tkz-grapheur [fr]

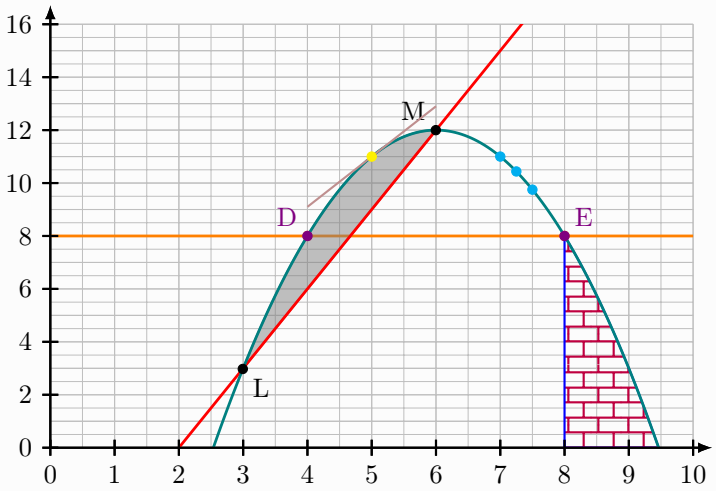
Un système de grapheur,

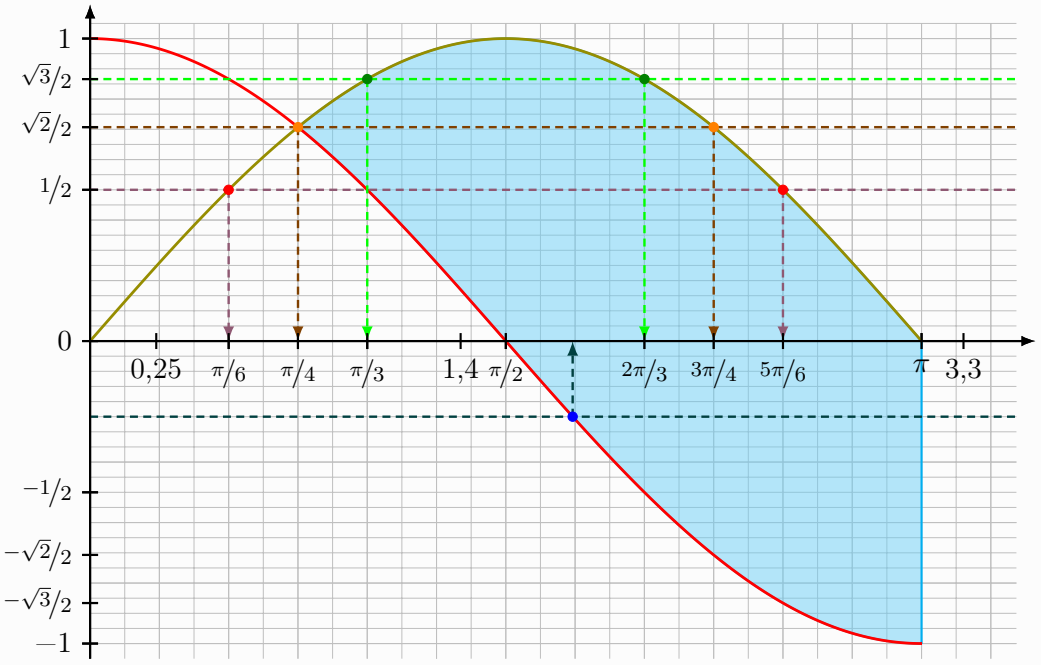
basé sur TikZ et xint.

Version 0.30f - 12 avril 2026

Cédric Pierquet
c pierquet - at - outlook . fr
https://github.com/cpierquet/latex-packages/tree/main/tkz-grapheur

---





À mon papa.

Table des matières

1 Introduction

1.1 Description et idées générales

Avec ce modeste package, loin des capacités offertes par exemple par les excellents packages `pgfplots`[1], `tkz-*`[2] (d'Alain Matthes) ou `tzplot`[3] (de In-Sung Cho), il est possible de travailler sur des graphiques de fonctions, en langage Ti*k*Z, de manière *intuitive* et *explicite*.

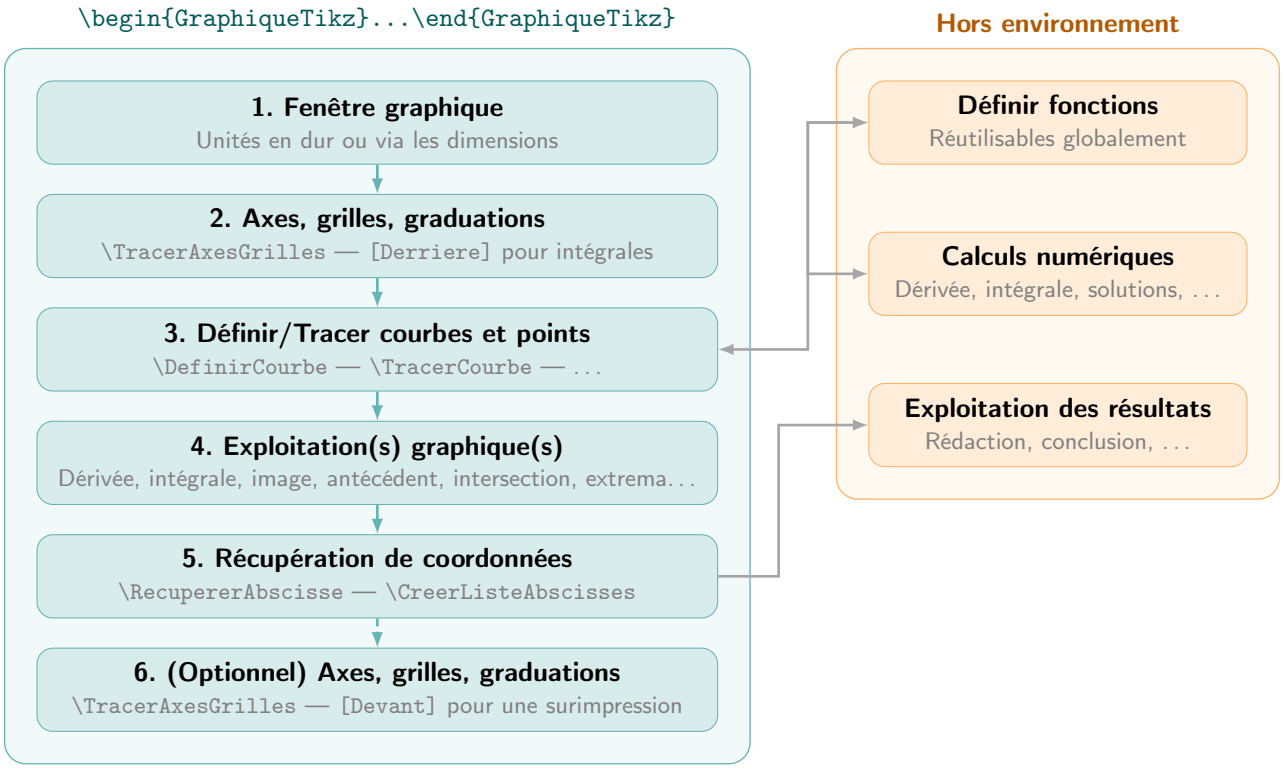
Concernant le fonctionnement global :

— des styles particuliers pour les objets utilisés ont été définis (modifiables localement) ;

— le nom des commandes est sous forme *opérationnelle*, de sorte que la construction des éléments graphiques a une forme quasi *algorithmique*.

1.2 Fonctionnement global

Pour schématiser le fonctionnement global du package :



1.3 Packages utilisés, et options du package

Le package utilise :

— `tikz`, avec les librairies `calc,intersections,patterns,patterns.meta,bbox` ;

— `simplekv`, `xintexpr`, `xstring`, `listofitems` ;

— `pgfplots`, avec la librairie `fillbetween` (désactivable via `[nonpgfplots]`) ;

— `xint-regression`[4] (pour les régressions, désactivable via `[nonxintreg]`).

Le package charge également `siunitx` avec les options classiques `[fr]`, mais il est possible de ne pas le charger en utilisant l'option `[nonsiunitx]`.

1. CTAN : https://ctan.org/pkg/pgfplots
2. par exemple tkz-base https://ctan.org/pkg/tkz-base et tkz-fct https://ctan.org/pkg/tkz-fct.
3. CTAN : https://ctan.org/pkg/tzplot.
4. CTAN : https://ctan.org/pkg/xint-regression.

1.4 Chargement du package

Le package charge également la librairie Ti*k*Z `babel`, mais il est possible de ne pas la charger en utilisant l'option `[nontikzbabel]`.

Les différentes options sont bien évidemment cumulables.

```
%chargement par défaut
\usepackage{tkz-grapheur}

%chargement sans sinuitx, à charger manuellement
\usepackage[nonsiunitx]{tkz-grapheur}

%chargement sans tikz.babel
\usepackage[nontikzbabel]{tkz-grapheur}

%chargement sans pgfplots + options compat
\usepackage[nonpgfplots]{tkz-grapheur}
\pgfplotsset{compat=...}
```

À noter également que certaines commandes peuvent utiliser des packages comme `nicefrac`, qui sera donc à charger le cas échéant.

Concernant la partie *calculs* et *tracés*, c'est le package `xint` qui s'en occupe.

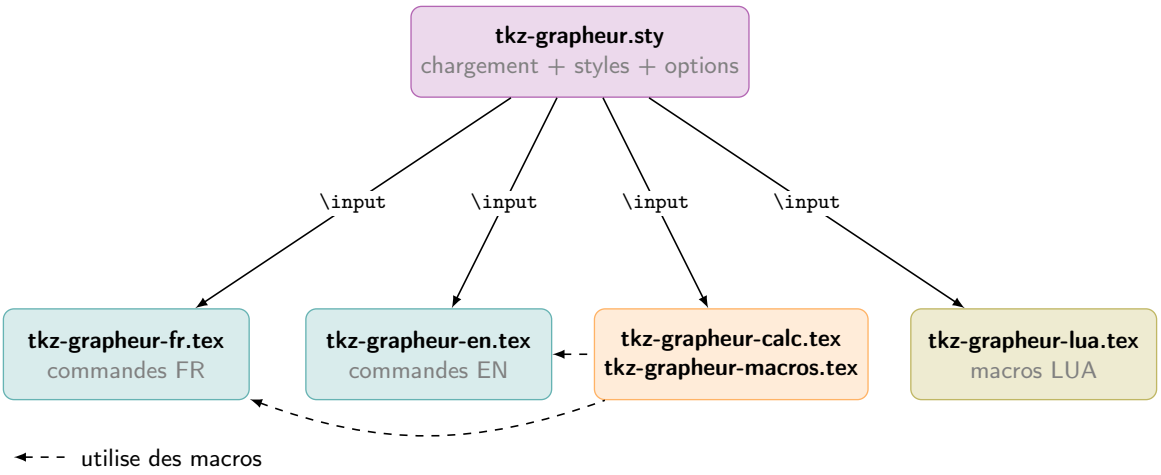
1.5 Avertissements

Il est possible, dû aux calculs (multiples) effectués en interne, que le temps de compilation soir un peu *allongé*.

La précision des résultats (de détermination) semble être aux environs de 10^{-4}, ce qui devrait normalement garantir des tracés et lectures *satisfaisantes*. Il est quand même conseillé d'être prudent quant aux résultats obtenus et ceux attendus.

1.6 Architecture du package

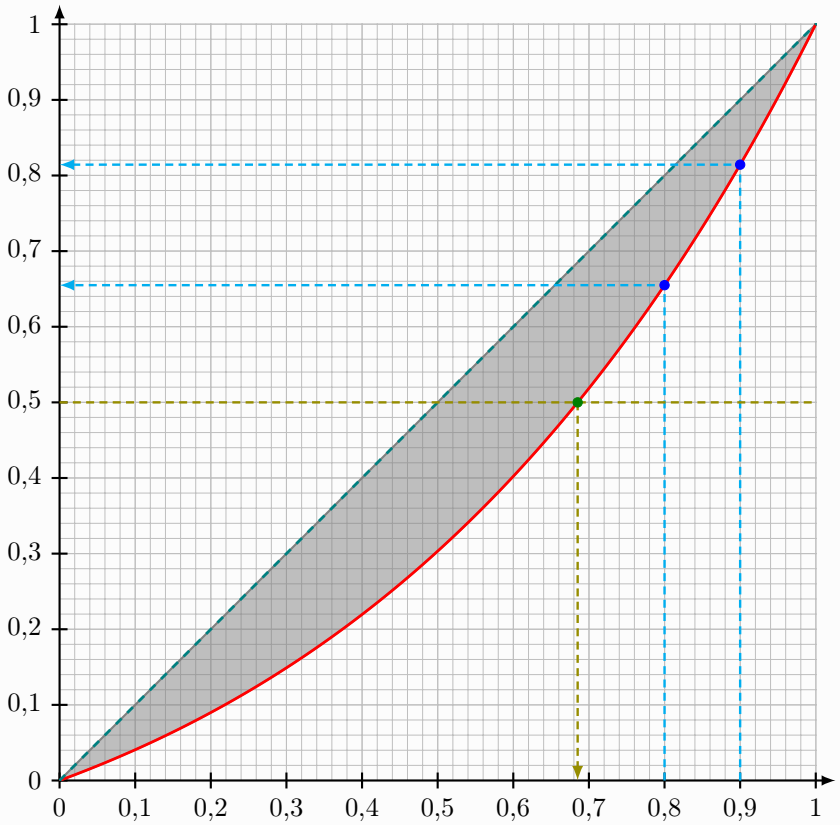
Afin de faciliter la maintenance et l'évolution du package, les commandes sont organisées en trois fichiers distincts, chargés automatiquement par `tkz-grapheur.sty` : les commandes graphiques en français et en anglais d'une part, les macros de calcul d'autre part.



1.7 Exemple introductif

On peut par exemple partir de l'exemple suivant, pour *illustrer* le cheminement des commandes de ce package. Les commandes et la syntaxe seront détaillées dans les sections suivantes !

```
\begin{GraphiqueTikz}%
  [x=10cm,y=10cm,Xmin=0,Xmax=1.001,Xgrille=0.1,Xgrilles=0.02,
  Ymin=0,Ymax=1.001,Ygrille=0.1,Ygrilles=0.02]
  \TracerAxesGrilles[Elargir=2.5mm,Police=\small]%
    {0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1}
    {0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1}
  \DefinirCourbe[Nom=cf,Debut=0,Fin=1]<f>{x*exp(x-1)}
  \DefinirCourbe[Nom=delta,Debut=0,Fin=1]<D>{x}
  \TracerIntegrale[Type=fct/fct]{f(x)}[D(x)]{0}{1}
  \TracerCourbe[Couleur=red]{f(x)}
  \TracerCourbe[Couleur=teal,StyleTrace=dashed]{D(x)}
  \PlacerImages[Couleurs=blue/cyan,Traits]{f}{0.8,0.9}
  \PlacerAntecedents[Couleurs=green!50!black/olive,Traits]{cf}{0.5}
\end{GraphiqueTikz}
```



2 Styles de base et création de l'environnement

2.1 Styles de base

Les styles utilisés pour les tracés sont donnés ci-dessous.

Dans une optique de *simplicité*, seule la couleur des éléments peut être paramétrée, mais si l'utilisateur le souhaite, il peut redéfinir les styles proposés.

```
%paramètres déclarés et stockés (utilisables dans l'environnement a posteriori)
\tikzset{
  Xmin/.store in=\pflxmin,Xmin/.default=-3,Xmin=-3,
  Xmax/.store in=\pflxmax,Xmax/.default=3,Xmax=3,
  Ymin/.store in=\pflymin,Ymin/.default=-3,Ymin=-3,
  Ymax/.store in=\pflymax,Ymax/.default=3,Ymax=3,
  Origx/.store in=\pflOx,Origx/.default=0,Origx=0,
  Origy/.store in=\pflOy,Origy/.default=0,Origy=0,
  Xgrille/.store in=\pflgrillex,Xgrille/.default=1,Xgrille=1,
  Xgrillei/.store in=\pflgrillexi,Xgrillei/.default=1,Xgrillei=1,
  Xgrilles/.store in=\pflgrillexs,Xgrilles/.default=0.5,Xgrilles=0.5,
  Ygrille/.store in=\pflgrilley,Ygrille/.default=1,Ygrille=1,
  Ygrillei/.store in=\pflgrilleyi,Ygrillei/.default=1,Ygrillei=1,
  Ygrilles/.store in=\pflgrilleys,Ygrilles/.default=0.5,Ygrilles=0.5
}
```

On retrouve donc :

— l'origine du repère (`Origx`/`Origy`) ;

— les valeurs extrêmes des axes (`Xmin`/`Xmax`/`Ymin`/`Ymax`) ;

— les paramètres des grilles principales et secondaires (`Xgrille`/`Xgrilles`/`Ygrille`/`Ygrilles`).

À noter que, depuis la version `0.2.8`, un troisième niveau de grille est accessible, via les valeurs `Xgrillei`/`Ygrillei`.

Concernant les styles des *objets*, ils sont donnés ci-dessous.

```
%styles grilles/axes
\tikzset{pflgrillep/.style={thin,lightgray}}
\tikzset{pflgrilles/.style={very thin,lightgray}}
\tikzset{pflaxes/.style={line width=0.8pt,->,>=latex}}
```

```
%style des points (courbe / nuage /labels / montecarlo)
\tikzset{pflpoint/.style={line width=0.95pt}}
\tikzset{pflpointc/.style={radius=1.75pt}}
\tikzset{pflpointnuage/.style={radius=1.75pt}}
\tikzset{pflpointmc/.style={radius=0.875pt}}
\tikzset{pflnoeud/.style={}} %pour les inner sep par exemple :-)
\tikzset{pflcourbediscont/.style={line width=1.1pt}}
```

```
%style des courbes
\tikzset{pflcourbe/.style={line width=1.05pt}}
```

```
%style des traits (normaux, antécédents, images)
\tikzset{pfltrait/.style={line width=0.8pt}}
\tikzset{pfltraitantec/.style={line width=0.95pt,densely dashed}}
\tikzset{pfltraitimg/.style={line width=0.95pt,densely dashed,->,>=latex}}

%style des flèches
\tikzset{pflflecheg/.style={<-,>=latex}}
\tikzset{pflfleched/.style={->,>=latex}}
\tikzset{pflflechegd/.style={<->,>=latex}}
```

```
%style des constructions ECC (courbe / paramètres)
\tikzset{pfltraitsparamecc/.style={line width=0.9pt,densely dashed}}
\tikzset{pflcourbeecc/.style={line width=1.05pt}}
```

```
%style des constructions récurrence
\tikzset{pfltraitrec/.style={line width=0.8pt}}
\tikzset{pfltraitrecpointill/.style={pfltraitrec,densely dashed}}
```

L'idée est donc de pouvoir redéfinir globalement ou localement les styles, et éventuellement de rajouter des éléments, en utilisant `monstyle/.append style={...}`.

2.2 Création de l'environnement

2.2.1 Valeurs manuelles

L'environnement proposé est basé sur Ti*k*Z, de sorte que toute commande *classique* liée à Ti*k*Z peut être utilisée en marge des commandes du package !

```
\begin{GraphiqueTikz}[options tikz]<clés>
  %code(s)
\end{GraphiqueTikz}
```

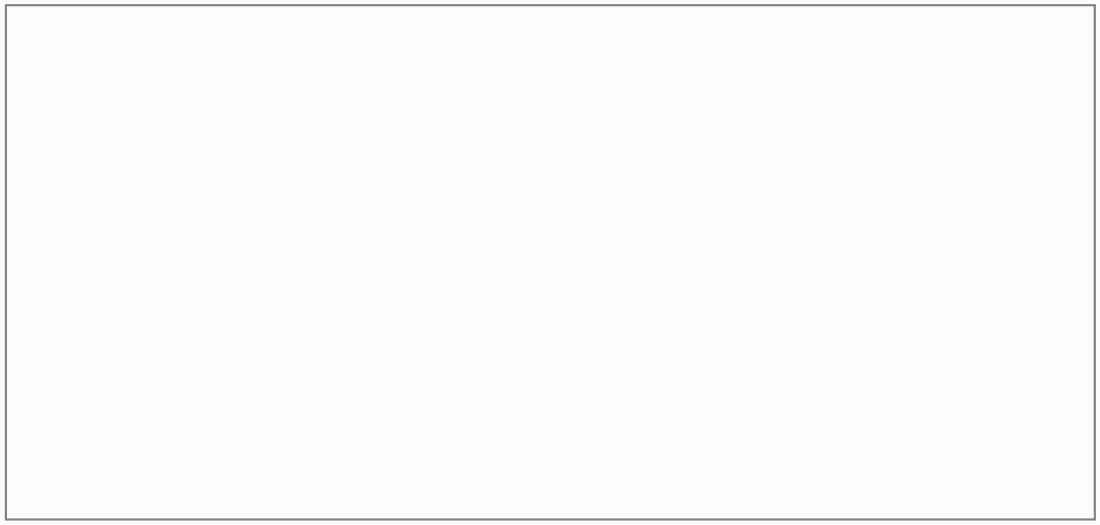
Les `[options tikz]` sont les options *classiques* qui peuvent être passées à un environnement Ti*k*Z, ainsi que les clés des axes/grilles/fenêtre présentées précédemment.

Les `<clés>` spécifiques (et optionnelles) sont :

- `TailleGrad` : taille des graduations des axes (`3pt` pour 3pt *dessus* et 3pt *dessous*) ;
- `Theme` : thème *couleurs* de la grille ;
- `NomFigure` : préfixe pour les nœuds de la fenêtre ;
- `AffCadre` : booléen (`false` par défaut) pour afficher un cadre qui délimite la fenêtre graphique (hors graduations éventuelles).

```
\begin{GraphiqueTikz}
  [x=0.075cm,y=0.03cm,Xmin=0,Xmax=160,Xgrille=20,Xgrilles=10,
  Origy=250,Ymin=250,Ymax=400,Ygrille=25,Ygrilles=5]
  <AffCadre>
\end{GraphiqueTikz}
```

```
\begin{GraphiqueTikz}%
  [x=0.9cm,y=0.425cm,Xmin=4,Xmax=20,Origx=4,
  Ymin=40,Ymax=56,Ygrille=2,Ygrilles=1,Origy=40]
  <AffCadre>
\end{GraphiqueTikz}
```



Ce sera bien évidemment plus parlant avec les éléments graphiques rajoutés !

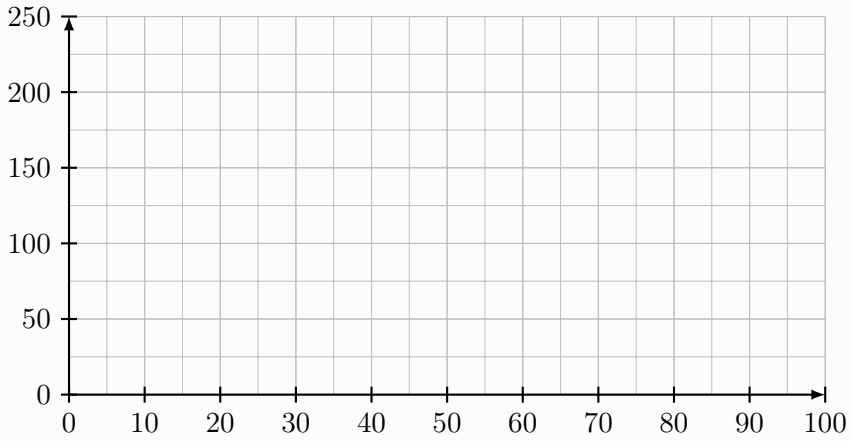
2.2.2 Avec choix des dimensions

Il est également possible (c'est en test) de spécifier les dimensions du graphique, en laissant le code déterminer les bonnes unités.

Les paramètres des axes et de la grille peuvent (et doivent) dans ce cas être spécifiés dans l'argument entre `<...>`.

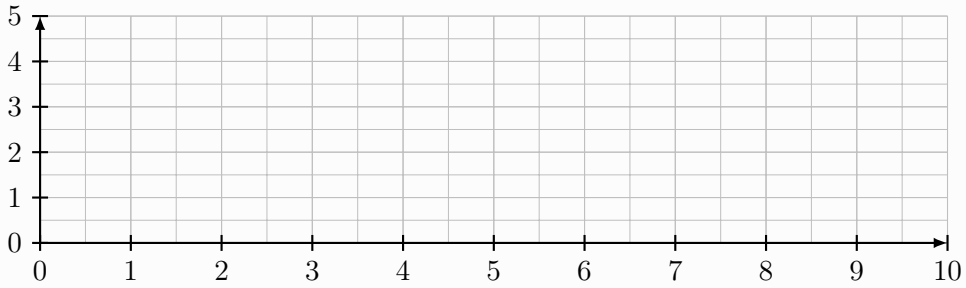
Dans ce cas, les clés et arguments diffèrent légèrement, notamment via les clés `<Taille=larg/haut>` ou `<Largeur=...,XYratio=...>` ou `<Hauteur=...,XYratio=...>`.

```
%dimensions fixées (hors graduations)
\begin{GraphiqueTikz}%
    [Xgrille=10,Xgrilles=5,Ygrille=50,Ygrilles=25]
    %Xgrille(s) et/ou Ygrille(s) dans ce cas
    <Xmin=0,Xmax=100,Ymin=0,Ymax=250,Taille={10cm/5cm}>
  \TracerAxesGrilles[]{auto}{auto}
\end{GraphiqueTikz}
```



Il est également possible de spécifier une `<Largeur>` et /ou une `<Hauteur>` et/ou un `<XYratio>` :

```
%largeur + ratio 0.5
\begin{GraphiqueTikz}%
    <Xmin=0,Xmax=10,Ymin=0,Ymax=5,Largeur=12cm,XYratio=0.5>
  \TracerAxesGrilles{auto}{auto}
\end{GraphiqueTikz}
```



2.3 Grilles et axes

2.3.1 Fonctionnement global

La première commande *utile* va permettre de créer les grilles, les axes et les graduations.

```
%dans l'environnement GraphiqueTikz
\TracerAxesGrilles[clés]{gradX}{gradY}
```

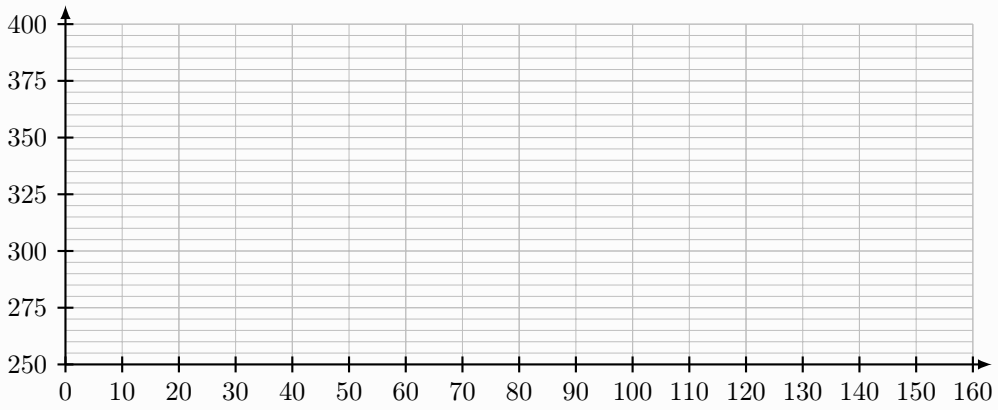
Les `[clés]`, optionnelles, disponibles sont :

- `Grille` : booléen (`true` par défaut) pour afficher les grilles (pour une grille unique, il suffit de mettre les paramètres identiques pour `Xgrille`/`Xgrilles` ou `Ygrille`/`Ygrilles`) ;
- `Elargir` : rajout à la fin des axes (`0` par défaut) ;
- `Grads` : booléen (`true` par défaut) pour les graduations ;
- `Police` : police globale des graduations `vide` par défaut ;
- `Format` : formatage particulier (voir en dessous) des valeurs des axes.

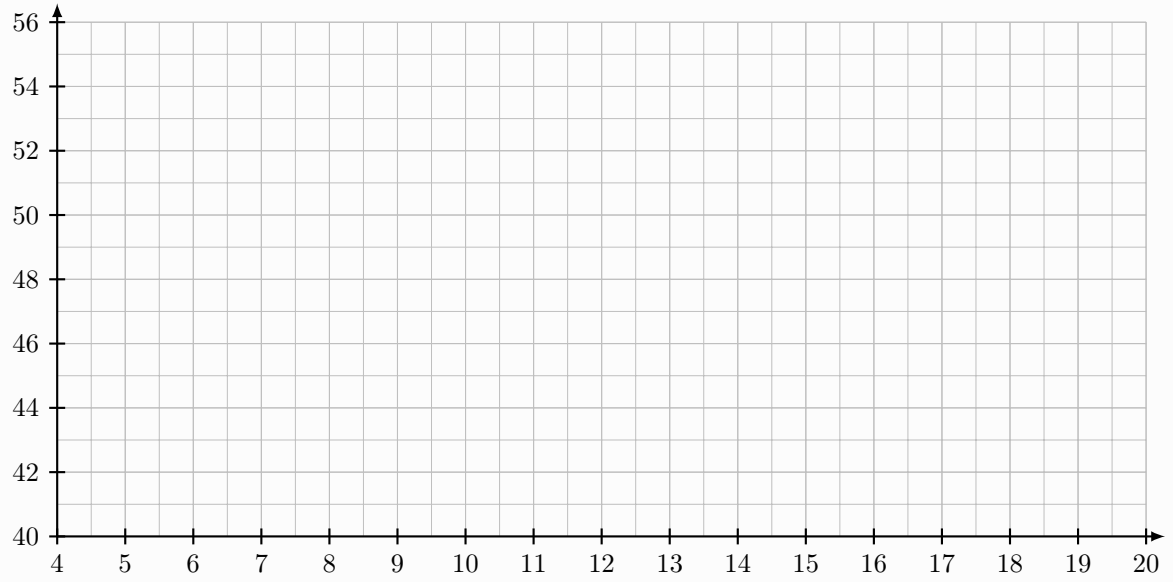
Concernant la clé `Format`, elle permet de spécifier un paramétrage spécifique pour les valeurs des axes. Elle peut être donnée sous la forme `fmt` pour un formatage combiné, ou sous la forme `fmtX/fmtY` pour différencier le formatage.
Les options possible sont :

- `num` : formater avec siunitx ;
- `annee` : formater en année ;
- `frac` : formater en fraction frac ;
- `dfrac` : formater en fraction dfrac ;
- `nfrac` : formater en fraction nicefrac ; (à charger !)
- `trig` : formater en trigo avec frac ;
- `dtrig` : formater en trigo avec dfrac ;
- `ntrig` : formater en trigo avec nfrac ;
- `sqrt` : formater en racine avec frac ;
- `dsqrt` : formater en racine avec dfrac ;
- `nsqrt` : formater en racine avec nicefrac.

```
\begin{GraphiqueTikz}
  [x=0.075cm,y=0.03cm,Xmin=0,Xmax=160,Xgrille=20,Xgrilles=10,
  Origy=250,Ymin=250,Ymax=400,Ygrille=25,Ygrilles=5]
  \TracerAxesGrilles[Elargir=2.5mm,Police=\small]{0,10,...,160}{250,275,...,400}
\end{GraphiqueTikz}
```

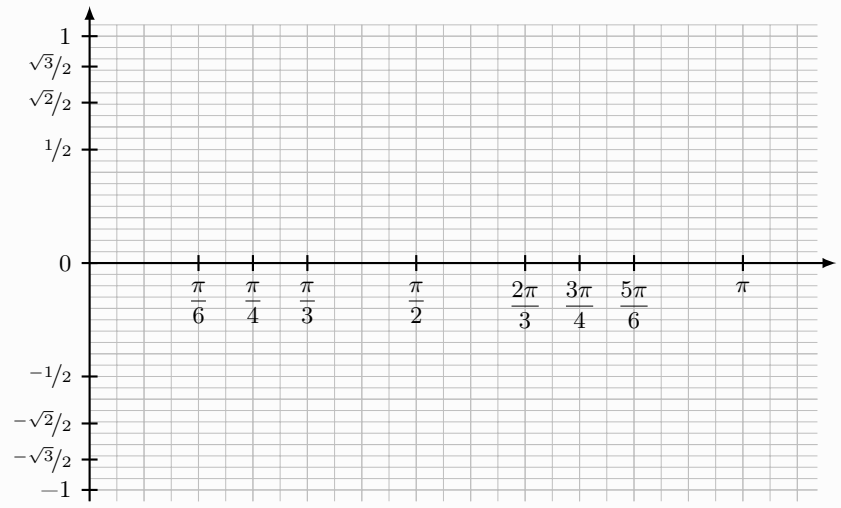


```
\begin{GraphiqueTikz}%
  [x=0.9cm,y=0.425cm,Xmin=4,Xmax=20,Origx=4,
  Ymin=40,Ymax=56,Ygrille=2,Ygrilles=1,Origy=40]
  \TracerAxesGrilles[Elargir=2.5mm,Police=\small]{4,5,...,20}{40,42,...,56}
\end{GraphiqueTikz}
```



À noter qu'il existe les clés booléennes `[Derriere]` (sans les graduations) et `[Devant]` (sans la grille) pour afficher les axes en mode *sous/sur*-impression dans le cas d'intégrales par exemple.

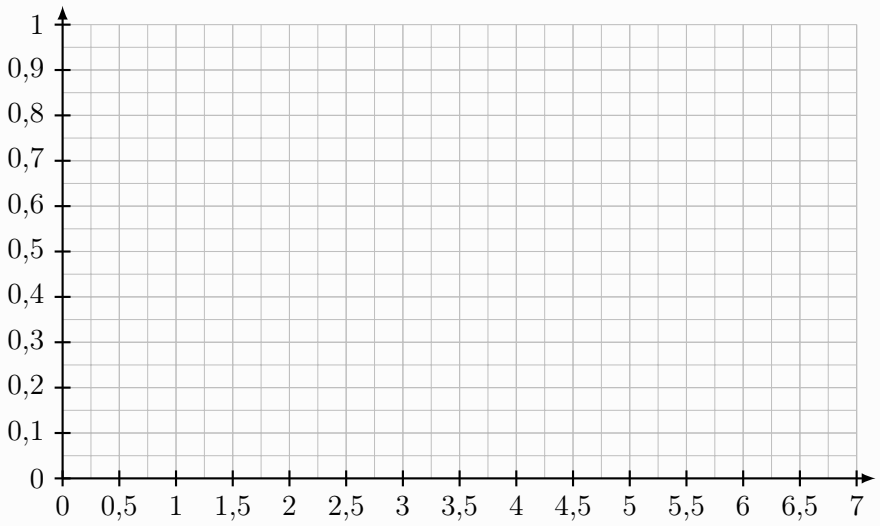
```
\begin{GraphiqueTikz}%
  [x=2.75cm,y=3cm,
  Xmin=0,Xmax=3.5,Xgrille=pi/12,Xgrilles=pi/24,
  Ymin=-1.05,Ymax=1.05,Ygrille=0.2,Ygrilles=0.05]
  \TracerAxesGrilles[Elargir=2.5mm,Format=dtrig/nsqrt,Police=\footnotesize]%
    {pi/6,pi/4,pi/3,pi/2,2*pi/3,3*pi/4,5*pi/6,pi}
    {0,sqrt(2)/2,1/2,sqrt(3)/2,1,-1,-sqrt(3)/2,-1/2,-sqrt(2)/2}
\end{GraphiqueTikz}
```



Dans le cas où le formatage ne donne pas de résultat(s) satisfaisant(s), il est possible d'utiliser une commande générique de placement des graduations.

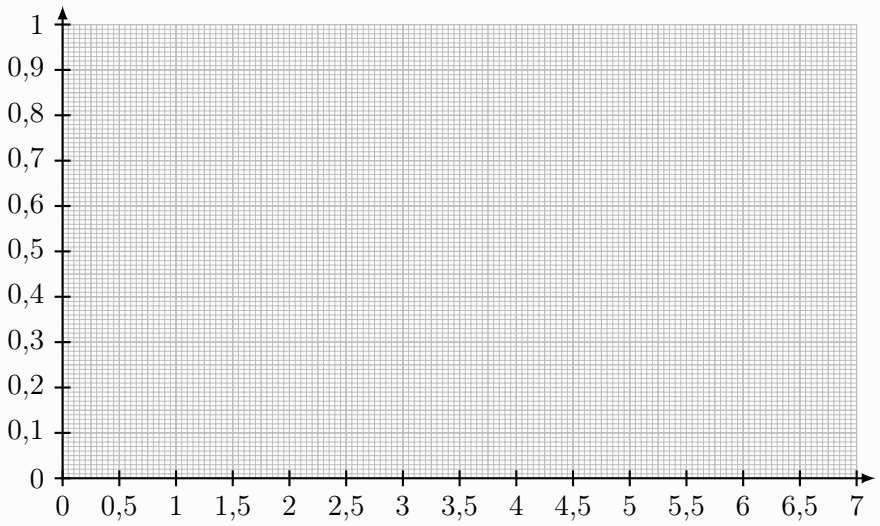
Dans le cas où les graduations sont *naturellement* définies par les données de la fenêtre et de la grille (principale), il est possible de préciser `auto` dans les arguments obligatoires (dans ce cas le formatage ne sera pas possible, et `Format=num` sera obligatoirement utilisé).

```
\begin{GraphiqueTikz}%
  [x=1.5cm,y=6cm,Xmin=0,Xmax=7,Xgrille=0.5,Xgrilles=0.25,
  Ymin=0,Ymax=1,Ygrille=0.1,Ygrilles=0.05]
  \TracerAxesGrilles[Elargir=2.5mm,Dernier]{auto}{auto}
\end{GraphiqueTikz}
```



À noter qu'une clé (expérimentale), [Grille Intermediaire], permet d'afficher un troisième niveau de grille, définie par `[Xgrillei=...,Ygrillei=...]`.

```
\begin{GraphiqueTikz}%
  [x=1.5cm,y=6cm,Xmin=0,Xmax=7,Xgrille=0.5,Xgrillei=0.25,Xgrilles=0.05,
  Ymin=0,Ymax=1,Ygrille=0.1,Ygrillei=0.05,Ygrilles=0.01]
  \TracerAxesGrilles[Elargir=2.5mm,Dernier,Grille Intermediaire]{auto}{auto}
\end{GraphiqueTikz}
```

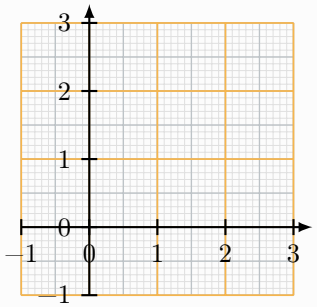


2.3.2 Thème de couleurs de la grille

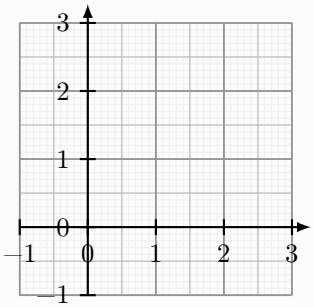
Il existe de plus 6 thèmes prédéfinis de couleurs pour les grilles, parmi : `<Theme=standard/gris/bleu/vert/chaud/contraste>`.

```
\begin{GraphiqueTikz}[options]<Theme=...>
  ...
\end{GraphiqueTikz}
```

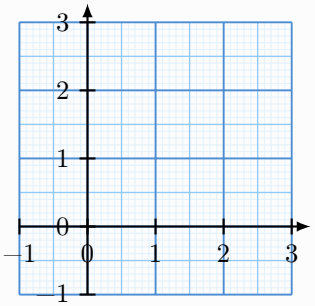
Theme=standard



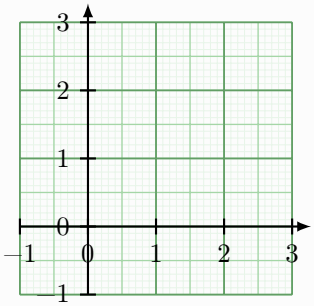
Theme=gris



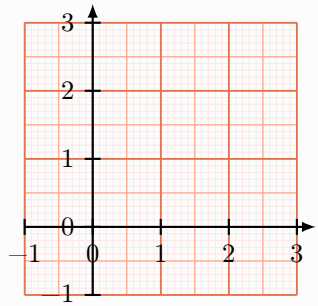
Theme=bleu



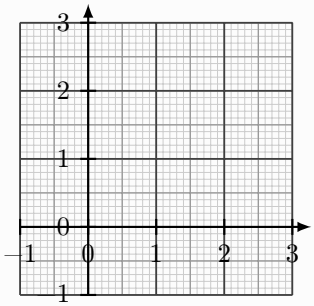
Theme=vert



Theme=chaud



Theme=contraste



Les épaisseurs des traits pour ce type de grille sont données (et modifiables) via :

```
\setlength\pflthickgridp{0.65pt}    %épaisseur grille principale
\setlength\pflthickgridi{0.5pt}     %épaisseur grille intermédiaire
\setlength\pflthickgrids{0.35pt}    %épaisseur grille secondaire
```

2.4 Ajout de valeurs manuellement

Il est également possible d'utiliser une commande spécifique pour placer des valeurs sur les axes, indépendamment d'un système *automatisé* de formatage.

```
%dans l'environnement GraphiqueTikz
\RajouterValeursAxeX[clés]{positions}{valeurs formatées}
\RajouterValeursAxeY[clés]{positions}{valeurs formatées}
```

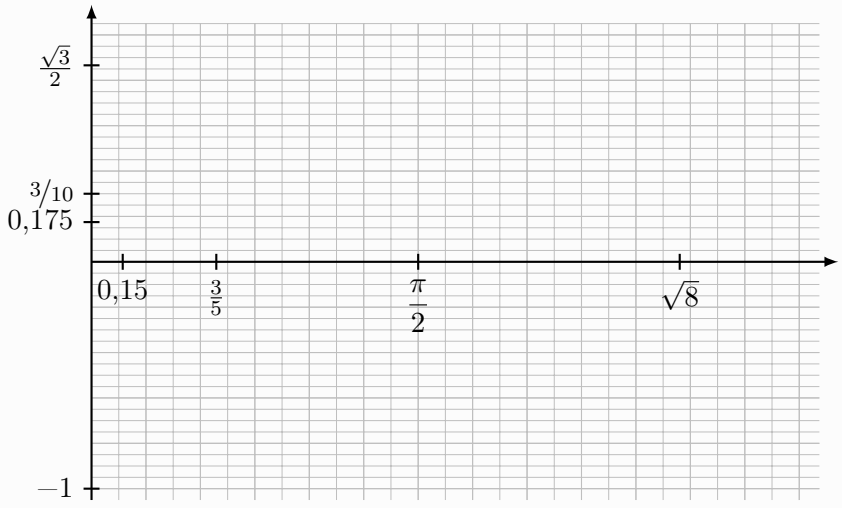
Les `[clés]`, optionnelles, disponibles sont :

— `Police` : police globale des graduations `vide` par défaut ;

— `Traits` : booléen pour ajouter les traits des graduations `true` par défaut.

Les arguments obligatoires correspondent aux abscisses (en langageTi*k*Z) et aux labels (en langage LaTeX) des graduations.

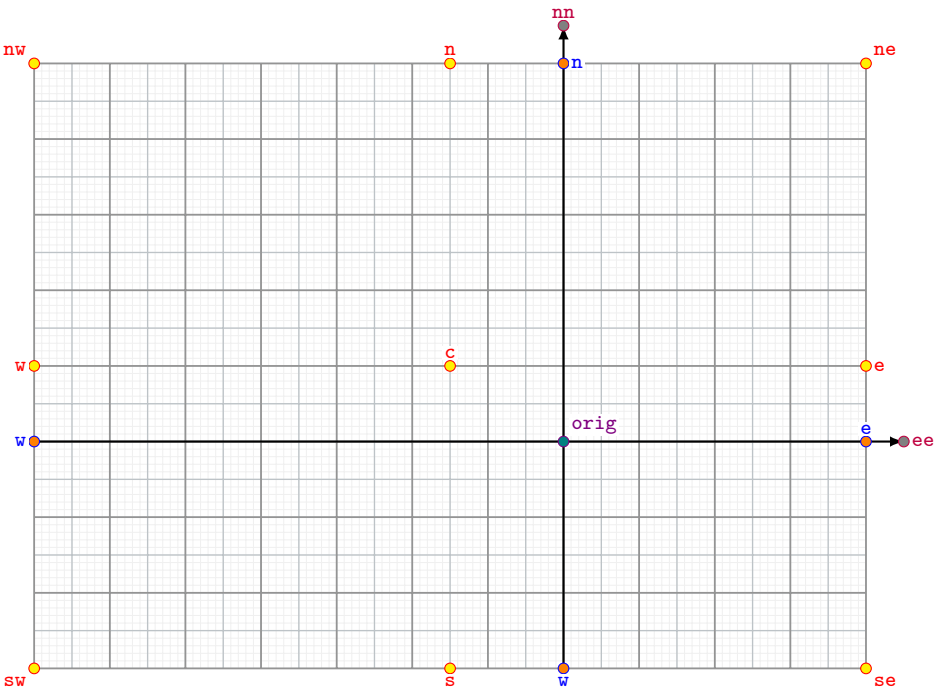
```
\begin{GraphiqueTikz}%
  [x=2.75cm,y=3cm,
  Xmin=0,Xmax=3.5,Xgrille=pi/12,Xgrilles=pi/24,
  Ymin=-1.05,Ymax=1.05,Ygrille=0.2,Ygrilles=0.05]
  \TracerAxesGrilles[Grad=false,Elargir=2.5mm,]{}{}
  \RajouterValeursAxeX
    {0.15,0.6,pi/2,2.8284}
    {\num{0.15},$\frac35$,$\displaystyle\frac{\pi}{2}$,$\sqrt{8}$}
  \RajouterValeursAxeY
    {-1,0.175,0.3,sqrt(3)/2}
    {\num{-1},\num{0.175},$\nicefrac{3}{10}$,$\frac{\sqrt{3}}{2}$}
\end{GraphiqueTikz}
```



2.5 Nœuds liés à la fenêtre et aux axes

En marge de la création de la fenêtre (et éventuellement des axes), des nœuds sont créés pour réutilisation ultérieure (placement de texte par exemple).

