tkz-grapheur [fr]

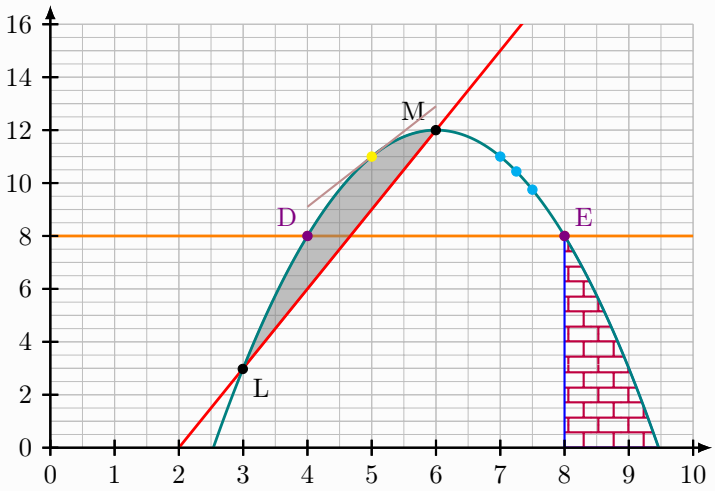
Un système de grapheur,

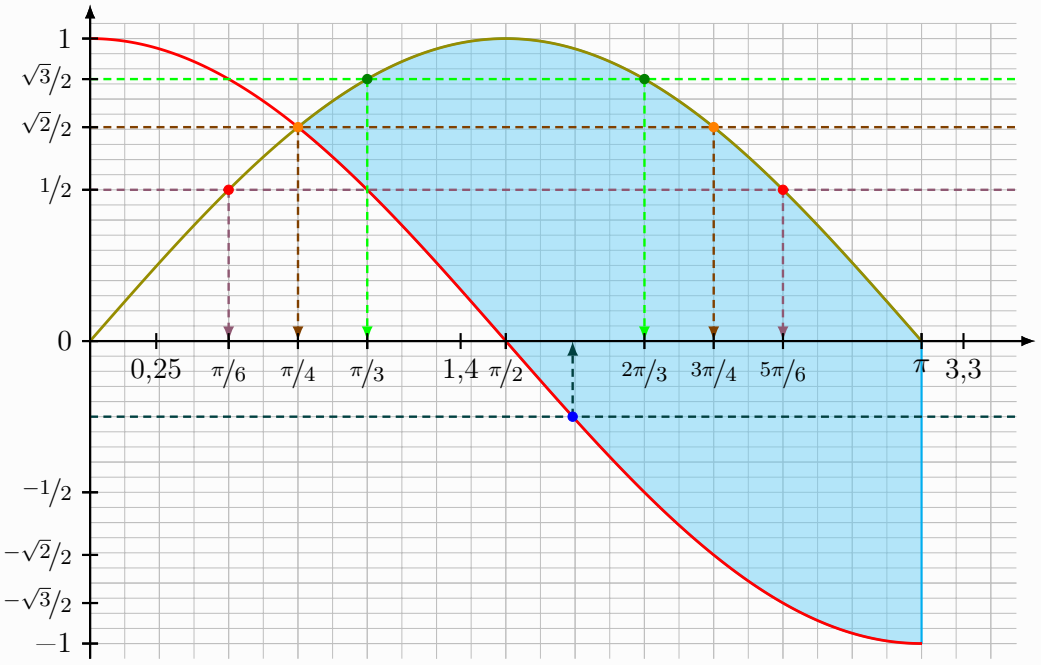
basé sur TikZ et xint.

Version 0.2.5 - 30/05/2025

Cédric Pierquet
c pierquet - at - outlook . fr
https://forge.apps.education.fr/pierquetcedric/packages-latex

---





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, il *suffit* :

— de déclarer les paramètres de la fenêtre graphique ;

— d'afficher grille/axes/graduations ;

— de déclarer les fonctions ou les courbes d'interpolation ;

— de déclarer éventuellement des points particuliers ;

— de placer un nuage de points.

