-, -, 4, 1, 1, 1, 1, 1, 1, 1, 1, 1, 4, -, -
-, -, 4, 1, 1, 1, 1, 1, 1, 1, 1, 1, 4, -, -
-, 4, 1, 9, 9, 1, 1, 1, 1, 1, 1, 9, 9, 1, 4, -
-, 4, 9, 9, 9, 9, 4, 4, 4, 4, 9, 9, 9, 9, 4, -
-, 4, 9, 4, 9, 9, 4, 4, 4, 4, 9, 4, 9, 9, 4, -
-, 4, 1, 9, 9, 9, 4, 4, 4, 4, 9, 9, 9, 1, 4, -
-, -, 4, 1, 1, 9, 4, 4, 4, 4, 9, 1, 1, 4, -, -
-, -, 4, 1, 1, 1, 4, 4, 4, 4, 1, 1, 1, 4, -, -
-, -, -, 4, 1, 1, 1, 4, 4, 1, 1, 1, 4, -, -, -
-, -, 4, 3, 1, 1, 1, 1, 1, 1, 1, 1, 3, 4, -, -
-, 4, 6, 3, 1, 1, 1, 1, 1, 1, 1, 1, 3, 6, 4, -
-, 4, 6, 6, 1, 1, 1, 1, 1, 1, 1, 1, 6, 6, 4, -
-, 4, 6, 6, 1, 1, 1, 1, 1, 1, 1, 1, 6, 6, 4, -
-, 4, 6, 4, 1, 1, 1, 4, 4, 1, 1, 1, 4, 6, 4, -
2, 2, 4, 2, 4, 4, 4, 2, 2, 4, 4, 4, 2, 4, 2, 2
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2
-, -, -, -, 4, 1, 1, 1, 1, 4, -, -, -, -, -
-, -, -, -, 4, 1, 1, 4, -, -, -, -, -
-, -, -, -, 4, 4, -, -, -, -, -, -, -
\end{filecontents*}

\begin{tikzpicture}[x=0.35cm,y=0.35cm]
  \PixelArtTikz[Codes=123469,Style=\ttfamily]{perroquet.csv}
\end{tikzpicture}