- 9 nœuds liés à la fenêtre :
  - `(graphe-nw)`, `(graphe-n)`, `(graphe-ne)` ;
  - `(graphe-w)`, `(graphe-c)`, `(graphe-e)` ;
  - `(graphe-sw)`, `(graphe-s)`, `(graphe-se)` ;
- 4 nœuds liés aux axes :
  - `(axeox-w)`, `(axeox-e)` ;
  - `(axeoy-s)`, `(axeoy-n)` ;
- 1 nœud liés à l'origine :
  - `(axes-orig)` ;
- 2 nœuds liés aux axes *élargis* (si nécessaire) :
  - `(axeox-ee)` ;
  - `(axeoy-nn)`.



À noter qu'avec la clé `<NomFigure=...>`, il est possible d'ajouter un préfixe pour les nœuds, pour des rajouts ultérieurs via `overlay` par exemple.
Les nœuds créés seront de ce fait accessibles via `(<nomfigure>-graphe-nw)`, etc.

3 Commandes spécifiques de définitions

3.1 Création de fonctions hors environnement

Ces commandes permettent de définir des fonctions xint accessibles en dehors d'un environnement `GraphiqueTikz`, notamment pour les calculs numériques ou les exercices sans graphique.
La version étoilée force une définition **globale**.

```
%fonction simple
\GenererFonction<nom>{expr}

%fonction globale (version étoilée) si besoin
\GenererFonction*<nom>{expr}
```

```
%définition hors environnement
\GenererFonction<f>{x^2+2*x-3}

%calculs sans graphique
$f(2) = \xintfloateval{f(2)}$

\tkzgCalcIntegrale*{f(x)}{0}{3}[\monres]
$\displaystyle\int_0^3 f(x)\,dx \approx \monres$

\tkzgResolApproch*{f(x)=0}{-5:0}[\myzero]
La racine négative est $x \approx \myzero$.
```

$f(2) = 5$
$\displaystyle\int_0^3 f(x)\,dx \approx 9$
La racine négative est $x \approx -2.999992370605469$.

```
%famille de fonctions
\GenererFamilleFonctions<nom famille>{expr, avec paramètre n}{début n}{fin n}

%famille globale (version étoilée)
\GenererFamilleFonctions*<nom famille>{expr, avec paramètre n}{début n}{fin n}
```

```
%définition hors environnement
\GenererFamilleFonctions<parab>{n*x^2}{1}{4}

%calculs sur un membre de la famille
$f_2(3) = \xintfloateval{parab_2(3)}$

\tkzgCalcIntegrale*{parab_3(x)}{0}{2}[\monres]
$\displaystyle\int_0^2 f_3(x)\,dx \approx \monres$
```

$f_2(3) = 18$
$\displaystyle\int_0^2 f_3(x)\,dx \approx 8$

3.2 Tracer une droite

L'idée est de proposer une commande pour tracer une droite, à partir :

— de deux points (ou nœuds) ;

— d'un point (ou nœud) et de la pente.

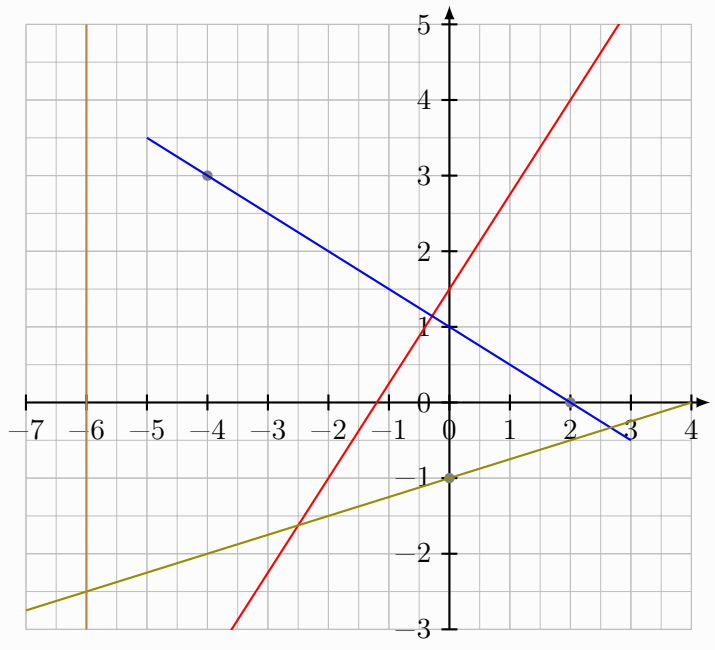
Il existe également une commande pour une asymptote verticale.

```
%dans l'environnement GraphiqueTikz
\TracerDroite[clés]{point ou nœud}{point ou noeud ou pente}
\TracerAsymptote[clés]{abscisse}
```

Les `[clés]`, optionnelles, disponibles sont :

— `Nom` : nom éventuel du tracé (pour réutilisation) ;
— `Pente` : booléen pour préciser que la pente est utilisée (`false` par défaut) ;
— `Debut` : début du tracé (`\pflxmin` par défaut) ;
— `Fin` : fin du tracé (`\pflxmax` par défaut) ;
— `StyleTrace` : style du tracé (`vide` par défaut) ;
— `Couleur` : couleur du tracé (`black` par défaut).

```
\begin{GraphiqueTikz}%
  [x=0.8cm,y=1cm,Xmin=-7,Xmax=4,Ymin=-3,Ymax=5]
  \TracerAxesGrilles[Elargir=2.5mm]{auto}{auto}
  \DefinirPts[Aff,Couleur=gray]{A/-4/3,B/2/0,C/0/-1}
  \TracerDroite[Couleur=red]{(-2,-1)}{(2,4)}
  \TracerDroite[Couleur=blue,Debut=-5,Fin=3]{(A)}{(B)}
  \TracerDroite[Couleur=olive,Pente]{(C)}{0.25}
  \TracerAsymptote[Couleur=brown]{-6}
\end{GraphiqueTikz}
```



3.3 Gestion de fonctions, de courbes

3.3.1 Définir une fonction, tracer la courbe d'une fonction

L'idée est de définir une fonction, pour réutilisation ultérieure. Cette commande *crée* la fonction, sans la tracer, car dans certains cas des éléments devront être tracés au préalable.

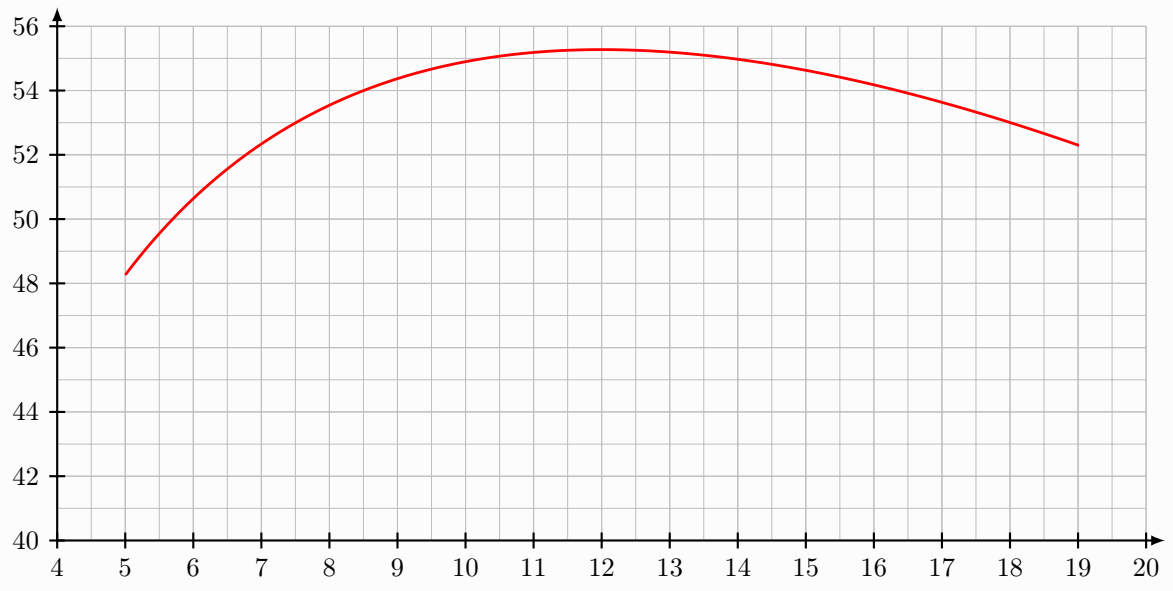
Il existe également une commande pour tracer la courbe d'une fonction précédemment définie.

```
%dans l'environnement GraphiqueTikz
\DefinirCourbe[clés]<nom fct>{formule xint}
\TracerCourbe[clés]{formule xint}
```

Les `[clés]` pour la définition ou le tracé, optionnelles, disponibles sont :

- `Debut` : borne inférieure de l'ensemble de définition (`\pflxmin` par défaut) ;
- `Fin` : borne inférieure de l'ensemble de définition (`\pflxmax` par défaut) ;
- `Nom` : nom de la courbe (important pour la suite !) ;
- `Couleur` : couleur du tracé (`black` par défaut) ;
- `Pas` : pas du tracé (il est déterminé *automatiquement* au départ mais peut être modifié) ;
- `StyleTrace` : style du tracé (`vide` par défaut) ;
- `Trace` : booléen pour tracer également la courbe (`false` par défaut) ;
- `DefGlobale` : booléen (`false` par défaut) pour rendre la définition de la fonction xint globale, accessible après l'environnement `GraphiqueTikz`.

```
\begin{GraphiqueTikz}%
  [x=0.9cm,y=0.425cm,Xmin=4,Xmax=20,Origx=4,
  Ymin=40,Ymax=56,Ygrille=2,Ygrilles=1,Origy=40]
  \TracerAxesGrilles[Elargir=2.5mm,Police=\small]{4,5,...,20}{40,42,...,56}
  %définition de la fonction + tracé de la courbe
  %la fonction ln a été créée pour xint !
  \DefinirCourbe[Nom=cf,Debut=5,Fin=19]<f>{-2*x+3+24*ln(2*x)}
  \TracerCourbe[Couleur=red,Debut=5,Fin=19]{f(x)}
  %ou en une seule commande si "suffisant"
  %\DefinirCourbe[Nom=cf,Debut=5,Fin=19,Trace]<f>{-2*x+3+24*ln(2*x)}
\end{GraphiqueTikz}
```



3.3.2 Définir/tracer une courbe d'interpolation (simple)

Il est également possible de définir une courbe via des points supports, donc une courbe d'interpolation simple.

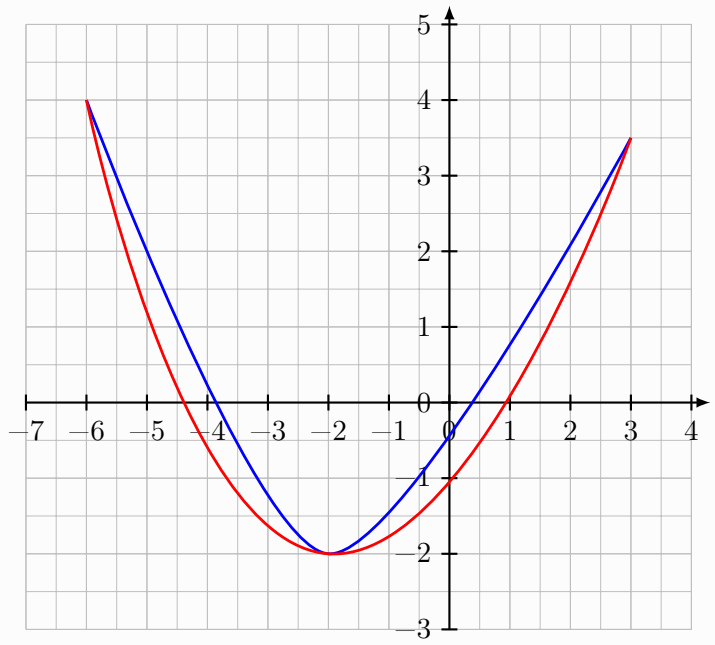
```
%dans l'environnement GraphiqueTikz
\DefinirCourbeInterpo[clés]{liste des points support}
\TracerCourbeInterpo[clés]{liste des points support}
```

Les `[clés]` pour la définition ou le tracé, optionnelles, disponibles sont :

— `Nom` : nom de la courbe d'interpolation (important pour la suite !) ;
— `Couleur` : couleur du tracé (`black` par défaut) ;
— `Tension` : paramétrage de la *tension* du tracé d'interpolation (`0.5` par défaut) ;
— `StyleTrace` : style du tracé (`vide` par défaut) ;
— `Trace` : booléen pour tracer également la courbe (`false` par défaut).

L'argument obligatoire permet quant à lui de spécifier la liste des points supports sous la forme `(x1,y1)(x2,y2)...`.

```
\begin{GraphiqueTikz}%
  [x=0.8cm,y=1cm,Xmin=-7,Xmax=4,Ymin=-3,Ymax=5]
  \TracerAxesGrilles[Elargir=2.5mm]{-7,-6,...,4}{-3,-2,...,5}
  %courbes d'interpolation simples (avec tension diff)
  \DefinirCourbeInterpo[Nom=interpotest,Couleur=blue,Trace]%
    {(-6,4)(-2,-2)(3,3.5)}
  \DefinirCourbeInterpo[Nom=interpotest,Couleur=red,Trace,Tension=1]%
    {(-6,4)(-2,-2)(3,3.5)}
\end{GraphiqueTikz}
```



3.3.3 Définir/tracer une courbe d'interpolation (Hermite)

Il est également possible de définir une courbe via des points supports, donc une courbe d'interpolation avec contrôle de la dérivée.

Certaines exploitations demandant des techniques différentes suivant le type de fonction utilisée, une clé booléenne `Spline` permettra au code d'adapter ses calculs suivant l'objet utilisé.

```
%dans l'environnement GraphiqueTikz
\DefinirCourbeSpline[clés]{liste des points support}[\macronomspline]
\TracerCourbeSpline[clés]{liste des points support}[\macronomspline]
```

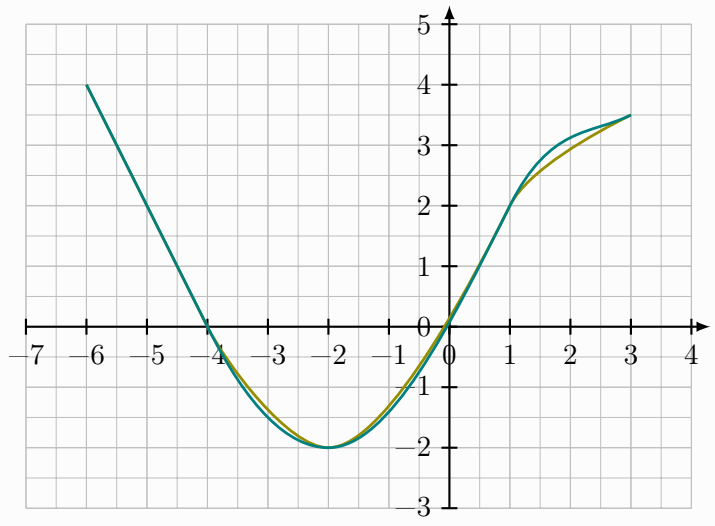
Les `[clés]` pour la définition ou le tracé, optionnelles, disponibles sont :

— `Nom` : nom de la courbe d'interpolation (important pour la suite !) ;
— `Coeffs` : modifier (voir la documentation de ProfLycee [5] pour les *coefficients* du spline) ;
— `Couleur` : couleur du tracé (`black` par défaut) ;
— `Trace` : booléen pour tracer également la courbe (`false` par défaut) ;
— `StyleTrace` : style du tracé (`vide` par défaut) ;
— `Alt` : booléen pour activer une autre *méthode de calcul* (`false` par défaut).

L'argument obligatoire permet quant à lui de spécifier la liste des points supports sous la forme `x1/y1/f'1§x2/y2/f'2§...` avec :

— `xi/yi` les coordonnées du point ;
— `f'i` la dérivé au point support.

```
\begin{GraphiqueTikz}%
  [x=0.8cm,y=0.8cm,Xmin=-7,Xmax=4,Ymin=-3,Ymax=5]
  \TracerAxesGrilles[Elargir=2.5mm]{-7,-6,...,4}{-3,-2,...,5}
  %définition de la liste des points support du spline
  \def\LISTETEST{-6/4/-2§-5/2/-2§-4/0/-2§-2/-2/0§1/2/2§3/3.5/0.5}
  %définition et tracé du spline cubique (x2)
  \DefinirCourbeSpline[Nom=splinetest,Trace,Couleur=olive]{\LISTETEST}
  \DefinirCourbeSpline[Alt,Nom=splinetest,Trace,Couleur=teal]{\LISTETEST}
\end{GraphiqueTikz}
```



5. CTAN : https://ctan.org/pkg/proflycee

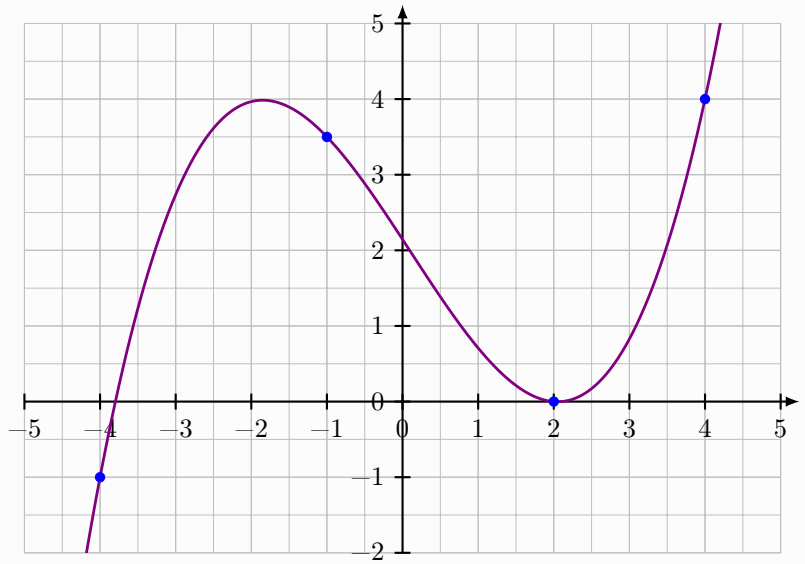
3.3.4 Définir/tracer une courbe d'interpolation (Lagrange)

Il est également possible de définir une courbe d'interpolation de Lagrange (merci à *JF Burnol* pour son aide !).
L'idée est d'utiliser une commande permettant de générer le polynôme de Lagrange, utilisable comme une fonction `xint` à l'aide des commandes classiques

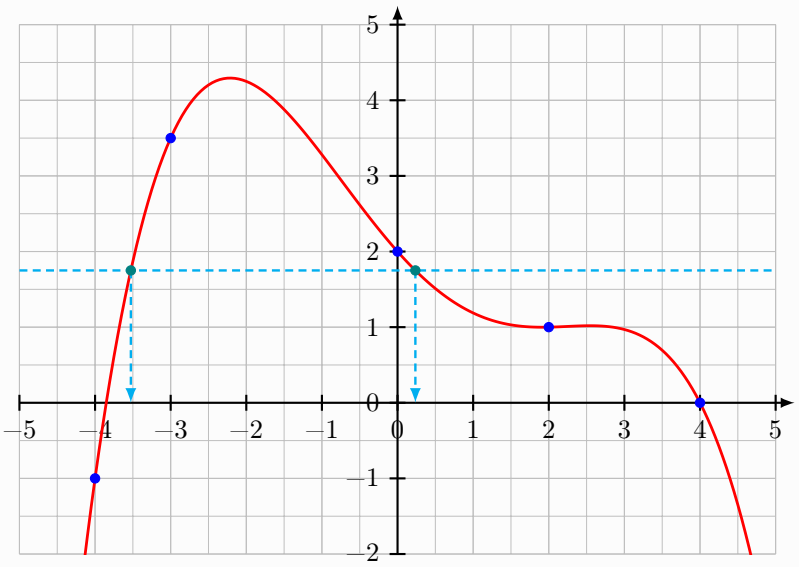
```
%dans l'environnement GraphiqueTikz
\GenererPolynomeLagrange[nom fonction]{liste X}{liste Y}
\TracerCourbe[clés]{f(x)}
```

par défaut, le nom de la fonction définie est `polylagrange`, mais il peut être modifié.
Une clé (booléenne) spécifique lors du tracé, `RestreindreY`, permet de limiter les valeurs verticales liées au phénomène de Runge.
Une commande spécifique de placements des points est également disponible, afin de conserver la syntaxe de la commande de génération.

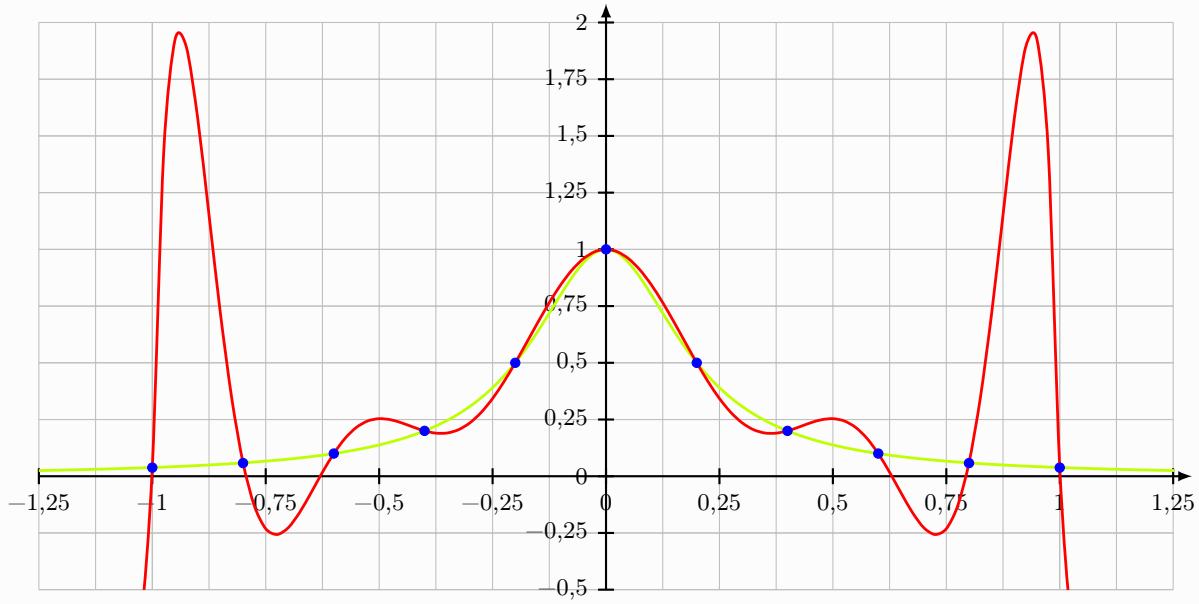
```
\begin{GraphiqueTikz}[Xmin=-5,Xmax=5,Ymin=-2,Ymax=5]
  \TracerAxesGrilles[Elargir=2.5mm,Police=\small]{auto}{auto}
  \GenererPolynomeLagrange{-4,-1,2,4}{-1,3.5,0,4}
  \TracerCourbe[Couleur=violet]{polylagrange(x)}
  \MarquerPtsLagrange*[Couleur=blue]{-4,-1,2,4}{-1,3.5,0,4}
\end{GraphiqueTikz}
```



```
\begin{GraphiqueTikz}[Xmin=-5,Xmax=5,Ymin=-2,Ymax=5]
  \TracerAxesGrilles[Elargir=2.5mm,Police=\small]{auto}{auto}
  \GenererPolynomeLagrange{-4,-3,0,2,4}{-1,3.5,2,1,0}
  \TracerCourbe[Couleur=red,Nom=cf]{polylagrange(x)}
  \MarquerPtsLagrange*[Couleur=blue]{-4,-3,0,2,4}{-1,3.5,2,1,0}
  \PlacerAntecedents[Couleurs=teal/cyan,Traits,Nom=PO]{cf}{1.75}
\end{GraphiqueTikz}
```



```
\begin{GraphiqueTikz}%
    [x=6cm,y=3cm,Xmin=-1.25,Xmax=1.25,Ymin=-0.5,Ymax=2,
    Xgrille=0.125,Xgrilles=1,Ygrille=0.25,Ygrilles=0.25]
  \TracerAxesGrilles[Elargir=2.5mm,Police=\footnotesize]%
    {-1.25,-1,...,1.25}%
    {auto}
  \GenererPolynomeLagrange%
    {seq(i,i=-1..[0.2]..1)}                %langage xint :-)
    {seq(1/(1+25*i^2),i=-1..[0.2]..1)}     %langage xint :-)
  \TracerCourbe[Couleur=lime]{1/(25*x^2+1)}
  \TracerCourbe[Couleur=red,RestreindreY]{polylagrange(x)}
  \MarquerPtsLagrange*[Couleur=blue]%
    {-1,-0.8,-0.6,-0.4,-0.2,0,0.2,0.4,0.6,0.8,1}%
    {0.038,0.058,0.1,0.2,0.5,1.0,0.5,0.2,0.1,0.058,0.038}
\end{GraphiqueTikz}
```



3.4 Gestion de points

3.4.1 Définir des points sous forme de nœuds

La seconde idée est de travailler avec des nœuds Ti*k*Z, qui pourront être utiles pour des tracés de tangentes, des représentations d'intégrales...

Il est également possible de définir des nœuds pour des points *image*.

Certaines commandes (explicités ultérieurement) permettent de déterminer des points particuliers des courbes sous forme de nœuds, donc il semble intéressant de pouvoir en définir directement.

```
%par les coordonnées
\DefinirPts[clés]{Nom1/x1/y1,Nom2/x2/y2,...}
```

Les `[clés]`, optionnelles, disponibles sont :

— `Aff` : booléen pour marquer les points (`false` par défaut) ;

— `Couleur` : couleur des points, si `Aff=true` (`black` par défaut).

```
%sous forme d'image
\DefinirImage[clés]{objet}{abscisse}
```

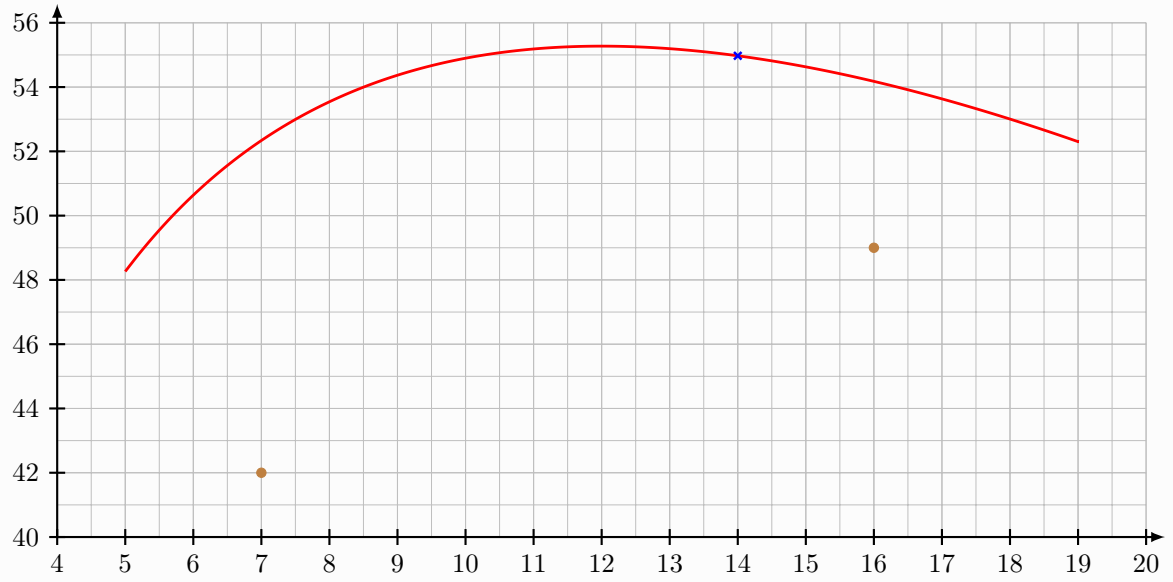
Les `[clés]`, optionnelles, disponibles sont :

— `Nom` : nom du nœud (`vide` par défaut) ;

— `Spline` : booléen pour spécifier qu'un spline est utilisé (`false` par défaut).

Le premier argument obligatoire est l'*objet* considéré (nom de la courbe pour le spline, fonction sinon) ; le second est l'abscisse du point considéré.

```
\begin{GraphiqueTikz}%
  [x=0.9cm,y=0.425cm,Xmin=4,Xmax=20,Origx=4,
  Ymin=40,Ymax=56,Ygrille=2,Ygrilles=1,Origy=40]
  \TracerAxesGrilles[Elargir=2.5mm,Police=\small]{4,5,...,20}{40,42,...,56}
  %définition de la fonction + tracé de la courbe
  \DefinirFonction[Nom=cf,Debut=5,Fin=19,Trace,Couleur=red]<f>{-2*x+3+24*log(2*x)}
  %nœuds manuels
  \DefinirPts[Aff,Couleur=brown]{A/7/42,B/16/49}
  %nœud image
  \DefinirImage[Nom=IMGf]{f}{14}
  \MarquerPts*[Style=x,Couleur=blue]{(IMGf)}
\end{GraphiqueTikz}
```



3.4.2 Marquage de points

L'idée est de proposer de quoi marquer des points avec un style particulier.

```
%dans l'environnement GraphiqueTikz
\MarquerPts(*)[clés]<police>{liste}
```

La version *étoilée* marque les points sans les « noms », alors que la version *non étoilée* les affiche :

— dans le cas de la version *étoilée*, la liste est à donner sous la forme `(ptA),(ptB),...` ;

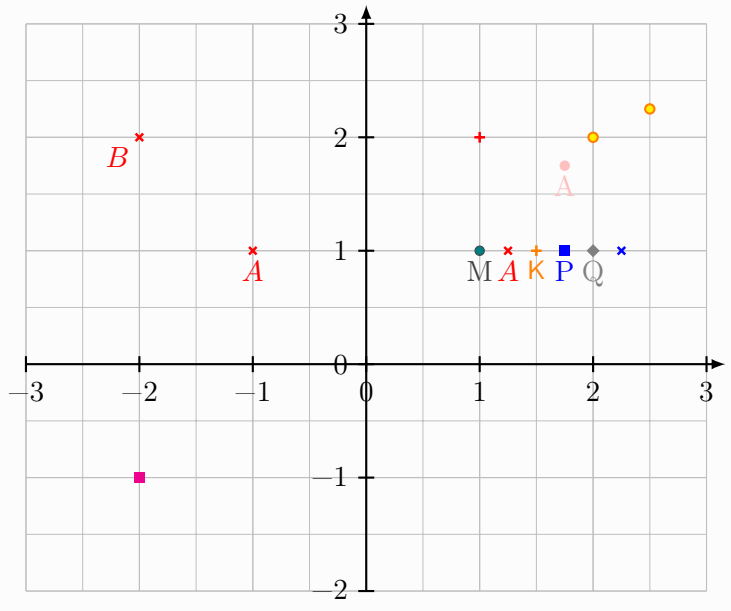
— sinon, la liste est à donner sous la forme `(ptA)/labelA/poslabelA,...`.

Les `[clés]`, optionnelles, disponibles sont :

— `Couleur` : couleur (`black` par défaut) ;

— `Style` : style des marques (`o` par défaut).

```
\begin{GraphiqueTikz}[x=1.5cm,y=1.5cm,Ymin=-2]
  \TracerAxesGrilles[Elargir=2.5mm]{auto}{auto}
  \DefinirPts{A/1.75,-1.25}\MarquerPts[Couleur=pink]{(A)/A/below} %rond (par défaut)
  \MarquerPts[Style=ggb,Couleur=darkgray/teal]{(1,1)/M/below}
  \MarquerPts[Couleur=red,Style=x]{(1.25,1)/$A$/below} %croix
  \MarquerPts[Couleur=orange,Style=+]<\small\sffamily>{(1.5,1)/K/below} %plus
  \MarquerPts[Couleur=blue,Style=c]{(1.75,1)/P/below} %carré
  \MarquerPts[Couleur=gray,Style=d]{(2,1)/Q/below} %diamant
  \MarquerPts*[Couleur=orange/yellow]{(2,2),(2.5,2.25)} %rond bicolore
  \MarquerPts*[Style=+,Couleur=red]{(1,2)}
  \MarquerPts*[Style=x,Couleur=blue]{(2.25,1)}
  \MarquerPts*[Style=c,Couleur=magenta]{(-2,-1)}
  \MarquerPts[Couleur=red,Style=x]{(-1,1)/$A$/below,(-2,2)/$B$/below left}
\end{GraphiqueTikz}
```



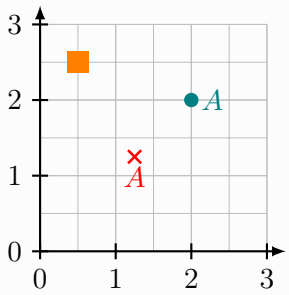
À noter qu'il est également possible de modifier la taille des marques `o/x/+/c` via les `[clés]` :

— `Taillex=...` (`2pt` par défaut) pour les points *croix* ;

— `Tailleo=...` (`1.75pt` par défaut) pour les points *cercle* ;

— `Taillec=...` (`2pt` par défaut) pour les points *carré*.

```
\begin{GraphiqueTikz}[x=1cm,y=1cm,Xmin=0,Ymin=0]
  \TracerAxesGrilles[Elargir=2.5mm]{auto}{auto}
  \MarquerPts[Couleur=red,Style=x,Taillex=3.5pt]{(1.25,1.25)/$A$/below}
  \MarquerPts[Couleur=teal,Tailleo=2.5pt]{(2,2)/$A$/right}
  \MarquerPts*[Couleur=orange,Style=c,Taillec=4pt]{(0.5,2.5)}
\end{GraphiqueTikz}
```



3.4.3 Marquer des points de discontinuité

Il est possible de marquer des points de discontinuité, mais c'est commande est *déconnectée* des commandes de tracé de courbes/splines.

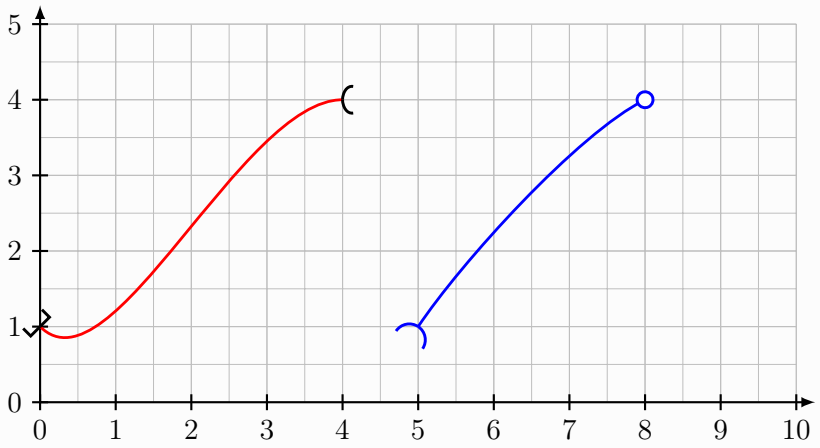
```
%dans l'environnement GraphiqueTikz
\AfficherPtsDiscont[clés]{liste}
```

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

- `Couleur=...` (`black` par défaut) ;
- `Pos=...` (`D` par défaut) pour choisir la position de la discontinuité (parmi `G/D`) ;
- `Echelle=...` (`1` par défaut) pour modifier l'échelle du symbole ;
- `Type=...` (`par` par défaut) pour choisir le type de symbole, parmi `par/cro/rond/demirond`.

Le second argument, obligatoire et entre `{...}` permet de préciser la liste des points en lesquels le symbole de discontinuité sera positionné, sous la forme `x1/y1/d1 § x2/y2/d2 § ...` avec les points `(xi;yi)` et `f'(xi)=di`.

```
\begin{GraphiqueTikz}[x=1cm,y=1cm,Xmin=0,Xmax=10,Ymin=0,Ymax=5]
  \TracerAxesGrilles[Elargir=2.5mm]{auto}{auto}
  \DefinirCourbeSpline[Trace,Couleur=red]{0/1/-1 § 4/4/0}
  \AfficherPtsDiscont{4/4/0}
  \AfficherPtsDiscont[Pos=G,Type=cro]{0/1/-1}
  \DefinirCourbeSpline[Trace,Couleur=blue]{5/1/1.5 § 8/4/0.5}
  \AfficherPtsDiscont[Couleur=blue,Type=rond]{8/4/0.5}
  \AfficherPtsDiscont[Couleur=blue,Pos=G,Type=demirond,Echelle=2]{5/1/1.5}
\end{GraphiqueTikz}
```



3.5 Récupérer les coordonnées de nœuds

Il est également possible, dans l'optique d'une réutilisation de coordonnées, de récupérer les coordonnées d'un nœud (défini ou déterminé).

Les calculs étant effectués en flottant en fonction des unités (re)calculées, les valeurs sont donc approchées !

```
%dans l'environnement GraphiqueTikz
\RecupererAbscisse{nœud}[\macrox]
\RecupererOrdonnee{nœud}[\macroy]
\RecupererCoordonnees{nœud}[\macrox][\macroy]
```

3.6 Placer du texte

À noter qu'une commande de placement de texte est disponible.

```
%dans l'environnement GraphiqueTikz
\PlacerTexte[clés]{(nœud ou coordonnées)}{texte}
```

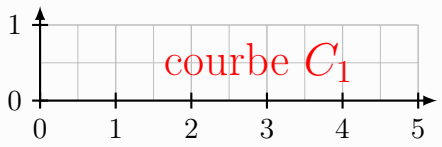
Les `[clés]` disponibles sont :

— `Police=...` (`\normalsize\normalfont` par défaut) pour la police ;

— `Couleur=...` (`black` par défaut) pour la couleur ;

— `Position=...` (`vide` par défaut) pour la position du texte par rapport aux coordonnées.

```
\begin{GraphiqueTikz}[x=1cm,y=1cm,Xmin=0,Xmax=5,Ymin=0,Ymax=1]
  \TracerAxesGrilles[Elargir=2.5mm]{auto}{auto}
  \PlacerTexte[Couleur=red,Police=\LARGE,Position=right]{(1.5,0.5)}{courbe $C_1$}
\end{GraphiqueTikz}
```



4 Commandes spécifiques d'exploitation des courbes

4.1 Placement d'images

Il est possible de la placer des points (images) sur une courbe, avec traits de construction éventuels. La fonction/courbe utilisée doit avoir été déclarée précédemment pour que cette commande fonctionne.

```
%dans l'environnement GraphiqueTikz
\PlacerImages[clés]{fonction ou courbe}{liste d'abscisses}
```

Les `[clés]`, optionnelles, disponibles sont :

— `Traits` : booléen pour afficher les traits de construction (`false` par défaut) ;

— `Couleurs` : couleur des points/traits, sous la forme `Couleurs` ou `CouleurPoint/CouleurTraits` ;

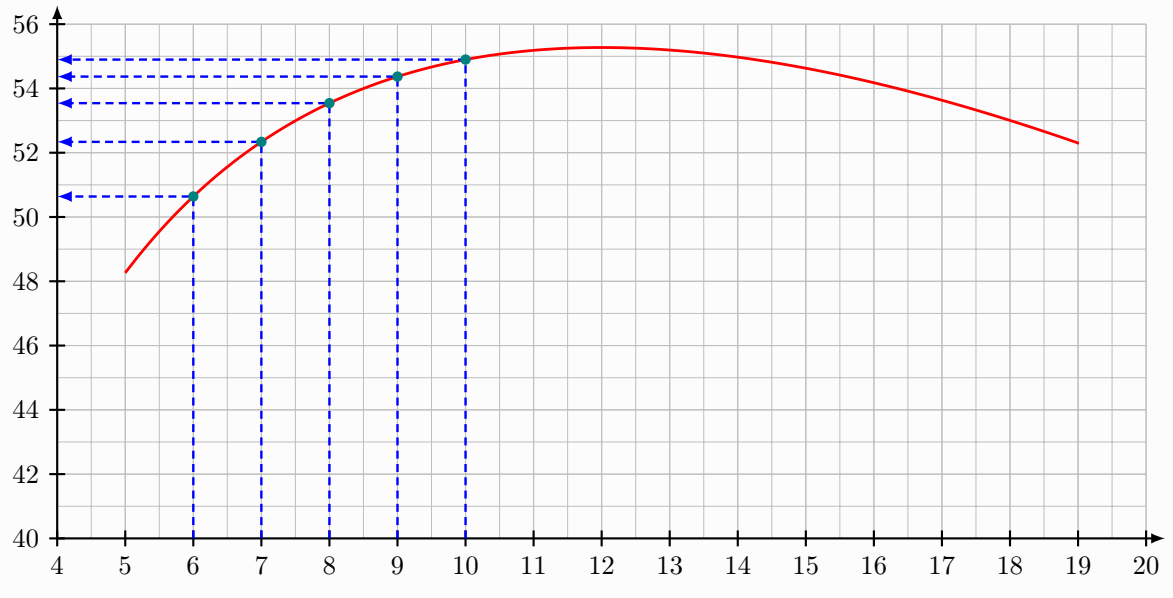
— `Spline` : booléen pour préciser que la courbe utilisée est définie comme un spline (`false` par défaut).

Le premier argument obligatoire, permet de spécifier :

— le nom de la courbe dans la cas `Spline=true` ;

— le nom de la fonction sinon.

```
\begin{GraphiqueTikz}%
  [x=0.9cm,y=0.425cm,Xmin=4,Xmax=20,Origx=4,
  Ymin=40,Ymax=56,Ygrille=2,Ygrilles=1,Origy=40]
  \TracerAxesGrilles[Elargir=2.5mm,Police=\small]{4,5,...,20}{40,42,...,56}
  %définition de la fonction + tracé de la courbe
  \DefinirCourbe[Nom=cf,Debut=5,Fin=19,Trace,Couleur=red]<f>{-2*x+3+24*log(2*x)}
  %images
  \PlacerImages[Traits,Couleurs=teal/blue]{f}{6,7,8,9,10}
\end{GraphiqueTikz}
```



4.2 Antécédents

4.2.1 Détermination d'antécédents

Il est possible de déterminer graphiquement les antécédents d'un réel donné.
La fonction/courbe utilisée doit avoir été déclarée précédemment pour que cette commande fonctionne.

```
%dans l'environnement GraphiqueTikz
\TrouverAntecedents[clés]{courbe}{k}<\myantec>
\TrouverRacines[clés]{courbe}<\myantec>
```

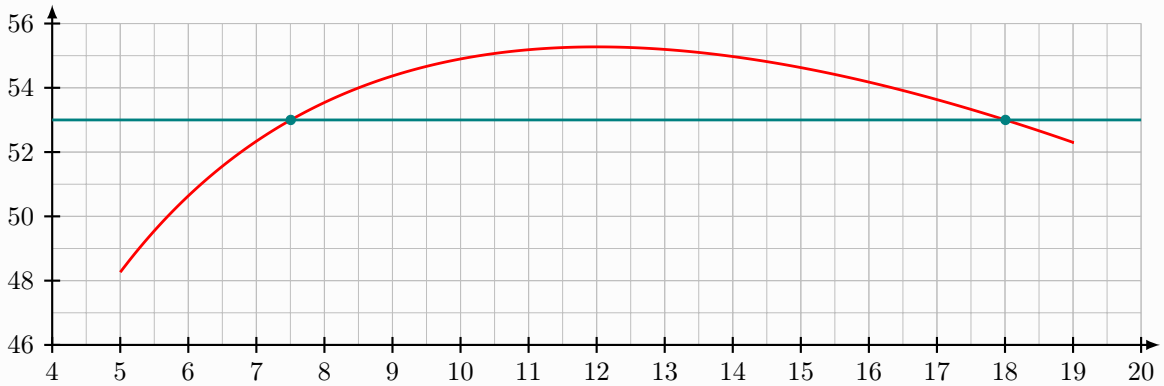
Les `[clés]`, optionnelles, disponibles sont :

- `Nom` : base du nom des **nœuds** intersection (`S` par défaut, ce qui donnera S-1, S-2, etc) ;
- `ListeRes` : permet (si non vide) de créer une macro contenant les différentes solutions ;
- `Aff` : booleen pour afficher les points (`true` par défaut) ;
- `Couleur` : couleur des points (`black` par défaut) ;
- `AffDroite` : booleen pour afficher la droite horizontale (`false` par défaut).