Il sera ensuite possible :

— de tracer des courbes ;

— de déterminer graphiquement des images ou des antécédents ;

— de rajouter des éléments de dérivation (tangentes) ou d'intégration (domaine) ;

— de tracer une droite d'ajustement linéaire ou la courbe d'un autre ajustement ;

— ...

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

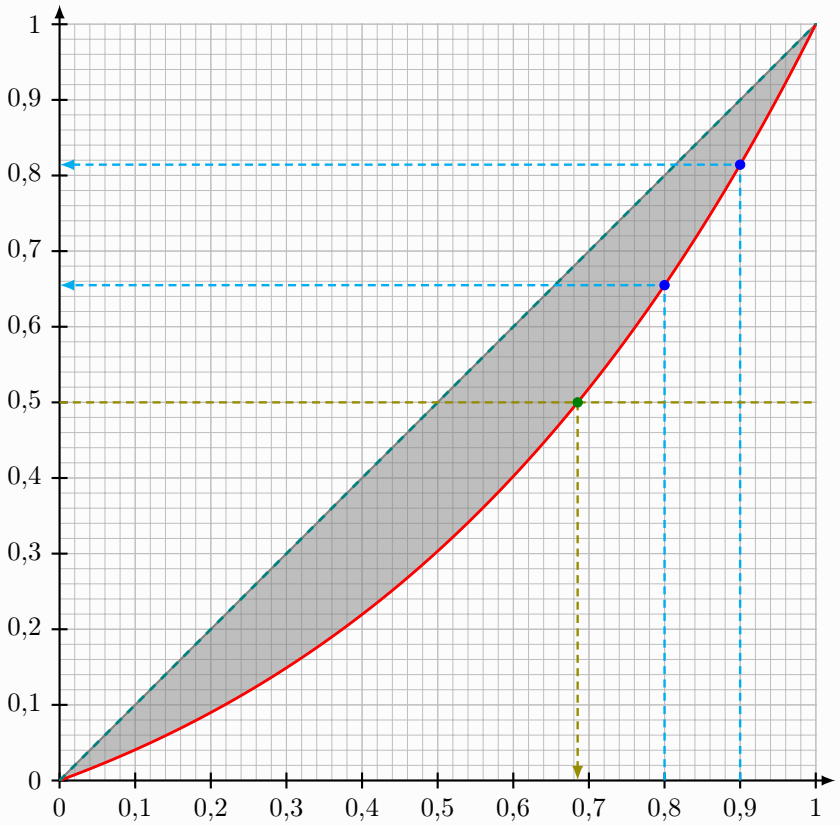
Il est possible, dû aux calculs (multiples) effectués en interne, que le temps de compilation soir un peu *allongé*.

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 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,
  Xgrilles/.store in=\pflgrillexs,Xgrilles/.default=0.5,Xgrilles=0.5,
  Ygrille/.store in=\pflgrilley,Ygrille/.default=1,Ygrille=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`).

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

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*) ;

— `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 !

2.3 Grilles et axes

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

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

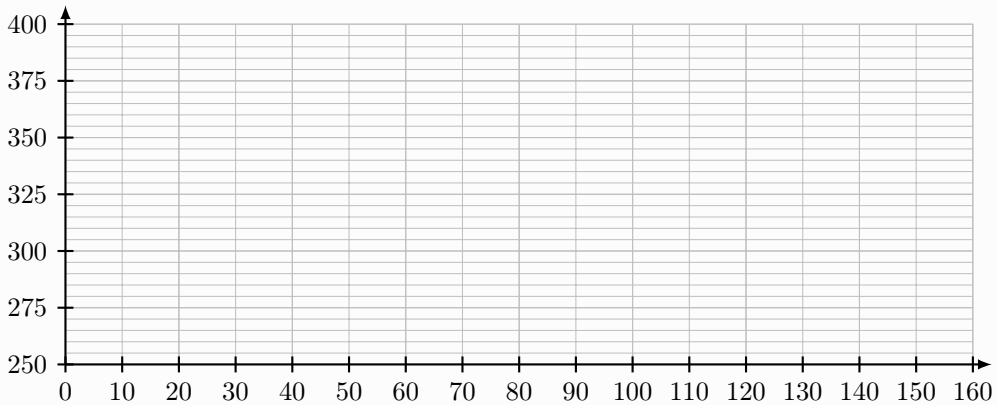
Les `[clés]`, optionnelles, disponibles sont :

- `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.