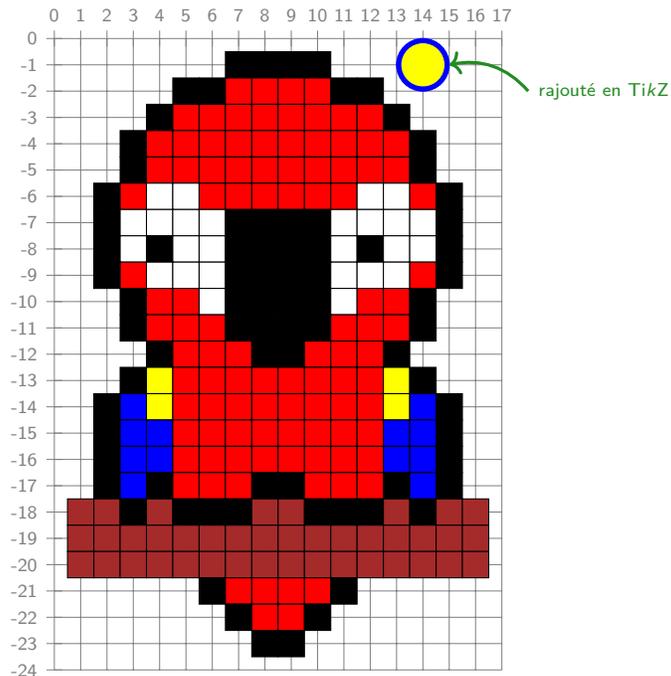
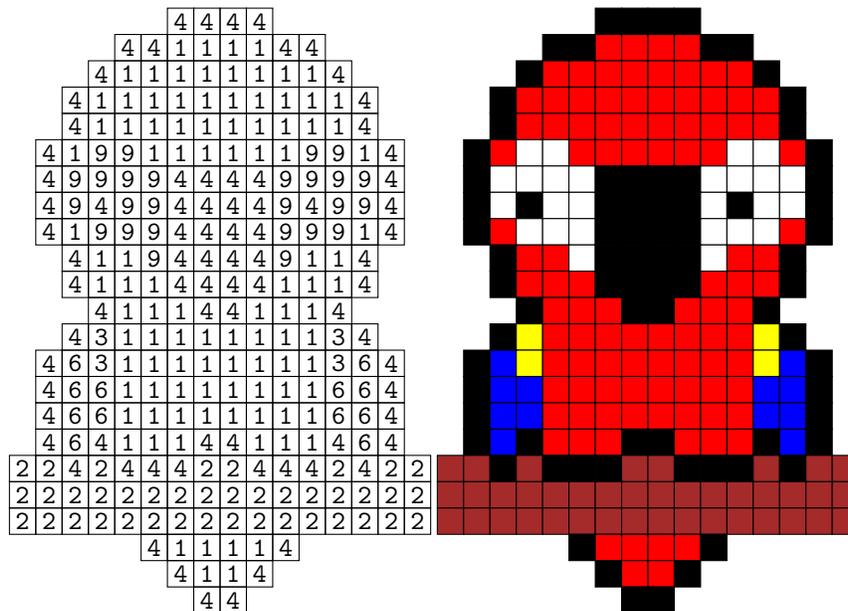
\begin{tikzpicture}[x=0.35cm,y=0.35cm]
  \PixelArtTikz[Codes=123469,Couleurs={Red,Brown,Yellow,Black,Blue,White},Correction]{perroquet.csv}
\end{tikzpicture}
```

31.4 Exemples complémentaires

📄 Information(s)

Les symboles affichés dans les cases sont situés aux nœuds de coordonnées $(c; -l)$ où l et c sont les numéros de ligne et de colonne correspondants à la position de la donnée dans le fichier csv.

Notice					
1	2	3	5	6	9
Rouge	Marron	Jaune	Noir	Bleu	Blanc



</> Code L^AT_EX

```

%code tikz et pixelart
\filldraw[Blue] (14,-1) circle[radius=1] ;
\filldraw[Yellow] (14,-1) circle[radius=0.8] ;
\draw[ForestGreen,very thick,<-] (15,-1) to[bend left=30] (18,-2) node[right,font=\scriptsize\sffamily]
- {rajouté en \TikZ} ;

```


32 SudoMaths, en TikZ

32.1 Introduction

💡 Idée(s)

L'idée est de *proposer* un environnement TikZ, une commande permettant de tracer des grilles de SudoMaths. L'environnement créé, lié à TikZ, trace la grille de SudoMaths (avec les blocs démarqués), et peut la remplir avec une liste d'éléments.

</> Code L^AT_EX

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

⊖ Sortie L^AT_EX

	a	b	c	d	e	f	g	h	i
A									
B									
C									
D									
E									
F									
G									
H									
I									

📌 Information(s)

La commande `\SudoMaths` crée donc la grille (remplie ou non), dans un environnement TikZ, c'est *c'est tout!*

Si on veut exploiter le tracé de la grille, on peut utiliser l'*environnement* `\EnvSudoMaths` dans lequel on peut rajouter toute commande en TikZ!

</> Code L^AT_EX