Le premier argument obligatoire, permet de spécifier le **nom** de la courbe.
Le second argument obligatoire, permet de spécifier la valeur à atteindre.
Le dernier argument, optionnel, permet de stocker le nombre d'antécédents.

À noter que `ListeRes` (nom de la **macro** dans laquelle sera stockée la liste CSV des abscisses des intersections) via `ListeRes=mesracines`, permet de créer la macro `\mesracines` exploitable ensuite via `\tkzgCalcElementListe`, `\tkzgCalcLongueurListe` ou `\xintFor*`.

```
\begin{GraphiqueTikz}%
  [x=0.9cm,y=0.425cm,Xmin=4,Xmax=20,Origx=4,
  Ymin=46,Ymax=56,Ygrille=2,Ygrilles=1,Origy=46]
  \TracerAxesGrilles[Elargir=2.5mm,Police=\small]{4,5,...,20}{46,48,...,56}
  %définition de la fonction + tracé de la courbe
  \DefinirCourbe[Nom=cf,Debut=5,Fin=19,Trace,Couleur=red]<f>{-2*x+3+24*log(2*x)}
  %antécédents
  \TrouverAntecedents%
    [Couleur=teal,AffDroite,Aff,ListeRes=listesols]%
    {cf}{53}<\nbsols>
  %les deux antécédents sont aux nœuds (S-1) et (S-2)
\end{GraphiqueTikz}

Il y a donc \nbsols\ solutions, qui sont
$\left\lbrace \listesols\right\rbrace$.
```



Il y a donc 2 solutions, qui sont $\{7.505389008644637, 18.00702237827507\}$.

4.2.2 Construction d'antécédents

Il est possible de construire graphiquement les antécédents d'un réel donné.
La fonction/courbe utilisée doit avoir été déclarée précédemment pour que cette commande fonctionne.

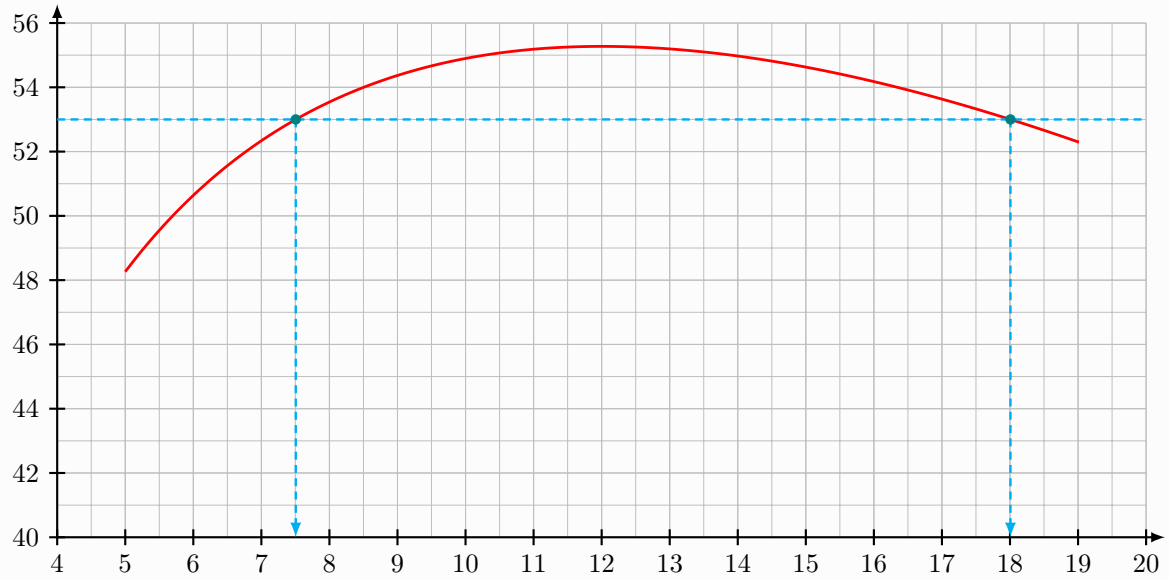
```
%dans l'environnement GraphiqueTikz
\PlacerAntecedents[clés]{courbe}{k}
```

Les `[clés]`, optionnelles, disponibles sont :

— `Couleurs` : couleur des points/traits, sous la forme `Couleurs` ou `CouleurPoint/CouleurTraits` ;
— `Nom` : nom *éventuel* pour les points d'intersection liés aux antécédents (`vide` par défaut) ;
— `Traits` : booleen pour afficher les traits de construction (`false` par défaut).

Le premier argument obligatoire, permet de spécifier le **nom** de la courbe.
Le second argument obligatoire, permet de spécifier la valeur à atteindre.

```
\begin{GraphiqueTikz}%
  [x=0.9cm,y=0.425cm,Xmin=4,Xmax=20,Origx=4,
  Ymin=40,Ymax=56,Ygrille=2,Ygrilles=1,Origy=40]
  \TracerAxesGrilles[Elargir=2.5mm,Police=\small]{4,5,...,20}{40,42,...,56}
  %définition de la fonction + tracé de la courbe
  \DefinirCourbe[Nom=cf,Debut=5,Fin=19,Trace,Couleur=red]<f>{-2*x+3+24*log(2*x)}
  %antécédents
  \PlacerAntecedents[Couleurs=teal/cyan,Traits,Nom=PO]{cf}{53}
  \RecupererAbscisse{(PO-1)}[\premsol]
  \RecupererAbscisse{(PO-2)}[\deuxsol]
\end{GraphiqueTikz}

Graphiquement, les antécédents de 53 sont (environ) :

\begin{itemize}
  \item \num{\premsol}
  \item \num{\deuxsol}
\end{itemize}
```



Graphiquement, les antécédents de 53 sont (environ) :

— 7,505 389 008 644 637
— 18,007 022 378 275 07

4.3 Intersections de deux courbes

Il est également possible de déterminer (sous forme de nœuds) les éventuels points d'intersection de deux courbes préalablement définies.

```
%dans l'environnement GraphiqueTikz
\TrouverIntersections[clés]{courbe1}{courbe2}<\myt>
```

Les `[clés]`, optionnelles, disponibles sont :

- `Nom` : base du nom des **nœuds** intersection (`S` par défaut, ce qui donnera S-1, S-2, etc) ;
- `ListeRes` : permet (si non vide) de créer une macro contenant les différentes solutions ;
- `Aff` : booléen pour afficher les points (`true` par défaut) ;
- `Couleur` : couleur des points (`black` par défaut).

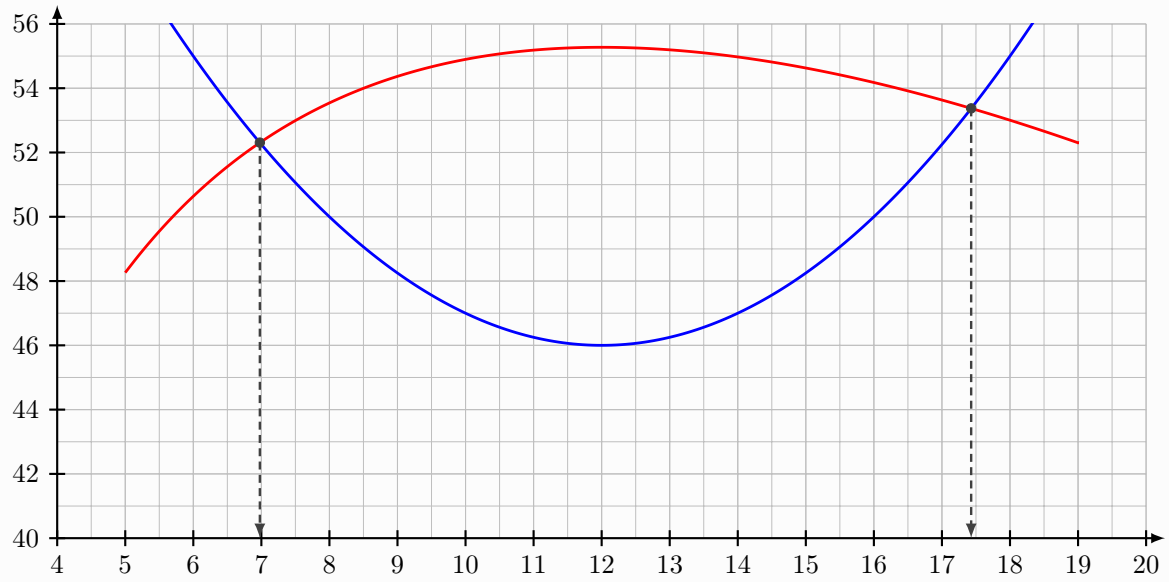
Le premier argument obligatoire, permet de spécifier le **nom** de la première courbe.

Le premier argument obligatoire, permet de spécifier le **nom** de la seconde courbe.

Le dernier argument, optionnel, permet de stocker le nombre d'antécédents.

À noter que `ListeRes` (nom de la **macro** dans laquelle sera stockée la liste CSV des abscisses des intersections) via `ListeRes=mesracines`, permet de créer la macro `\mesracines` exploitable ensuite via `\tkzgCalcElementListe`, `\tkzgCalcLongueurListe` ou `\xintFor*`.

```
\begin{GraphiqueTikz}%
  [x=0.9cm,y=0.425cm,Xmin=4,Xmax=20,Origx=4,
  Ymin=40,Ymax=56,Ygrille=2,Ygrilles=1,Origy=40]
  \TracerAxesGrilles[Elargir=2.5mm,Police=\small]{4,5,...,20}{40,42,...,56}
  \DefinirCourbe[Nom=cf,Debut=5,Fin=19,Trace,Couleur=red]<f>{-2*x+3+24*log(2*x)}
  \DefinirCourbe[Nom=cg,Debut=5,Fin=19,Trace,Couleur=blue]<g>{0.25*(x-12)^2+46}
  %intersections, nommées (TT-1) et (TT-2)
  \TrouverIntersections[Nom=TT,Couleur=darkgray,Aff,Traits]{cf}{cg}
  %récupération des points d'intersection
  \RecupererCoordonnees{(TT-1)}[\alphaA][\betaA]
  \RecupererCoordonnees{(TT-2)}[\alphaB][\betaB]
\end{GraphiqueTikz}\\
Les solutions de $f(x)=g(x)$ sont $\alpha \approx \num{\alphaA}$ et
$\beta \approx \num{\alphaB}$.\\
Les points d'intersection des courbes de $f$ et de $g$ sont donc
$(\ArrondirNum[2]{\alphaA};\ArrondirNum[2]{\betaA})$ et
$(\ArrondirNum[2]{\alphaB};\ArrondirNum[2]{\betaB})$.
```



Les solutions de $f(x) = g(x)$ sont $\alpha \approx 6{,}977\,766\,172\,581\,613$ et $\beta \approx 17{,}429\,687\,326\,385\,03$.
Les points d'intersection des courbes de f et de g sont donc $(6{,}98; 52{,}31)$ et $(17{,}43; 53{,}37)$.

4.4 Extremums

L'idée (encore *expérimentale*) est de proposer des commandes pour extraire les extremums d'une courbe définie par le package.
La commande crée le nœud correspondant, et il est du coup possible de récupérer ses coordonnées pour exploitation ultérieure.

Il est possible, en le spécifiant, de travailler sur les différentes courbes gérées par le package (fonction, interpolation, spline).
Pour des courbes singulières, il est possible que les résultats ne soient pas tout à fait ceux attendus...

💣 Pour le moment, les *limitations* sont :

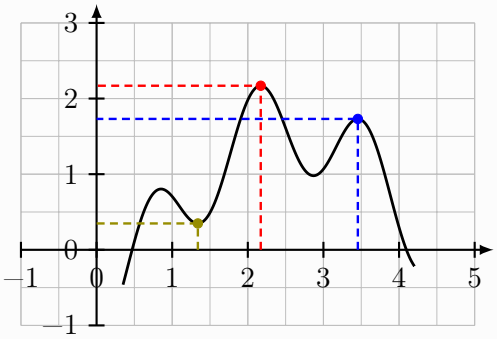
- pas de gestion d'extremums multiples (seul le premier sera traité)...
- pas de gestion d'extremums aux bornes du tracé...
- pas de récupération automatique des paramètres de définition des courbes...
- le temps de compilation peut être plus long...

```
%dans l'environnement GraphiqueTikz
\TrouverMaximum[clés]{objet}[nœud créé]
\TrouverMinimum[clés]{objet}[nœud créé]
```

Les `[clés]`, optionnelles, disponibles sont :

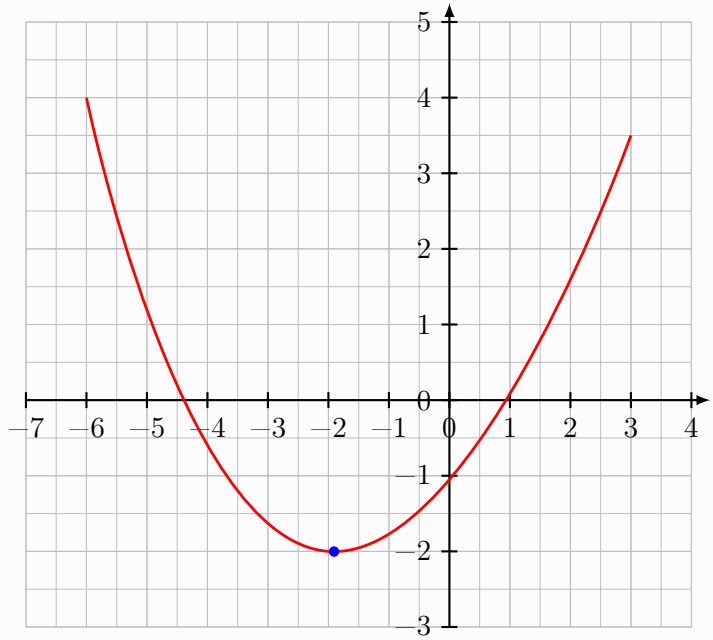
- `Methode` : méthode, parmi `fonction/interpo/spline` pour les calculs (`fonction` par défaut) ;
- `Debut` : début du tracé (`\pflxmin` par défaut) ;
- `Fin` : fin du tracé (`\pflxmax` par défaut) ;
- `Pas` : pas du tracé si `fonction` (il est déterminé *automatiquement* au départ mais peut être modifié) ;
- `Coeffs` : modifier les *coefficients* du spline si `spline` ;
- `Tension` : paramétrage de la *tension* du tracé d'interpolation si `interpo`(`0.5` par défaut).

```
\begin{GraphiqueTikz}[x=1cm,y=1cm,Xmin=-1,Xmax=5,Ymin=-1,Ymax=3]
  \TracerAxesGrilles[Elargir=2.5mm]{auto}{auto}
  \DefinirCourbe[Nom=cf,Debut=0.35,Fin=4.2,Trace]%
    <f>{0.6*cos(4.5*(x-4)+2.1)-1.2*sin(x-4)+0.1*x+0.2}
  \TrouverMaximum[Debut=0.35,Fin=4.2]{f}[cf-max]
  \TrouverMaximum[Debut=3,Fin=4]{f}[cf-maxlocal]
  \TrouverMinimum[Debut=1,Fin=2]{f}[cf-minlocal]
  \MarquerPts*[Couleur=red,Traits]{(cf-max)}
  \MarquerPts*[Couleur=blue,Traits]{(cf-maxlocal)}
  \MarquerPts*[Couleur=olive,Traits]{(cf-minlocal)}
  \RecupererCoordonnees{(cf-max)}[\MonMaxX][\MonMaxY]
\end{GraphiqueTikz}\\
Le maximum est $M\approx\ArrondirNum{\MonMaxY}$, atteint en
↪  $x\approx\ArrondirNum{\MonMaxX}$
```



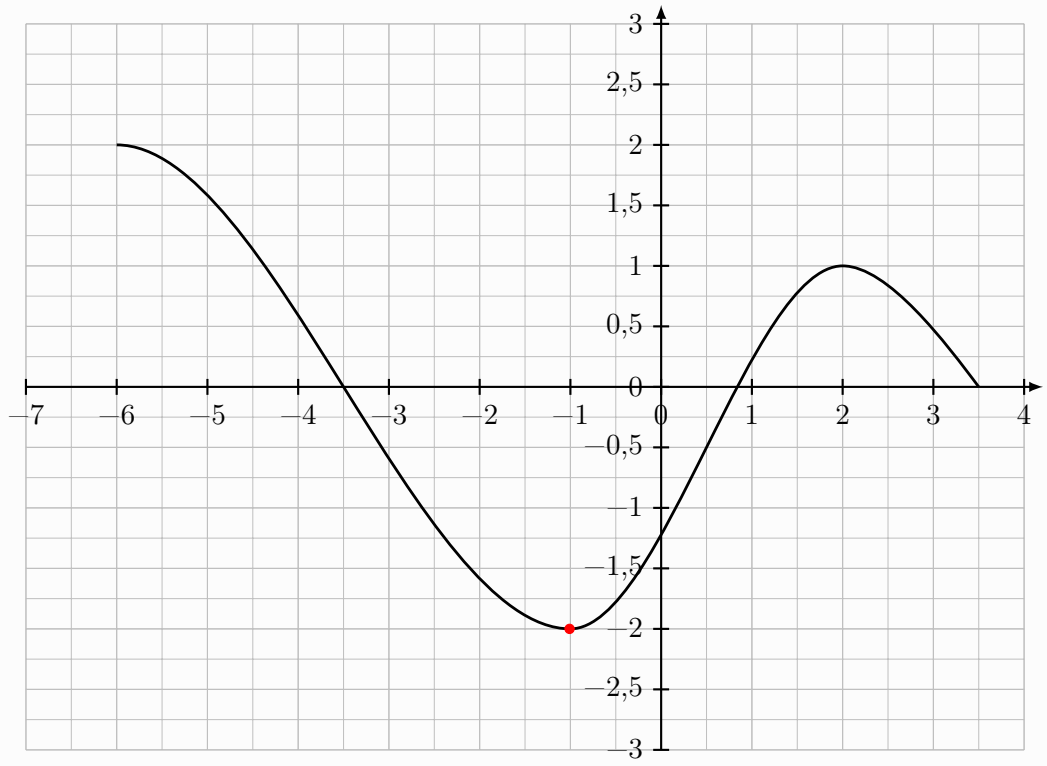
Le maximum est $M \approx 2{,}17$, atteint en $x \approx 2{,}17$

```
\begin{GraphiqueTikz}[x=0.8cm,y=1cm,Xmin=-7,Xmax=4,Ymin=-3,Ymax=5]
  \TracerAxesGrilles[Elargir=2.5mm]{-7,-6,...,4}{-3,-2,...,5}
  \DefinirCourbeInterpo[Nom=interpotest,Couleur=red,Trace,Tension=1]%
  {(-6,4)(-2,-2)(3,3.5)}
  \TrouverMinimum[Methode=interpo,Tension=1]{(-6,4)(-2,-2)(3,3.5)}[interpo-min]
  \MarquerPts*[Couleur=blue]{(interpo-min)}
  \RecupererCoordonnees{(interpo-min)}[\MinInterpoX][\MinInterpoY]
\end{GraphiqueTikz}\\
Le minimum est $M\approx\ArrondirNum[3]{\MinInterpoY}$, atteint en
↪  $x\approx\ArrondirNum[3]{\MinInterpoX}$
```



Le minimum est $M \approx -2{,}003$, atteint en $x \approx -1{,}908$

```
\begin{GraphiqueTikz}%
  [x=1.2cm,y=1.6cm,Xmin=-7,Xmax=4,Ymin=-3,Ymax=3,Ygrille=0.5,Ygrilles=0.25]
  \TracerAxesGrilles[Elargir=2.5mm]{auto}{auto}
  \def\LISTETEST{-6/2/0§-1/-2/0§2/1/0§3.5/0/-1}
  \DefinirCourbeSpline[Nom=splinetest,Trace]{\LISTETEST}
  \TrouverMinimum[Methode=spline]{\LISTETEST}[spline-min]
  \MarquerPts*[Couleur=red]{(spline-min)}
\end{GraphiqueTikz}
```



4.5 Intégrales (version améliorée)

On peut également travailler avec des intégrales.
Dans ce cas il est préférable de mettre en évidence le domaine **avant** les tracés, pour éviter la surimpression par rapport aux courbes/points.

Il est possible de :

— représenter une intégrale **sous** une courbe définie ;

— représenter une intégrale **entre** deux courbes ;

— les bornes d'intégration peuvent être des abscisses et/ou des nœuds.

💣 Compte-tenu des différences de traitement entre les courbes par formule, les courbes par interpolation simple ou les courbes par interpolation cubique, les arguments et clés peuvent différer suivant la configuration !

```
%dans l'environnement GraphiqueTikz
\TracerIntegrale[clés]<options spécifiques>{objet1}[objet2]{A}{B}
```

Les `[clés]` pour la définition ou le tracé, optionnelles, disponibles sont :

— `Couleurs` = : couleurs du remplissage, sous la forme `Couleur` ou `CouleurBord/CouleurFond` (`gray` par défaut) ;

— `Style` : type de remplissage, parmi `remplissage`/`hachures` (`remplissage` par défaut) ;

— `Opacite` : opacité (`0.5` par défaut) du remplissage ;

— `Hachures` : style (`north west lines` par défaut) du remplissage hachures ;

— `Type` : type d'intégrale parmi
  — `fct` (défaut) pour une intégrale sous une courbe définie par une formule ;
  — `spl` pour une intégrale sous une courbe définie par un spline cublique ;
  — `fct/fct` pour une intégrale entre deux courbes définie par une formule ;
  — `fct/spl` pour une intégrale entre une courbe (dessus) définie par une formule et une courbe (dessous) définie par un spline cubique ;
  — etc

— `Pas` : pas (calculé par défaut sinon) pour le tracé ;

— `Jonction` : jonction des segments (`bevel` par défaut) ;

— `Bornes` : type des bornes parmi :
  — `abs` pour les bornes données par les abscisses ;
  — `noeuds` pour les bornes données par les nœuds ;
  — `abs/noeud` pour les bornes données par abscisse et nœud ;
  — `noeud/abs` pour les bornes données par nœud et abscisse ;

— `Bord` : booléen (`true` par défaut) pour afficher les traits latéraux,

— `NomSpline` : macro (important !) du spline généré précédemment pour un spline en version supérieure ;

— `NomSplineB` : macro (important !) du spline généré précédemment pour un spline en version inférieure ;

— `NomInterpo` : nom (important !) de la courbe d'interpolation générée précédemment, en version supérieure ;

— `NomInterpoB` : nom (important !) de la courbe d'interpolation générée précédemment, en version inférieure ;

— `Tension` : tension pour la courbe d'interpolation générée précédemment, en version supérieure ;

— `TensionB` : tension de la courbe d'interpolation générée précédemment, en version inférieure.

Le premier argument obligatoire est la fonction ou la courbe du spline ou la liste de points d'interpolation.

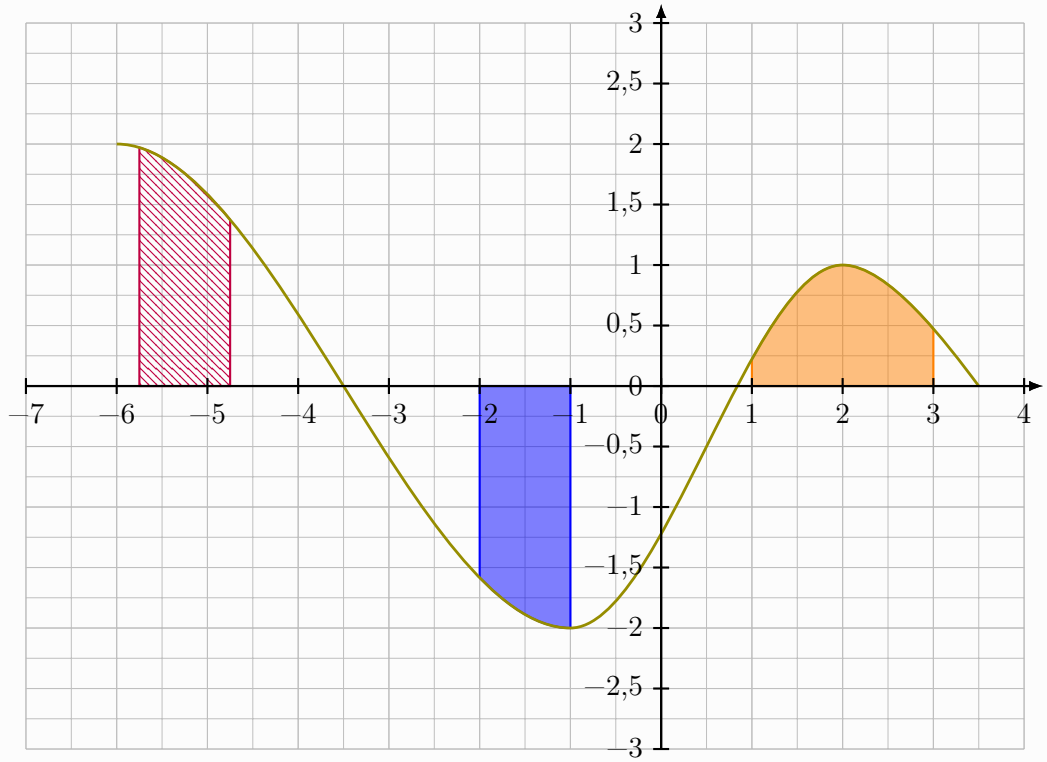
L'argument suivant, optionnel, est la fonction ou la courbe du spline ou la liste de points d'interpolation.

Les deux derniers arguments obligatoires sont les bornes de l'intégrale, données sous une forme en adéquation avec la clé `Bornes`.

Dans le cas de courbes définies par des *points*, il est nécessaire de travailler sur des intervalles sur lesquels la première courbe est **au-dessus** de la deuxième.
Il sera sans doute intéressant de travailler avec les *intersections* dans ce cas.

```
\begin{GraphiqueTikz}%
  [x=0.6cm,y=0.06cm,
  Xmin=0,Xmax=21,Xgrille=1,Xgrilles=0.5,
  Ymin=0,Ymax=155,Ygrille=10,Ygrilles=5]
  \TracerAxesGrilles%
    [Grads=false,Elargir=2.5mm]{}{}
  \DefinirCourbe[Nom=cf,Debut=1,Fin=20,Couleur=red]<f>{80*x*exp(-0.2*x)}
  \TracerIntegrale
    [Bornes=abs,Couleurs=blue/cyan!50]%
    {f(x)}{3}{12}
  \TracerCourbe[Couleur=red,Debut=1,Fin=20]{f(x)}
  \TracerAxesGrilles%
    [Grille=false,Elargir=2.5mm,Police=\small]{0,1,...,20}{0,10,...,150}
\end{GraphiqueTikz}
```

```
\begin{GraphiqueTikz}%
  [x=1.2cm,y=1.6cm,Xmin=-7,Xmax=4,Ymin=-3,Ymax=3,Ygrille=0.5,Ygrilles=0.25]
  \TracerAxesGrilles[Grads=false,Elargir=2.5mm]{}{}
  \def\LISTETEST{-6/2/0§-1/-2/0§2/1/0§3.5/0/-1}
  \DefinirCourbeSpline[Nom=splinetest]{\LISTETEST}
  \TracerIntegrale[Type=spl,Style=hachures,Couleurs=purple]{splinetest}{-5.75}{-4.75}
  \TracerIntegrale[Type=spl,Couleurs=blue]{splinetest}{-2}{-1}
  \TracerIntegrale[Type=spl,Couleurs=orange]{splinetest}{1}{3}
  \TracerCourbeSpline[Couleur=olive]{\LISTETEST}
  \TracerAxesGrilles[Grille=false,Elargir=2.5mm]
    {-7,-6,...,4}%
    {-3,-2.5,...,3}
\end{GraphiqueTikz}
```



4.6 Tangentes

L'idée de cette commande est de tracer la tangente à une courbe précédemment définie, en spécifiant :

— le point (abscisse ou nœud) en lequel on souhaite travailler ;
— éventuellement le direction (dans le cas d'une discontinuité ou d'une borne) ;
— éventuellement le pas (h) du calcul ;
— les *écartements latéraux* pour tracer la tangente.

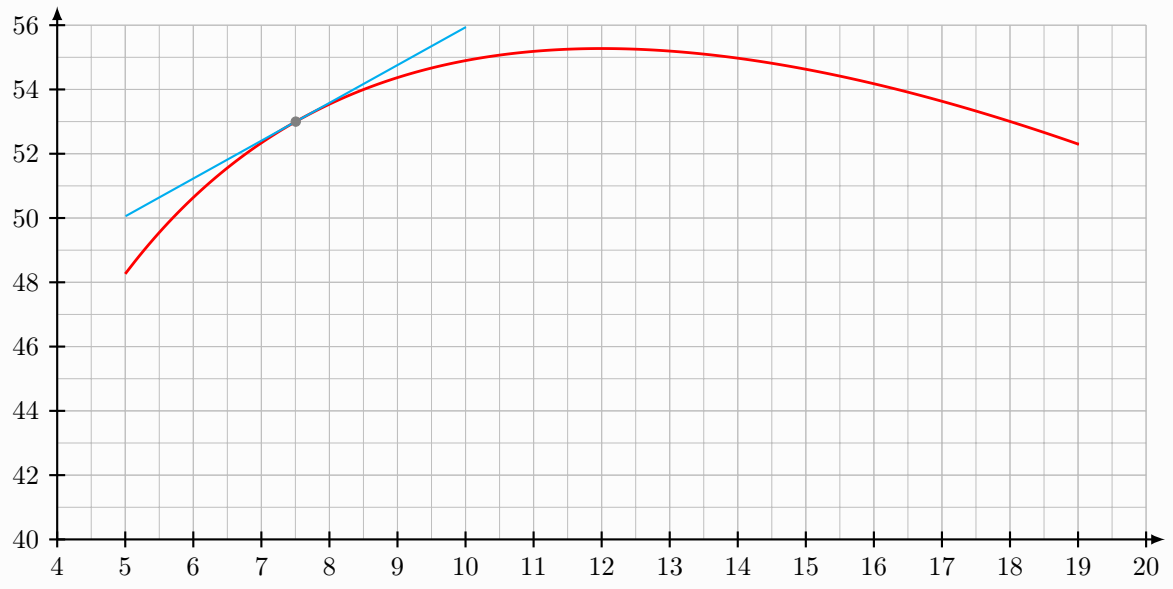
```
%dans l'environnement GraphiqueTikz
\TracerTangente[clés]{fonction ou courbe}{point}<options traits>
```

Les `[clés]` pour la définition ou le tracé, optionnelles, disponibles sont :

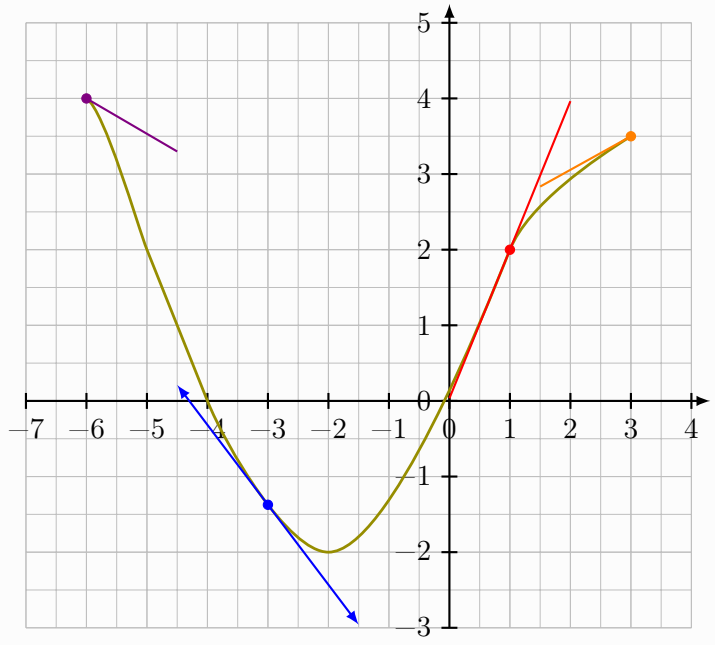
— `Couleurs` = : couleurs des tracés, sous la forme `Couleur` ou `CouleurLigne/CouleurPoint` (`black` par défaut) ;
— `DecG` = : écartement horizontal gauche pour débuter le tracé (`1` par défaut) ;
— `DecD` = : écartement horizontal gauche pour débuter le tracé (`1` par défaut) ;
— `AffPoint` : booléen pour afficher le point support (`false` par défaut) ;
— `Spline` : booléen pour préciser qu'un spline est utilisé (`false` par défaut) ;
— `h` : pas h utilisé pour les calculs (`0.01` par défaut) ;
— `Sens` : permet de sprécifier le *sens* de la tangente, parmi `gd`/`g`/`d` (`gd` par défaut) ;
— `Noeud` : booléen pour préciser qu'un nœud est utilisé (`false` par défaut).

Le premier argument obligatoire est la fonction ou la courbe du spline (le cas échéant).
Le dernier argument obligatoire est le point de travail (version abscisse ou nœud suivant la clé `Noeud`).

```
\begin{GraphiqueTikz}%
  [x=0.9cm,y=0.425cm,Xmin=4,Xmax=20,Origx=4,
  Ymin=40,Ymax=56,Ygrille=2,Ygrilles=1,Origy=40]
  \TracerAxesGrilles[Elargir=2.5mm,Police=\small]{4,5,...,20}{40,42,...,56}
  \DefinirCourbe[Nom=cf,Debut=5,Fin=19,Couleur=red,Trace]<f>{-2*x+3+24*log(2*x)}
  \TrouverAntecedents[Couleur=teal,Nom=JKL,Aff=false]{cf}{53}
  %tangente
  \TracerTangente%
    [Couleurs=cyan/gray,DecG=2.5,DecD=2.5,Noeud,AffPoint]{f}{(JKL-1)}
\end{GraphiqueTikz}
```



```
\begin{GraphiqueTikz}%
  [x=0.8cm,y=1cm,Xmin=-7,Xmax=4,Ymin=-3,Ymax=5]
  \TracerAxesGrilles[Elargir=2.5mm]{-7,-6,...,4}{-3,-2,...,5}
  \def\LISTETEST{-6/4/-0.5§-5/2/-2§-4/0/-2§-2/-2/0§1/2/2§3/3.5/0.5}
  \DefinirCourbeSpline[Nom=splinetest,Trace,Couleur=olive]{\LISTETEST}
  \TracerTangente[Couleurs=red,Spline,AffPoint]{splinetest}{1}
  \TracerTangente%
    [Couleurs=blue,Spline,DecG=1.5,DecD=1.5,AffPoint]{splinetest}{-3}%
    <pflflechegd>
  \TracerTangente[Sens=g,Couleurs=orange,Spline,DecG=1.5,AffPoint]{splinetest}{3}
  \TracerTangente[Sens=d,Couleurs=violet,Spline,DecD=1.5,AffPoint]{splinetest}{-6}
\end{GraphiqueTikz}
```



4.7 Suites

4.7.1 Nuage de points d'une suite

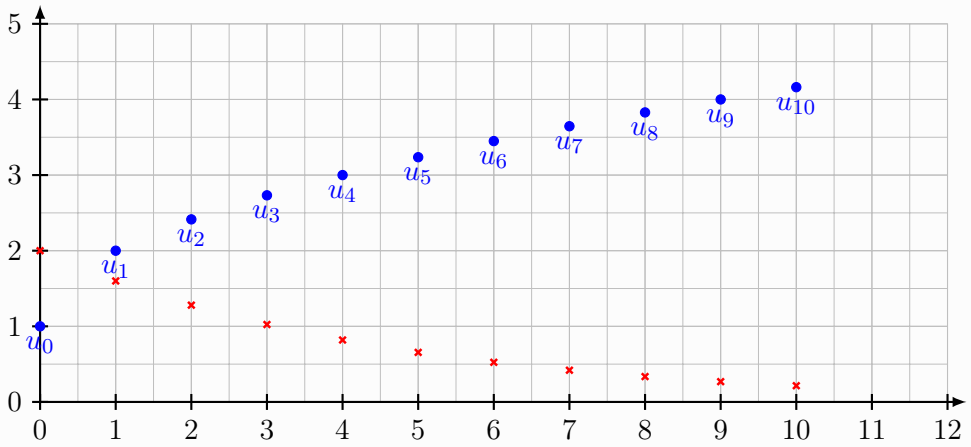
Il est possible de représenter les premiers termes d'une suite sous forme de nuage de points.

```
\TracerNuageSuite(*)[clés]{formule xint en n}
```

La version étoilée n'affiche pas le label des termes, sous la forme u_i en dessous (expérimental).
Les clés disponibles sont :

— `Debut` : indice initial (`0` par défaut) ;
— `Fin` : indice final (`10` par défaut) ;
— `Style` : style des points (`o` par défaut) ;
— `Nom` : nom de la suite (`u` par défaut) ;
— `TaillePts` : taille des points (`1.75pt` par défaut) ;
— `CouleurNuage` : couleur des points (`black` par défaut).

```
\begin{GraphiqueTikz}[Xmin=0,Xmax=12,Ymin=0,Ymax=5]
  \TracerAxesGrilles[Elargir=2.5mm]{auto}{auto}
  \TracerNuageSuite[CouleurNuage=blue,Fin=10]{sqrt(n)+1}
  \TracerNuageSuite*[CouleurNuage=red,Style=x,Fin=10]{0.8^n*2}
\end{GraphiqueTikz}
```



4.7.2 Suites récurrentes et toiles

L'idée est d'obtenir une commande pour tracer la « toile » permettant d'obtenir – graphiquement – les termes d'une suite récurrente définie par une relation $u_{n+1} = f(u_n)$.
La commande est compatible avec une fonction précédemment définie, mais également avec une courbe type *spline* précédemment définie.

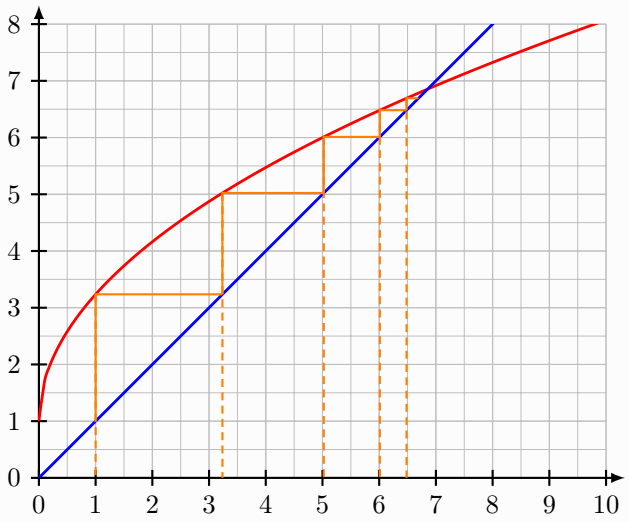
```
%dans l'environnement GraphiqueTikz
\TracerToileRecurrence[clés]{fct ou courbe}
```

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

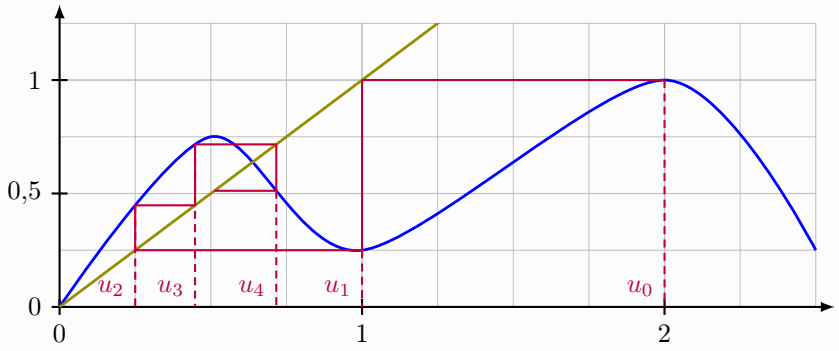
- `Couleur=...` (`black` par défaut) ;
- `Spline=...` (`false` par défaut) pour spécifier qu'une courbe *spline* est utilisée ;
- `No=...` (`0` par défaut) est l'indice initial ;
- `Uno=...` est qui est la valeur du terme initial (à donner obligatoirement !) ;
- `Nom=...` (`u` par défaut) est le nom de la suite ;
- `Nb=...` (`5` par défaut) ;
- `AffTermes=...` (`false` par défaut) qui est un booléen pour afficher les termes ;
- `AffPointilles=...` (`true` par défaut) pour afficher les pointillés ;
- `TailleLabel=...` (`\small` par défaut) ;
- `PosLabel=...` (`below` par défaut).