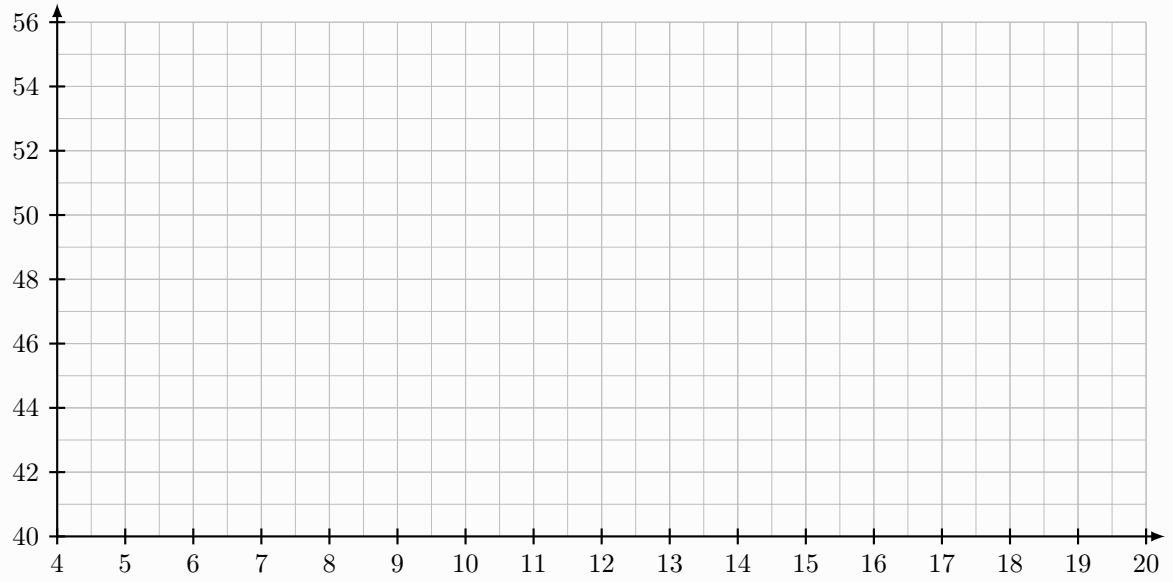
Concernant la clé `Format`, elle permet de spécifier un paramétrage spécifique pour les valeurs des axes. Elle peut être donnée sous la forme `fmt` pour un formatage combiné, ou sous la forme `fmtX/fmtY` pour différencier le formatage.
Les options possible sont :

- `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}
```

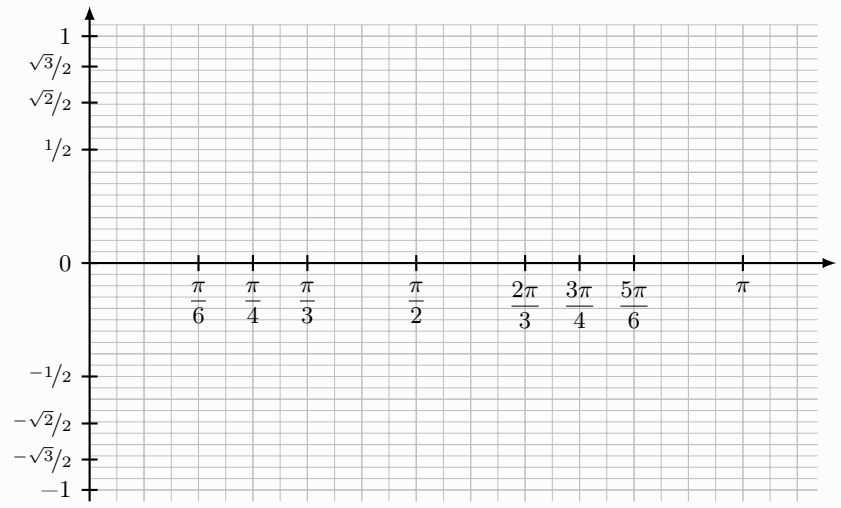