```
%grille "toute seule"
\SudoMaths[<options>]{<liste>}

%grille avec ajout de code
\begin{EnvSudoMaths}[<options>]{<grille>}
  <commandes tikz> ;
\end{EnvSudoMaths}
```

32.2 Clés et options

Clés et options

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

- la clé **⟨Epaisseur⟩** pour gérer l'épaisseur des traits épais; défaut **⟨1.5pt⟩**
- la clé **⟨Epaisseur⟩** pour gérer l'épaisseur des traits fins; défaut **⟨0.5pt⟩**
- la clé **⟨Unite⟩** qui est l'unité graphique de la figure; défaut **⟨1cm⟩**
- la clé **⟨CouleurCase⟩** pour la couleur (éventuelles) des cases; défaut **⟨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 formate le label des cases; défaut **⟨\normalfont\normalsize⟩**
- le booléen **⟨Legendes⟩** qui affiche ou non les légendes (H et V) des cases; défaut **⟨true⟩**
- la clé **⟨PoliceLeg⟩** qui formate le label des légendes; défaut **⟨\normalfont\normalsize⟩**
- la clé **⟨ListeLegV⟩** qui est la liste de la légende verticale; défaut **⟨ABCD...WXYZ⟩**
- la clé **⟨ListeLegH⟩** qui est la liste de la légende horizontale; défaut **⟨abcd...wxyz⟩**
- la clé **⟨DecalLegende⟩** qui est le décalage de la légende par rapport à la grille. défaut **⟨0.45⟩**

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 L^AT_EX

```
%grille 6x6 avec blocs 2x3, avec coloration de cases (présentée sous forme de "cases")
\def\grilleSuMa%
(a)* / (b)* /      /      / (c)* / (d)* $%
(e)* /      /      / (f)* / (g)* / (h)* $%
      /      / (i)* /      /      / (j)* $%
      /      / (k)* /      / (l)* / (m)* $%
(n)* /      / (o)* /      /      / (p)* $%
      /      /      / (q)* /      /      $%
}

\PLsudomaths[unite=0.75cm,nbcol=6,nbsubcol=2,nblig=6,nbsublig=3,police=\small\bfseries\ttfamily,%
couleurtexte=red,couleurcase=yellow!50,legendes=false]{\grilleSuMa}
```

Sortie L^AT_EX

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

Information(s)

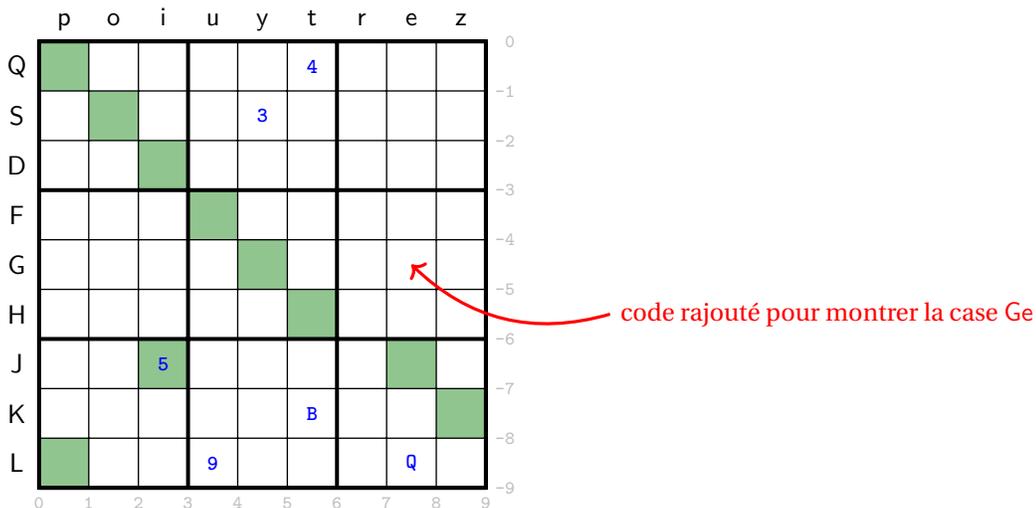
La grille, créée en TikZ, est portée par le rectangle de « coins » (0; 0) et (nbcot; -nblig), de sorte que les labels des cases sont situés au nœuds de coordonnées (x,5; -y,5).

Code L^AT_EX

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

\begin{EnvSudoMaths}[%
  Unite=0.66cm,Police=\footnotesize\bfseries\ttfamily,CouleurCase=ForestGreen!50,%
  ListeLegV=QSDFGHJKL,ListeLegH=poiuytrez]{\grilleSuMaB}
  \draw[red,very thick,<-] (7.5,-4.5) to[bend right] ++ (4,-1) node[right] {code rajouté...} ;
\end{EnvSudoMaths}
```