Le second argument, obligatoire et entre `{...}` permet de préciser l'objet avec lequel il faut effectuer les tracés (fonction ou nom_courbe).

```
\begin{GraphiqueTikz}%
  [x=0.75cm,y=0.75cm,Xmin=0,Xmax=10,Xgrille=1,Xgrilles=0.5,
  Ymin=0,Ymax=8,Ygrille=1,Ygrilles=0.5]
  \TracerAxesGrilles[Elargir=2.5mm,Police=\small]{auto}{auto}
  \DefinirCourbe[Couleur=red,Nom=cf,Debut=0,Fin=10,Trace]<f>{sqrt(5*x)+1}
  \TracerCourbe[Couleur=blue]{x}
  \TracerToileRecurrence[Couleur=orange,No=1,Uno=1]{f}
\end{GraphiqueTikz}
```



```
\begin{GraphiqueTikz}[x=4cm,y=3cm,Xmin=0,Xmax=2.5,Xgrille=1,Xgrilles=0.25,
  Ymin=0,Ymax=1.25,Ygrille=0.5,Ygrilles=0.25]
  \TracerAxesGrilles[Elargir=2.5mm,Police=\small]{auto}{auto}
  \DefinirCourbeInterpo[Nom=interpotest,Couleur=blue,Trace]%
    {(0,0)(0.5,0.75)(1,0.25)(2,1)(2.5,0.25)}
  \TracerCourbe[Couleur=olive]{x}
  \TracerToileRecurrence%
    [AffTermes,Couleur=purple,Spline,No=0,Uno=2,PosLabel=above left]%
    {interpotest}
\end{GraphiqueTikz}
```



4.8 Inégalité linéaire

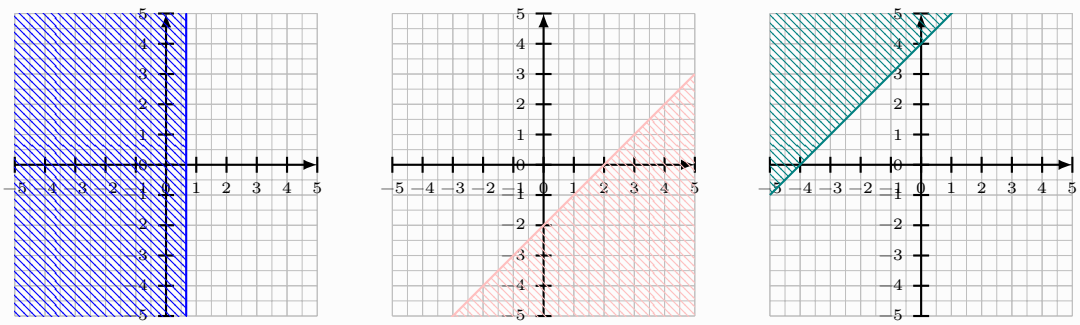
L'idée (en test) est d'obtenir une commande pour représenter graphiquement une inégalité linéaire.

```
%dans l'environnement GraphiqueTikz
\InegaliteLineaire[clés]<nom fct>{expression}{sens inégalité}
```

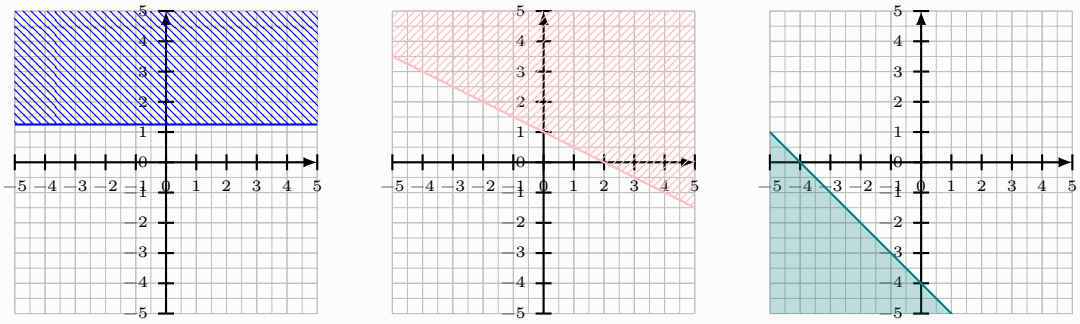
Le premier argument, *optionnel* et entre `[...]`, contient les `Clés` suivantes :

— `Opacite=...` (`0.25` par défaut) ;
— `Couleur=...` (`black` par défaut) ;
— `Style=...` (`hachures` par défaut) ;
— `Hachures=...` (`north west lines` par défaut).

```
\begin{GraphiqueTikz}[x=0.4cm,y=0.4cm,Xmin=-5,Xmax=5,Ymin=-5,Ymax=5]
  \TracerAxesGrilles[Police=\tiny]{auto}{auto}
  \InegaliteLineaire[Couleur=blue]{-3x+2}{>0}
\end{GraphiqueTikz}
~~
\begin{GraphiqueTikz}[x=0.4cm,y=0.4cm,Xmin=-5,Xmax=5,Ymin=-5,Ymax=5]
  \TracerAxesGrilles[Police=\tiny]{auto}{auto}
  \InegaliteLineaire[Couleur=pink]{-x+y+2}{<=0}
\end{GraphiqueTikz}
~~
\begin{GraphiqueTikz}[x=0.4cm,y=0.4cm,Xmin=-5,Xmax=5,Ymin=-5,Ymax=5]
  \TracerAxesGrilles[Police=\tiny]{auto}{auto}
  \InegaliteLineaire[Couleur=teal]{-x+y-4}{>0}
\end{GraphiqueTikz}
```



```
\begin{GraphiqueTikz}[x=0.4cm,y=0.4cm,Xmin=-5,Xmax=5,Ymin=-5,Ymax=5]
  \TracerAxesGrilles[Police=\tiny]{auto}{auto}
  \InegaliteLineaire[Couleur=blue]{-4*y+5}{<=0}
\end{GraphiqueTikz}
~~
\begin{GraphiqueTikz}[x=0.4cm,y=0.4cm,Xmin=-5,Xmax=5,Ymin=-5,Ymax=5]
  \TracerAxesGrilles[Police=\tiny]{auto}{auto}
  \InegaliteLineaire[Couleur=pink,Hachures={north east lines}]{-x-2y+2}{<=0}
\end{GraphiqueTikz}
~~
\begin{GraphiqueTikz}[x=0.4cm,y=0.4cm,Xmin=-5,Xmax=5,Ymin=-5,Ymax=5]
  \TracerAxesGrilles[Police=\tiny]{auto}{auto}
  \InegaliteLineaire[Style=remplissage,Couleur=teal]{-x-y-4}{>0}
\end{GraphiqueTikz}
```



4.9 Dérivée, primitive(s)

Il est également (c'est expérimental) possible d'afficher la courbe de la dérivée d'une fonction, ou la courbe d'une primitive particulière d'une fonction.

```
%courbe de la dérivée
\TracerDerivee[clés]{objet}
```

```
%courbe d'une primitive (uniquement pour des courbes via fonction f(x) !!)
%la primitive tracée sera celle telle que F(a)=b
\TracerPrimitive[clés]{nom fonction}{a}{b}
```

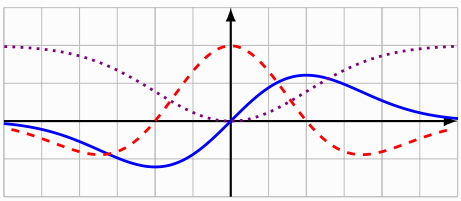
Les clés disponibles sont les mêmes que pour la définition des courbes, avec quelques spécificités :

- `Debut` : borne inférieure (**important**) de l'ensemble de définition (`\pflxmin` par défaut) ;
- `Fin` : borne supérieure (**important**) de l'ensemble de définition (`\pflxmax` par défaut) ;
- `Nom` : nom éventuel de la courbe ;
- `Couleur` : couleur du tracé (`black` par défaut) ;
- `Pas` : pas du tracé (il est déterminé *automatiquement* au départ mais peut être modifié) ;
- `StyleTrace` : style du tracé (`vide` par défaut) ;
- `Spline` : booléen pour spécifier qu'une courbe type `[Spline]` est utilisée ;
- `h` : pas (interne) pour effectuer les calculs (`vide` par défaut).

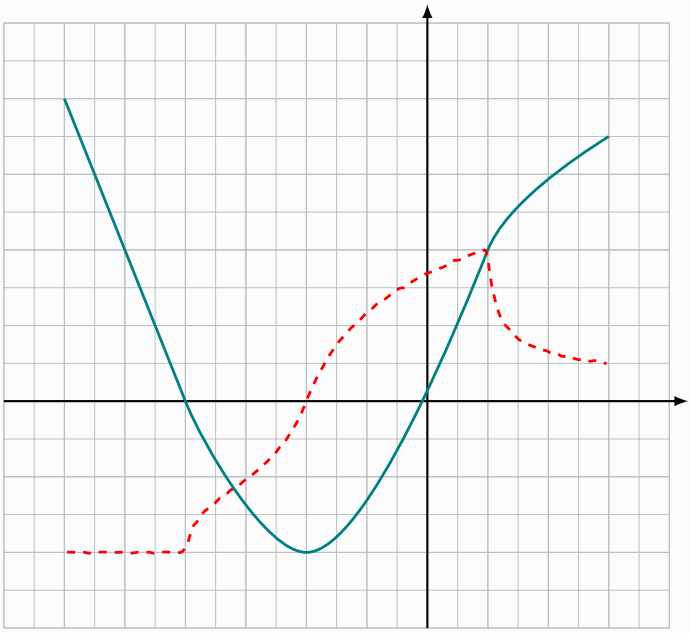
💣 Quelques remarques importantes :

- le pas `h` est déterminé automatiquement par le code, mais peut être modifié pour améliorer le redu (attention à la multiplicité des calculs en interne qui sont effectués !) ;
- la détermination via la clé `[Spline]` se fait grâce à la librairie `intersections`, et de ce fait la précision des résultats et le rendu de la courbe peut être altéré...
- il est conseillé de tracer la courbe dérivée via la clé `[StyleTrace=dashed]` pour éviter les rendus chaotiques.

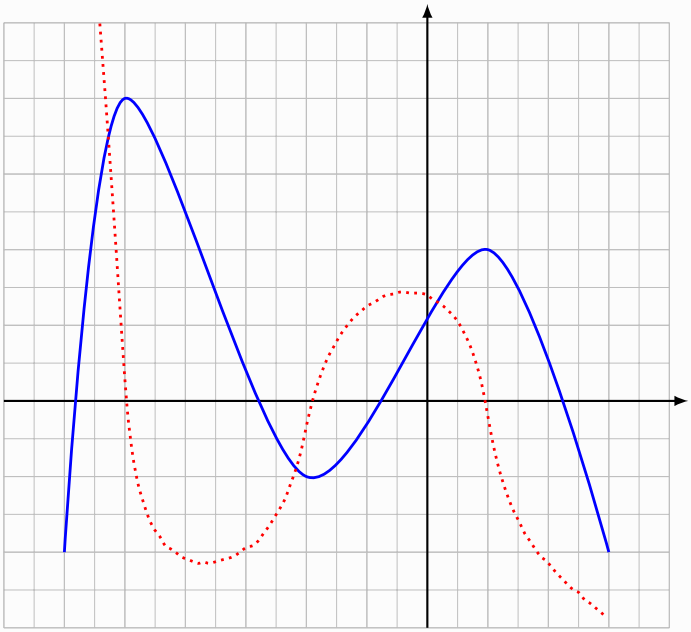
```
\begin{GraphiqueTikz}[Ymin=-1,Ymax=1.5]
  \TracerAxesGrilles{}{}
  \DefinirCourbe[Trace,Couleur=blue]<f>{x*exp(-x^2/2)}
  \TracerDerivee[Couleur=red,StyleTrace=dashed]{f}
  \TracerPrimitive[Couleur=violet,StyleTrace=dotted]{f}{0}{0}
\end{GraphiqueTikz}
```



```
\begin{GraphiqueTikz}[x=0.8cm,y=1cm,Xmin=-7,Xmax=4,Ymin=-3,Ymax=5]
  \TracerAxesGrilles[Grads=false,Elargir=2.5mm]{}{}
  \DefinirListeSpline%
    {-6/4/-2§-5/2/-2§-4/0/-2§-2/-2/0§1/2/2§3/3.5/0.5}%
    [\lstsplineA]
  \DefinirCourbeSpline%
    [Nom=splinetest,Trace,Couleur=teal]%
    {\lstsplineA}
  % dérivée
  \TracerDerivee%
    [Couleur=red,Spline,Debut=-6,Fin=3,StyleTrace=dashed]%
    {splinetest}
\end{GraphiqueTikz}
```



```
\begin{GraphiqueTikz}[x=0.8cm,y=1cm,Xmin=-7,Xmax=4,Ymin=-3,Ymax=5]
  \TracerAxesGrilles[Grads=false,Elargir=2.5mm]{}{}
  \DefinirListeInterpo%
    {(-6,-2)(-5,4)(-2,-1)(1,2)(3,-2)}%
    [\interpoA]
  \DefinirCourbeInterpo%
    [Nom=interpotest,Couleur=blue,Trace]{\interpoA}
  %dérivée
  \TracerDerivee%
    [Couleur=red,Spline,Debut=-6,Fin=3,StyleTrace=dotted,h=0.05]%
    {interpotest}
\end{GraphiqueTikz}
```



5 Commandes spécifiques des lois de probabillités

5.1 Lois continues

5.1.1 Aires sous les courbes continues

Pour toutes les lois continues (normale, exponentielle, Student, Fischer), il est possible de colorer une aire sous la courbe via la commande `\RepresenterProbaContinue`, alias de `\TracerIntegrale`.

```
%dans l'environnement GraphiqueTikz
\RepresenterProbaContinue[clés]{fct(x)}{a}{b}
```

5.1.2 Loi normale

L'idée est de proposer de quoi travailler avec des lois normales.

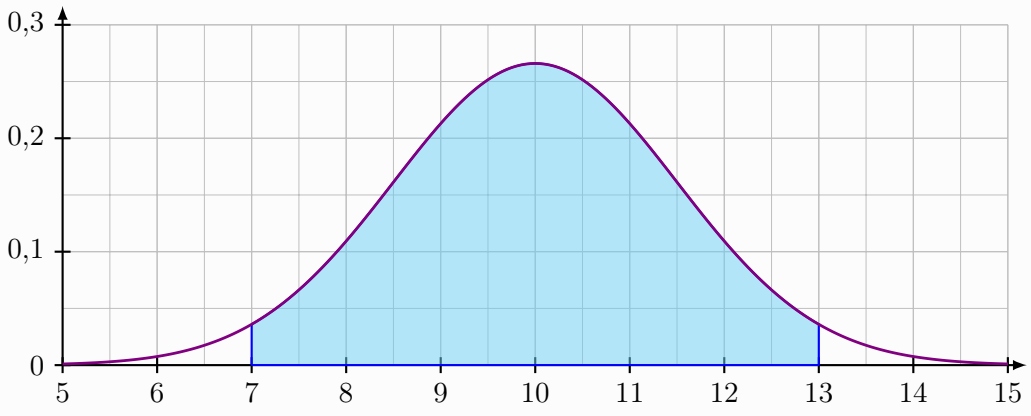
```
%dans l'environnement GraphiqueTikz
\DefinirLoiNormale[clés]<nom fct>{mu}{sigma}
\TracerLoiNormale[clés]{fct(x)}
```

Les `[clés]`, optionnelles, disponibles sont :

— `Nom` : nom du tracé (`gaussienne` par défaut) ;
— `Trace` : booléen pour tracer la courbe(`false` par défaut) ;
— `Couleur` : couleur du tracé, si demandé (`black` par défaut) ;
— `Debut` : borne inférieure de l'ensemble de définition (`\pflxmin` par défaut) ;
— `Fin` : borne inférieure de l'ensemble de définition (`\pflxmax` par défaut) ;
— `Pas` : pas du tracé (il est déterminé *automatiquement* au départ mais peut être modifié).

À noter que l'axe vertical est à adapter en fonction des paramètres de la loi normale.

```
\begin{GraphiqueTikz}%
    [x=1.25cm,y=15cm,Origx=5,Xmin=5,Xmax=15,Ymin=0,Ymax=0.3,
    Ygrille=0.1,Ygrilles=0.05]
  \TracerAxesGrilles[Elargir=2.5mm]{auto}{auto}
  \DefinirLoiNormale[Nom=gaussienne]<phi>{10}{1.5}
  \TracerIntegrale
    [Bornes=abs,Couleurs=blue/cyan!50]%
    {phi(x)}{7}{13}
  \TracerLoiNormale[Couleur=violet,Debut=5,Fin=15]{phi(x)}
\end{GraphiqueTikz}
```



5.1.3 Loi exponentielle

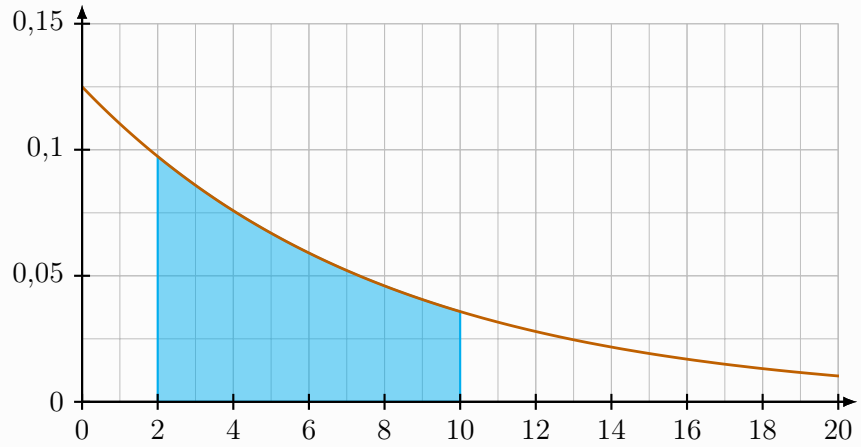
La loi exponentielle de paramètre λ est définie sur $[0\,;+\infty[$. La densité est $f(x)=\lambda e^{-\lambda x}$.

```
%dans l'environnement GraphiqueTikz
\DefinirLoiExpo[clés]<nom fct>{lambda}
\TracerLoiExpo[clés]{fct(x)}
```

Le premier argument, optionnel et entre `[...]` propose les clés suivantes pour `\DefinirLoiExpo` :

— `Nom` : nom du tracé (`expo` par défaut) ;
— `Debut` : borne inférieure (`0` par défaut) ;
— `Fin` : borne supérieure (`\pflxmax` par défaut) ;
— `Pas` : pas du tracé (déterminé automatiquement).

```
\begin{GraphiqueTikz}%
    [Xgrille=2,Xgrilles=1,Ygrille=0.05,Ygrilles=0.025]%
    <Xmin=0,Xmax=20,Ymin=0,Ymax=0.15,Taille={10cm/5cm}>
  \TracerAxesGrilles[Elargir=2.5mm,Dernier,Derriere]{}{}
  \DefinirLoiExponentielle[]<fdexpo>{0.125}
  \RepresenterProbaContinue[Couleurs=cyan]{fdexpo(x)}{2}{10}
  \TracerLoiExponentielle[Couleur=orange!75!black]{fdexpo(x)}
  \TracerAxesGrilles[Elargir=2.5mm,Dernier,Devant]{auto}{auto}
\end{GraphiqueTikz}
```



5.1.4 Loi du khi deux

L'idée est de proposer de quoi travailler avec des lois normales.

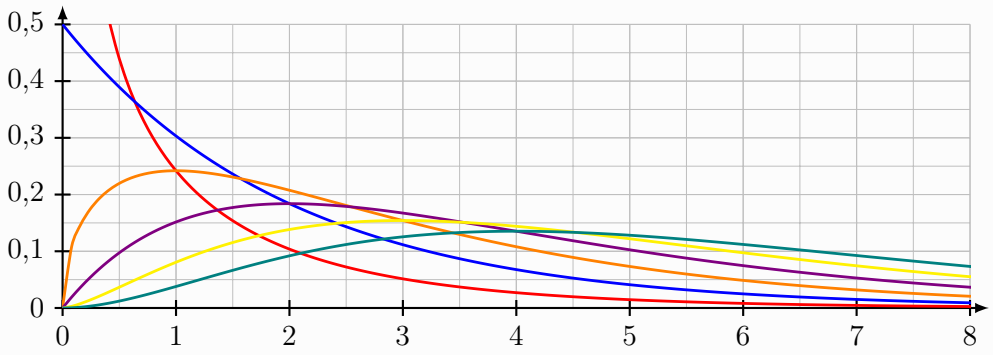
```
%dans l'environnement GraphiqueTikz
\DefinirLoiKhiDeux[clés]<nom fct>{k}
\TracerLoiKhiDeux[clés]{fct(x)}
```

Les `[clés]`, optionnelles, disponibles sont :

— `Nom` : nom du tracé (`gaussienne` par défaut) ;
— `Trace` : booléen pour tracer la courbe(`false` par défaut) ;
— `Couleur` : couleur du tracé, si demandé (`black` par défaut) ;
— `Debut` : borne inférieure de l'ensemble de définition (`\pflxmin` par défaut) ;
— `Fin` : borne inférieure de l'ensemble de définition (`\pflxmax` par défaut) ;
— `Pas` : pas du tracé (il est déterminé *automatiquement* au départ mais peut être modifié).

À noter que l'axe vertical est à adapter en fonction du paramètre de la loi du khi deux.

```
\begin{GraphiqueTikz}[
    x=1.5cm,y=7.5cm,
    Xmin=0,Xmax=8,Xgrille=1,Xgrilles=0.5,
    Ymin=0,Ymax=0.5,Ygrille=0.1,Ygrilles=0.05
    ]
  \TracerAxesGrilles[Elargir=2.5mm]{auto}{auto}
  \DefinirLoiKhiDeux[Couleur=red,Debut=0.25,Trace]<phiA>{1}
  \DefinirLoiKhiDeux[Couleur=blue,Trace]<phiB>{2}
  \DefinirLoiKhiDeux[Couleur=orange,Trace]<phiC>{3}
  \DefinirLoiKhiDeux[Couleur=violet,Trace]<phiD>{4}
  \DefinirLoiKhiDeux[Couleur=yellow,Trace]<phiE>{5}
  \DefinirLoiKhiDeux[Couleur=teal,Trace]<phiF>{6}
\end{GraphiqueTikz}
```



5.1.5 Loi de Student

La loi de Student à ν degrés de liberté est une loi continue, symétrique, dont la densité fait intervenir la fonction Γ.

```
%dans l'environnement GraphiqueTikz
\DefinirLoiStudent[clés]<nom fct>{nu}
\TracerLoiStudent[clés]{fct(x)}
```

Le premier argument, optionnel et entre `[...]` propose les clés suivantes pour `\DefinirLoiStudent` :

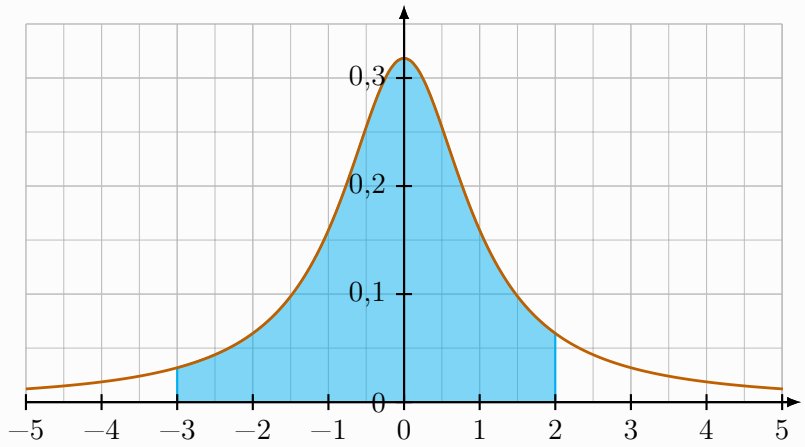
— `Nom` : nom du tracé (`student` par défaut) ;

— `Debut`, `Fin` : bornes de définition (`\pflxmin`, `\pflxmax` par défaut) ;

— `Pas` : pas du tracé (déterminé automatiquement).

À noter que la fenêtre est à adapter en fonction de ν — typiquement $x \in [-4\,;\,4]$ et $y \in [0\,;\,0.40]$ pour $\nu = 5$.

```
%avec mise en évidence d'une proba/intégrale
\begin{GraphiqueTikz}%
    [Xgrille=1,Xgrilles=0.5,Ygrille=0.1,Ygrilles=0.05]%
    <Xmin=-5,Xmax=5,Ymin=0,Ymax=0.35,Taille={10cm/5cm}>
  \TracerAxesGrilles[Elargir=2.5mm,Dernier,Derriere]{}{}
  \DefinirLoiStudent[]<fdstudent>{1}
  \RepresenterProbaContinue[Couleurs=cyan]{fdstudent(x)}{-3}{2}
  \TracerLoiStudent[Couleur=orange!75!black]{fdstudent(x)}
  \TracerAxesGrilles[Elargir=2.5mm,Dernier,Devant]{auto}{auto}
\end{GraphiqueTikz}
```



5.1.6 Loi de Fischer

La loi de Fischer à $(d_1\,;d_2)$ degrés de liberté est définie sur $]0;+\infty[$. La fenêtre est à adapter en conséquence — typiquement $x \in [0;6]$ pour $F_{5;10}$.

```
%dans l'environnement GraphiqueTikz
\DefinirLoiFischer[clés]<nom fct>{d1}{d2}
\TracerLoiFischer[clés]{fct(x)}
```

Le premier argument, optionnel et entre `[...]` propose les clés suivantes pour `\DefinirLoiFischer` :

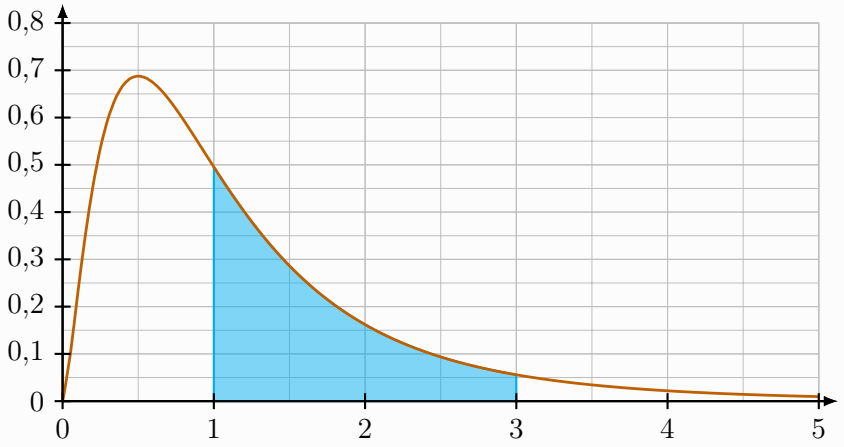
— `Nom` : nom du tracé (`fischer` par défaut) ;

— `Debut` : borne inférieure — forcée à $\max(0.001, \text{Debut})$ pour éviter la singularité en 0 ;

— `Fin` : borne supérieure (`\pflxmax` par défaut) ;

— `Pas` : pas du tracé (déterminé automatiquement).

```
\begin{GraphiqueTikz}%
    [Xgrille=1,Xgrilles=0.5,Ygrille=0.1,Ygrilles=0.05]%
    <Xmin=0,Xmax=5,Ymin=0,Ymax=0.8,Taille={10cm/5cm}>
  \TracerAxesGrilles[Elargir=2.5mm,Dernier,Derriere]{}{}
  \DefinirLoiFischer[]<fdfischer>{5}{10}
  \RepresenterProbaContinue[Couleurs=cyan]{fdfischer(x)}{1}{3}
  \TracerLoiFischer[Couleur=orange!75!black]{fdfischer(x)}
  \TracerAxesGrilles[Elargir=2.5mm,Dernier,Devant]{auto}{auto}
\end{GraphiqueTikz}
```



5.2 Lois discrètes

5.2.1 Loi binomiale

Il est également possible (d'une manière moins explicite que dans `ProfLycee`) de représenter l'histogramme d'une loi binomiale (`ProfLycee` permet de déterminer les unités automatiquement, ici elles doivent être précisées et connues).
Il est également possible de rajouter la loi normale « associée ».

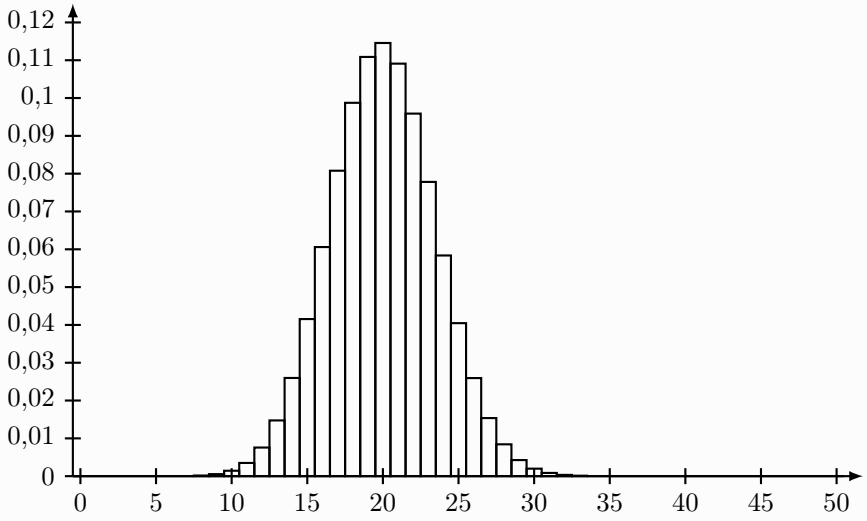
```
%dans l'environnement GraphiqueTikz
\TracerHistoBinomiale[clés]<nom fct normale>{n}{p}
```

Le premier argument, optionnel et entre `[...]` propose les clés suivantes :

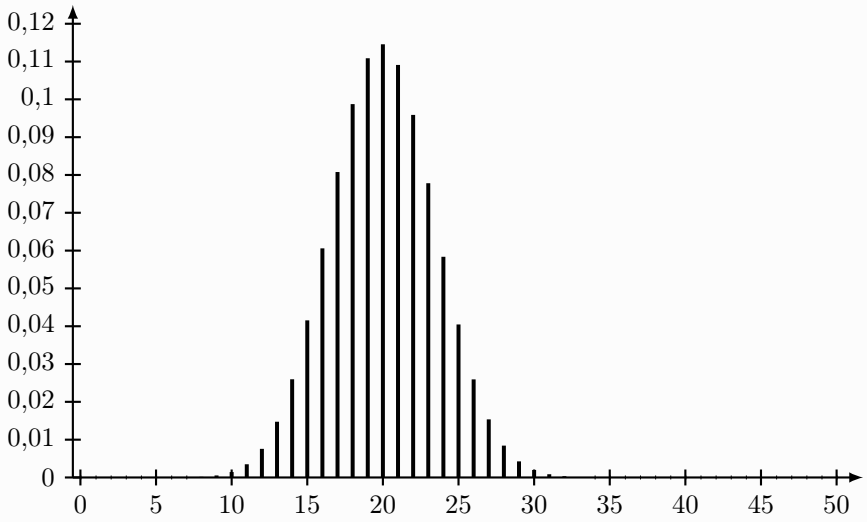
- `CouleurTraits` : couleur des traits/bâtons (`black` par défaut) ;
- `Plage` : plage, sous la forme `a-b` du coloriage éventuel ;
- `CouleurPlage` : couleur de la plage éventuelle ;
- `ClipX` : restriction de l'axe Ox, sous la forme `a-b` ;
- `AffNormale` : booléen (`true` par défaut) pour rajouter la loi normale ;
- `CouleurNormale` : couleur pour la loi normale ;
- `Style` : style de représentation, `histo` (rectangles, par défaut) ou `batons` (segments verticaux).

Les arguments obligatoires et entre `{...}` permettent de spécifier les paramètres de la loi binomiale.

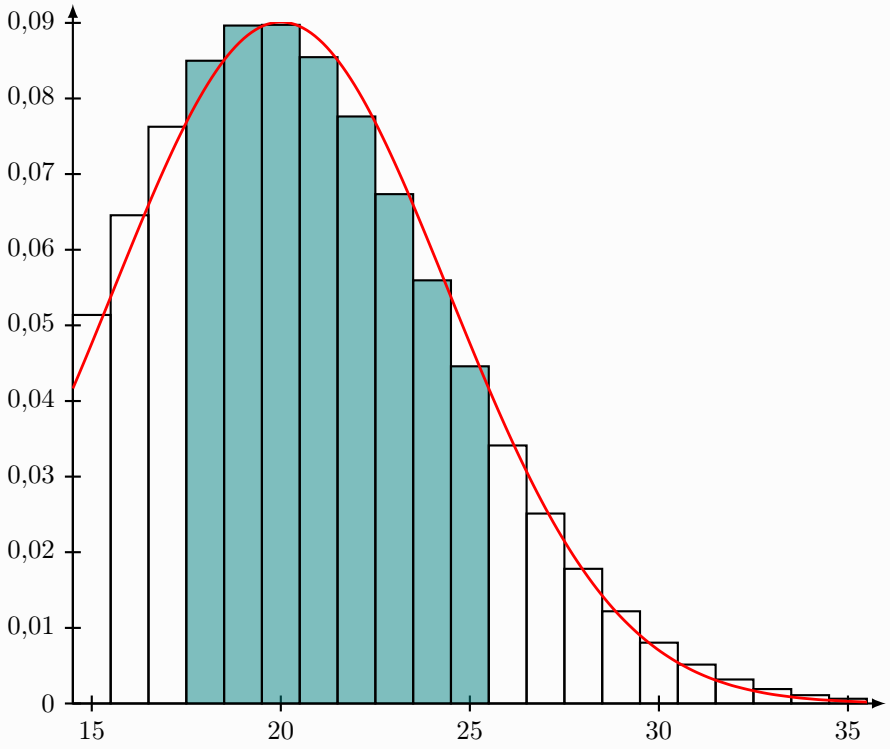
```
%les unités ont été déterminées au préalable...
\begin{GraphiqueTikz}[x=0.2cm,y=50cm,Origx=-0.5,Xmin=-0.5,Xmax=50.5,
  Xgrille=5,Xgrilles=1,Ymin=0,Ymax=0.12,Ygrille=0.01,Ygrilles=0.001]
  \TracerAxesGrilles[Elargir=2.5mm,Police=\small,Grille=false]%
    {0,5,...,50}{auto}
  \TracerHistoBinomiale{50}{0.4}
\end{GraphiqueTikz}
```




```
%les unités ont été déterminées au préalable...
\begin{GraphiqueTikz}[x=0.2cm,y=50cm,Origx=-0.5,Xmin=-0.5,Xmax=50.5,
  Xgrille=5,Xgrilles=1,Ymin=0,Ymax=0.12,Ygrille=0.01,Ygrilles=0.001]
  \TracerAxesGrilles[Elargir=2.5mm,Police=\small,Grille=false]%
    {0,5,...,50}{auto}
  \TracerHistoBinomiale[Style=batons]{50}{0.4}
\end{GraphiqueTikz}
```



```
%les unités ont été déterminées au préalable...
\begin{GraphiqueTikz}[x=0.5cm,y=100cm,Origx=14.5,Xmin=14.5,Xmax=35.5,
  Xgrille=5,Xgrilles=1,Ymin=0,Ymax=0.09,Ygrille=0.01,Ygrilles=0.001]
  \TracerAxesGrilles[Elargir=2.5mm,Police=\small,Grille=false]%
    {15,20,...,35}{auto}
  \TracerHistoBinomiale%
    [ClipX=15-35,Plage=18-25,CouleurPlage=teal,AffNormale,CouleurNormale=red]%
    {1000}{0.02}
\end{GraphiqueTikz}
```



5.2.2 Loi de Poisson

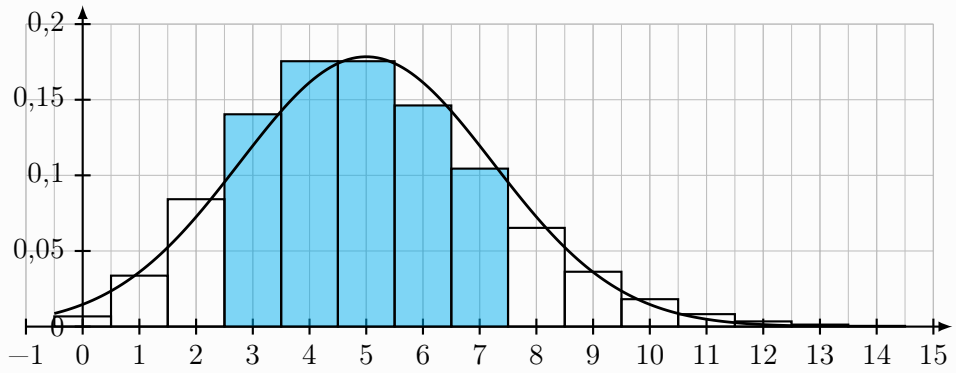
La loi de Poisson peut être représentée de façon similaire à la loi binomiale. La borne supérieure de représentation est déterminée automatiquement en fonction de λ.

```
%dans l'environnement GraphiqueTikz
\TracerLoiPoisson[clés]<nom fct>{lambda}
```

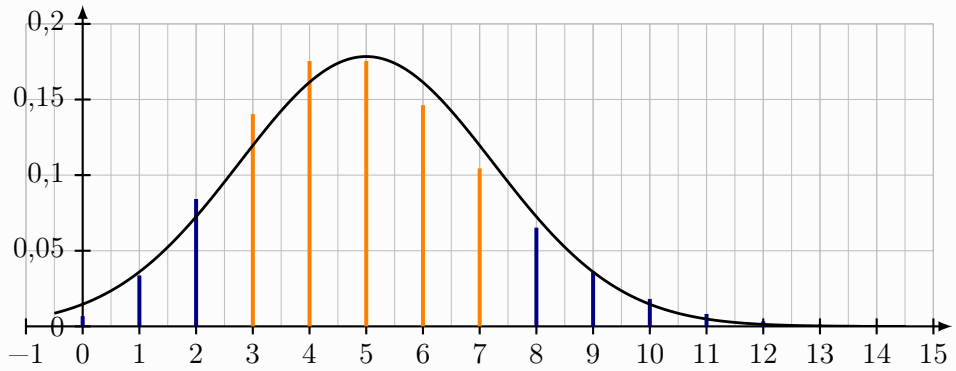
Le premier argument, optionnel et entre `[...]` propose les clés suivantes :

- `Plage` : plage, sous la forme `a-b` du coloriage éventuel ;
- `CouleurPlage` : couleur de la plage éventuelle ;
- `CouleurTraits` : couleur des traits/bâtons (`black` par défaut) ;
- `Style` : style de représentation, `histo` (par défaut) ou `batons` ;
- `ClipX` : restriction de l'axe Ox, sous la forme `a-b` ;
- `Nmax` : borne supérieure de représentation (déterminée automatiquement par défaut).

```
\begin{GraphiqueTikz}%
    [x=0.75cm,y=20cm,Xgrille=1,Ygrille=0.05,Xmin=-1,Xmax=15,Ymin=0,Ymax=0.20]
  \TracerAxesGrilles[Elargir=2.5mm,Dernier]{auto}{auto}
  \TracerLoiPoisson[Style=histo,Plage=3-7,CouleurPlage=cyan,AffNormale]{5}
\end{GraphiqueTikz}
```




```
\begin{GraphiqueTikz}%
    [x=0.75cm,y=20cm,Xgrille=1,Ygrille=0.05,Xmin=-1,Xmax=15,Ymin=0,Ymax=0.20]
  \TracerAxesGrilles[Elargir=2.5mm,Dernier]{auto}{auto}
  \TracerLoiPoisson[CouleurTraits=blue!50!black,Style=batons,
  ↪  Plage=3-7,CouleurPlage=orange,AffNormale]{5}
\end{GraphiqueTikz}
```



5.2.3 Loi géométrique

La loi géométrique modélise le rang du premier succès dans une suite d'épreuves de Bernoulli indépendantes. La borne supérieure est déterminée automatiquement.

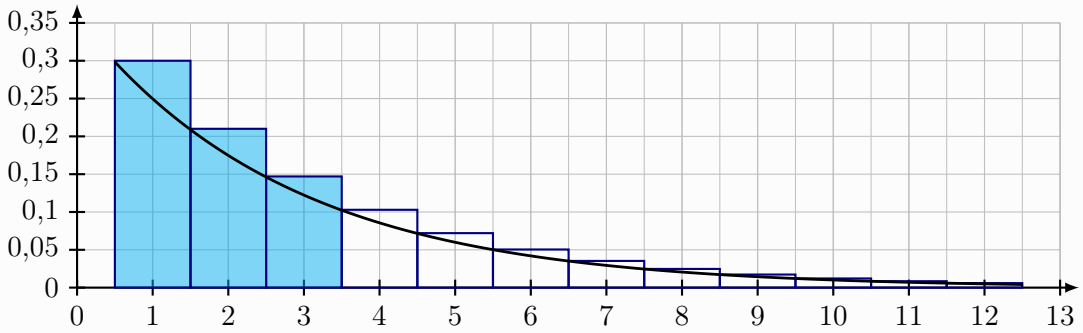
```
%dans l'environnement GraphiqueTikz
\TracerLoiGeo[clés]{p}
```

Le premier argument, optionnel et entre `[...]` propose les clés suivantes :

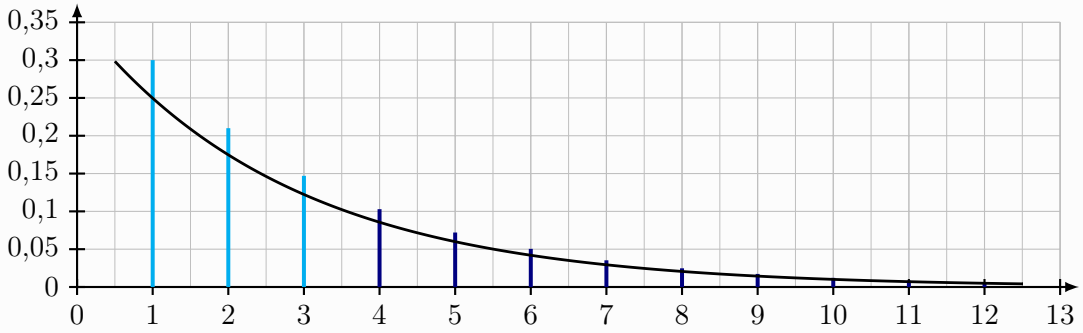
- `Plage` : plage, sous la forme `a-b` du coloriage éventuel ;
- `CouleurPlage` : couleur de la plage éventuelle ;
- `CouleurTraits` : couleur des traits/bâtons (`black` par défaut) ;
- `Style` : style de représentation, `histo` (par défaut) ou `batons` ;
- `AffExpo` : booléen (`false` par défaut) pour superposer la loi exponentielle associée (de paramètre $\lambda = -\ln(1-p)$) ;

— `CouleurExpo` : couleur de la loi exponentielle associée.

```
\begin{GraphiqueTikz}[y=10cm,Xgrille=1,Ygrille=0.05,Xmin=0,Xmax=13,Ymin=0,Ymax=0.35]
  \TracerAxesGrilles[Elargir=2.5mm,Dernier]{auto}{auto}
  \TracerLoiGeo[Style=histo,Plage=*-3,CouleurPlage=cyan,
  ↪  Nmax=12,CouleurTraits=blue!50!black,AffExpo]{0.3}
\end{GraphiqueTikz}
```




```
\begin{GraphiqueTikz}[y=10cm,Xgrille=1,Ygrille=0.05,Xmin=0,Xmax=13,Ymin=0,Ymax=0.35]
  \TracerAxesGrilles[Elargir=2.5mm,Dernier]{auto}{auto}
  \TracerLoiGeo[Style=batons,Plage=*-3,CouleurPlage=cyan,
  ↪  Nmax=12,CouleurTraits=blue!50!black,AffExpo]{0.3}
\end{GraphiqueTikz}
```



5.2.4 Loi hypergéométrique

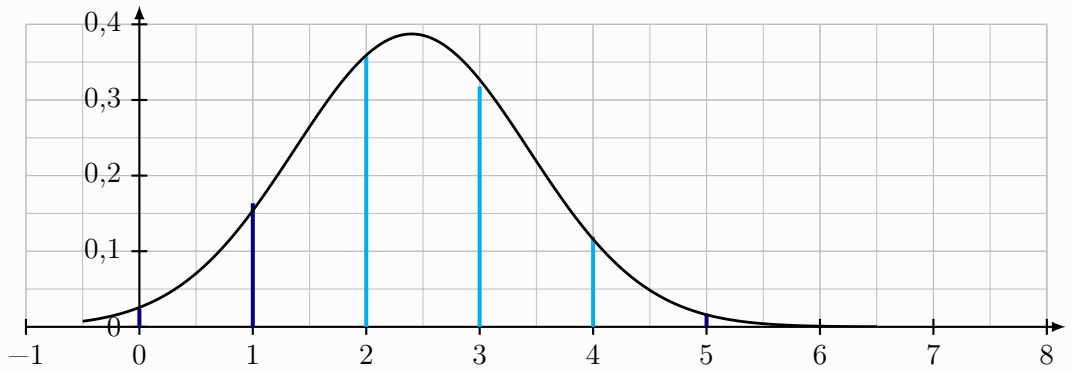
```
%dans l'environnement GraphiqueTikz
%N=population totale  K=succès dans population  n=taille tirage
\TracerLoiHyperGeo[clés]<nom fct normale>{N}{K}{n}
```

Le premier argument, optionnel et entre `[...]` propose les clés suivantes :

— `Plage` : plage, sous la forme `a-b` du coloriage éventuel ;
— `CouleurPlage` : couleur de la plage éventuelle ;
— `CouleurTraits` : couleur des traits/bâtons (`black` par défaut) ;
— `Style` : style de représentation, `histo` (par défaut) ou `batons` ;
— `AffNormale` : booléen (`false` par défaut) pour superposer la loi normale associée ;
— `CouleurNormale` : couleur pour la loi normale.

Les bornes de représentation sont déterminées automatiquement : $k_{\min} = \max(0; n + K - N)$ et $k_{\max} = \min(n; K)$.

```
\begin{GraphiqueTikz}%
    [x=1.5cm,y=10cm,Xgrille=1,Ygrille=0.1,Ygrilles=0.05,
    ↪  Xmin=-1,Xmax=8,Ymin=0,Ymax=0.40]
  \TracerAxesGrilles[Elargir=2.5mm,Dernier]{auto}{auto}
  \TracerLoiHyperGeo%

    ↪  [Style=batons,Plage=2-4,CouleurPlage=cyan,CouleurTraits=blue!50!black,AffNormale]%
   {20}{8}{6}
\end{GraphiqueTikz}
```



6 Commandes spécifiques des méthodes intégrales

6.1 Méthodes géométriques

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

— une méthode des rectangles (Gauche, Droite ou Milieu) ;

— la méthode des trapèzes.

```
%dans l'environnement GraphiqueTikz
\RepresenterMethodeIntegrale[clés]<fonction>{a}{b}
```

Les `Clés` disponibles sont :

— `Spline` : booléen pour préciser qu'un spline est utilisé, `false` par défaut ;

— `Couleur` : couleur des tracés, `red` par défaut ;

— `NbSubDiv` : nombre de subdivisions, `10` par défaut ;

— `Methode` : méthode géométrique utilisée, parmi parmi `RectanglesGauche / RectanglesDroite / RectanglesMilieu / Trapezes` pour spécifier la méthode utilisée, `RectanglesGauche` par défaut ;

— `Remplir` : booléen, `true` par défaut, pour remplir les éléments graphiques ;

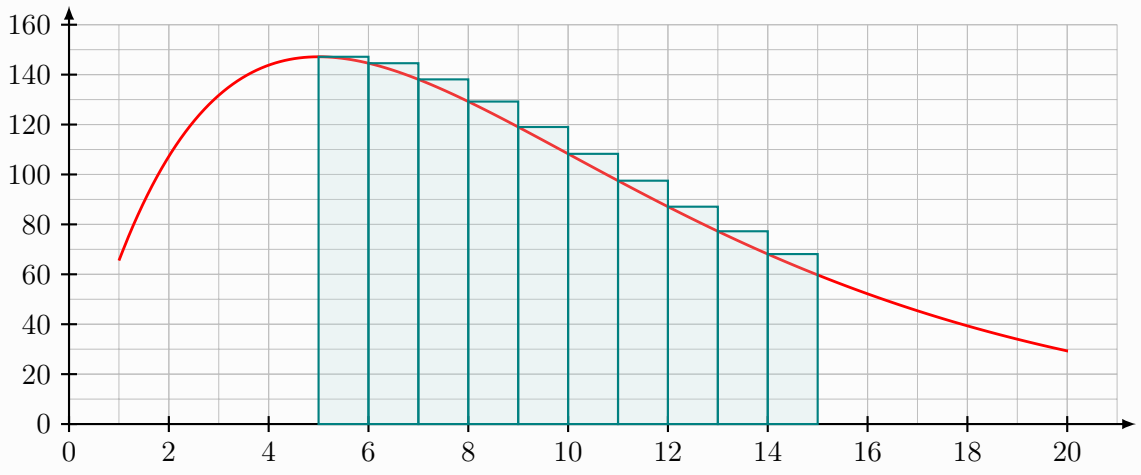
— `CouleurRemplissage` : couleur de remplissage, définie par rapport à la couleur principale par défaut ;

— `Opacite` : opacité, `0.25` par défaut, du remplissage.