```
\begin{GraphiqueTikz}%
  [x=0.9cm,y=0.425cm,Xmin=4,Xmax=20,Origx=4,
  Ymin=40,Ymax=56,Ygrille=2,Ygrilles=1,Origy=40]
  \TracerAxesGrilles[Elargir=2.5mm,Police=\small]{4,5,...,20}{40,42,...,56}
\end{GraphiqueTikz}
```



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.

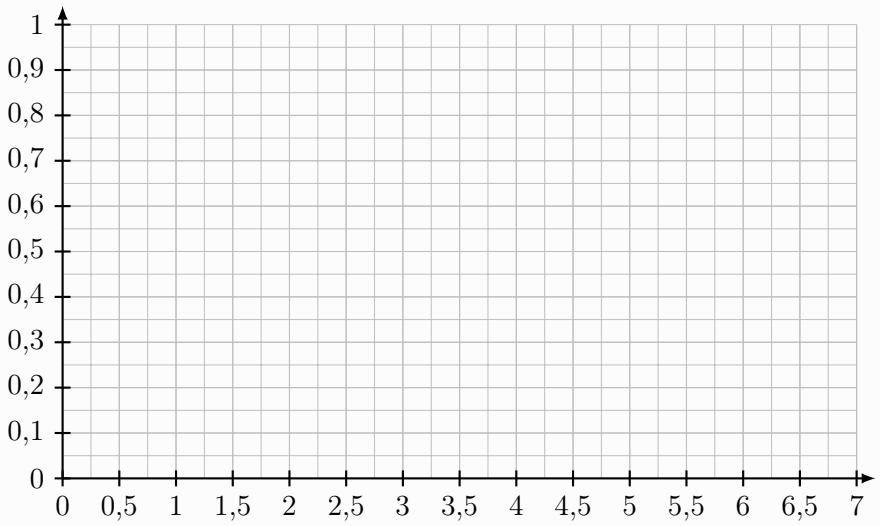
```
\begin{GraphiqueTikz}%
  [x=2.75cm,y=3cm,
  Xmin=0,Xmax=3.5,Xgrille=pi/12,Xgrilles=pi/24,
  Ymin=-1.05,Ymax=1.05,Ygrille=0.2,Ygrilles=0.05]
  \TracerAxesGrilles[Elargir=2.5mm,Format=dtrig/nsqrt,Police=\footnotesize]%
    {pi/6,pi/4,pi/3,pi/2,2*pi/3,3*pi/4,5*pi/6,pi}
    {0,sqrt(2)/2,1/2,sqrt(3)/2,1,-1,-sqrt(3)/2,-1/2,-sqrt(2)/2}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



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}
```

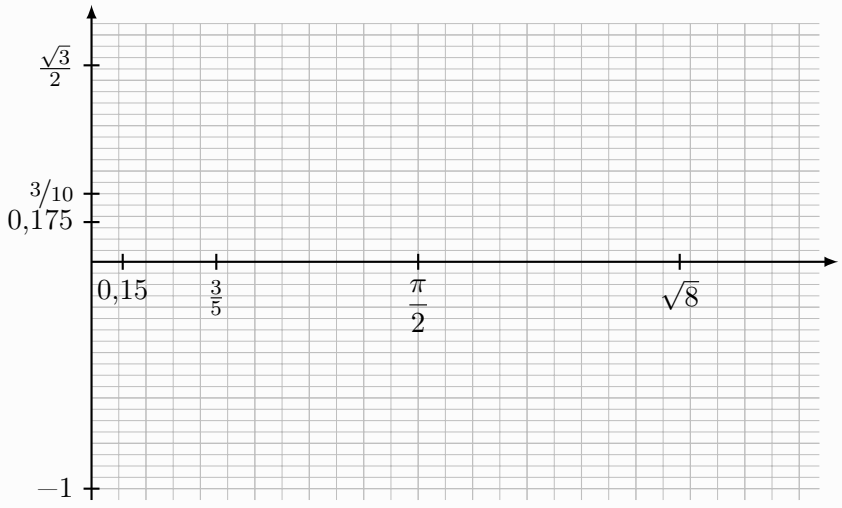
Les `[clés]`, optionnelles, disponibles sont :

— `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}
```