Sortie L^AT_EX



Onzième partie

Historique

- 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 17)
 - : Correction d'un bug avec Arbre=3x3 (page 65)
 - 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 28)
 - v 2.0.1: Chargement du package `piton` uniquement si compilation en Lua \TeX (page 28)
 - v 2.0.0: Refonte du code source avec modification des commandes
 - : Refonte de la documentation
-
- v 1.3.7: Commandes pour du code python via `piton`, en compilation Lua \TeX (page 28)
 - : Corrections et modifications mineures de la documentation
 - v 1.3.6: Présentation de l'algorithme d'Euclide pour le PGCD (page 75)
 - : Affichage d'un trinôme par coefficients, aléatoires ou non (page 83)
 - v 1.3.5: Correction d'un bug avec la loi géométrique (page 61)
 - v 1.3.4: Ajout de petits schémas, en `TikZ`, de lois normales et exponentielles (page 68)
 - : Calculs de probas avec les lois géométriques et hypergéométriques (page 61)
 - v 1.3.3: Ajout d'un environnement pour des arbres de probas classiques, en `TikZ` (page 65)
 - v 1.3.2: Correction d'un bug sur les conversions `bin-to-hex` avec `lualatex` (page 70)
 - v 1.3.1: Ajout d'une option pour ne pas afficher les bordures des corrections de pixelart (page 84)
 - v 1.3.0: Commande pour présenter une conversion depuis la base 10 (page 73)
 - v 1.2.9: Correction des commandes avec `simplekv`
 - v 1.2.8: Ajout d'une librairie `TikZ` oubliée, et remise en forme de la documentation
 - v 1.2.7: Ajout de commandes pour des calculs de probabilités (page 61)
 - v 1.2.6: Ajout d'un environnement pour des `SudoMaths` (page 89)
 - v 1.2.5: Ajout de commandes pour des boîtes à moustaches (page 58)
 - 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 25)
 - v 1.2.2: Commandes pour travailler sur des stats à 2 variables (page 50)
 - v 1.2.1: Amélioration de la gestion du `csv` pour `Pixelart`
 - v 1.2.0: Correction d'un *méchant* bug sur `Pixelart`
 - v 1.1.9: `Pixelart` en `TikZ` (page 84)
 - v 1.1.8: Style "Mainlevée" basique pour `TikZ`(page 82)
 - v 1.1.7: Conversions bin/hex/dec (basées sur `xintbinhex`) avec quelques détails (page 70)
 - v 1.1.6: Ajout d'une commande pour déterminer les paramètres d'une régression linéaire par moindres carrés (page 46)
 - v 1.1.5: Ajout de deux commandes pour, en `TikZ`, créer des petits schémas « de signe » (page 79)
 - v 1.1.4: Ajout d'une commande pour, en `TikZ`, créer facilement un cercle trigo avec *options* (page 43)
 - v 1.1.3: Ajout des commandes pour fractions, ensembles et récurrence (pages 77, 78 et 20)
 - 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 41)
 - v 1.0.9: Ajout d'une commande pour créer des pavés droits (avec nœuds) en `TikZ` (page 39)
 - v 1.0.8: Ajout d'une commande pour créer des cartouches de lien "comme capytale" (page 38)
 - 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 31)
 - v 1.0.4: Ajout des environnements pour Terminal (`win`, `osx`, `unix`) (page 36)
 - v 1.0.3: Ajout des environnements pour `PseudoCode` (page 34)
 - v 1.0.2: Ajout des environnements pour Python (`pythontex`) (page 30)
 - v 1.0.1: Modification mineure liée au chargement de `xcolor`
 - v 1.0 : Version initiale