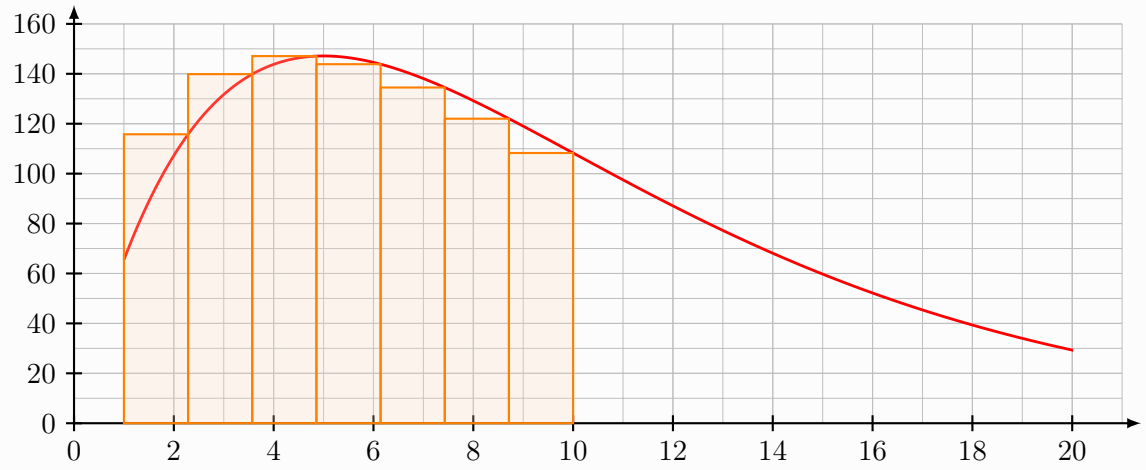
Le deuxième argument, optionnel et entre `<...>`, correspond à la fonction ou le spline **précédemment définie** !

Les deux derniers arguments, obligatoires, correspondent aux bornes de l'intégrale.

```
\begin{GraphiqueTikz}
  [x=0.66cm,y=0.033cm,Xmin=0,Xmax=21,Xgrille=2,Xgrilles=1,
  Ymin=0,Ymax=160,Ygrille=20,Ygrilles=10]
  \TracerAxesGrilles[Elargir=2.5mm]{auto}{auto}
  \DefinirCourbe[Couleur=red,Nom=cf,Debut=1,Fin=20,Trace]<f>{80*x*exp(-0.2*x)}
  \RepresenterMethodeIntegrale[Couleur=teal]<f>{5}{15}
\end{GraphiqueTikz}
```

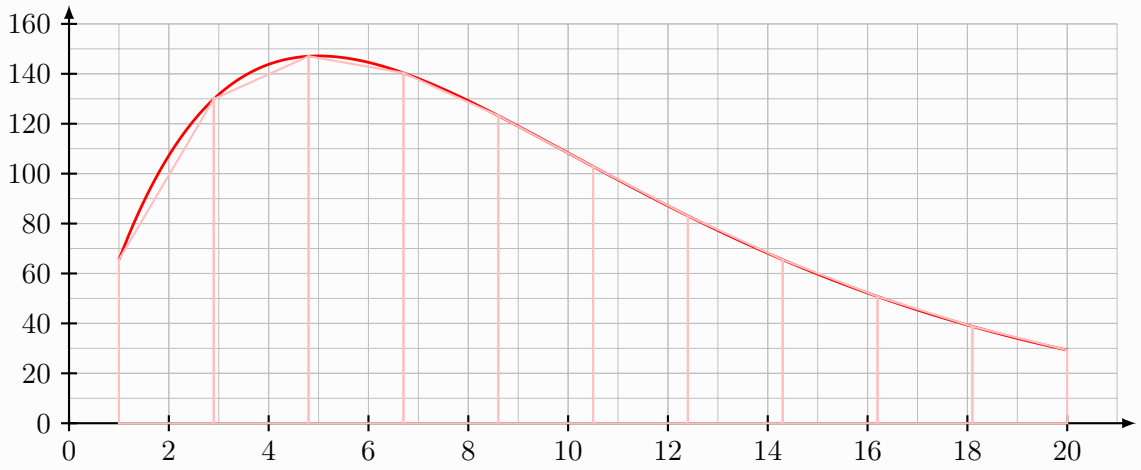


```
\begin{GraphiqueTikz}
  [x=0.66cm,y=0.033cm,Xmin=0,Xmax=21,Xgrille=2,Xgrilles=1,
  Ymin=0,Ymax=160,Ygrille=20,Ygrilles=10]
  \TracerAxesGrilles[Elargir=2.5mm]{auto}{auto}
  \DefinirCourbe[Couleur=red,Nom=cf,Debut=1,Fin=20,Trace]<f>{80*x*exp(-0.2*x)}
  \RepresenterMethodeIntegrale
    [Methode=RectanglesDroite,Couleur=orange,NbSubDiv=7]<f>{1}{10}
\end{GraphiqueTikz}
```

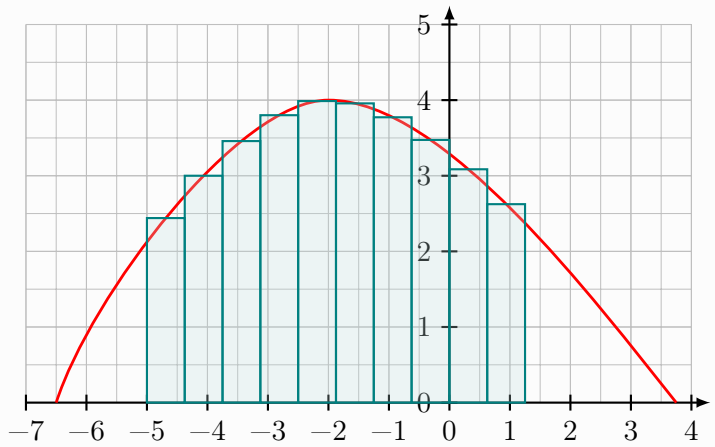



```
\begin{GraphiqueTikz}
  [x=0.66cm,y=0.033cm,Xmin=0,Xmax=21,Xgrille=2,Xgrilles=1,
  Ymin=0,Ymax=160,Ygrille=20,Ygrilles=10]
  \TracerAxesGrilles[Elargir=2.5mm]{auto}{auto}
  \DefinirCourbe[Couleur=red,Nom=cf,Debut=1,Fin=20,Trace]<f>{80*x*exp(-0.2*x)}
  \RepresenterMethodeIntegrale
  [Methode=RectanglesMilieu,Couleur=yellow,NbSubDiv=25]<f>{1}{20}
\end{GraphiqueTikz}
```

```
\begin{GraphiqueTikz}
  [x=0.66cm,y=0.033cm,Xmin=0,Xmax=21,Xgrille=2,Xgrilles=1,
  Ymin=0,Ymax=160,Ygrille=20,Ygrilles=10]
  \TracerAxesGrilles[Elargir=2.5mm]{auto}{auto}
  \DefinirCourbe[Couleur=red,Nom=cf,Debut=1,Fin=20,Trace]<f>{80*x*exp(-0.2*x)}
  \RepresenterMethodeIntegrale
  [Methode=Trapezes,Couleur=pink,Remplir=false]<f>{1}{20}
\end{GraphiqueTikz}
```




```
\begin{GraphiqueTikz}%
  [x=0.8cm,y=1cm,Xmin=-7,Xmax=4,Ymin=0,Ymax=5]
  \TracerAxesGrilles[Elargir=2.5mm]{auto}{auto}
  \DefinirListeSpline{-6.5/0/2.5§-2/4/0§3.75/0/-1}[\lstsplineB]
  \DefinirCourbeSpline[Nom=splinered]{\lstsplineB}
  \TracerCourbeSpline[Couleur=red]{\lstsplineB}
  \RepresenterMethodeIntegrale%
    [Methode=RectanglesMilieu,Spline,Couleur=teal]%
    <splinered>{-5}{1.25}
\end{GraphiqueTikz}
```



6.2 Méthode de Monte-Carlo

L'idée est de proposer une commande pour simuler un calcul intégral via la méthode de Monte-Carlo. Le code se charge de simuler les *tirages*, et les résultats peuvent être stockés dans des macros.

```
%dans l'environnement GraphiqueTikz
\SimulerMonteCarlo[clés]<fonction>{nb essais}[\nbptsmcok][\nbptsmcko]
```

Les `Clés` disponibles sont :

— `Couleurs` : couleurs des points, `blue/red` par défaut ;

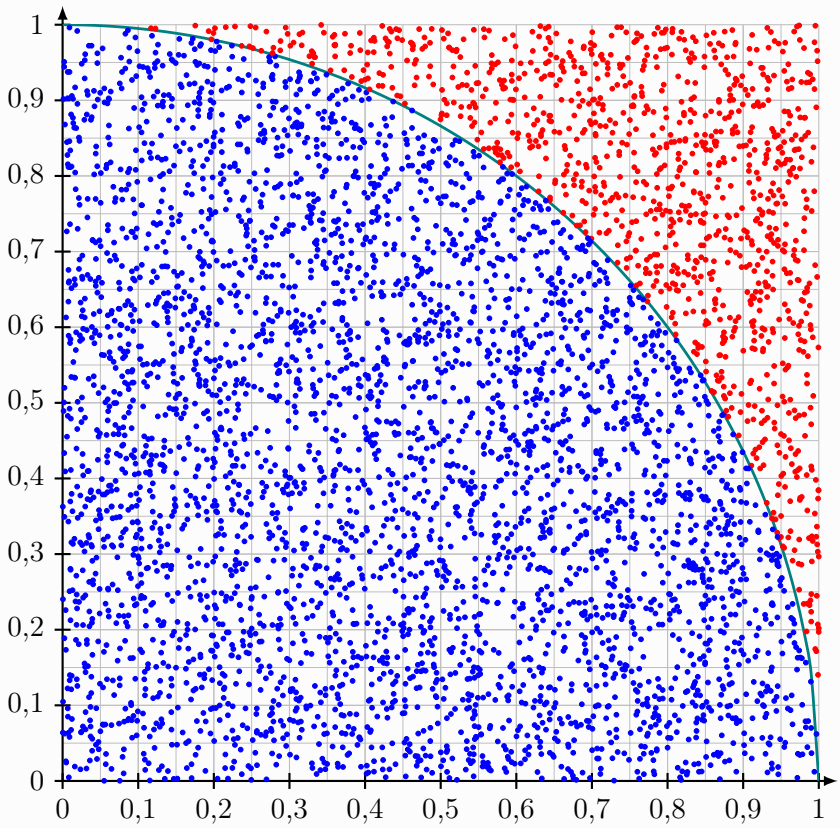
— `BornesX` : bornes *horizontales* pour la simulation, valant `\pflxmin,\pflxmax` par défaut ;

— `BornesY` : bornes *verticales* pour la simulation, valant `\pflymin,\pflymax` par défaut.

Le deuxième argument, optionnel et entre `<...>`, est la fonction **précédemment définie** à utiliser.

Les deux derniers arguments, optionnels et entre `[...]`, sont les macros dans lesquelles sont stockées les résultats de la simulation. Ces macros sont `\nbptsmcok` et `\nbptsmcko` par défaut.
À noter que la macro `\nbptsmc` permet de récupérer le nombre de points utilisés.

```
%avec \sisetup{group-minimum-digits=4} pour le formatage des "milliers"

\begin{GraphiqueTikz}%
  [x=10cm,y=10cm,Xmin=0,Xmax=1,Xgrille=0.1,Xgrilles=0.05,
  Ymin=0,Ymax=1,Ygrille=0.1,Ygrilles=0.05]
  \TracerAxesGrilles[Elargir=2.5mm,Dernier]{auto}{auto}
  \DefinirCourbe[Trace,Couleur=teal,Pas=0.001]<f>{sqrt(1-x^2)}
  \SimulerMonteCarlo<f>{5000}
\end{GraphiqueTikz}

Le nombre de points bleus est de \textcolor{blue}{\num{\nbptsmcok}},
le nombre de points rouges est de \textcolor{red}{\num{\nbptsmcko}}.

La proportion de points bleus est de $\frac{\num{\nbptsmcok}}{\num{\nbptsmc}}
\approx \ArrondirNum[4]{\nbptsmcok/\nbptsmc}$
et $\frac{\pi}{4} \approx \ArrondirNum[4]{pi/4}$.
```



Le nombre de points bleus est de 3 921, le nombre de points rouges est de 1 079.
La proportion de points bleus est de $\frac{3\,921}{5\,000} \approx 0{,}784\,2$ et $\frac{\pi}{4} \approx 0{,}785\,4$.

7 Commandes spécifiques des statistiques

7.1 Limitations

Compte-tenu des spécificités de Ti*k*Z, il est conseillé de ne pas utiliser de valeurs trop *grandes* au niveau de axes (cela peut coincer avec des année par exemple...), ou bien il faudra *transformer* les valeurs des axes et/ou des données pour que tout s'affiche comme il faut (attention également aux régressions, aux calculs...).

7.2 Courbe des ECC/FCC (1 variable)

Il est possible de travailler sur une représentation de la courbe des ECC/FCC.

```
\TracerCourbeECC[clés]{liste valeurs}{liste effectifs}
```

Le code se charge de déterminer une valeur des paramètres, pour utilisation ultérieure (avec arrondis éventuels car ils sont obtenus par *conversions*) :

— le premier quartile, Q_1, est stocké dans la macro `\ValPremQuartile` ;

— la médiane, méd, est stocké dans la macro `\ValMed` ;

— le troisième quartile, Q_3, est stocké dans la macro `\ValTroisQuartile`.

Les `Clés` disponibles sont :

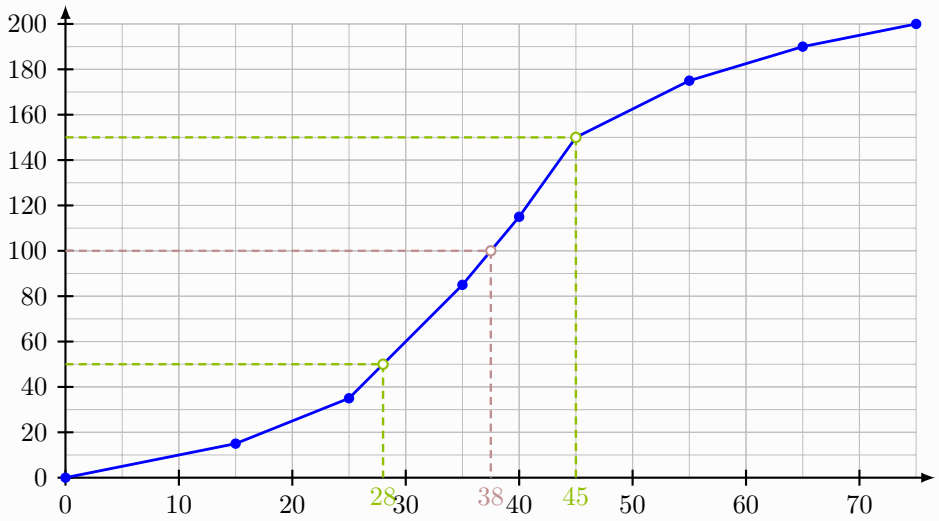
— `Couleur=...` : couleur du tracé, `black` par défaut ;

— `AffParams` : booléen, `true` par défaut, pour afficher les paramètres ;

— `CouleursParams=...` : couleur des paramètres, `black` par défaut ;

— `TraitsComplets` : booléen, `true` par défaut, pour afficher les pointillés en entier

```
\begin{GraphiqueTikz}[x=0.15cm,y=0.03cm,Xmin=0,Xmax=75,Xgrille=10,Xgrilles=5,
  Ymin=0,Ymax=200,Ygrille=20,Ygrilles=10]
  \TracerAxesGrilles[Elargir=2.5mm,Police=\small]{auto}{auto}
  \TracerCourbeECC%
    [Couleur=blue,CouleursParams={lime!75!black/pink!75!black},
    TraitsComplets=false]%
    {0,15,25,35,40,45,55,65,75}%
    {15,20,50,30,35,25,15,10}
  %ajouts 'manuels'
  \PlacerTexte[Couleur=lime!75!black,Police=\small,Position=below]%
    {(\ValPremQuartile,0)}{\ArrondirNum[0]{\ValPremQuartile}}
  \PlacerTexte[Couleur=lime!75!black,Police=\small,Position=below]%
    {(\ValTroisQuartile,0)}{\ArrondirNum[0]{\ValTroisQuartile}}
  \PlacerTexte[Couleur=pink!75!black,Police=\small,Position=below]%
    {(\ValMed,0)}{\ArrondirNum[0]{\ValMed}}
\end{GraphiqueTikz}
```



7.3 Le nuage de points (2 variables)

En marge des commandes liées aux fonctions, il est également possible de représenter des séries statistiques doubles.

Le paragraphe suivant montre que l'ajout d'une clé permet de rajouter la droite d'ajustement linéaire.

```
%dans l'environnement GraphiqueTikz
\TracerNuage[clés]{ListeX}{ListeY}
```

La `[clé]` optionnelle est :

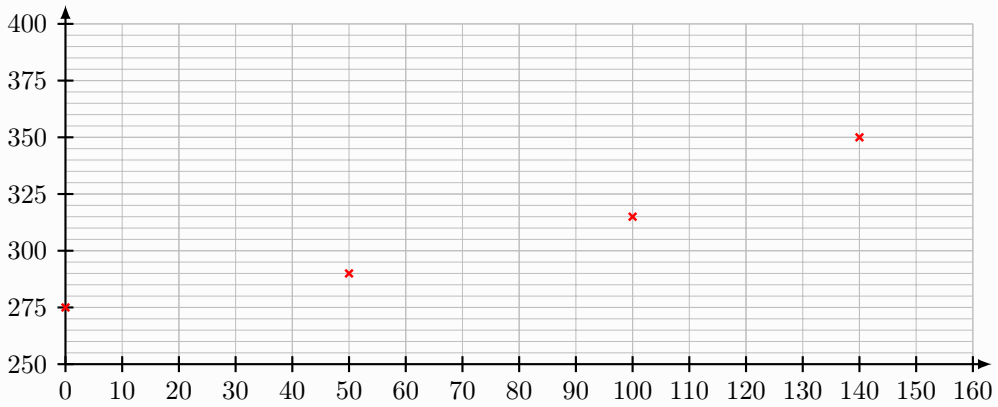
— `CouleurNuage` : couleur des points du nuage (`black` par défaut).

Les arguments, obligatoires, permettent de spécifier :

— la liste des abscisses ;

— la liste des ordonnées.

```
\begin{GraphiqueTikz}%
  [x=0.075cm,y=0.03cm,Xmin=0,Xmax=160,Xgrille=20,Xgrilles=10,
  Origy=250,Ymin=250,Ymax=400,Ygrille=25,Ygrilles=5]
  %préparation de la fenêtre
  \TracerAxesGrilles[Elargir=2.5mm,Police=\small]{0,10,...,160}{250,275,...,400}
  %nuage de points
  \TracerNuage[Style=x,CouleurNuage=red]{0,50,100,140}{275,290,315,350}
\end{GraphiqueTikz}
```



7.4 La droite de régression (2 variables)

La droite de régression linéaire (obtenue par la méthode des moindres carrés) peut facilement être rajoutée, en utilisant la clé `TracerDroite`.

Dans ce cas, de nouvelles clés sont disponibles :

— `CouleurDroite` : couleur de la droite (`black` par défaut) ;

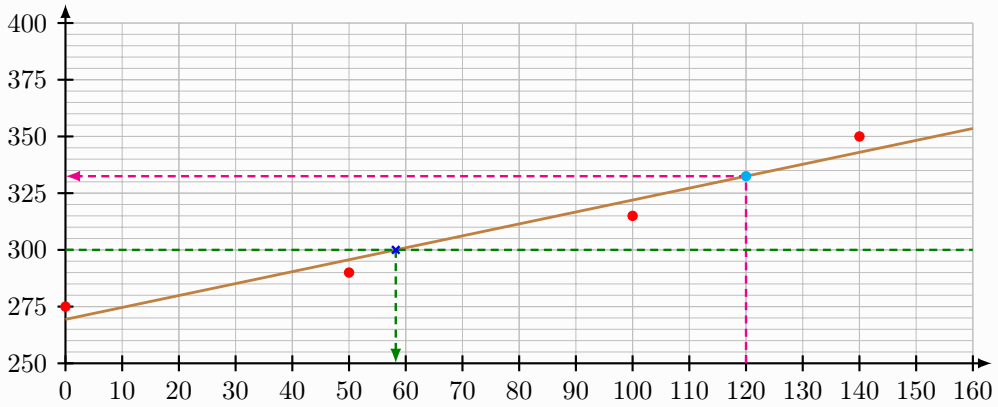
— `Arrondis` : précision des coefficients (`vide` par défaut) ;

— `Debut` : abscisse initiale du tracé (`\pflxmin` par défaut) ;

— `Fin` : abscisse terminale du tracé (`\pflxmax` par défaut) ;

— `Nom` : nom du tracé, pour exploitation ultérieure (`reglin` par défaut).

```
\begin{GraphiqueTikz}%
  [x=0.075cm,y=0.03cm,Xmin=0,Xmax=160,Xgrille=20,Xgrilles=10,
  Origy=250,Ymin=250,Ymax=400,Ygrille=25,Ygrilles=5]
  \TracerAxesGrilles[Elargir=2.5mm,Police=\small]{0,10,...,160}{250,275,...,400}
  %nuage et droite
  \TracerNuage%
    [CouleurNuage=red,CouleurDroite=brown,TracerDroite]%
    {0,50,100,140}{275,290,315,350}
  %image
  \PlacerImages[Couleurs=cyan/magenta,Traits]{d}{120}
  %antécédents
  \PlacerAntecedents[Style=x,Couleurs=blue/green!50!black,Traits]{reglin}{300}
\end{GraphiqueTikz}
```



7.5 Autres régressions (2 variables)

En partenariat avec le package `xint-regression`, chargé par le package (mais *désactivable* via l'option `[nonxintreg]`), il est possible de travailler sur d'autres types de régression :

- linéaire $\boxed{ax+b}$;
- quadratique $\boxed{ax^2+bx+c}$;
- cubique $\boxed{ax^3+bx^2+cx+d}$;
- puissance $\boxed{ax^b}$;
- exponentielle $\boxed{ab^x}$ ou $\boxed{e^{ax+b}}$ ou $\boxed{be^{ax}}$ ou $\boxed{C+be^{ax}}$;
- logarithmique $\boxed{a+b\ln(x)}$;
- hyperbolique $\boxed{a+\dfrac{b}{x}}$.

La commande, similaire à celle de définition d'une courbe, est :

```
\TracerAjustement[clés]<non fct>{type}<arrondis>{listex}{listey}
```

Les `[clés]` disponibles sont, de manière classique :

- `Debut` : borne inférieure de l'ensemble de définition (`\pflxmin` par défaut) ;
- `Fin` : borne inférieure de l'ensemble de définition (`\pflxmax` par défaut) ;
- `Nom` : nom de la courbe (important pour la suite !) ;
- `Couleur` : couleur du tracé (`black` par défaut) ;
- `Pas` : pas du tracé (il est déterminé *automatiquement* au départ mais peut être modifié).

Le deuxième argument, optionnel et entre `<...>` permet de nommer la fonction de régression.
Le troisième argument, obligatoire et entre `{...}` permet de choisir le type de régression, parmi :

— `lin` : linéaire $\boxed{ax+b}$;

— `quad` : quadratique $\boxed{ax^2+bx+c}$;

— `cub` : cubique $\boxed{ax^3+bx^2+cx+d}$;

— `pow` : puissance $\boxed{ax^b}$;

— `expab` : exponentielle $\boxed{ab^x}$

— `hyp` : hyperbolique $\boxed{a+\dfrac{b}{x}}$;

— `log` : logarithmique $\boxed{a+b\ln(x)}$;

— `exp` : exponentielle $\boxed{e^{ax+b}}$;

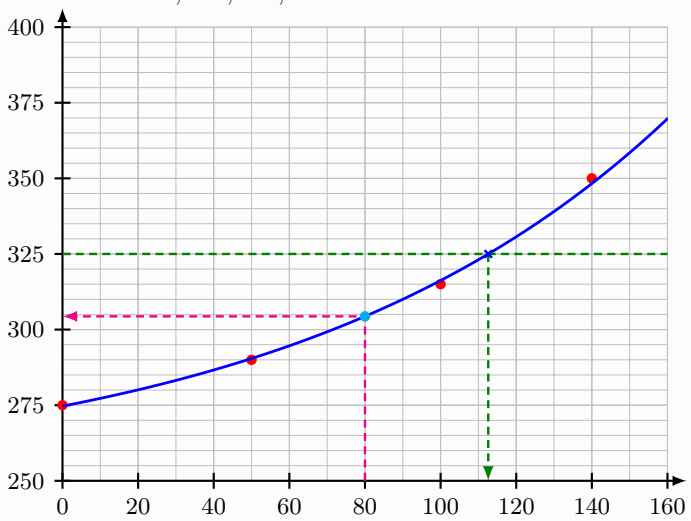
— `expalt` : exponentielle $\boxed{be^{ax}}$;

— `expoff=C` : exponentielle $\boxed{C+be^{ax}}$.

Le quatrième argument, optionnel et entre `<...>` permet de spécifier le ou les arrondis pour les coefficients de la fonction de régression.
Les deux derniers arguments sont les listes des valeurs de X et de Y.

```
\def\LISTEXX{0,50,100,140}\def\LISTEYY{275,290,315,350}%
ListeX := \LISTEXX\\
ListeY := \LISTEYY

\begin{GraphiqueTikz}
  [x=0.05cm,y=0.04cm,Xmin=0,Xmax=160,Xgrille=20,Xgrilles=10,
  Origy=250,Ymin=250,Ymax=400,Ygrille=25,Ygrilles=5]
  %préparation de la fenêtre
  \TracerAxesGrilles[Elargir=2.5mm,Police=\footnotesize]{auto}{auto}
  %nuage de points
  \TracerNuage[Style=o,CouleurNuage=red]{\LISTEXX}{\LISTEYY}
  %ajustement expoffset
  \TracerAjustement[Couleur=blue,Nom=ajust]<ajust>{expoff=250}{\LISTEXX}{\LISTEYY}
  %exploitations
  \PlacerImages[Couleurs=cyan/magenta,Traits]{ajust}{80}
  \PlacerAntecedents[Style=x,Couleurs=blue/green!50!black,Traits]{ajust}{325}
\end{GraphiqueTikz}

\xintexpoffreg[offset=250,round=3/1]{\LISTEXX}{\LISTEYY}%
On obtient $y=250+\num{\expregoffb}\text{e}^{\num{\expregoffa}x}$
```

ListeX := 0,50,100,140
ListeY := 275,290,315,350



On obtient $y = 250 + 24{,}7\text{e}^{0{,}01x}$

7.6 Diagramme en bâtons (1 variable)

L'idée est de représenter une série statistique à l'aide d'un diagramme en bâtons ou de rectangles, dans l'environnement `GraphiqueTikz`.

```
%dans l'environnement GraphiqueTikz
\TracerDiagrammeBatons[clés]{série}
```

La `série` est spécifiée sous la forme `x1/y1,x2/y2,...` (valeur/effectif).

Les `[clés]` disponibles sont :

— `Couleurs=...` : liste de couleurs (barre par barre), `black` par défaut ;

— `Style=...` : style de représentation, `batons` (trait vertical) ou `histo` (rectangle) ;

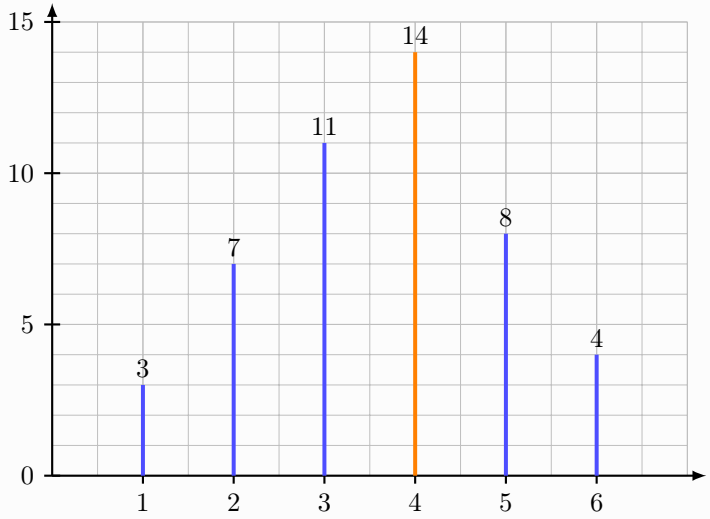
— `Largeur=...` : largeur des rectangles en mode `histo`, `0.8` par défaut ;

— `AffValeurs` : booléen, `false` par défaut, pour afficher les effectifs au-dessus des barres ;

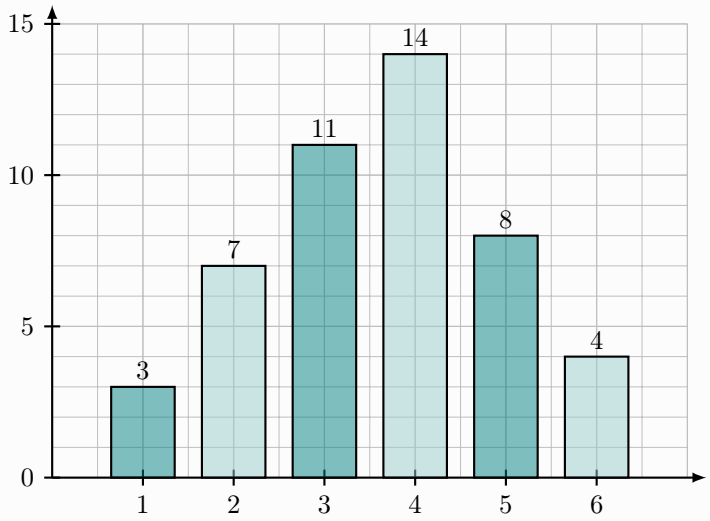
— `Arrondi=...` : nombre de décimales pour l'affichage des valeurs, `0` par défaut ;

— `CouleurValeurs=...` : couleur des labels de valeurs, `black` par défaut.

```
\begin{GraphiqueTikz}%
  [x=1.2cm,y=0.4cm,Xmin=0,Xmax=7,Xgrille=1,Ymin=0,Ymax=15,Ygrille=5,Ygrilles=1]
  \TracerAxesGrilles[Elargir=2.5mm,Police=\small]{1,2,...,6}{0,5,...,15}
  %bâtons, couleur par barre (la mode en orange)
  \TracerDiagrammeBatons%
    [Couleurs={blue!70,blue!70,blue!70,orange,blue!70,blue!70},AffValeurs]%
    {1/3,2/7,3/11,4/14,5/8,6/4}
\end{GraphiqueTikz}
```



```
\begin{GraphiqueTikz}%
  [x=1.2cm,y=0.4cm,Xmin=0,Xmax=7,Xgrille=1,Ymin=0,Ymax=15,Ygrille=5,Ygrilles=1]
  \TracerAxesGrilles[Elargir=2.5mm,Police=\small]{1,2,...,6}{0,5,...,15}
  %style histo, deux couleurs alternées
  \TracerDiagrammeBatons%
    [Style=histo,Largeur=0.7,Couleurs={teal,teal!40},AffValeurs]%
    {1/3,2/7,3/11,4/14,5/8,6/4}
\end{GraphiqueTikz}
```



7.7 Histogramme (1 variable)

L'idée est de représenter un histogramme à classes régulières ou non, dans l'environnement `GraphiqueTikz`. En mode densité (classes non-régulières), la hauteur de chaque barre correspond à effectif/amplitude.

```
%dans l'environnement GraphiqueTikz
\TracerHistogramme[clés]{données}
```

Les `données` sont spécifiées sous la forme `bi1/bs1/eff1,bi2/bs2/eff2,...` (borne inférieure/borne supérieure/effectif).

Les `[clés]` disponibles sont :

— `Couleurs=...` : liste de couleurs cycliques pour les fonds, `{black}` par défaut ;

— `CouleurBordure=...` : couleur unique des bordures, `black` par défaut ;

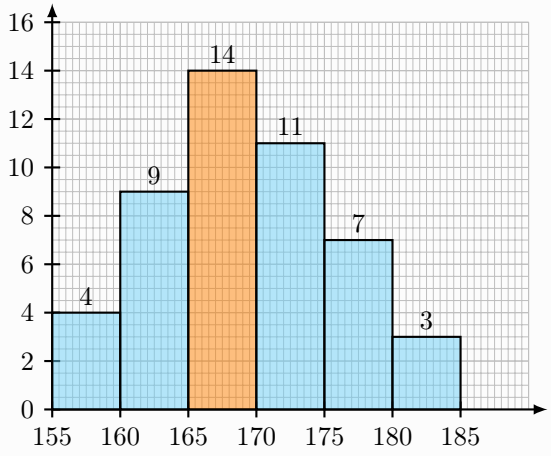
— `Densite` : booléen, `false` par défaut, pour afficher les densités (effectif/amplitude) en ordonnée ;

— `AffValeurs` : booléen, `false` par défaut, pour afficher les effectifs bruts au-dessus des barres ;

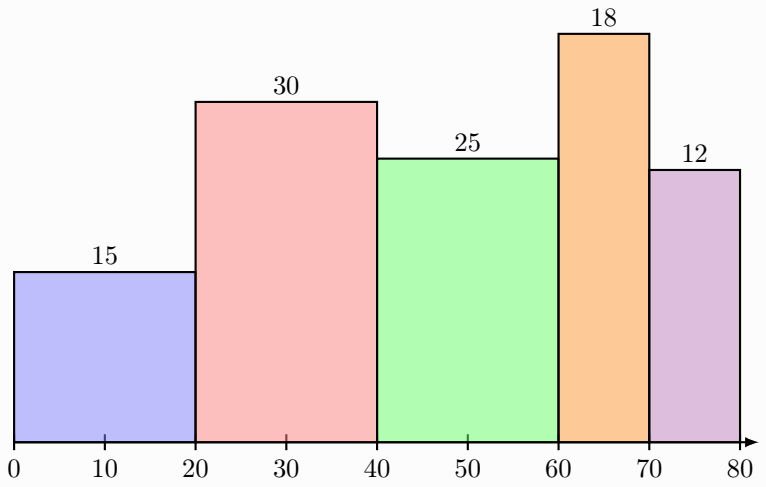
— `PosValeurs=...` : position du label (`above` par défaut) ;

— `CouleurValeurs=...` : couleur des labels, `black` par défaut.

```
%classes régulières - ordonnée = effectif, classe modale en orange
\begin{GraphiqueTikz}%
  [x=0.18cm,y=0.32cm,Xmin=155,Xmax=190,Origx=155,Xgrille=5,Ymin=0,Ymax=16,Ygrille=2]
  \TracerAxesGrilles[Elargir=2.5mm,Police=\small]{155,160,...,185}{0,2,...,16}
  \TracerHistogramme%
    [Couleurs={cyan!50,cyan!50,orange,cyan!50,cyan!50,cyan!50},AffValeurs]%
    {155/160/4,160/165/9,165/170/14,170/175/11,175/180/7,180/185/3}
\end{GraphiqueTikz}
```



```
%classes non-régulières, mode densité - axe Oy masqué
\begin{GraphiqueTikz}%
  [x=0.12cm,y=3cm,Xmin=0,Xmax=80,Xgrille=10,Ymin=0,Ymax=2,Ygrille=0.5,Ygrilles=0.25]
  \TracerAxesGrilles%
    [Elargir=2.5mm,Police=\small,AxeOy=false,Grille=false]{0,10,...,80}{}
  \TracerHistogramme%
    [Densite,Couleurs={blue!50,red!50,green!60,orange!80,violet!50},AffValeurs]%
    {0/20/15,20/40/30,40/60/25,60/70/18,70/80/12}
\end{GraphiqueTikz}
```



La clé `AxeOy=false` dans `\TracerAxesGrilles` permet de supprimer l'axe des ordonnées, ses tirets et ses labels — utile pour les histogrammes en densité où cet axe n'a pas de sens direct.

7.8 Boîte à moustaches

L'idée est de représenter une boîte à moustaches dans l'environnement `GraphiqueTikz`, soit à partir de valeurs explicites, soit directement depuis une série de données.

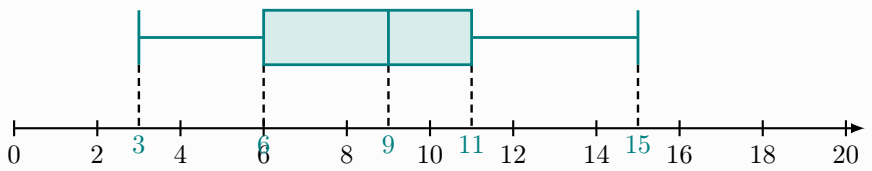
```
%dans l'environnement GraphiqueTikz
%version normale : paramètres saisis explicitement
\TracerBoiteMoustaches[clés]{min/Q1/mediane/Q3/max}
%version étoilée : calcul automatique depuis une liste de données
\TracerBoiteMoustaches*[clés]{liste}
```

En version étoilée, la `liste` est une liste CSV de valeurs individuelles. La clé `Regroup` permet de passer une liste de type `valeur/effectif,...`.

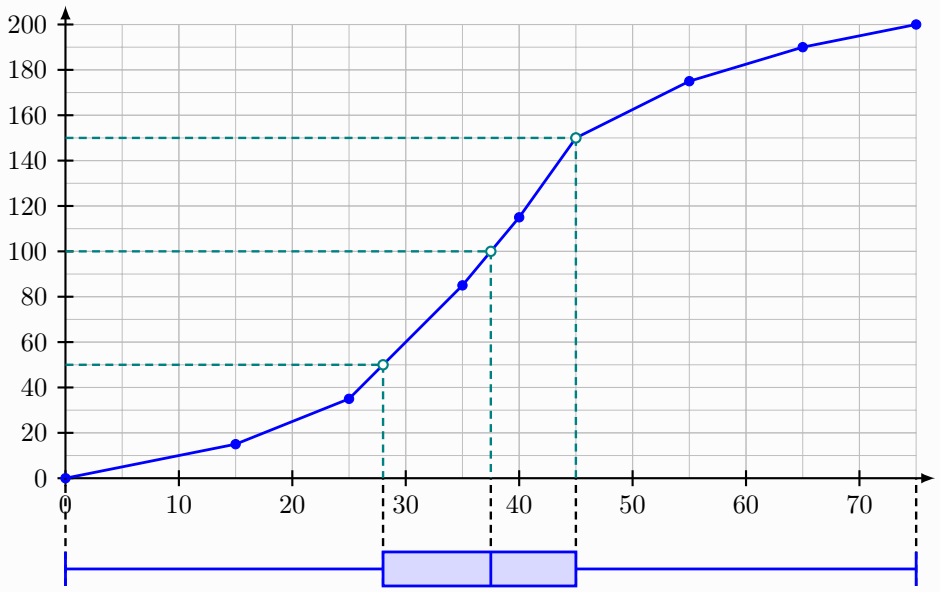
Les `[clés]` disponibles sont :

- `Couleur=...` : couleur des traits, `black` par défaut ;
- `Remplissage=...` : couleur du fond de la boîte, `white` par défaut ;
- `Elevation=...` : ordonnée du centre de la boîte, `0` par défaut ;
- `Hauteur=...` : hauteur de la boîte (en unités y), `1` par défaut ;
- `AffMoyenne` : booléen, `false` par défaut, pour afficher la moyenne (point plein) ;
- `Moyenne=...` : valeur de la moyenne (utilisée avec `AffMoyenne`) ;
- `Pointilles` : booléen, `false` par défaut, pour afficher des pointillés vers l'axe Ox ;
- `Valeurs` : booléen, `false` par défaut, pour afficher les 5 paramètres sur l'axe Ox ;
- `Arrondi=...` : décimales pour l'affichage des valeurs, `0` par défaut ;
- `Regroup` : booléen, `false` par défaut, pour une liste `valeur/effectif` (version étoilée uniquement).

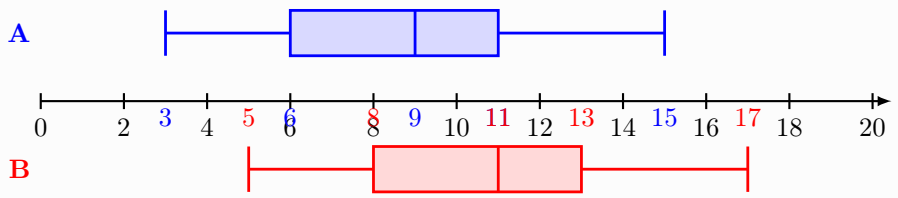
```
%version étoilée, liste brute
\begin{GraphiqueTikz}%
  [x=0.55cm,y=1.2cm,Xmin=0,Xmax=20,Xgrille=2,Ymin=0,Ymax=1,Ygrille=1]
  \TracerAxesGrilles%
    [Elargir=2.5mm,Police=\small,AxeOy=false,Grille=false]{0,2,...,20}{}
  \TracerBoiteMoustaches*%
    [Hauteur=0.6,Elevation=1,Couleur=teal,Remplissage=teal!15,Pointilles,Valeurs]%
    {4,7,11,3,15,9,12,6,14,8,5,10,13,7,9,11,6,12,8,10}
\end{GraphiqueTikz}
```



```
%combinaison ECC + boîte à moustaches (version normale)
%\TracerCourbeECC stocke \ValPremQuartile, \ValMed, \ValTroisQuartile
\begin{GraphiqueTikz}%
  [x=0.15cm,y=0.03cm,Xmin=0,Xmax=75,Xgrille=10,Xgrilles=5,
  Ymin=0,Ymax=200,Ygrille=20,Ygrilles=10]
  \TracerAxesGrilles%
    [Elargir=2.5mm,Police=\small]{0,10,...,70}{0,20,...,200}
  \TracerCourbeECC%
    [Couleur=blue,CouleursParams={teal/teal},TraitsComplets=false]%
    {0,15,25,35,40,45,55,65,75}{15,20,50,30,35,25,15,10}
  %boîte sous l'axe Ox (Ymin négatif requis)
  \TracerBoiteMoustaches%
    [Elevation=-40,Hauteur=15,Couleur=blue,Remplissage=blue!15,Pointilles]%
    {0/\ValPremQuartile/\ValMed/\ValTroisQuartile/75}
\end{GraphiqueTikz}
```



```
%comparaison de deux séries (deux boîtes superposées)
\begin{GraphiqueTikz}%
  [x=0.55cm,y=1.2cm,Xmin=0,Xmax=20,Xgrille=2,Ymin=-2,Ymax=2,Ygrille=1]
  \TracerAxesGrilles%
    [Elargir=2.5mm,Police=\small,AxeOy=false,Grille=false]{0,2,...,20}{}
  % Série A (en haut)
  \TracerBoiteMoustaches*%
    [Elevation=0.75,Hauteur=0.5,Couleur=blue,Remplissage=blue!15,Valeurs]%
    {4,7,11,3,15,9,12,6,14,8,5,10,13,7,9,11,6,12,8,10}
  % Série B (en bas)
  \TracerBoiteMoustaches*%
    [Elevation=-0.75,Hauteur=0.5,Couleur=red,Remplissage=red!15,Valeurs]%
    {6,9,13,5,17,11,14,8,16,10,7,12,15,9,11,13,8,14,10,12}
  % Labels des séries
  \PlacerTexte[Couleur=blue,Police=\small\bfseries,Position=left]{(0,0.75)}{A}
  \PlacerTexte[Couleur=red, Police=\small\bfseries,Position=left]{(0,-0.75)}{B}
\end{GraphiqueTikz}
```



8 Courbes paramétriques, courbes polaires

8.1 Courbes paramétriques

Il est possible de tracer des courbes définies paramétriquement par $(x(t);y(t))$.

```
%définition des focntions paramétriques
\DefinirCourbeParam[clés]<nom x(t)><nom y(t)>{x(t)}{y(t)}

%tracé des fonctions paramétriques
\TracerCourbeParam[clés]{x(t)}{y(t)}
```

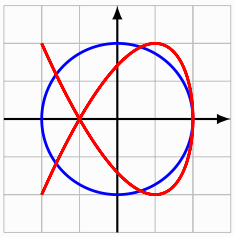
Les clés disponibles sont les mêmes que pour `\DefinirCourbe` ou `\TracerCourbe`, avec quelques spécificités :

- `Debut` : valeur initiale du paramètre t (**obligatoire**) ;
- `Fin` : valeur finale du paramètre t (**obligatoire**) ;
- `Nom` : nom éventuel du chemin (pour réutilisation) ;
- `Couleur` : couleur du tracé (`black` par défaut) ;
- `Pas` : pas en t ;
- `Trace` : booléen pour tracer la courbe (`false` par défaut) ;
- `Restreindre` : booléen pour éviter les débordements (`false` par défaut).