3 Commandes spécifiques de définitions

3.1 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) ;

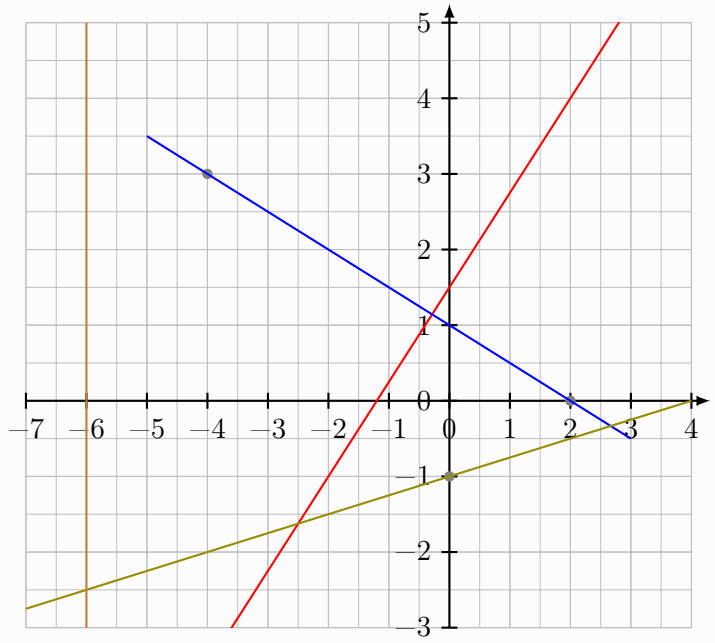
— `Debut` : début du tracé (`\pflxmin` par défaut) ;

— `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}
```



3.2 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.

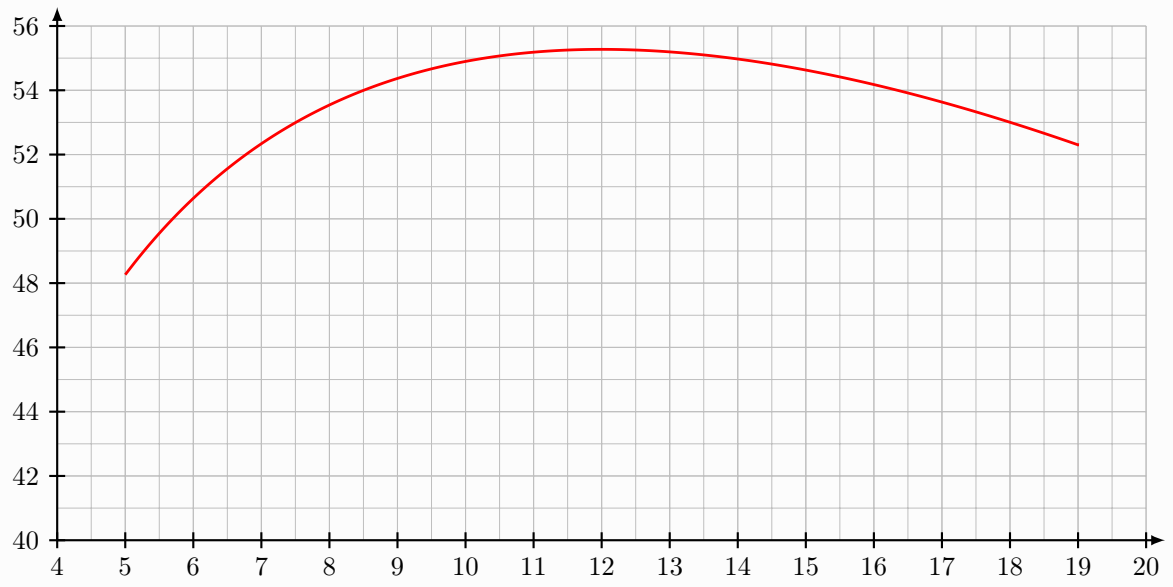
Il existe également une commande pour tracer la courbe d'une fonction précédemment définie.