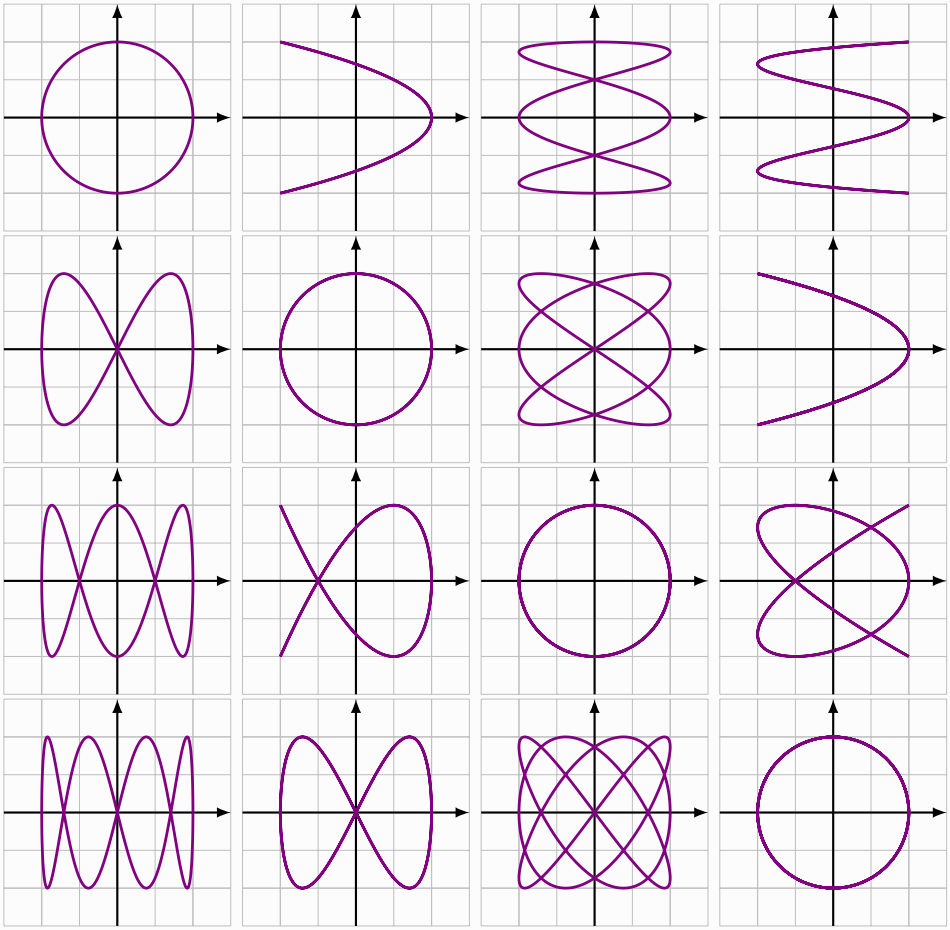
💣 Quelques remarques importantes :

- les noms `<x>` et `<y>` par défaut sont écrasés à chaque appel — penser à les nommer différemment si plusieurs courbes paramétriques coexistent ;
- le pas automatique peut être insuffisant pour des courbes très sinueuses — réduire `Pas` si nécessaire ;
- `Fin=6.28` (ou `2*pi`) pour les courbes fermées.

```
\begin{GraphiqueTikz}[Xmin=-1.5,Xmax=1.5,Ymin=-1.5,Ymax=1.5]
  \TracerAxesGrilles{}{}
  % Cercle
  \TracerCourbeParam[Couleur=blue,Debut=0,Fin=6.28]{cos(t)}{sin(t)}
  % Lissajous
  \TracerCourbeParam[Couleur=red,Debut=0,Fin=6.28]{cos(2*t)}{sin(3*t)}
\end{GraphiqueTikz}
```



```
% Planche de courbes de Lissajous
\foreach \i in {1,...,4}{%
  \noindent%
  \foreach \j in {1,...,4}{%
    \begin{GraphiqueTikz}[Xmin=-1.5,Xmax=1.5,Ymin=-1.5,Ymax=1.5]
      \TracerAxesGrilles{}{}
      \TracerCourbeParam%
      [Couleur=violet,Debut=0,Fin=6.28]%
      {cos(\j*t)}{sin(\i*t)}
    \end{GraphiqueTikz}
  }%
  \par
}
```



8.2 Courbes polaires

Il est également possible de travailler sur des courbes polaires, au quel cas des clés et macros complémentaires sont disponibles.

Pour la création de l'environnement (et la grille notamment), sont disponibles :

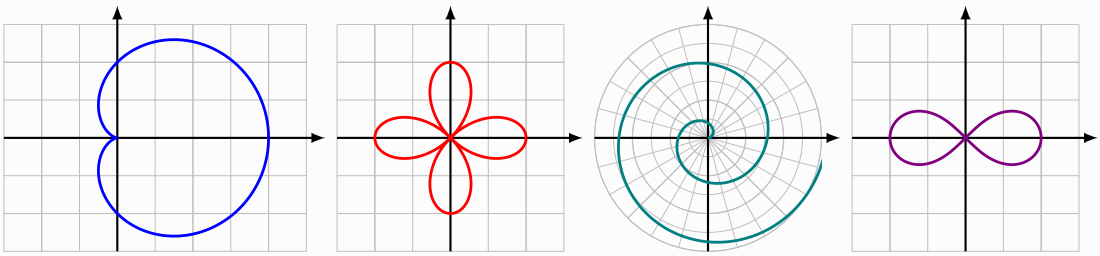
— `Rmax` ;

— `Rgrille` et `Rgrilles` ;

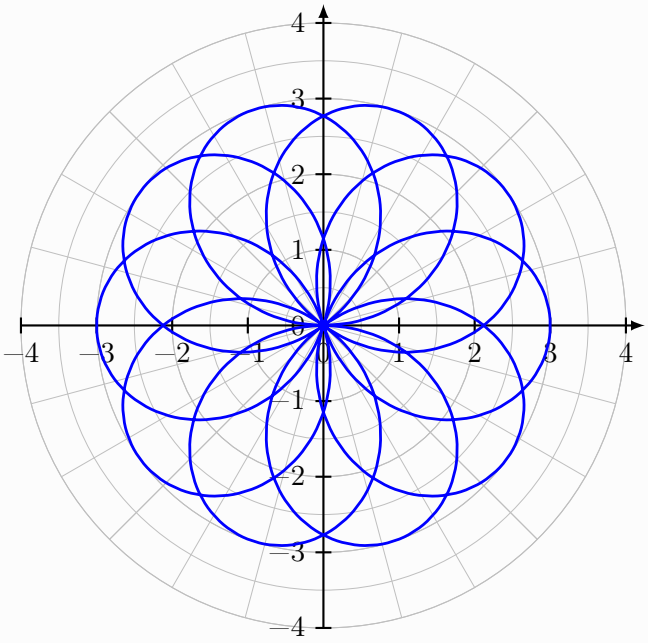
— `Tgrille` et `Tgrilles` (grille radiale) ;

```
%tracés des axes et des grilles radiales
\TracerAxesGrillePol[clés]{grad r}{grad theta}
```

```
% Cardioïde  r = 1 + cos(theta)
\begin{GraphiqueTikz}[Xmin=-1.5,Xmax=2.5,Ymin=-1.5,Ymax=1.5]
  \TracerAxesGrilles[Elargir=2.5mm]{}{} %grille cartésienne
  \TracerCourbePol[Couleur=blue,Debut=0,Fin=6.28]{1+cos(theta)}
\end{GraphiqueTikz}
% Rose à 4 pétales  r = cos(2theta)
\begin{GraphiqueTikz}[Xmin=-1.5,Xmax=1.5,Ymin=-1.5,Ymax=1.5]
  \TracerAxesGrilles[Elargir=2.5mm]{}{} %grille cartésienne
  \TracerCourbePol[Couleur=red,Debut=0,Fin=6.28]{cos(2*theta)}
\end{GraphiqueTikz}
% Spirale d'Archimède  r = theta/4
\begin{GraphiqueTikz}%
    [x=0.5cm,y=0.5cm,Xmin=-3,Xmax=3,Ymin=-3,Ymax=3,Rmax=3]
  \TracerAxesGrillePol[Elargir=2.5mm]{}{} %grille radiale
  \TracerCourbePol[Couleur=teal,Debut=0,Fin=12.56]{theta/4}
\end{GraphiqueTikz}
% Lemniscate de Bernoulli  r² = cos(2theta)
\begin{GraphiqueTikz}[Xmin=-1.5,Xmax=1.5,Ymin=-1.5,Ymax=1.5]
  \TracerAxesGrilles[Elargir=2.5mm]{}{} %grille cartésienne
  \TracerCourbePol[Couleur=violet,Debut=0,Fin=6.28]%
    {sqrt(max(cos(2*theta),0))}
\end{GraphiqueTikz}
```



```
\begin{GraphiqueTikz}[Xmin=-4,Xmax=4,Ymin=-4,Ymax=4,Rmax=4,]
  \TracerAxesGrillePol[Elargir=2.5mm]{auto}{auto}
  \TracerCourbePol[Couleur=blue,Debut=0,Fin=8*pi]{3sin(5theta/4)}
\end{GraphiqueTikz}
```



9 Commandes complémentaires (expérimentales)

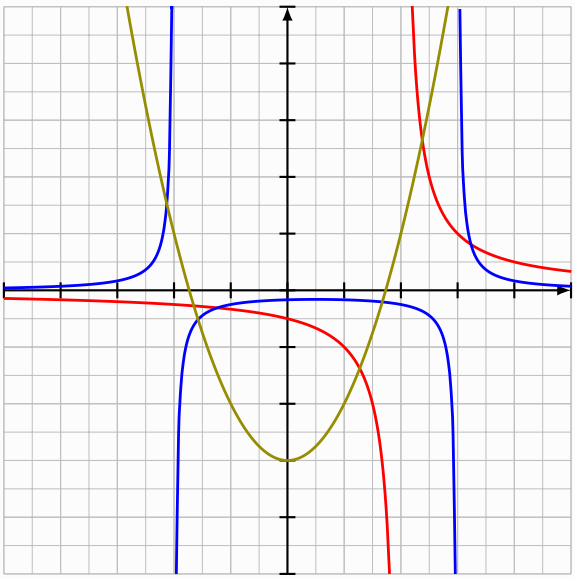
9.1 Valeurs interdites

De manière expérimentale, il est possible de travailler avec des fonctions possédant des valeurs interdites.
Dans ce cas, l'intégralité de la courbe est tracée/stockée, avec une gestion en *interne* des valeurs interdites (le *pas* de calcul est alors réduit pour la gestion des asymptotes éventuelles, et un contrôle des valeurs en y est effectué en amont !).

💣 Logiquement les commandes d'exploitation sont compatibles avec cette nouvelle *méthode*, mais il n'est pas impossible de que des dysfonctionnements existent, auquel cas une déclaration des courbes/fonctions par *branches* sera nécessaire...

Une nouvelle clé, `[ValeursInterdites=...]`, fait donc son apparition pour la définition ou le tracé d'une courbe, ainsi que la clé `[DeltaVI=...]` qui permet de paramétrer le *delta* autour des asymptotes (`auto` par défaut).

```
\begin{GraphiqueTikz}[x=0.75cm,y=0.75cm,Xmin=-5,Xmax=5,Ymin=-5,Ymax=5]
  \TracerAxesGrilles[Grads=false]{auto}{auto}
  % 1/(x-2) avec VI en x=2
  \TracerCourbe[Couleur=red, ValeursInterdites={2}]{1/(x-2)}
  % 1/((x+2)(x-3)) avec VI en x=-2 et x=3
  \TracerCourbe[Couleur=blue, ValeursInterdites={-2,3}]{1/((x+2)*(x-3))}
  % x*x-3, sans VI
  \TracerCourbe[Couleur=olive]{x^2-3}
\end{GraphiqueTikz}
```



9.2 Voisinages

De manière expérimentale, il est possible de travailler avec des *voisinages*, pour illustrer des limites notamment.
Il est dans ce cas nécessaire que les courbes aient été déclarées avec un `[Nom=...]` afin de permettre au code de réaliser les calculs.

```
\AfficherVoisinage[clés]<courbe>{val x/y}{epsilon}
```

Les `Clés` disponibles sont :

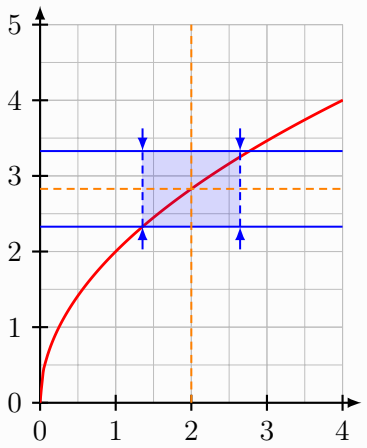
- `Couleurs=...` : couleur du voisinage et des tracés, `blue` par défaut ;
- `CouleurAsymptote=...` : couleur de l'asymptote, `orange` ;
- `Opacite=...` : opacité de la zone, `0.15` par défaut ;
- `TypeLimite=` : choix du type de voisinage, parmi :
  - `L en +inf` ; `L en -inf` ;
  - `+inf en +inf` ; `-inf en +inf` ; `+inf en -inf` ; `-inf en -inf` ;
  - `+inf en a` ; `+inf en a+` ; `+inf en a-` ;
  - `-inf en a` ; `-inf en a+` ; `-inf en a-` ;
  - `L en a`, avec la clé `LimiteEn=...` pour préciser le a ;
- `Asymptote` : booléen (`true` par défaut) pour afficher l'asymptote éventuelle ;
- `Fleches` : booléen (`true` par défaut) pour afficher les flèches de *rentrée* ;
- `DebutX=...` : pour spécifier une valeur de début pour les calculs ;
- `FinX=...` : pour spécifier une valeur de fin pour les calculs.

Concernant les autres arguments :

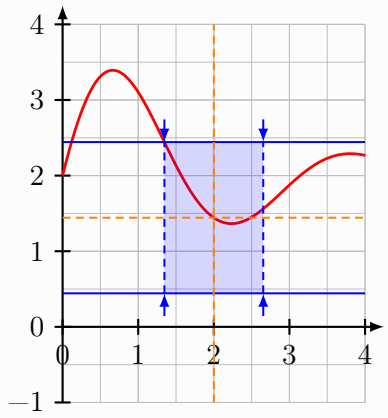
- celui entre `<...>` est le nom de l'objet à utiliser ;
- le premier argument obligatoire est :
  - la valeur de L ou de a (en fonction de l'asymptote considérée) ;
  - ou la valeur de b dans le cas `L en a`, avec la clé `LimiteEn=...` pour préciser le a ;
- le deuxième argument obligatoire permet de préciser la valeur de ϵ le cas échéant.

💣 Cette commande est à utiliser avec prudence, car de multiples dysfonctionnements ou cas particuliers peuvent exister, suivant la complexité des courbes considérées...

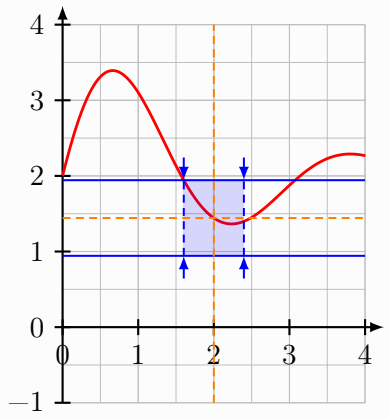
```
\begin{GraphiqueTikz}[Xmin=0,Xmax=4,Ymin=0,Ymax=5]
  \TracerAxesGrilles[Elargir=2.5mm]{auto}{auto}
  \DefinirCourbe[Nom=cf, Trace, Couleur=red]<f>{2*sqrt(x)}
  \AfficherVoisinage%
    [TypeLimite={L en a}, LimiteEn=2, Couleur=green!50!black]%
    <cf>{\xintfloateval{f(2)}}{0.5}
\end{GraphiqueTikz}
```



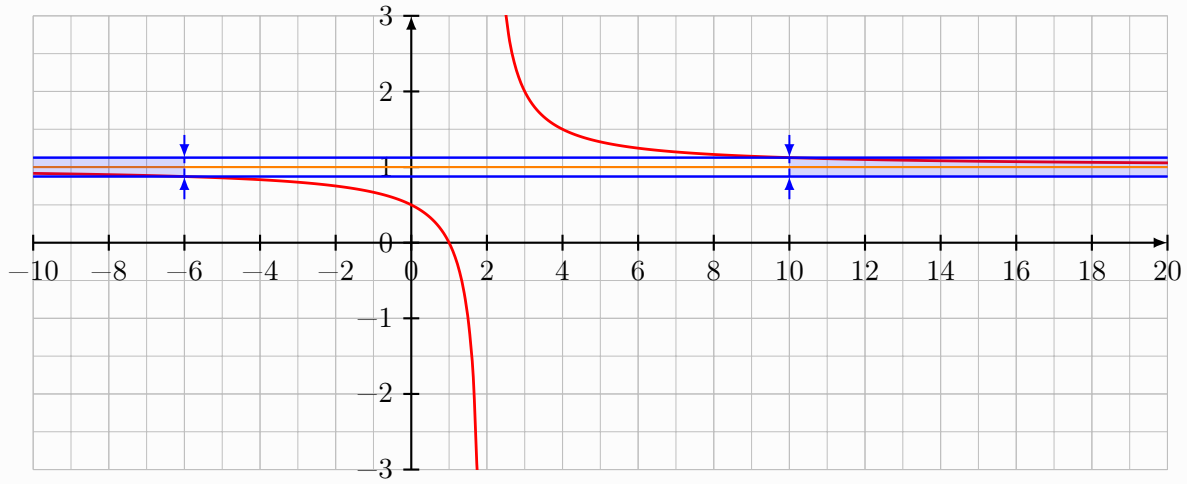

```
\begin{GraphiqueTikz}[Xmin=0,Xmax=4,Ymin=-1,Ymax=4]
  \TracerAxesGrilles[Elargir=2.5mm]{auto}{auto}
  \DefinirCourbe[Nom=cf, Trace, Couleur=red]<f>{2+2*sin(2*x)*exp(-x/2)}
  \AfficherVoisinage%
    [TypeLimite={L en a}, LimiteEn=2]%
    <cf>{\xintfloateval{f(2)}}{1}
\end{GraphiqueTikz}
```



```
\begin{GraphiqueTikz}[Xmin=0,Xmax=4,Ymin=-1,Ymax=4]
  \TracerAxesGrilles[Elargir=2.5mm]{auto}{auto}
  \DefinirCourbe[Nom=cf, Trace, Couleur=red]<f>{2+2*sin(2*x)*exp(-x/2)}
  \AfficherVoisinage%
    [TypeLimite={L en a}, LimiteEn=2]%
    <cf>{\xintfloateval{f(2)}}{0.5}
\end{GraphiqueTikz}
```




```
\begin{GraphiqueTikz}[x=0.5cm,Xmin=-10,Xmax=20,Xgrille=2,Xgrilles=1]
  \TracerAxesGrilles{auto}{auto}
  \DefinirCourbe[ValeursInterdites={2},Nom=montest,Trace,Couleur=red]<f>{1/(x-2)+1}
  \AfficherVoisinage[Entree=auto,TypeLimite={L en +inf},DebutX=2]<montest>{1}{0.125}
  \AfficherVoisinage[Entree=auto,TypeLimite={L en -inf},FinX=2]<montest>{1}{0.125}
\end{GraphiqueTikz}
```



9.3 Coniques

De manière expérimentale, il est possible de travailler sur des coniques :

— tracé d'une ellipse, d'une parabole, d'une hyperbole ;

— mise en avant des éléments caractéristiques.

```
%ellipse
\TracerEllipse[clés,Nom=...]{centre}{a}{b}
%parabole
\TracerParabole[clés,Nom=...,Sens=H/V]{centre}{p}
%hyperbole
\TracerHyperbole[clés,Nom=...,Sens=H/V]{centre}{a}{b}

%les clés sont les mêmes que pour les définitions/tracés de courbes
```

```
%ellipse
\AfficherElementsEllipse[clés]{nom courbe}{centre}{a}{b}
%parabole
\AfficherElementsParabole[clés]{nom courbe}{centre}{p}
%hyperbole
\AfficherElementsHyperbole[clés]{nom courbe}{centre}{a}{b}
```

Les styles disponibles pour les éléments caractéristiques sont :

```
\tikzset{pfltraitvoisinageasympt/.style={%
  densely dashed,line width=0.75pt}%
}
```

Les `Clés` disponibles pour les éléments caractéristiques sont :

— `CouleurPoints=...` : couleur des points particuliers ;

— `CouleurDroites=...` : couleur des droites particulières ;

— `AffCentre` : booléen ;

— `AffFoyers` : booléen ;

— `AffSommets` : booléen ;

— `AffAxes` : booléen ;

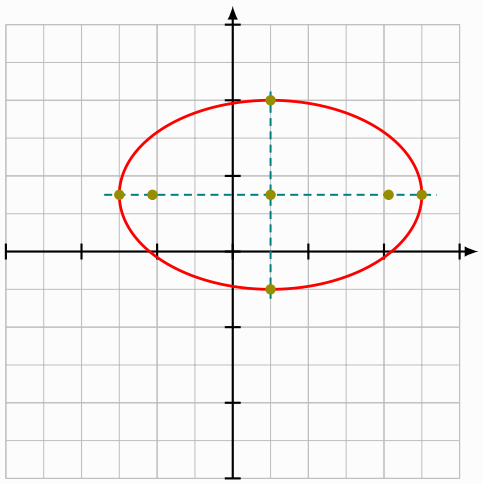
— `AffDirectrice` : booléen (parabole) ;

— `AffAsymptotes` : booléen (hyperbole) ;

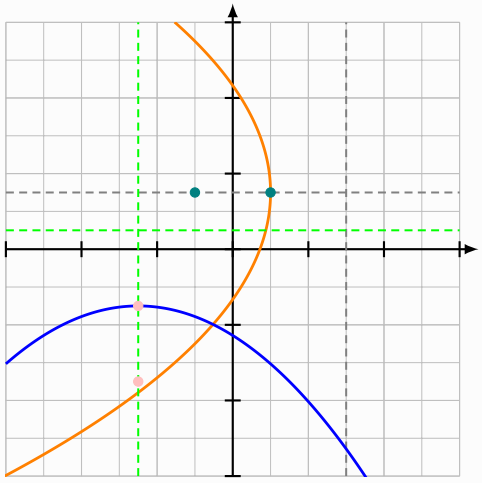
— `AffTout` : booléen (tous les éléments).

— `Sens=H/V` : préciser l'orientation de la conique (parabole/hyperbole).

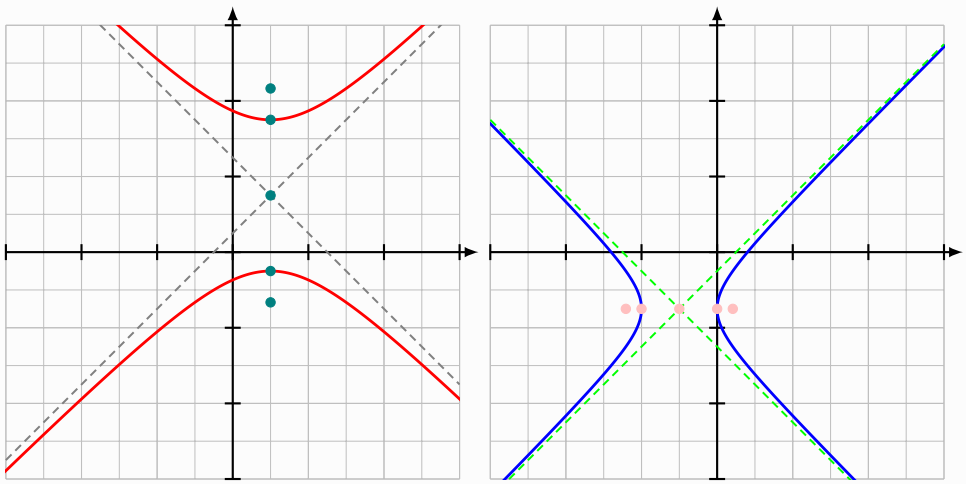
```
\begin{GraphiqueTikz}[]
  \TracerAxesGrilles[Grads=false,Elargir=2.5mm]{auto}{auto}
  \TracerEllipse[Couleur=red,Debut=0,Fin=2*pi,Nom=ellipse]{(0.5,0.75)}{2}{1.25}
  \AfficherElementsEllipse%
    [AffTout,CouleurPoints=olive,CouleurDroites=teal]%
    {ellipse}{(0.5,0.75)}{2}{1.25}
\end{GraphiqueTikz}
```



```
\begin{GraphiqueTikz}[]
  \TracerAxesGrilles[Grads=false,Elargir=2.5mm]{auto}{auto}
  \TracerParabole[Sens=H,Couleur=orange,Nom=paraB]{(0.5,0.75)}{-1}
  \AfficherElementsParabole%
    [Sens=H,AffTout,CouleurPoints=teal,CouleurDroites=gray]%
    {paraB}{(0.5,0.75)}{-1}
  \TracerParabole[Sens=V,Couleur=blue,Nom=paraD]{(-1.25,-0.75)}{-1}
  \AfficherElementsParabole%
    [Sens=V,AffTout,CouleurPoints=pink,CouleurDroites=green]%
    {paraD}{(-1.25,-0.75)}{-1}
\end{GraphiqueTikz}
```



```
\begin{GraphiqueTikz}[]
  \TracerAxesGrilles[Grads=false,Elargir=2.5mm]{auto}{auto}
  \TracerHyperbole[Couleur=red,Nom=hyperb]{(0.5,0.75)}{1}{1}
  \AfficherElementsHyperbole%
    [AffTout,CouleurPoints=teal,CouleurDroites=gray]%
    {hyperb}{(0.5,0.75)}{1}{1}
\end{GraphiqueTikz}
\begin{GraphiqueTikz}[]
  \TracerAxesGrilles[Grads=false,Elargir=2.5mm]{auto}{auto}
  \TracerHyperbole[Couleur=blue,Nom=hyperbol,Sens=H]{(-0.5,-0.75)}{0.5}{0.5}
  \AfficherElementsHyperbole%
    [AffTout,CouleurPoints=pink,CouleurDroites=green,Sens=H]%
    {hyperbol}{(-0.5,-0.75)}{0.5}{0.5}
\end{GraphiqueTikz}
```



9.4 Transformations

9.4.1 Translation

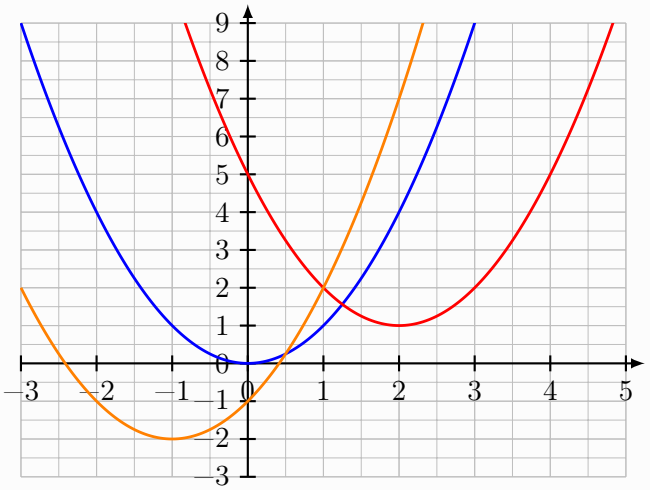
Il est possible de tracer la courbe translatée $x \mapsto f(x-a)+b$ d'une fonction, d'un spline ou d'une interpolation.

```
%avec une formule xint
\TracerTransformee[clés]{nom_fct}{a}{b}
%avec un spline cubique
\TracerTransformee[Spline,clés]{liste_spline}{a}{b}
%avec une interpolation
\TracerTransformee[Interpo,clés]{liste_interpo}{a}{b}
```

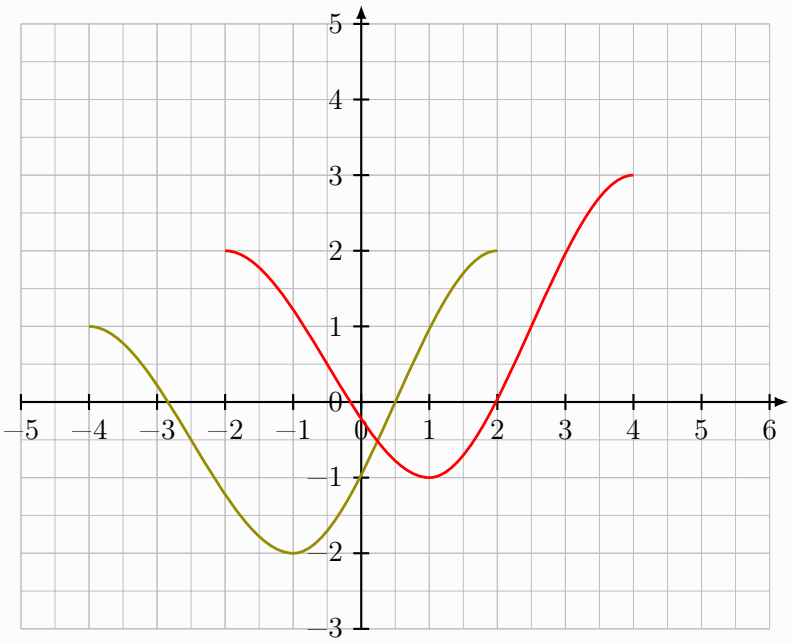
Les `Clés` disponibles sont :

— `Couleur=...` : couleur du tracé (`black` par défaut) ;
— `StyleTrace=...` : style Ti*k*Z supplémentaire ;
— `Nom=...` : nom du *path* pour réutilisation ;
— `Debut=.../Fin=...` : bornes du tracé (`\pflxmin`/`\pflxmax` par défaut) ;
— `Pas=...` : pas du tracé (calculé automatiquement) ;
— `Clip=true/false` : clipping sur la fenêtre (`true` par défaut) ;
— `Spline=true/false` : mode spline cubique (`false` par défaut) ;
— `Interpo=true/false` : mode interpolation (`false` par défaut) ;
— `Alt=true/false` : mode alternatif pour les splines (`false` par défaut) ;
— `Coeffs=...` : coefficient(s) de lissage pour les splines (`3` par défaut) ;
— `Tension=...` : tension pour les interpolations (`0.5` par défaut).

```
\begin{GraphiqueTikz}[x=1cm,y=0.5cm,Xmin=-3,Xmax=5,Ymin=-3,Ymax=9]
  \TracerAxesGrilles[Elargir=2.5mm]{auto}{auto}
  \DefinirCourbe[Nom=cf,Trace,Couleur=blue]<f>{x^2}
  \TracerTransformee[Couleur=red,Nom=cg]{f}{2}{1}        %f(x-2)+1
  \TracerTransformee[Couleur=orange,Nom=ch]{f}{-1}{-2}   %f(x+1)-2
\end{GraphiqueTikz}
```



```
\def\LISTESPLINE{-4/1/0 § -1/-2/0 § 2/2/0}
\begin{GraphiqueTikz}[x=0.9cm,y=1cm,Xmin=-5,Xmax=6,Ymin=-3,Ymax=5]
  \TracerAxesGrilles[Elargir=2.5mm]{auto}{auto}
  % courbe originale
  \DefinirCourbeSpline[Trace,Couleur=olive]{\LISTESPLINE}
  % courbe transformée f(x-2)+1
  \TracerTransformee[Spline,Couleur=red]{\LISTESPLINE}{2}{1}
\end{GraphiqueTikz}
```



9.4.2 Réflexion

Il est possible de tracer la courbe symétrique d'une fonction, d'un spline ou d'une interpolation par rapport aux axes Ox, Oy ou à la droite $y = x$ (réciproque).

```
%avec une formule xint
\TracerReflexion[clés]{nom_fct}
%avec un spline cubique
\TracerReflexion[Spline,clés]{liste_spline}
%avec une interpolation
\TracerReflexion[Interpo,clés]{liste_interpo}
```

Les `Clés` disponibles sont :

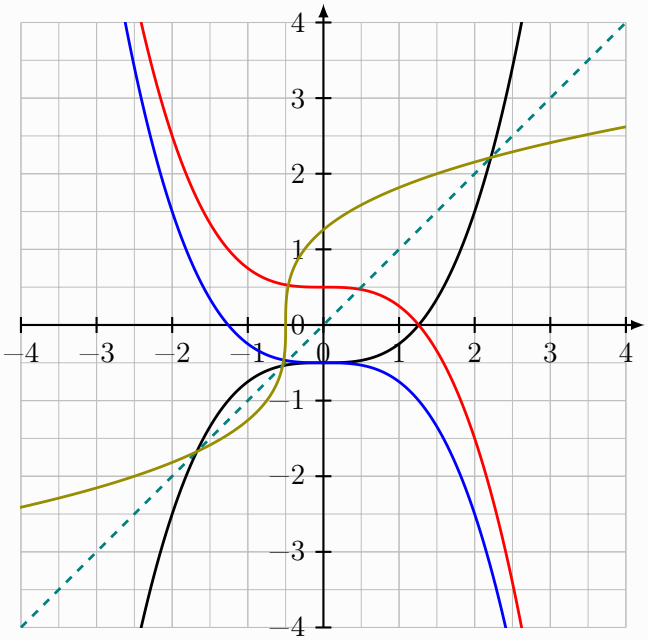
- `Axe=x/y/yx` : axe de symétrie — `x` pour $-f(x)$, `y` pour $f(-x)$, `yx` pour la réciproque (`x` par défaut) ;
- `Couleur=...` : couleur du tracé (`black` par défaut) ;
- `StyleTrace=...` : style Ti*k*Z supplémentaire ;
- `Nom=...` : nom du *path* pour réutilisation ;
- `Debut=.../Fin=...` : bornes du tracé (paramètre t pour `Axe=yx`) ;
- `Clip=true/false` : clipping sur la fenêtre (`true` par défaut) ;
- `Spline=true/false` : mode spline cubique (`false` par défaut) ;
- `Interpo=true/false` : mode interpolation (`false` par défaut) ;
- `Alt=true/false` : mode alternatif pour les splines (`false` par défaut) ;

— `Coeffs=...` : coefficient(s) de lissage pour les splines (`3` par défaut) ;

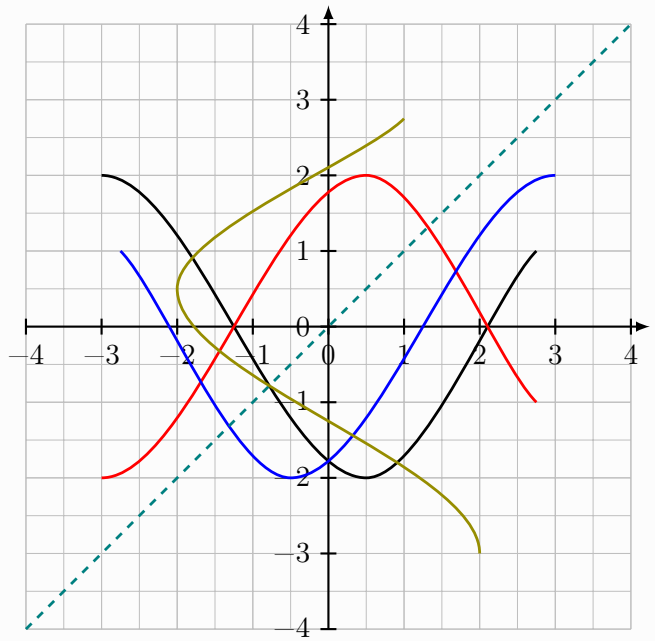
— `Tension=...` : tension pour les interpolations (`0.5` par défaut).

Remarque : pour `Axe=yx` avec `Spline` ou `Interpo`, la symétrie est correcte uniquement si les unités x et y sont égales. Un avertissement est émis dans le `log` sinon.

```
\begin{GraphiqueTikz}[x=1cm,y=1cm,Xmin=-4,Xmax=4,Ymin=-4,Ymax=4]
  \TracerAxesGrilles[Elargir=2.5mm]{auto}{auto}
  \DefinirCourbe[Nom=cf,Trace,Couleur=black]<f>{x^3/4-0.5}
  \TracerReflexion[Axe=x,Couleur=red]{f}
  \TracerReflexion[Axe=y,Couleur=blue]{f}
  \TracerCourbe[Couleur=teal,StyleTrace=dashed]{x}
  \TracerReflexion[Axe=yx,Couleur=olive,Debut=-4,Fin=4]{f}
\end{GraphiqueTikz}
```



```
\def\LISTESPLINE{-3/2/0 § 0.5/-2/0 § 2.75/1/1}
\begin{GraphiqueTikz}[x=1cm,y=1cm,Xmin=-4,Xmax=4,Ymin=-4,Ymax=4]
  \TracerAxesGrilles[Elargir=2.5mm]{auto}{auto}
  \DefinirCourbeSpline[Trace,Couleur=black]{\LISTESPLINE}
  \TracerReflexion[Axe=x,Spline,Couleur=red]{\LISTESPLINE}
  \TracerReflexion[Axe=y,Spline,Couleur=blue]{\LISTESPLINE}
  \TracerCourbe[Couleur=teal,StyleTrace=dashed]{x}
  \TracerReflexion[Axe=yx,Spline,Couleur=olive]{\LISTESPLINE}
\end{GraphiqueTikz}
```



9.5 Taylor

Il est possible de tracer le polynôme de Taylor (ou de Maclaurin) d'ordre N d'une fonction en un point a, calculé par différences finies centrées.
Remarque : cette commande est uniquement disponible pour les fonctions définies via une expression `xint` (pas de spline ni d'interpolation).

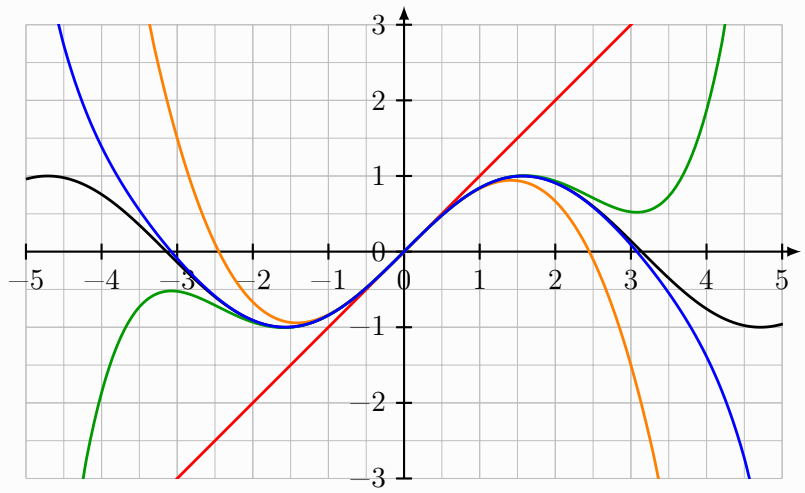
```
%tracé direct
\TracerTaylor[clés]{nom_fct}{a}
%définition + tracé optionnel (Trace=true)
\DefinirTaylor[clés]<nom_taylor>{nom_fct}{a}
```

Les `Clés` disponibles sont :

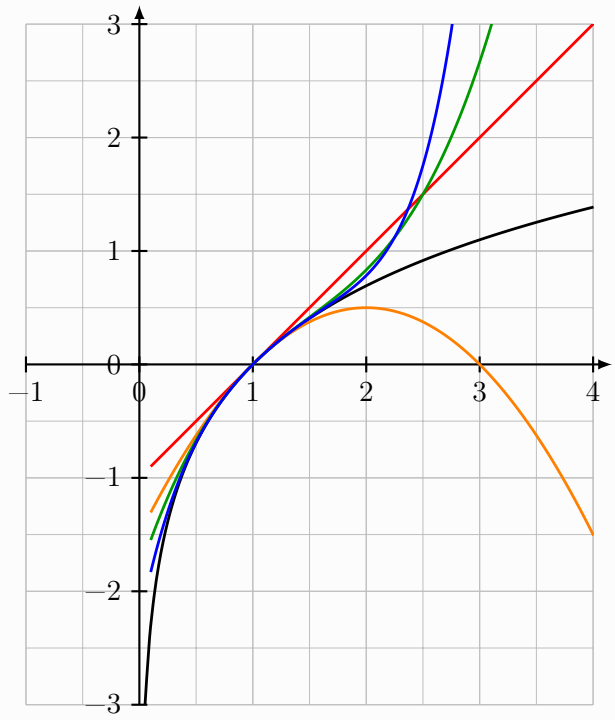
— `Ordre=...` : degré N du polynôme (`5` par défaut) ;
— `h=...` : pas de dérivation numérique (`0.01` par défaut) ;
— `Couleur=...` : couleur du tracé (`black` par défaut) ;
— `StyleTrace=...` : style TikZ supplémentaire ;
— `Nom=...` : nom du *path* pour réutilisation ;
— `Debut=.../Fin=...` : bornes du tracé (`\pflxmin`/`\pflxmax` par défaut) ;
— `Pas=...` : pas du tracé (calculé automatiquement) ;
— `Clip=true/false` : clipping sur la fenêtre (`true` par défaut) ;
— `Trace=true/false` : tracé immédiat pour `\DefinirTaylor` (`false` par défaut).

Remarque : les valeurs du polynôme sont automatiquement bornées entre `\pflMinoffsetV` et `\pflMaxoffsetV` pour éviter les débordements TikZ hors de la fenêtre, notamment pour les ordres élevés hors du rayon de convergence. Pour $N > 7$, les erreurs numériques peuvent devenir significatives.

```
%sin(x) autour de 0, ordres 1, 3, 5, 7
\begin{GraphiqueTikz}[x=1cm,y=1cm,Xmin=-5,Xmax=5,Ymin=-3,Ymax=3]
  \TracerAxesGrilles[Elargir=2.5mm]{auto}{auto}
  \DefinirCourbe[Trace,Couleur=black]<f>{sin(x)}
  \TracerTaylor[Ordre=1,Couleur=red]{f}{0}
  \TracerTaylor[Ordre=3,Couleur=orange]{f}{0}
  \TracerTaylor[Ordre=5,Couleur=green!60!black]{f}{0}
  \TracerTaylor[Ordre=7,Couleur=blue]{f}{0}
\end{GraphiqueTikz}
```



```
%ln(x) autour de 1 - rayon de convergence = 1
\begin{GraphiqueTikz}[x=1.5cm,y=1.5cm,Xmin=-1,Xmax=4,Ymin=-3,Ymax=3]
  \TracerAxesGrilles[Elargir=2.5mm]{auto}{auto}
  \TracerCourbe[Couleur=black,Epaisseur=1.5pt,Debut=0.05]{ln(x)}
  \TracerTaylor[Ordre=1,Couleur=red,Debut=0.1,Fin=4]{ln}{1}
  \TracerTaylor[Ordre=2,Couleur=orange,Debut=0.1,Fin=4]{ln}{1}
  \TracerTaylor[Ordre=3,Couleur=green!60!black,Debut=0.1,Fin=4]{ln}{1}
  \TracerTaylor[Ordre=5,Couleur=blue,Debut=0.1,Fin=4]{ln}{1}
\end{GraphiqueTikz}
```



9.6 Familles de courbes

— Les commandes de familles de courbes sont **expérimentales** et leur syntaxe pourra évoluer dans les versions futures.

— Le paramètre de la famille est noté `n` dans l'expression — il sera substitué par chaque valeur entière de l'intervalle `{début n}{fin n}`.

9.6.1 Définition et tracé dans l'environnement

```
\DefinirFamilleCourbes[clés]<nom>{expr avec n}{début}{fin}
```

— `<nom>` : nom de la famille, utilisé comme base des fonctions xint (`nom_1`, `nom_2`...) et des *name paths* (`famille-nom-1`, `famille-nom-2`...) ;

— `{expr}` : expression avec `n` comme paramètre entier ;

— `{début}{fin}` : bornes entières de l'intervalle ;

— `Trace` : booléen pour tracer les courbes (`false` par défaut) ;

— `Couleurs` : liste CSV de couleurs explicites ;

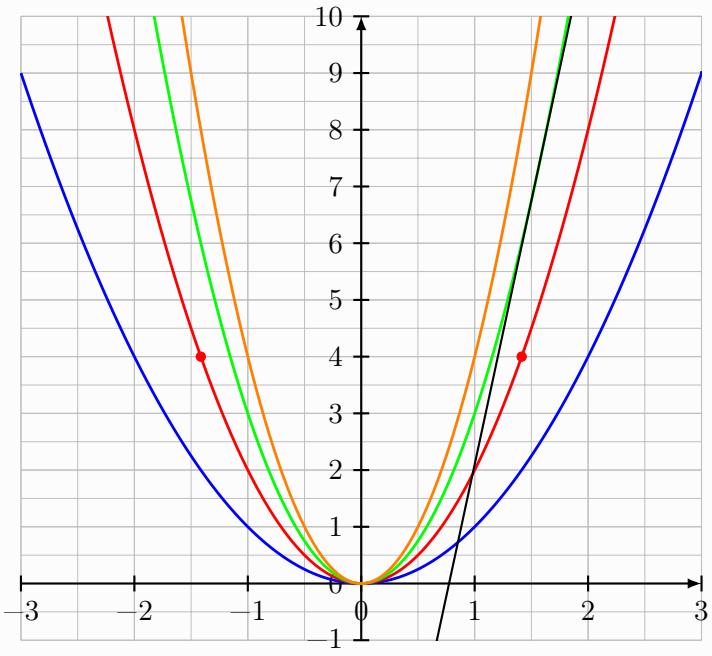
— `CouleurBase` : couleur de base pour un dégradé automatique ;

— `Couleur` : couleur unique pour toutes les courbes (`black` par défaut) ;

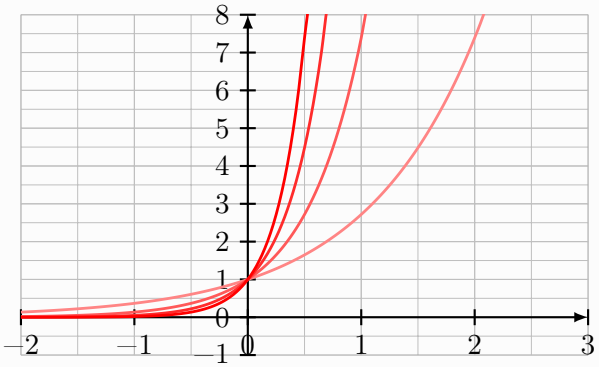
— `Debut`/`Fin` : bornes de tracé (valeurs de l'environnement par défaut) ;

— `DefGlobale` : booléen pour rendre les fonctions xint globales (`false` par défaut).

```
\begin{GraphiqueTikz}[x=1.5cm,y=0.75cm,Xmin=-3,Xmax=3,Ymin=-1,Ymax=10]
  \TracerAxesGrilles{auto}{auto}
  % famille n*x^2 pour n=1..4, couleurs explicites
  \DefinirFamilleCourbes%
    [Trace,Couleurs={blue,red,green,orange}]%
    <parab>{n*x^2}{1}{4}
  % exploitable dans l'environnement
  \TrouverAntecedents[Traits,Couleur=red]{famille-parab-2}{4}
  \TracerTangente[Couleur=blue]{parab_3}{1.5}
\end{GraphiqueTikz}
```



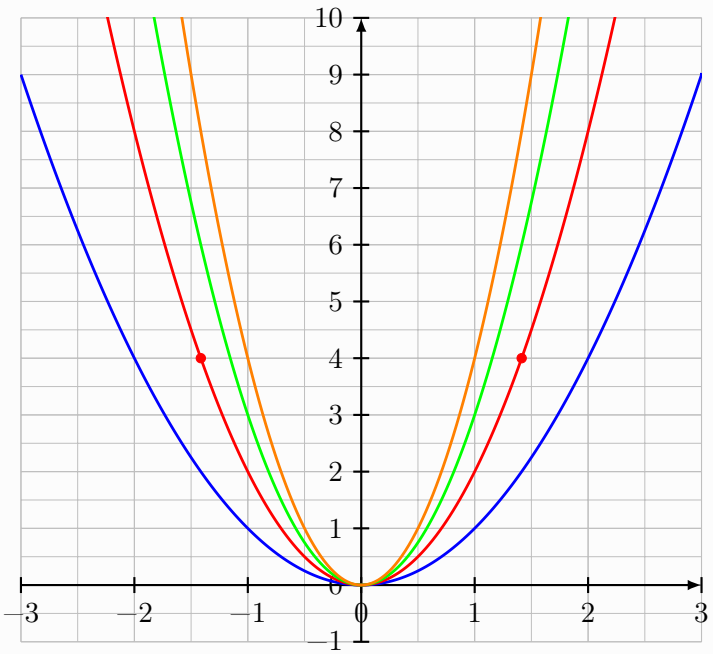
```
\begin{GraphiqueTikz}[x=1.5cm,y=0.5cm,Xmin=-2,Xmax=3,Ymin=-1,Ymax=8]
  \TracerAxesGrilles{auto}{auto}
  % famille exp(n*x) avec dégradé automatique
  \DefinirFamilleCourbes%
    [Trace,CouleurBase=red,RestreindreY]%
    <expo>{exp(n*x)}{1}{4}
\end{GraphiqueTikz}
```



9.6.2 Fonctionnement hors/dans environnement

```
%hors environnement → fonctions xint globales
\GenererFamilleFonctions<parab>{n*x^2}{1}{4}

%dans environnement → tracé uniquement
\TracerFamilleCourbes[clés]<parab>
```

```
\begin{GraphiqueTikz}[x=1.5cm,y=0.75cm,Xmin=-3,Xmax=3,Ymin=-1,Ymax=10]
  \TracerAxesGrilles{auto}{auto}
  \DefinirFamilleCourbes[Trace,Couleurs={blue,red,green,orange},DefGlobale]%
    <parab>{n*x^2}{1}{4}
  \TrouverAntecedents[Traits,Couleur=red]{famille-parab-2}{4}
\end{GraphiqueTikz}

%après - fonctions accessibles globalement
$f_2(3) = \xintfloateval{parab_2(3)}$

\tkzgCalcIntegrale*{parab_2(x)}{0}{2}[\monres]
$\displaystyle\int_0^2 f_2(x)\,\text{d}x \approx \monres$
```



$f_2(3) = 18$

$$\int_0^2 f_2(x)\,\mathrm{d}x \approx 5.333333333333334$$

9.7 Graphiques semi-logarithmiques et log-log (expérimental)

9.7.1 Environnements

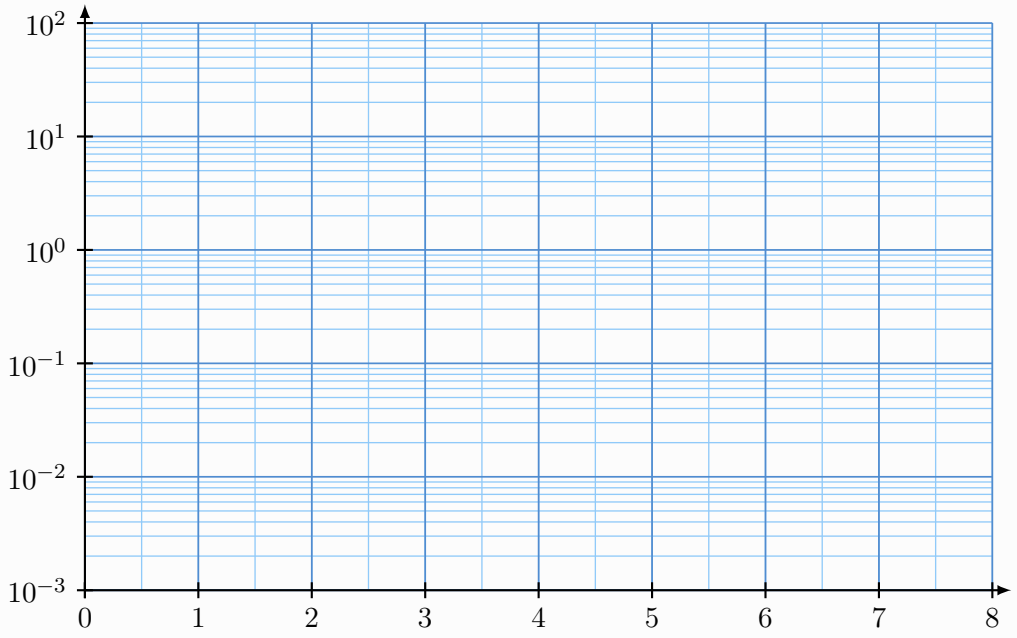
```
% semi-log (axe Y logarithmique)
\begin{GraphiqueTikzSemiLog}[...]<...>
  \TracerAxesGrillesSemiLog[clés]{valeurs X}
  \TracerCourbeSemiLog[clés]{expression}
  \TracerNuageSemiLog[clés]{liste X}{liste Y}
\end{GraphiqueTikzSemiLog}

% log-log (deux axes logarithmiques)
\begin{GraphiqueTikzLogLog}[...]<...>
  \TracerAxesGrillesLogLog[clés]
  \TracerCourbeLogLog[clés]{expression}
  \TracerNuageLogLog[clés]{liste X}{liste Y}
\end{GraphiqueTikzLogLog}
```

— `Ymin`/`Ymax` : valeurs réelles (ex : `Ymin=0.01,Ymax=1000`) qui **doivent** être des puissances de 10 ;

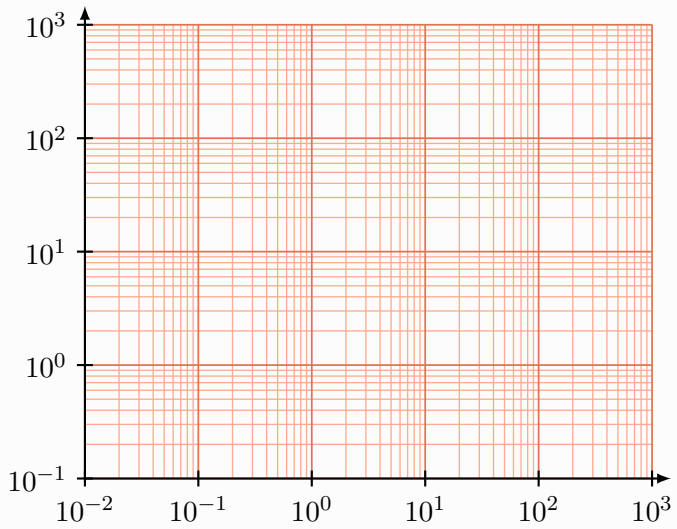
— `Puissance` : booléen (`true` par défaut) qui affiche 10^n sur l'axe `Y`, sinon la valeur décimale.

9.7.2 Exemples

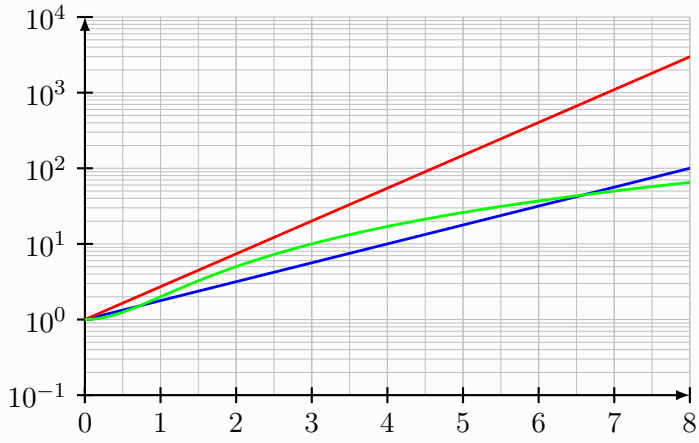
```
\begin{GraphiqueTikzSemiLog}%
    [x=1.5cm,y=1.5cm,Xmin=0,Xmax=8,Ymin=0.001,Ymax=100,Xgrille=1,Xgrilles=0.5]%
    <Theme=bleu>
  \TracerAxesGrillesSemiLog[Elargir=2.5mm]{0,...,8}
\end{GraphiqueTikzSemiLog}
```



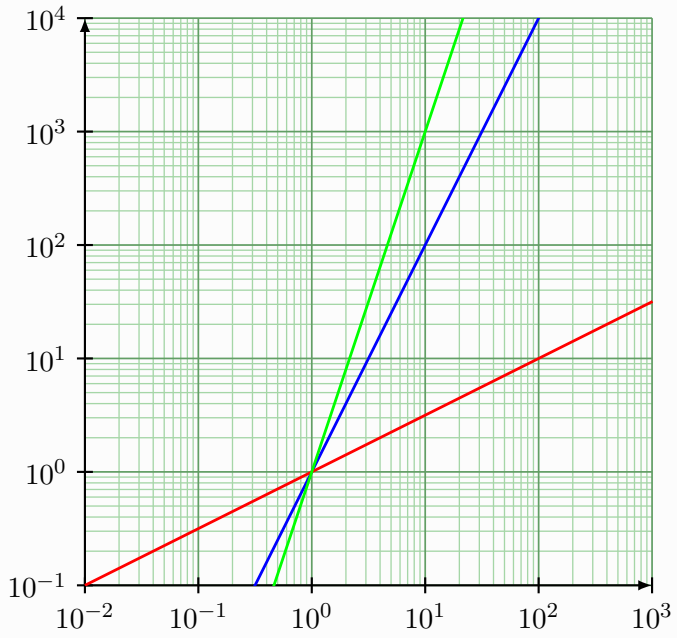
```
\begin{GraphiqueTikzLogLog}%
    [x=1.5cm,y=1.5cm,Xmin=0.01,Xmax=1000,Ymin=0.1,Ymax=1000]<Theme=chaud>
  \TracerAxesGrillesLogLog[Elargir=2.5mm]
\end{GraphiqueTikzLogLog}
```



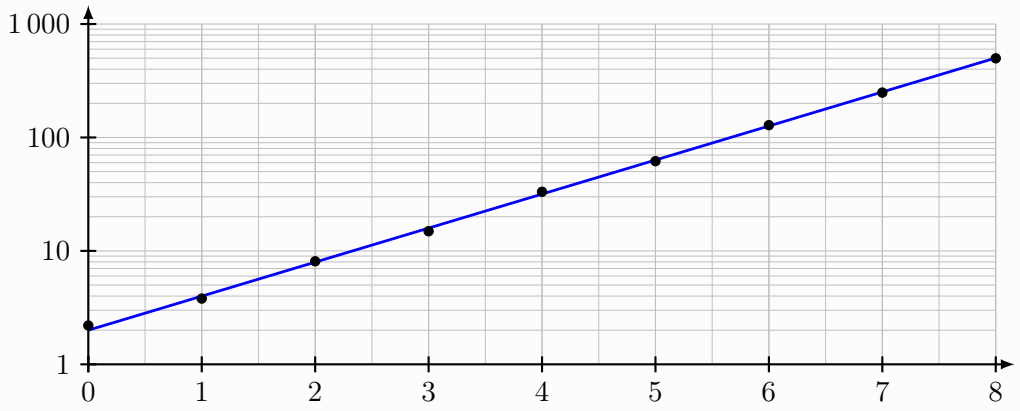

```
\begin{GraphiqueTikzSemiLog}%
    [x=1cm,y=1cm,Xmin=0,Xmax=8,Ymin=0.1,Ymax=10000,Xgrille=1,Xgrilles=0.5]
  \TracerAxesGrillesSemiLog{auto}
  % f(x) = 10^(x/4) → droite en semi-log
  \TracerCourbeSemiLog[Couleur=blue]{10^(x/4)}
  % g(x) = exp(x)
  \TracerCourbeSemiLog[Couleur=red]{exp(x)}
  % h(x) = x^2+1
  \TracerCourbeSemiLog[Couleur=green]{x^2+1}
\end{GraphiqueTikzSemiLog}
```



```
\begin{GraphiqueTikzLogLog}%
    [x=1.5cm,y=1.5cm,Xmin=0.01,Xmax=1000,Ymin=0.1,Ymax=10000]<Theme=vert>
  \TracerAxesGrillesLogLog
  % f(x) = x^2 → droite de pente 2 en log-log
  \TracerCourbeLogLog[Couleur=blue]{x^2}
  % g(x) = x^(0.5) → droite de pente 0.5
  \TracerCourbeLogLog[Couleur=red]{sqrt(x)}
  % h(x) = x^3 → droite de pente 3
  \TracerCourbeLogLog[Couleur=green]{x^3}
\end{GraphiqueTikzLogLog}
```




```
\begin{GraphiqueTikzSemiLog}%
    [x=1.5cm,y=1.5cm,Xmin=0,Xmax=8,Ymin=1,Ymax=1000,Xgrille=1,Xgrilles=0.5]
  \TracerAxesGrillesSemiLog[Elargir=2.5mm,Puissance=false]{auto}
  % courbe théorique
  \TracerCourbeSemiLog[Couleur=blue]{2*10^(0.3*x)}
  % nuage de points (avec un peu de bruit)
  \TracerNuageSemiLog
  {0,1,2,3,4,5,6,7,8}%
  {2.2, 3.8, 8.1, 14.9, 33.2, 61.8, 128.4, 248.7, 498.3}
\end{GraphiqueTikzSemiLog}
```



9.8 Nuages de points avec transformation

Les commandes suivantes permettent de tracer un nuage de points en appliquant une transformation aux données avant le tracé — utile pour la linéarisation de modèles non linéaires.

```
\TracerNuagePoints[ModeTransfo=...,Clés points]{liste X}{liste Y}
```

— `ModeTransfo` : transformation appliquée aux données avant le tracé (`lin` par défaut) ;

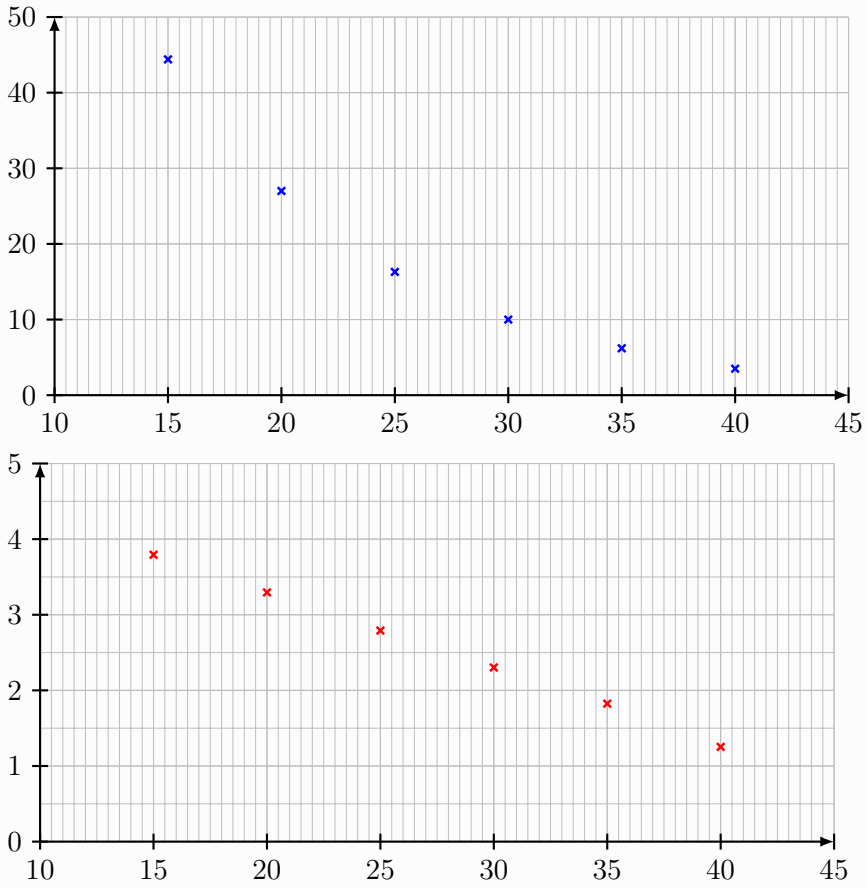
— `[Clés points]` : clés passées à `\MarquerPts` (style, couleur...).

Les modes `semilogx`, `semilogy` et `loglog` nécessitent d'être dans un environnement `GraphiqueTikzSemiLog` ou `GraphiqueTikzLogLog` : les variables de translation `\pflslogexpomin` et `\pflslogexpominx` doivent être définies.

Le tableau suivant récapitule les modes disponibles :

Mode	X tracé	Y tracé
`lin`	x	y
`lnx` / `lny` / `lnxy`	$\ln(x)$	$\ln(y)$
`log10x` / `log10y` / `log10xy`	$\log_{10}(x)$	$\log_{10}(y)$
`expx` / `expy` / `expxy`	e^x	e^y
`invx` / `invy` / `invxy`	$1/x$	$1/y$
`sqrtx` / `sqrty` / `sqrtxy`	$\sqrt{x}$	$\sqrt{y}$
`carrex` / `carrey` / `carrexy`	x^2	y^2
`semilogx` / `semilogy`	$\log_{10}$ + translation (SemiLog)	
`loglog`	$\log_{10}$ + translation (LogLog)	

```
\def\listeX{15,20,25,30,35,40}
\def\listeY{44.4,27.0,16.3,10.0,6.2,3.5}

%nuage brut
\begin{GraphiqueTikz}%
    [x=0.3cm,y=0.1cm,Origx=10,Xmin=10,Xmax=45,%
    Ymin=0,Ymax=50,Xgrille=5,Ygrille=10,Ygrilles=10]
  \TracerAxesGrilles{auto}{auto}
  \TracerNuagePoints[Style=x,Couleur=blue]{\listeX}{\listeY}
\end{GraphiqueTikz}

%nuage transformé z=ln(y)
\begin{GraphiqueTikz}%
    [x=0.3cm,y=1cm,Origx=10,Xmin=10,Xmax=45,Ymin=0,Ymax=5,Xgrille=5,Ygrille=1]
  \TracerAxesGrilles{auto}{auto}
  \TracerNuagePoints[ModeTransfo=lny,Style=x,Couleur=red]{\listeX}{\listeY}
\end{GraphiqueTikz}
```



9.9 Lecture de fichiers CSV

Ces commandes nécessitent le chargement préalable de `csvsimple`, un avertissement est émis sinon.

9.9.1 Tracer directement depuis un CSV

```
%usepackage{csvsimple}
\TracerNuageDepuisCSV[Clés]<Clés points>{fichier.csv}{colX}{colY}
```

— `ModeTransfo` : même modes que `\TracerNuagePoints` (`lin` par défaut) ;

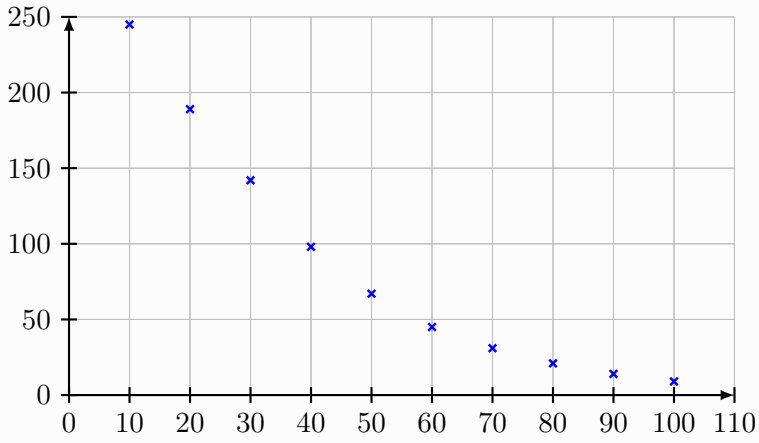
— `Separateur` : séparateur CSV (`semicolon` par défaut) ;

— `MaxLignes` : nombre maximum de lignes traitées (`100000` par défaut) — utile pour limiter le temps de compilation ;

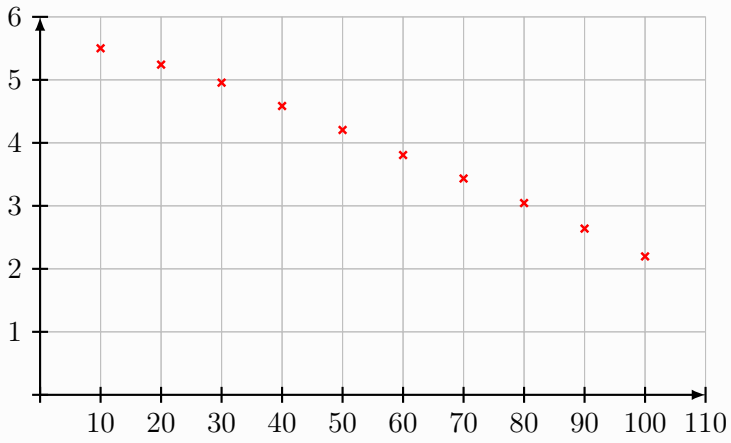
— `<Clés points>` : clés passées à `\MarquerPts`.

```
\begin{filecontents*}{datatestgrapheur.csv}
n;x;y
1;10;245
2;20;189
3;30;142
4;40;98
5;50;67
6;60;45
7;70;31
8;80;21
9;90;14
10;100;9
\end{filecontents*}
```

```
%nuage brut
\begin{GraphiqueTikz}%
    [x=0.08cm,y=0.02cm,Xmin=0,Xmax=110,%
    Ymin=0,Ymax=250,Xgrille=10,Xgrilles=10,Ygrille=50,Ygrilles=50]
  \TracerAxesGrilles{auto}{auto}
  \TracerNuageDepuisCSV<Style=x,Couleur=blue>{datatestgrapheur.csv}{x}{y}
\end{GraphiqueTikz}
```



```
% nuage transformé lny
\begin{GraphiqueTikz}%

    ↪  [x=0.08cm,y=0.833cm,Xmin=0,Xmax=110,Ymin=0,Ymax=6,Xgrille=10,Xgrilles=10,Ygrilles=1]
  \TracerAxesGrilles*{auto}{auto}
  \TracerNuageDepuisCSV%
    [ModeTransfo=lny]%
    <Style=x,Couleur=red>{datatestgrapheur.csv}{x}{y}
\end{GraphiqueTikz}
```



9.9.2 Créer des listes depuis un CSV

Si l'on préfère stocker les données dans des macros avant de les utiliser :

```
\CreerListesDepuisCSV[clés]{fichier.csv}{colX}{colY}{nomMacroX}{nomMacroY}
```

— `{nomMacroX}/{nomMacroY}` : noms des macros de sortie (sans \), qui créent `\nomMacroX` et `\nomMacroY` ;

— mêmes clés `ModeTransfo`, `Separateur`, `MaxLignes` que `\TracerNuageDepuisCSV`.

9.10 Courbes implicites (LuaLaTeX)

9.10.1 Principe

La commande `\TracerCourbeImplicite` permet de tracer une courbe définie implicitement par $f(x, y) = 0$, en utilisant l'algorithme des *marching squares* : le repère est découpé en une grille de cellules, et pour chaque cellule, on détecte les changements de signe de f sur les arêtes par interpolation linéaire.

Prérequis : cette commande nécessite une compilation en LuaLaTeX (`luacode` est chargé par `tkz-grapheur`).

Cette commande est *alpha*-expérimentale, donc beaucoup de prudence sur son utilisation :

— pas de gestion des valeurs interdites...

— possibilité de *bruit* suivant la précision des calculs et de la tolérance...

— temps de compilation qui peut être elevé pour des *grands* graphique...

9.10.2 Syntaxe

```
\TracerCourbeImplicite[clés]{expression f(x,y)}
```

Les `<clés>` disponibles sont :

— `Pas` : pas de la grille — `auto` pour un calcul automatique depuis l'épaisseur, sinon une valeur numérique (`0.05` par exemple) ; défaut `auto` ;

— `CoeffPasAuto` : coefficient multiplicateur du pas automatique ; défaut `1` ;

— `Couleur` : couleur du tracé ; défaut `black` ;

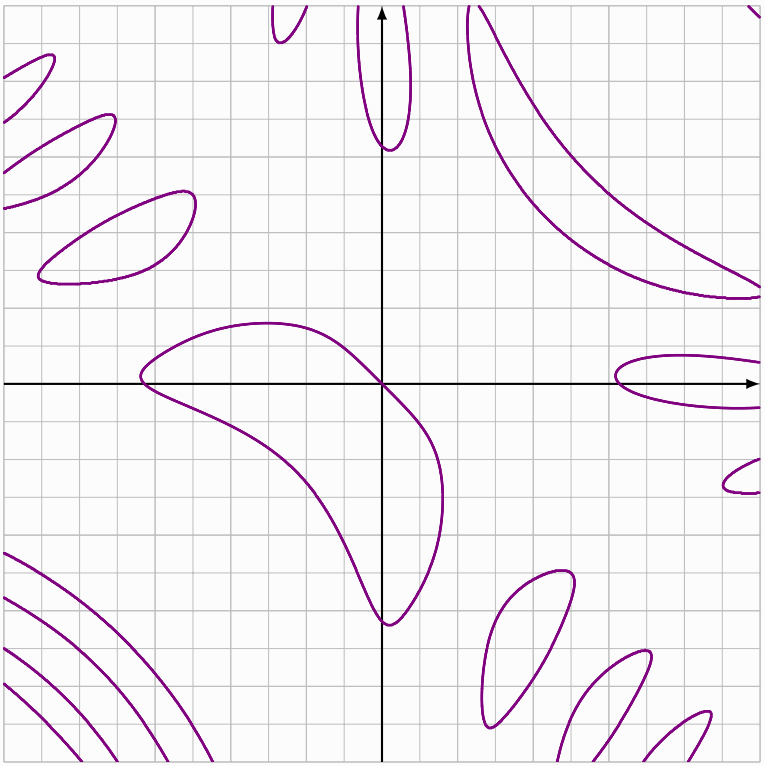
— `Epaisseur` : épaisseur des points de la courbe ; défaut `0.2875pt` ;

— `Tolerance` : seuil de détection des zéros exacts ; défaut `1e-10`.

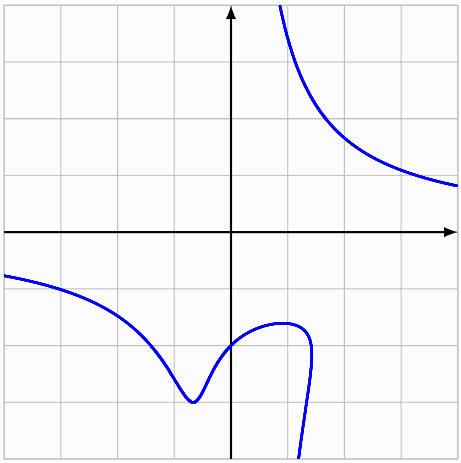
L'expression `f(x,y)` est saisie avec la syntaxe habituelle : `x^2+y^2-1` pour $x^2 + y^2 - 1 = 0$, `sin(x+y)-cos(x*y)` etc. Les fonctions disponibles sont `sin`, `cos`, `tan`, `exp`, `sqrt`, `log`, `abs` ainsi que les constantes `pi` et `e`.

9.10.3 Exemples

```
\begin{GraphiqueTikz}[Xmin=-5,Xmax=5,Ymin=-5,Ymax=5]
  \TracerAxesGrilles{}{}
  \TracerCourbeImplicite%
    [Couleur=violet,CoeffPasAuto=2]%
    {sin(x+y)-cos(x*y)+1}
\end{GraphiqueTikz}
```



```
\begin{GraphiqueTikz}%
    [x=1.5cm,y=1.5cm,Xmin=-2,Xmax=2,Ymin=-2,Ymax=2]
  \TracerAxesGrilles{}{}
  \TracerCourbeImplicite[Couleur=blue]{x*y^2+2*x^3*y^3-y-1}
\end{GraphiqueTikz}
```

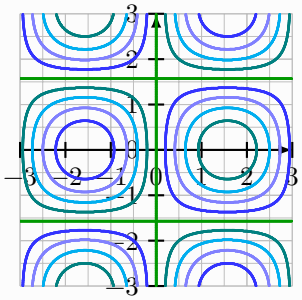


9.10.4 Courbes de niveaux

En complément de la commande précédente, une commande de tracé de lignes de niveau est proposé, sous la forme $f(x, y) = k$, sur la même base.
À noter qu'une clé `[Couleurs=...]` a été rajoutée pour permettre de proposer une liste (qui sera traitée comme cyclique) de couleurs.

```
\TracerLignesNiveau[clés]{expression f(x,y)}{valeurs de k}
```

```
\begin{GraphiqueTikz}%
    [x=1.2cm,y=1.2cm,Xmin=-3,Xmax=3,Ymin=-3,Ymax=3]
  \TracerAxesGrilles{auto}{auto}
  % sin(x)*cos(y) = k  →  lignes de niveau d'une surface ondulée
  \TracerLignesNiveau%
    [Couleurs={blue!80,blue!50,cyan,teal,green!60!black}]%
    {sin(x)*cos(y)}%
    {-0.8,-0.6,-0.4,-0.2,0,0.2,0.4,0.6,0.8}
\end{GraphiqueTikz}
```



9.11 Diagramme circulaire, diagramme annulaire

De manière expérimentale, il est possible de tracer un diagramme circulaire (camembert), dans un environnement `tikzpicture` standard (et non dans `GraphiqueTikz`).

💣 Ces commandes sont expérimentales. La couleur de chaque secteur est **obligatoire** dans la liste de données.

```
%dans un environnement tikzpicture
\TracerDiagrammeCirculaire[clés]{liste}
\TracerDiagrammeAnneau[clés]{liste}
\LegendeCirculaire[clés]{liste}
```

La `liste` est spécifiée sous la forme `label/valeur/couleur/style,...`.

Les `[clés]` sont :

— `Rayon=...` : rayon du cercle (en unités TikZ), `3` par défaut ;

— `RayonInt=...` : rayon intérieur du cercle (en unités TikZ), `1.75` par défaut ;

— `Titre=...` : titre affiché au-dessus du diagramme, vide par défaut.

Les `[clés]` de `\LegendeCirculaire` sont :

— `TailleCarreau=...` : taille des carreaux colorés, `0.28` par défaut ;

— `Espacement=...` : espacement vertical entre les entrées, `0.55` par défaut ;

— `DecalEst=...` : décalage horizontal depuis `cam-est`, `0.4` par défaut ;

— `Pct` : booléen, `true` par défaut, pour afficher les pourcentages (sinon les valeurs brutes) ;

— `Arrondi=...` : décimales pour l'affichage, `0` par défaut.

Après `\DiagrammeCirculaire`, les nœuds suivants sont disponibles pour placer des éléments manuellement :

— `cam-centre`, `cam-nord`, `cam-sud`, `cam-est`, `cam-ouest` : nœuds cardinaux ;

— `cam-i-debut`, `cam-i-fin` : points sur le cercle aux bornes du secteur i ;

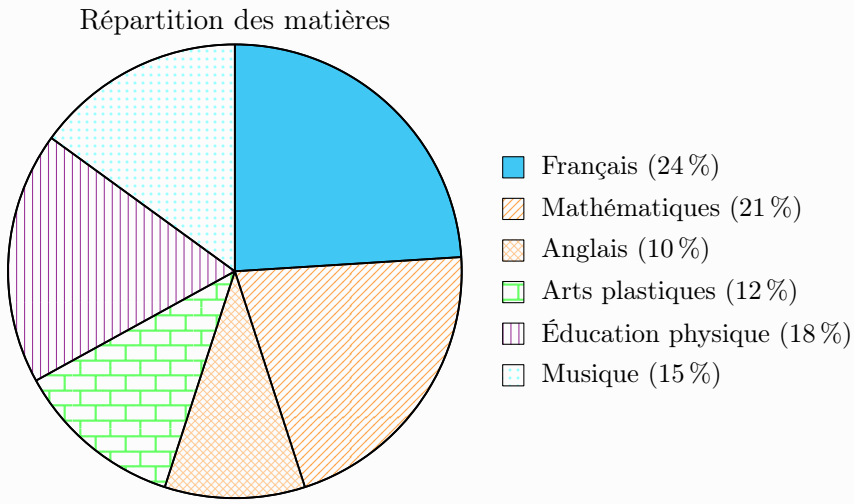
— `cam-i-mil` : point sur le cercle à l'angle médian du secteur i ;

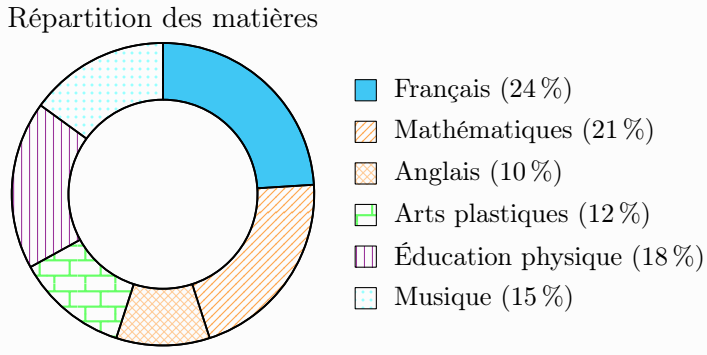
— `cam-i-int` : point intérieur (68 % du rayon), pour labels dans le secteur ;

— `cam-i-ext` : point extérieur (115 % du rayon), pour ligne et label externe.

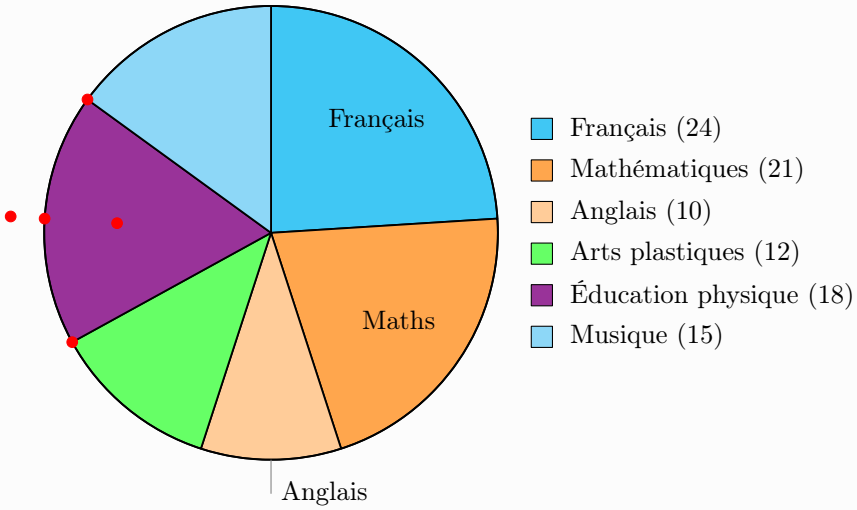
`\LegendeCirculaire` s'ancre automatiquement sur `cam-est`, elle doit donc être appelée après `\DiagrammeCirculaire`, dans le même `tikzpicture`.

```
\def\DONNEES{%
  Français/24/cyan!60/solid,%
  Mathématiques/21/orange!70/north east lines,%
  Anglais/10/orange!40/crosshatch,%
  Arts plastiques/12/green!60/bricks,%
  Éducation physique/18/violet!80/vertical lines,%
  Musique/15/cyan!40/dots%
}

\begin{tikzpicture}[line join=bevel]
  \TracerDiagrammeCirculaire%
    [Titre={Répartition des matières}]{\DONNEES}
  \LegendeCirculaire{\DONNEES}
\end{tikzpicture}

\begin{tikzpicture}[line join=bevel]
  \TracerDiagrammeAnneau%
    [Rayon=2,RayonInt=1.25,Titre={Répartition des matières}]%
   {\DONNEES}
  \LegendeCirculaire{\DONNEES}
\end{tikzpicture}
```





```
\def\DONNEES{%
  Français/24/cyan!60/,%
  Mathématiques/21/orange!70/,%
  Anglais/10/orange!40/,%
  Arts plastiques/12/green!60/,%
  Éducation physique/18/violet!80/,%
  Musique/15/cyan!40/%
}

%effectifs bruts + labels manuels sur les grands secteurs
\begin{tikzpicture}[line join=bevel]
  \TracerDiagrammeCirculaire[]{\DONNEES}
  \LegendeCirculaire%
    [Pct=false,DecalEst=0.3]{\DONNEES}
  %labels manuels via les nœuds
  \node[align=center,font=\small] at (cam-1-int) {Français} ;
  \node[align=center,font=\small] at (cam-2-int) {Maths} ;
  \draw[gray,thin] (cam-3-mil) -- (cam-3-ext) ;
  \node[right,font=\small] at (cam-3-ext) {Anglais} ;
  %visualisation des nœuds créés (secteur 5)
  \filldraw[red] (cam-5-debut) circle[radius=2pt]
                 (cam-5-fin) circle[radius=2pt]
                 (cam-5-mil) circle[radius=2pt]
                 (cam-5-int) circle[radius=2pt]
                 (cam-5-ext) circle[radius=2pt] ;
\end{tikzpicture}
```



10 Commandes numériques

10.1 Introduction

En marge des commandes graphiques, des commandes numériques ont été intégrées au package `tkz-grapheur`.
Les résultats obtenus sont *bruts*, afin de pouvoir être réutilisés (arrondis, formatage) ultérieurement.

Les commandes présentées dans cette section sont des *moteurs de calcul numérique* utilisables dans et hors de l'environnement `GraphiqueTikz`.
Elles stockent leurs résultats dans des macros et/ou les affichent directement. La convention est :

— sans étoile `\commande{...}` : affiche le résultat ;

— avec étoile `\commande*{...}[\macro]` : stocke le résultat dans `\macro` (défaut : `\mycalcres`).

10.2 Analyse

10.2.1 Maximum, minimum

```
\tkzgCalcMax[precision]{expr}{deb}{fin}[\macromax][\macroxmax]
\tkzgCalcMax[precision]{expr}{deb}{fin}[\macromin][\macroxmin]
```

— `[précision]` : précision décimale.

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

147.1517764685769 en 5

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

0.5623351587104050 en 1.099

10.2.2 Taux d'accroissement, nombre dérivée

```
\tkzgTauxAccroiss*{expression}{a}{h}[\macro]

\tkzgDeriveeNum*[précision]<gd>{expr}{a}[\macro]
```

— `[précision]` : précision décimale.

```
% entre 2 et 2 + 0.5, avec f(x)=x^2
\tkzgTauxAccroiss{x**2}{2}{0.5}
```

4.5

```
% en x=3 : f'(3) = 6
\tkzgDeriveeNum{x^2}{3}           % symétrique → 6.000...
\tkzgDeriveeNum<d>{x^2}{3}        % droite → 6.0001...
\tkzgDeriveeNum<g>{x^2}{3}        % gauche → 5.9999...

% en x=0 (borne gauche) : utiliser <d>
\tkzgDeriveeNum<d>{x^2}{0}        % → 0.0001...

% f(x) = sin(x), f'(0) = 1
\tkzgDeriveeNum{sin(x)}{0}        % → 1.000...
```

6 6.0001 5.9999
0.0001
0.9999999983333335

10.2.3 Intégrale numérique (méthode de Simpson)

```
\tkzgCalcIntegrale[N]{expression}{a}{b}
\tkzgCalcIntegrale*[N]{expression}{a}{b}[\macro]
```

— `[N]` : nombre de subdivisions (`50` par défaut).

```
$\displaystyle\int_0^{\pi} \sin(x)\,\mathrm{d}x \approx
↪  \tkzgCalcIntegrale{sin(x)}{0}{pi}$

\tkzgCalcIntegrale*{sin(x)}{0}{pi}[\monres]
$\displaystyle\int_0^{\pi} \sin(x)\,\mathrm{d}x \approx \monres$
```

$$\int_0^{\pi} \sin(x)\,\mathrm{d}x \approx 2.000000010824505$$
$$\int_0^{\pi} \sin(x)\,\mathrm{d}x \approx 2.000000010824505$$

10.2.4 Résolution approchée par dichotomie

```
\tkzgResolApproch[precision]{equation}{a:b}
\tkzgResolApproch*[precision]{equation}{a:b}[\macro]

%avec vérification TVI (f(a)*f(b)<0)
\tkzgResolApprochTVI[precision]{equation}{a:b}
\tkzgResolApprochTVI*[precision]{equation}{a:b}[\macro]
```

— `[precision]` : précision en valeur absolue (`0.001` par défaut) ;
— `{equation}` : équation de la forme `f(x)=valeur` ou simplement `f(x)` (résolution = 0) ;
— `{a:b}` : intervalle de recherche.

La version `TVI` vérifie que $f(a) \times f(b) < 0$ avant de lancer la dichotomie — si ce n'est pas le cas, la macro retourne `???`.

```
%résolution de e^x = 3 sur [0;2]
$x \approx \tkzgResolApproch{exp(x)=3}{0:2}$

%stockage
\tkzgResolApproch*{exp(x)=3}{0:2}[\monx]
$x \approx \monx$
```

$x \approx 1.098602294921875$
$x \approx 1.098602294921875$

10.2.5 Seuil d'une suite récurrente

```
\tkzgCalcSeuilSuite[Nmax]{f(x)}{n0}{u0}{signe}{seuil}
\tkzgCalcSeuilSuite*[Nmax]{f(x)}{n0}{u0}{signe}{seuil}[\macro]
```

— `{f(x)}` : relation de récurrence $u_{n+1} = f(u_n)$;
— `{n0}`, `u0` : rang et valeur initiaux ;
— `{signe}` : condition d'arrêt parmi `>`, `<`, `>=`, `<=` ;
— `{seuil}` : valeur seuil à atteindre ;
— `[Nmax]` : nombre maximal d'itérations (`1000` par défaut) — sécurité anti-boucle infinie.

```
%premier n tel que 1.05^n > 2
\tkzgCalcSeuilSuite*{1.05*x}{0}{1}{>}{2}[\monrang]
$n_0 = \monrang$
```

$n_0 = 16$

10.3 Probabilités

10.3.1 Lois discrètes

– Loi binomiale

```
%P(X=k)
\tkzgCalcBinomP{n}{p}{k}
\tkzgCalcBinomP*{n}{p}{k}[\macro]

%P(a<=X<=b)  (a/b = * => borne naturelle, 0 ou n)
\tkzgCalcBinomC{n}{p}{a}{b}
\tkzgCalcBinomC*{n}{p}{a}{b}[\macro]
```

```
% X -> B(10 ; 0.3)
$P(X=3) \approx \tkzgCalcBinomP{10}{0.3}{3}$

$P(X \leqslant 4) \approx \tkzgCalcBinomC{10}{0.3}{*}{4}$

$P(2 \leqslant X \leqslant 5) \approx \tkzgCalcBinomC{10}{0.3}{2}{5}$
```

$P(X = 3) \approx 0.266827932$
$P(X \leqslant 4) \approx 0.8497316674$
$P(2 \leqslant X \leqslant 5) \approx 0.8033426667$

– Loi de Poisson

```
%P(X=k)
\tkzgCalcPoissP{lambda}{k}
\tkzgCalcPoissP*{lambda}{k}[\macro]

%P(a<=X<=b)
\tkzgCalcPoissC{lambda}{a}{b}
\tkzgCalcPoissC*{lambda}{a}{b}[\macro]
```

```
% Y -> P(5)
$P(Y=3) \approx \tkzgCalcPoissP{5}{3}$

$P(Y \geqslant 2) \approx \tkzgCalcPoissC{5}{2}{*}$
```

$P(Y = 3) \approx 0.1403738958142806$
$P(Y \geqslant 2) \approx 0.9595723180054873$

– Loi géométrique

```
%P(X=k)
\tkzgCalcGeomP{p}{k}
\tkzgCalcGeomP*{p}{k}[\macro]

%P(a<=X<=b)
\tkzgCalcGeomC{p}{a}{b}
\tkzgCalcGeomC*{p}{a}{b}[\macro]
```

```
% T -> G(0.3)
$P(T=4) \approx \tkzgCalcGeomP{0.3}{4}$

$P(T \leqslant 5) \approx \tkzgCalcGeomC{0.3}{*}{5}$
```

$P(T = 4) \approx 0.1029$
$P(T \leqslant 5) \approx 0.83193$

– Loi hypergéométrique

```
%P(X=k)   N=population  n=tirage  m=succès dans population

\tkzgCalcHypergeomP{N}{n}{m}{k}
\tkzgCalcHypergeomP*{N}{n}{m}{k}[\macro]

%P(a<=X<=b)
\tkzgCalcHypergeomC{N}{n}{m}{a}{b}
\tkzgCalcHypergeomC*{N}{n}{m}{a}{b}[\macro]
```

```
% W -> H(50,10,6)
$P(W=3) \approx \tkzgCalcHypergeomP{50}{6}{10}{3}$

$P(1 \leqslant W \leqslant 4) \approx \tkzgCalcHypergeomC{50}{6}{10}{1}{4}$
```

$P(W = 3) \approx 0.07460967735845495$
$P(1 \leqslant W \leqslant 4) \approx 0.7578036209858597$

10.3.2 Lois continues

– Loi normale

```
%P(a<=X<=b)  (a/b = * => borne infinie)
\tkzgCalcNormC{mu}{sigma}{a}{b}
\tkzgCalcNormC*{mu}{sigma}{a}{b}[\macro]
```

```
% X -> N(100 ; 10)
$P(X \leqslant 115) \approx \tkzgCalcNormC{100}{10}{*}{115}$

$P(90 \leqslant X \leqslant 110) \approx \tkzgCalcNormC{100}{10}{90}{110}$

$P(X \geqslant 120) \approx \tkzgCalcNormC{100}{10}{120}{*}$
```

$P(X \leqslant 115) \approx 0.933192899994106$
$P(90 \leqslant X \leqslant 110) \approx 0.682689361682794$
$P(X \geqslant 120) \approx 0.0227502618904134$

– Loi exponentielle

```
%P(a<=X<=b)  (a/b = * => 0 pour a, +inf approché pour b)
\tkzgCalcExpoC{lambda}{a}{b}
\tkzgCalcExpoC*{lambda}{a}{b}[\macro]
```

```
% Z -> Exp(0.1)
$P(Z \geqslant 5) \approx \tkzgCalcExpoC{0.1}{5}{*}$

$P(3 \leqslant Z \leqslant 8) \approx \tkzgCalcExpoC{0.1}{3}{8}$
```

$P(Z \geqslant 5) \approx 0.6065306597126334$
$P(3 \leqslant Z \leqslant 8) \approx 0.2914892565644963$

– Loi de Student

```
%P(a<=T_nu<=b)  [N] = subdivisions Simpson (50 par défaut)
\tkzgCalcStudentC[N]{nu}{a}{b}
\tkzgCalcStudentC*[N]{nu}{a}{b}[\macro]
```

```
% T -> Student(5)
$P(-2 \leqslant T \leqslant 2) \approx \tkzgCalcStudentC{5}{-2}{2}$

$P(T \geqslant 1.5) \approx \tkzgCalcStudentC{5}{1.5}{*}$
```

$P(-2 \leqslant T \leqslant 2) \approx 0.8980605179981787$
$P(T \geqslant 1.5) \approx 0.09695044329706689$

– Loi de Fischer

```
%P(a<=F_{d1,d2}<=b)  attention : a > 0 !
\tkzgCalcFischerC[N]{d1}{d2}{a}{b}
\tkzgCalcFischerC*[N]{d1}{d2}{a}{b}[\macro]
```

```
% F -> Fischer(5,10)
$P(1 \leqslant F \leqslant 3) \approx \tkzgCalcFischerC{5}{10}{1}{3}$
```

$P(1 \leqslant F \leqslant 3) \approx 0.3995618628587974$

10.3.3 Quantiles

– Loi normale

```
%P(X<=k)=p
\tkzgQuantileNormal{mu}{sigma}{p}
\tkzgQuantileNormal*{mu}{sigma}{p}[\macro]

%P(X>k)=p
\tkzgQuantileNormalSup{mu}{sigma}{p}
\tkzgQuantileNormalSup*{mu}{sigma}{p}[\macro]

%P(mu-h<X<mu+h)=p  -> retourne h
\tkzgQuantileNormalCentre{mu}{sigma}{p}
\tkzgQuantileNormalCentre*{mu}{sigma}{p}[\macro]
```

```
% X -> N(100 ; 10)
$k$ tel que $P(X \leqslant k)=0{,}95$ :
$k \approx \tkzgQuantileNormal{100}{10}{0.95}$

$k$ tel que $P(X > k)=0{,}05$ :
$k \approx \tkzgQuantileNormalSup{100}{10}{0.05}$

$h$ tel que $P(100-h \leqslant X \leqslant 100+h)=0{,}95$ :
$h \approx \tkzgQuantileNormalCentre{100}{10}{0.95}$
```

k tel que $P(X \leqslant k) = 0{,}95$: $k \approx 116.4485549926758$
k tel que $P(X > k) = 0{,}05$: $k \approx 116.4485549926758$
h tel que $P(100 - h \leqslant X \leqslant 100 + h) = 0{,}95$: $h \approx 19.59964752197266$

– Loi binomiale

```
%plus petit k tel que P(X<=k)>=p
\tkzgQuantileBinom{n}{p}{seuil}
\tkzgQuantileBinom*{n}{p}{seuil}[\macro]

%plus grand k tel que P(X>=k)>=p
\tkzgQuantileBinomSup{n}{p}{seuil}
\tkzgQuantileBinomSup*{n}{p}{seuil}[\macro]
```

```
% X -> B(20 ; 0.3)
plus petit $k$ tel que $P(X \leqslant k) \geqslant 0{,}95$ :
$k = \tkzgQuantileBinom{20}{0.3}{0.95}$

plus grand $k$ tel que $P(X \geqslant k) \geqslant 0{,}05$ :
$k = \tkzgQuantileBinomSup{20}{0.3}{0.05}$
```

plus petit k tel que $P(X \leqslant k) \geqslant 0{,}95$: $k = 9$
plus grand k tel que $P(X \geqslant k) \geqslant 0{,}05$: $k = 9$

10.3.4 Intervalles de fluctuation et de confiance

```
%intervalle de fluctuation : p=proportion théorique, n=taille échantillon
\tkzgCalcIntFluctu<seuil>[niveau]{p}{n}[\borneinf][\bornesup]
\tkzgCalcIntFluctu*<seuil>[niveau]{p}{n}[\borneinf][\bornesup]

%intervalle de confiance : f=fréquence observée, n=taille échantillon
\tkzgCalcIntConf<seuil>[niveau]{f}{n}[\borneinf][\bornesup]
\tkzgCalcIntConf*<seuil>[niveau]{f}{n}[\borneinf][\bornesup]
```

— `<seuil>` : niveau de probabilité parmi `90`, `95` (défaut), `99`, ou `val` ;
— `[niveau]` : niveau scolaire parmi `2de` et `Term` ;
— `\borneinf`, `\bornesup` : macros de stockage des bornes (défaut : `\myborneinf`, `\mybornesup`).

```
%IF à 95% niveau Term, p=0.6, n=500
\tkzgCalcIntFluctu*{0.6}{500}[\myif][\myifs]
$I_f = [\myif;\myifs]$

%IC à 95% niveau Term, f=0.63, n=500
\tkzgCalcIntConf*{0.63}{500}[\myic][\myics]
$I_c = [\myic;\myics]$
```

$I_f = [0.557058551491595; 0.642941448508405]$
$I_c = [0.5876803364852696; 0.6723196635147304]$

10.4 Statistiques

10.4.1 Paramètres statistiques

```
%liste simple
\tkzgCalcParamStats{liste}[min][Q1][mediane][Q3][max]

%liste avec effectifs (valeur/effectif, séparateur .)
\tkzgCalcParamStats*{liste}[min][Q1][mediane][Q3][max]
```

Des résultats sont également stockés automatiquement dans :

— `\mamoyenne` : moyenne ;
— `\monecarttype` : écart-type.

```
\tkzgCalcParamStats{2,5,3,8,4,7,6,1,9,5}
Min : \monmin, Q1 : \monquartileun, Médiane : \mamediane,
Q3 : \monquartiletrois, Max : \monmax,
Moyenne : \mamoyenne, Écart-type : \monecarttype
```

Min : 1, Q1 : 3, Médiane : 5, Q3 : 7, Max : 9, Moyenne : 5, Écart-type : 2.449489742783178

10.4.2 Moyenne et écart-type

```
%liste simple
\tkzgCalcMoyEctype{liste}[\moyenne][\ecarttype]

%liste avec effectifs
\tkzgCalcMoyEctype*{liste}[\moyenne][\ecarttype]
```

```
\tkzgCalcMoyEctype{2,5,3,8,4,7,6,1,9,5}[\mamoy][\monect]
Moyenne : \mamoy, Écart-type : \monect

%avec effectifs : valeur/effectif
\tkzgCalcMoyEctype*{10/3,20/5,30/2}[\mamoy][\monect]
Moyenne : \mamoy, Écart-type : \monect
```

Moyenne : 5, Écart-type : 2.449489742783178
Moyenne : 19, Écart-type : 7

10.5 Sommes de termes de suites

10.5.1 Somme générale

```
\tkzgCalcSommeSuite{expr(k)}{deb}{fin}
\tkzgCalcSommeSuite*{expr(k)}{deb}{fin}[\macro]
```

— `{expr(k)}` : expression du terme général en fonction de `k` ;
— `{deb}`, `{fin}` : bornes inférieure et supérieure de la somme.

```
%somme des 10 premiers entiers
$\displaystyle\sum_{k=1}^{10} k = \tkzgCalcSommeSuite{k}{1}{10}$

%somme avec stockage
\tkzgCalcSommeSuite*{k^2}{1}{5}[\monres]
$\displaystyle\sum_{k=1}^{5} k^2 = \monres$
```

$$\sum_{k=1}^{10} k = 55$$
$$\sum_{k=1}^{5} k^2 = 55$$

10.5.2 Somme d'une suite arithmétique

```
\tkzgCalcSommeArithm{p}{a}{r}{n}
\tkzgCalcSommeArithm*{p}{a}{r}{n}[\macro]
```

— `{p}` : rang initial ;
— `{a}` : valeur initiale u_p ;
— `{r}` : raison ;
— `{n}` : rang final.

```
%u0=3, raison=2, somme de u0 à u10
$S = \tkzgCalcSommeArithm{0}{3}{2}{10}$

\tkzgCalcSommeArithm*{0}{3}{2}{10}[\monres]
$S = \monres$
```

$S = 143$
$S = 143$

10.5.3 Somme d'une suite géométrique

```
\tkzgCalcSommeGeo{p}{a}{q}{n}
\tkzgCalcSommeGeo*{p}{a}{q}{n}[\macro]
```

— `{p}` : rang initial ;
— `{a}` : valeur initiale u_p ;
— `{q}` : raison ;
— `{n}` : rang final.

```
%u0=1, raison=0.5, somme de u0 à u8
$S = \tkzgCalcSommeGeo{0}{1}{0.5}{8}$

\tkzgCalcSommeGeo*{0}{1}{0.5}{8}[\monres]
$S = \monres$
```

$S = 1.99609375$
$S = 1.99609375$

10.5.4 Somme d'une suite récurrente

```
\tkzgCalcSommeSuiteRecurr{f(x)}{p}{up}{n}
\tkzgCalcSommeSuiteRecurr*{f(x)}{p}{up}{n}[\macro]
```

— `{f(x)}` : relation de récurrence $u_{n+1} = f(u_n)$;
— `{p}`, `{up}` : rang et valeur initiaux ;
— `{n}` : rang final.

```
%u0=100, un+1=1.03*un, somme de u0 à u5
$S = \tkzgCalcSommeSuiteRecurr{1.03*x}{0}{100}{5}$

\tkzgCalcSommeSuiteRecurr*{1.03*x}{0}{100}{5}[\monres]
$S \approx \monres$
```

$S = 646.84098843$
$S \approx 646.84098843$

10.6 Taux d'évolution

10.6.1 Taux d'évolution

```
\tkzgCalcTauxEvol{VA}{VD}
\tkzgCalcTauxEvol*{VA}{VD}[\macro]
```

— `{VA}` : valeur d'arrivée ;
— `{VD}` : valeur de départ.

Le résultat est le taux $t = \dfrac{V_A - V_D}{V_D}$, exprimé sous forme décimale.

```
%passage de 80 à 92
$t = \tkzgCalcTauxEvol{92}{80}$

\tkzgCalcTauxEvol*{92}{80}[\montaux]
$t \approx \num{\montaux}$
```

$t = 0.15$
$t \approx 0{,}15$

10.6.2 Taux réciproque

```
\tkzgCalcTauxReciproque{t}
\tkzgCalcTauxReciproque*{t}[\macro]
```

— `{t}` : taux d'évolution (sous forme décimale).

Le résultat est le taux réciproque $t' = \dfrac{1}{1+t} - 1$.

```
%taux réciproque d'une augmentation de 15%
$t' = \tkzgCalcTauxReciproque{0.15}$

\tkzgCalcTauxReciproque*{0.15}[\treciproque]
$t' \approx \num{\treciproque}$
```

$t' = -0.1304347826086957$
$t' \approx -0{,}130\,434\,782\,608\,695\,7$

10.6.3 Taux moyen

```
\tkzgCalcTauxMoyen{t}{n}
\tkzgCalcTauxMoyen*{t}{n}[\macro]
```

— `{t}` : taux global d'évolution (sous forme décimale) ;

— `{n}` : nombre de périodes.

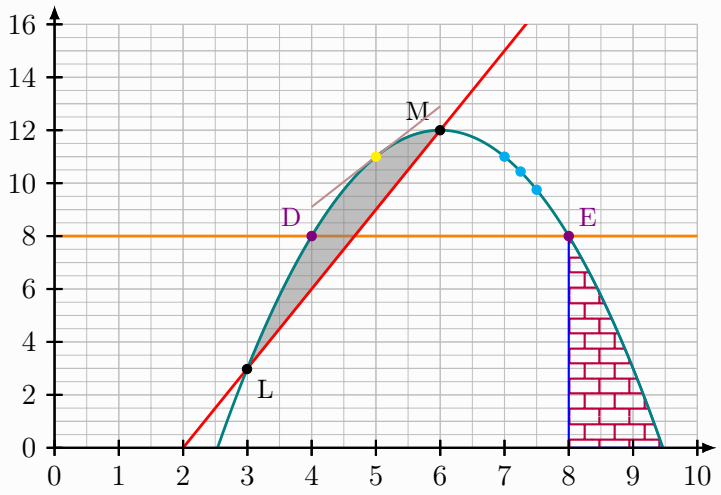
Le résultat est le taux moyen $t_m = (1+t)^{1/n} - 1$.

```
%taux moyen d'une augmentation globale de 30% sur 5 ans
$t_m = \tkzgCalcTauxMoyen{0.30}{5}$

\tkzgCalcTauxMoyen*{0.30}{5}[\tmoy]
$t_m \approx \num{\tmoy}$
```

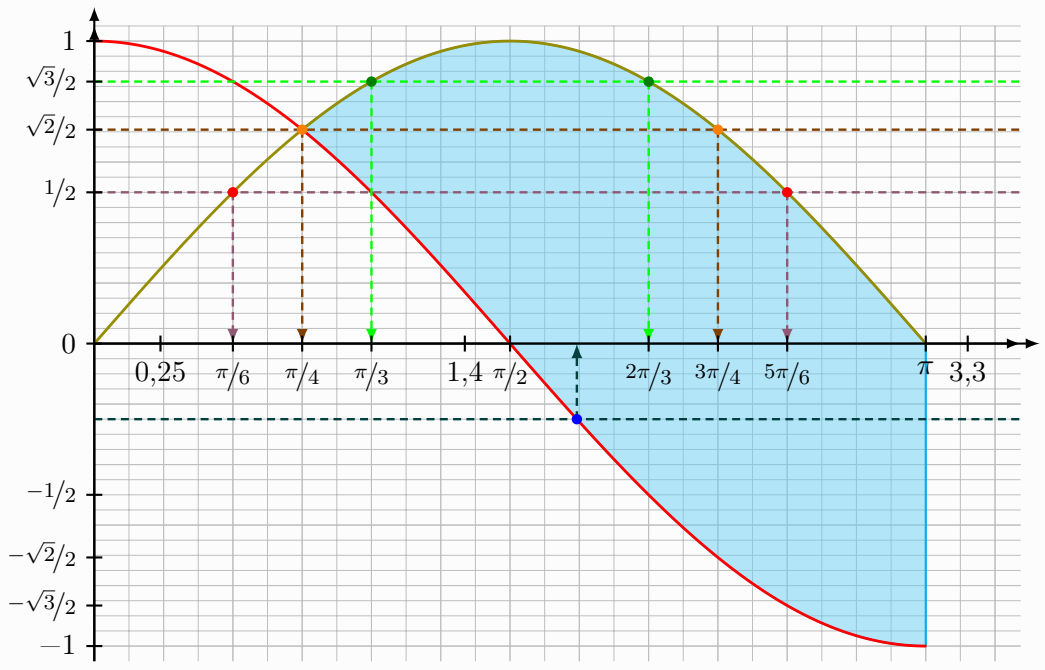
$t_m = 0.053873952061783$
$t_m \approx 0{,}053\,873\,952\,061\,783$

11 Codes source des exemples de la page d'accueil

```
\begin{GraphiqueTikz}[x=0.85cm,y=0.35cm,Xmin=0,Xmax=10,Ymin=0,Ymax=16]
  %préparation de la fenêtre
  \TracerAxesGrilles[Derriere,Elargir=2.5mm,Police=\small]{0,1,...,10}{0,2,...,16}
  %déf des fonctions avec nom courbe + nom fonction + expression (tracés à la fin !)
  \DefinirCourbe[Nom=cf]<f>{3*x-6}
  \DefinirCourbe[Nom=cg]<g>{-(x-6)^2+12}
  %antécédents et intersection
  \TrouverIntersections[Aff=false,Nom=K]{cf}{cg}
  \TrouverAntecedents[AffDroite,Couleur=orange,Nom=I]{cg}{8}
  \TrouverAntecedents[Aff=false,Nom=J]{cg}{0}
  %intégrale sous une courbe, avec intersection
  \TracerIntegrale%
    [Couleurs=blue/purple,Bornes=noeuds,Style=hachures,Hachures=bricks]%
    {g(x)}%
    {(I-2)}{(J-2)}
  %intégrale entre les deux courbes
  \TracerIntegrale[Bornes=noeuds,Type=fct/fct]%
    {f(x)}[g(x)]%
    {(K-1)}{(K-2)}
  %tracé des courbes et des points
  \TracerCourbe[Couleur=red]{f(x)}
  \TracerCourbe[Couleur=teal]{g(x)}
  \PlacerPoints<\small>{(K-1)/below right/L,(K-2)/above left/M}%
  \PlacerPoints[violet]<\small>{(I-1)/above left/D,(I-2)/above right/E}%
  %tangente
  \TracerTangente[Couleurs=pink!75!black/yellow,kl=2,kr=2,AffPoint]{g}{5}
  %images
  \PlacerImages[Couleurs=cyan]{g}{7,7.25,7.5}
  %surimpression des axes
  \TracerAxesGrilles[Devant,Elargir=2.5mm]{0,1,...,10}{0,2,...,16}
\end{GraphiqueTikz}
```



```
\begin{GraphiqueTikz}%
  [x=3.5cm,y=4cm,
  Xmin=0,Xmax=3.5,Xgrille=pi/12,Xgrilles=pi/24,
  Ymin=-1.05,Ymax=1.05,Ygrille=0.2,Ygrilles=0.05]
  %préparation de la fenêtre
  \TracerAxesGrilles[Derriere,Elargir=2.5mm,Format=ntrig/nsqrt]{}{}
  %rajouter des valeurs
  \RajouterValeursAxeX{0.25,1.4,3.3}{\num{0.25},\num{1.4},\num{3.3}}
  %fonction trigo (déf + tracé)
  \DefinirCourbe[Nom=ccos,Debut=0,Fin=pi]<fcos>{cos(x)}
  \DefinirCourbe[Nom=csin,Debut=0,Fin=pi]<fsin>{sin(x)}
  %intégrale
  \TrouverIntersections[Aff=false,Nom=JKL]{ccos}{csin}
  \TracerIntegrale%
    [Bornes=noeud/abs,Type=fct/fct,Couleurs=cyan/cyan!50]%
    {fsin(x)}[fcos(x)]%
    {(JKL-1)}{pi}
  %tracé des courbes
  \TracerCourbe[Couleur=red,Debut=0,Fin=pi]{fcos(x)}
  \TracerCourbe[Couleur=olive,Debut=0,Fin=pi]{fsin(x)}
  %antécédent(s)
  \PlacerAntecedents[Couleurs=blue/teal!50!black,Traits]{ccos}{-0.25}
  \PlacerAntecedents[Couleurs=red/magenta!50!black,Traits]{csin}{0.5}
  \PlacerAntecedents[Couleurs=orange/orange!50!black,Traits]{csin}{sqrt(2)/2}
  \PlacerAntecedents[Couleurs=green!50!black/green,Traits]{csin}{sqrt(3)/2}
  %surimpression axes
  \TracerAxesGrilles[Devant,Format=ntrig/nsqrt]%
    {pi/6,pi/4,pi/3,pi/2,2*pi/3,3*pi/4,5*pi/6,pi}
    {0,sqrt(2)/2,1/2,sqrt(3)/2,1,-1,-sqrt(3)/2,-1/2,-sqrt(2)/2}
\end{GraphiqueTikz}
```



12 Commandes auxiliaires

12.1 Intro

En marge des commandes purement *graphiques*, quelques commandes auxiliaires sont disponibles :

— une pour formater un nombre avec une précision donnée ;

— une pour travailler sur des nombres aléatoires, avec contraintes.

12.2 Arrondi formaté

La commande `\ArrondirNum` permet de formater, grâce au package `siunitx`, un nombre (ou un calcul), avec une précision donnée. Cela peut être *utile* pour formater des résultats obtenus grâce aux commandes de récupération des coordonnées, par exemple.

```
\ArrondirNum[précision]{calcul xint}
```

```
\ArrondirNum{1/3}\\
\ArrondirNum{16.1}\\
\ArrondirNum[3]{log(10)}\\
```

0,33
16,1
2,303

12.3 Gestion de listes

Les commandes suivantes permettent de travailler sur les listes CSV générées automatiquement via la clé `ListeRes` ou manuellement via `\CreerListeAbscisses`.

`\CreerListeAbscisses{préfixe}{nb}{nom}` Construit une liste CSV des abscisses des nœuds préfixe-1, préfixe-2, ..., préfixe-nb, et la stocke dans la macro `\nom`. Si `nb` vaut `0`, la macro est laissée vide sans erreur.

`\tkzgCalcLongueurListe*{liste}[macro]` Retourne le nombre d'éléments de la liste CSV `liste`. La version étoilée stocke le résultat dans `macro` sans l'afficher ; la version normale affiche également le résultat.

`\tkzgCalcElementListe*{liste}{index}[macro]` Retourne l'élément d'index `index` de la liste CSV `liste`. L'index peut être positif (depuis le début) ou négatif (depuis la fin) : `-1` désigne ainsi le dernier élément. La version étoilée stocke sans afficher.

Ces trois commandes s'articulent naturellement avec les itérateurs de xint, notamment `\xintFor*` et `\xintNthElt`, pour parcourir ou exploiter les listes générées.

12.4 Nombre aléatoire sous contraintes

L'idée de cette deuxième commande est de pouvoir déterminer un nombre aléatoire :

— entier ou décimal ;

— sous contraintes (entre deux valeurs fixées).

Cela peut permettre, par exemple, de travailler sur des courbes avec points *aléatoires*, mais respectant certaines contraintes.

```
\ChoisirNbAlea(*)[precision (déf 0)]{borne inf}{borne sup}[\macro]
```

La version étoilée prend les contraintes sous forme stricte (borne inf $<$ macro $<$ borne sup) alors que la version normale prend les contraintes sous forme large (borne inf $\leqslant$ macro $\leqslant$ borne sup).

À noter que les *bornes* peuvent être des *macros* existantes !

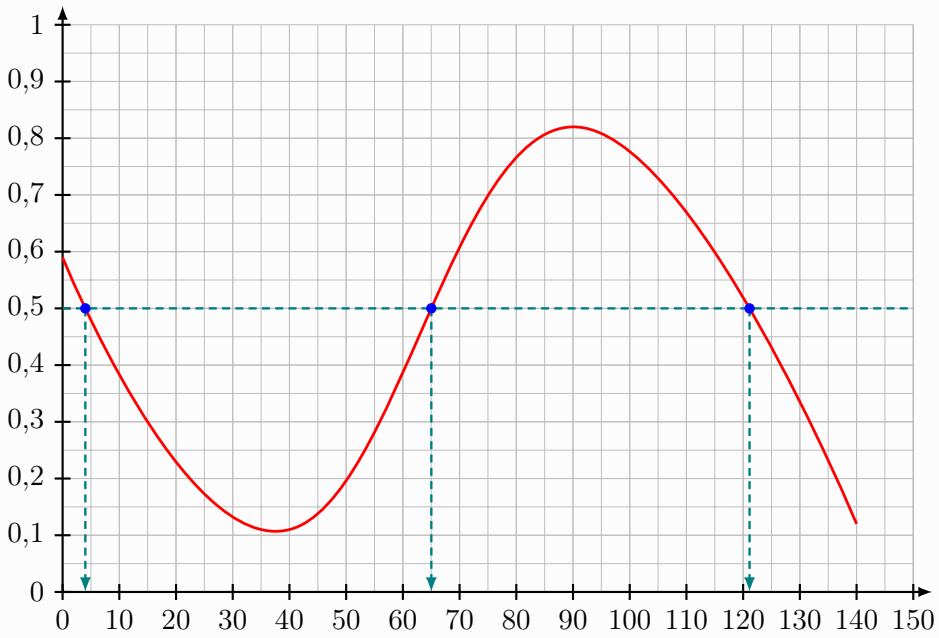
```
%un nombre (2 chiffres après la virgule) entre 0.75 et 0.95
%un nombre (2 chiffres après la virgule) entre 0.05 et 0.25
%un nombre (2 chiffres après la virgule) entre 0.55 et \YrandMax
%un nombre (2 chiffres après la virgule) entre \YrandMin et 0.45
\ChoisirNbAlea[2]{0.75}{0.95}[\YrandMax]%
\ChoisirNbAlea[2]{0.05}{0.25}[\YrandMin]%
\ChoisirNbAlea*[2]{0.55}{\YrandMax}[\YrandA]%
\ChoisirNbAlea*[2]{\YrandMin}{0.45}[\YrandB]%
%vérification
\num{\YrandMax} \& \num{\YrandMin} \& \num{\YrandA} \& \num{\YrandB}
```

0,78 & 0,18 & 0,75 & 0,38

```
%un nombre (2 chiffres après la virgule) entre 0.75 et 0.95
%un nombre (2 chiffres après la virgule) entre 0.05 et 0.25
%un nombre (2 chiffres après la virgule) entre 0.55 et \YrandMax
%un nombre (2 chiffres après la virgule) entre \YrandMin et 0.45
\ChoisirNbAlea[2]{0.75}{0.95}[\YrandMax]%
\ChoisirNbAlea[2]{0.05}{0.25}[\YrandMin]%
\ChoisirNbAlea*[2]{0.55}{\YrandMax}[\YrandA]%
\ChoisirNbAlea*[2]{\YrandMin}{0.45}[\YrandB]%
%vérification
\num{\YrandMax} \& \num{\YrandMin} \& \num{\YrandA} \& \num{\YrandB}
```

0,75 & 0,16 & 0,57 & 0,18

```
%la courbe est prévue pour qu'il y ait 3 antécédents
\ChoisirNbAlea[2]{0.75}{0.95}[\YrandMax]%
\ChoisirNbAlea[2]{0.05}{0.25}[\YrandMin]%
\ChoisirNbAlea*[2]{0.55}{\YrandMax}[\YrandA]%
\ChoisirNbAlea*[2]{\YrandMin}{0.45}[\YrandB]%

\begin{GraphiqueTikz}
  [x=0.075cm,y=7.5cm,Xmin=0,Xmax=150,Xgrille=10,Xgrilles=5,
  Ymin=0,Ymax=1,Ygrille=0.1,Ygrilles=0.05]
  \TracerAxesGrilles[Dernier,Elargir=2.5mm]{auto}{auto}
  \DefinirCourbeInterpo[Couleur=red,Trace,Nom=fonctiontest,Tension=0.75]
  {(0,\YrandA)(40,\YrandMin)(90,\YrandMax)(140,\YrandB)}
  \TrouverAntecedents[Aff=false,Nom=ANTECED]{fonctiontest}{0.5}
  \PlacerAntecedents[Couleurs=blue/teal,Traits]{fonctiontest}{0.5}
  \RecupererAbscisse{(ANTECED-1)}[\Aalpha]
  \RecupererAbscisse{(ANTECED-2)}[\Bbeta]
  \RecupererAbscisse{(ANTECED-3)}[\Cgamma]
\end{GraphiqueTikz}

Les solutions de $f(x)=\num{0.5}$ sont, par lecture graphique :
$\begin{cases}
  \alpha \approx \ArrondirNum[0]{\Aalpha} \\
  \beta \approx \ArrondirNum[0]{\Bbeta} \\
  \gamma \approx \ArrondirNum[0]{\Cgamma}
\end{cases}$.
```



Les solutions de $f(x) = 0{,}5$ sont, par lecture graphique : $\begin{cases} \alpha \approx 4 \\ \beta \approx 65 \\ \gamma \approx 121 \end{cases}$.

12.5 Tableaux de valeurs

Les commandes suivantes permettent de générer un tableau de valeurs pour une fonction xint, avec un choix de présentation horizontale ou verticale.

— `\CalcTableauValeurs` génère uniquement le *corps* du tableau, stocké dans une macro — l'utilisateur gère ensuite la mise en forme avec son environnement préféré (`tblr`, `NiceTabular`, `tabularray`...) ;

— `\ConstruireTableauValeurs` fait tout en un avec `tabularray` (nécessite de ce fait `\usepackage{tabularray}` !).

12.5.1 Génération du corps

```
%horizontal (défaut)
\CalcTableauValeurs[clés]<nom/expr>{valeurs}{nomMacro}

%vertical (version étoilée)
\CalcTableauValeurs*[clés]<nom/expr>{valeurs}{nomMacro}
```

— `<nom/expr>` : nom de fonction xint (défaut) ou expression directe si `Expr=true` ;

— `{valeurs}` : liste des valeurs de `x` (syntaxe `\foreach` : `{-2,-1,0,1,2}` ou `{-2,-1,...,2}`) ;

— `{nomMacro}` : nom de la macro de sortie (sans \) — crée `\nomMacro` (le body) et `\nomMacronbvals` (le nombre de colonnes/lignes) ;

— `Arrondi` : précision de l'arrondi (`2` par défaut) ;

— `NomFct` : label de la fonction dans le tableau (`f` par défaut) ;

— `NomVar` : label de la variable dans le tableau (`x` par défaut) ;

— `Expr` : booléen (`false` par défaut) — si `true`, `<...>` est une expression directe avec `x` comme variable.

```
%usepackage{tabularray}
\GenererFonction<f>{x^2+1}

%horizontal
\CalcTableauValeurs[Arrondi=1]<f>{-2,-1,0,1,2}[montab]

\begin{tblr}%
  [expand=\montab]%
  {%
    colspec={*{\montabnbvals}{c}},hlines,%
    row{1}={bg=gray!25,font=\bfseries}%
  }
  \montab
\end{tblr}
```

x	-2	-1	0	1	2
$f(x)$	5	2	1	2	5

```
%vertical + expression directe
\CalcTableauValeurs*%
  [Expr,NomFct=g,NomVar=t]%
  <x^2+1>{0,1,2,3}[montabv]

\begin{tblr}%
  [expand=\montabv]%
  {%
    colspec={*{2}{c}},hlines,%
    row{1}={bg=blue!15,font=\bfseries}%
  }
  \montabv
\end{tblr}
```

t	$g(t)$
0	1
1	2
2	5
3	10

12.5.2 Tableau complet avec tabularray

Cette commande nécessite le chargement préalable de `tabularray` — un avertissement est émis sinon.

```
%horizontal (défaut, Presentation=H)
\ConstruireTableauValeurs[clés]<options tblr>{nom/expr}{valeurs}

%vertical (Presentation=V)
\ConstruireTableauValeurs[clés,Presentation=V]<options tblr>{nom/expr}{valeurs}
```

— `[clés]` : mêmes clés que `\CalcTableauValeurs` ;

— `<options tblr>` : options passées directement à l'environnement ;

— `Presentation` : `H` (horizontal, défaut) ou `V` (vertical) ;

— `\tdvnbvals` : macro exposée contenant le nombre de colonnes — utile pour `colspec`.

```
\GenererFonction<f>{x^2+1}

%horizontal
\ConstruireTableauValeurs%
  [Arrondi=2]%
  <colspec={*{\tdvnbvals}{c}},hlines,row{1}={bg=gray!25,font=\bfseries}>%
  {f}{-2,-1,0,1,2}

%vertical
\ConstruireTableauValeurs%
  [Arrondi=2,Presentation=V]%
  <colspec={*{2}{c}},hlines,row{1}={bg=gray!25,font=\bfseries}>%
  {f}{0,0.5,...,2}
```

x	-2	-1	0	1	2
$f(x)$	5	2	1	2	5

x	$f(x)$
0	1
0,5	1,25
1	2
1,5	3,25
2	5

13 Liste des commandes

13.1 Environnement, axes et grilles

13.2 Courbes, suites

13.3 Outils graphiques

13.4 Intégrales, probabilités, statistiques

13.5 Courbes polaires, paramétriques

13.6 Coniques

13.7 Commandes diverses

14 Quelques commandes liées à pgfplots

14.1 Introduction

Pour des graphiques avec des fenêtres d'affichage *particulières*, il est fort possible que les commandes *classiques* de `tkz-grapheur` coincent, avec notamment des `dimension too large`...

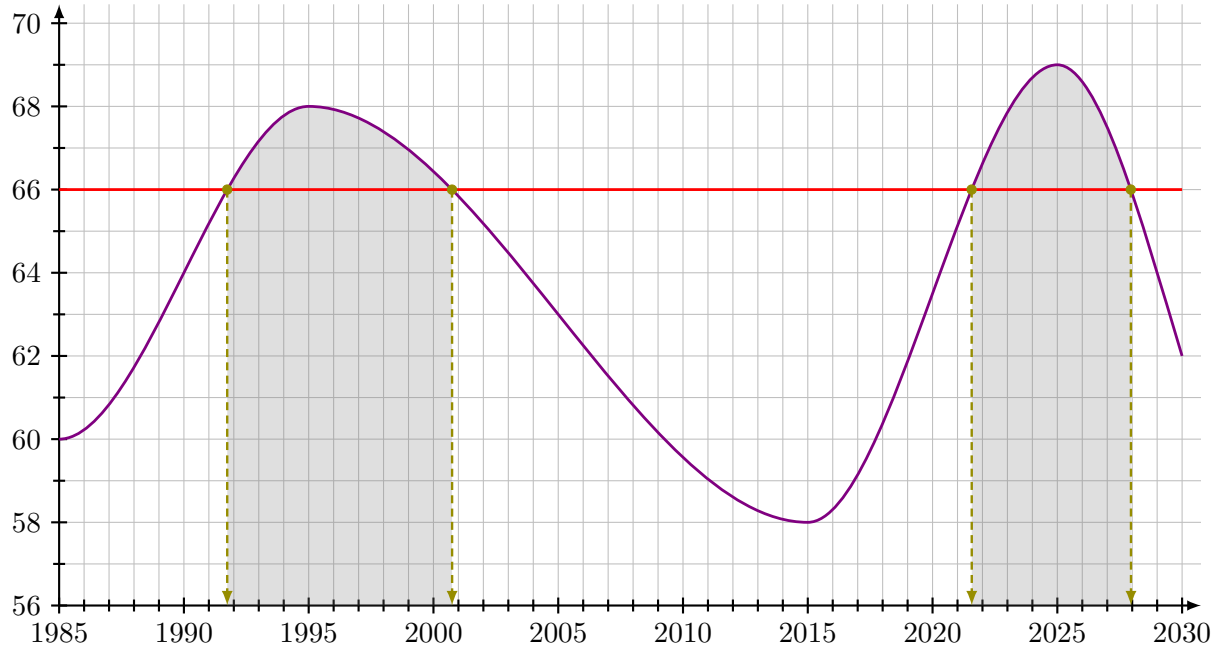
Dans ce cas, il est possible d'utiliser plutôt l'environnement `axis` de `pgfplots`, qui, de plus, permet *souvent* de pallier ce problème *interne*...
`tkz-grapheur` ne fournit pas d'environnement dédié pour la création de la fenêtre, mais quelques commandes spécifiques ont été intégrées pour certains points, avec un fonctionnement assez semblable (donc se référer aux paragraphes précédents) à celui des commandes *classiques*.

14.2 Macros spécifique pgfplots/axis

```
%déterminer l'intersection de deux objets préalablement définis via [name path]
\findintersectionspgf[base nom nœuds]{objet1}{objet2}[macro nb total]

%extraction (globale, non limitée à l'environnement) et stockage de coordonnées
\gextractxnodepgf{nœud}[\myxcoord]
\gextractynodepgf{nœud}[\myycoord]
\gextractxynodepgf{nœud}[\myxcoord][\myycoord]

%domaine entre courbes
\fillbetweencurvespgf[options tikz]{courbe1}{courbe2}<options soft domain>

%splines cubiques
\addplotspline(*)[options tikz]<coeffs>{liste des points support}[\monspline]
```

14.3 Exemple illustré

```
%\usepackage{alphalph}

\begin{tikzpicture}
  \begin{axis}%
    [%
    axis y line=center,axis x line=middle,                                    %axes
    axis line style={line width=0.8pt,-latex},
    x=0.33cm,y=0.55cm,xmin=1985,xmax=2030,ymin=56,ymax=70,                    %unités
    grid=both,xtick distance=5,ytick distance=2,                              %grillep
    minor x tick num=4,minor y tick num=1,                                    %grilles
    extra x ticks={1985},extra x tick style={grid=none},                      %origx
    extra y ticks={56},extra y tick style={grid=none},                        %origy
    x tick label style={/pgf/number format/.cd,use comma,1000 sep={}},        %année
    major tick length={2*3pt},minor tick length={1.5*3pt},                    %grads
    every tick/.style={line width=0.8pt},enlargelimits=false,                 %style
    enlarge x limits={abs=2.5mm,upper},enlarge y limits={abs=2.5mm,upper},    %élargir
    ]
    %spline + y=66
    \addplot[name path global=eqtest,mark=none,red,line width=1.05pt,domain=1985:2030]
    ↪  {66} ;
    \def\LISTETEST{1985/60/0§1995/68/0§2015/58/0§2025/69/0§2030/62/-2}
    \addplotspline*[line width=1.05pt,violet,name path
    ↪  global=splinecubtest]{\LISTETEST}[\monsplineviolet]
    %recherche d'antécédents
    \findintersectionspgf[MonItsc]{eqtest}{splinecubtest}
    %extraction des coordonnées
    \gextractxnodepgf{(MonItsc-1)}[\xMonItscA]
    \gextractxnodepgf{(MonItsc-2)}[\xMonItscB]
    \gextractxnodepgf{(MonItsc-3)}[\xMonItscC]
    \gextractxnodepgf{(MonItsc-4)}[\xMonItscD]
    %visualisation
    \xintFor* #1 in {\xintSeq{1}{4}}\do{%
      \draw[line width=0.9pt,densely dashed,olive,->,>=latex] (MonItsc-#1) -- (\csname
      ↪  xMonItsc\AlphAlph{#1}\endcsname,56) ;
      \filldraw[olive] (MonItsc-#1) circle[radius=1.75pt] ;
    }
    %intégrale
    \path [name path=xaxis] (1985,56) -- (2030,56);
    \fillbetweencurvespgf{splinecubtest}{xaxis}<domain={\xMonItscB:\xMonItscA}>
    \fillbetweencurvespgf{splinecubtest}{xaxis}<domain={\xMonItscD:\xMonItscC}>
  \end{axis}
\end{tikzpicture}

Les solutions de $f(x)=66$ sont d'environ \ArrondirNum*[0]{\xMonItscA} \&\
↪  \ArrondirNum*[0]{\xMonItscB} \&\ \ArrondirNum*[0]{\xMonItscC} \&\
↪  \ArrondirNum*[0]{\xMonItscD}.
```



Les solutions de $f(x) = 66$ sont d'environ 1992 & 2001 & 2022 & 2028.

15 Historique

```
0.30f : Ajouts en statistiques (barres/bâtons/histogrammes/moustaches/camembert)
0.30e : Meilleure gestion de la taille du graphique
-----   Améliorations dans les valeurs interdites
-----   Courbe de niveaux (expérimental)
0.30d : Ajout d'une clé pour créer une liste d'abscisse + exploitation de listes
-----   Famille de courbes (expérimental)
-----   Définition globales de fonctions (hors environnement ou via clé [DefGlobale])
-----   Tableau de valeurs (expérimental)
-----   Graphiques semi-log ou log-log (expérimental)
-----   Nuage de points avec transformations et/ou par csv (expérimental)
-----   Macros de calculs pour des évolutions en %, des sommes de suites
-----   Courbe implicite (très expérimental, lua)
0.30c : Refonte du fonctionnement avec séparation commandes [en/fr]
-----   Transformations, Taylor (expérimental)
-----   Ajout de lois de probabilités (continues + discrètes)
-----   Outils numériques
0.30b : Dérivée/primitive (expérimental)
-----   Nuage suites
-----   Courbes paramétrées, polaires
-----   Voisinages (expérimental)
-----   Valeurs interdites (expérimental)
-----   Coniques (expérimental)
0.30a : Création des nœuds (fenêtre + axes)
0.2.9 : Ajout de thèmes de couleurs pour les grilles
0.2.7 : Possibilité de spécifier les dimensions du graphique (en test)
0.2.6 : Inegalité linéaire (en test)
0.2.5 : Interpolation de Lagrange + améliorations
0.2.4 : Clé [StyleTrace] pour des pointillés par exemple
0.2.3 : Bugfix avec une longueur
0.2.0 : Méthode alternative des splines cubiques + commandes auxiliaires pgfplots
0.1.9 : Correction d'un bug avec la détermination d'unités
0.1.8 : Courbes ECC/FCC + Toile récurrence + Points discontinuité + HistoBinom
0.1.7 : Méthodes intégrales avec des splines
0.1.6 : Asymptote verticale + Méthodes intégrales (géom + Monte Carlo)
0.1.5 : Correction d'un bug sur les rajouts de valeurs + Nœud pour une image + [en] version !
0.1.4 : Placement de texte
0.1.3 : Ajout de régressions avec le package xint-regression
0.1.2 : Droites + Extremums
0.1.1 : Densité loi normale et khi deux + Marquage points + Améliorations
0.1.0 : Version initiale
```