```
%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).

```
\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 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.

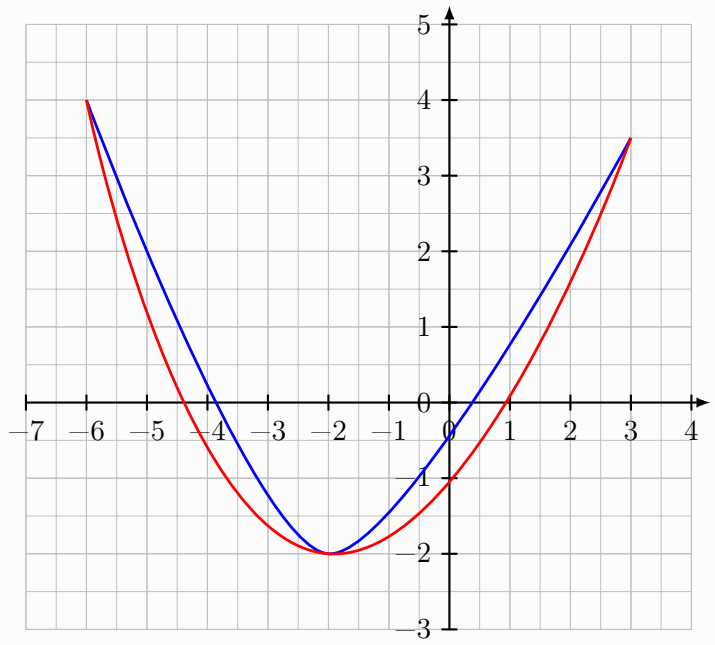
```
%dans l'environnement GraphiqueTikz
\DefinirCourbeInterpo[clés]{liste des points support}
\TracerCourbeInterpo[clés]{liste des points support}
```

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}
```



3.4 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]
```

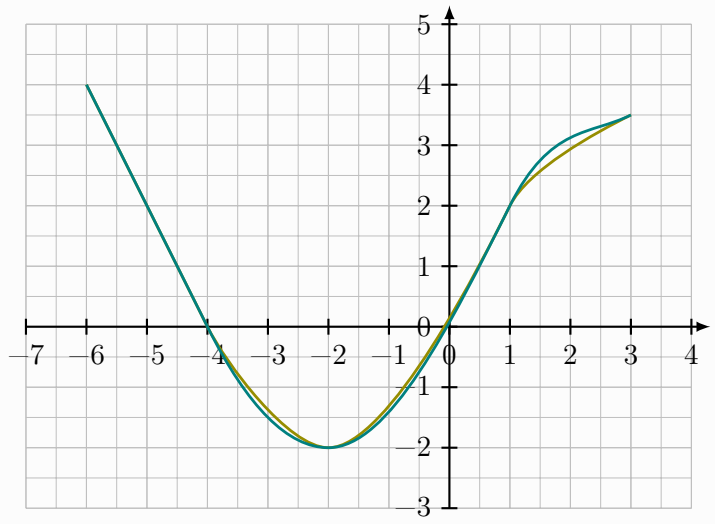
Les `[clés]` pour la définition ou le tracé, optionnelles, disponibles sont :

— `Nom` : nom de la courbe d'interpolation (important pour la suite !) ;
— `Coeffs` : modifier (voir la documentation de ProfLycee [5] 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

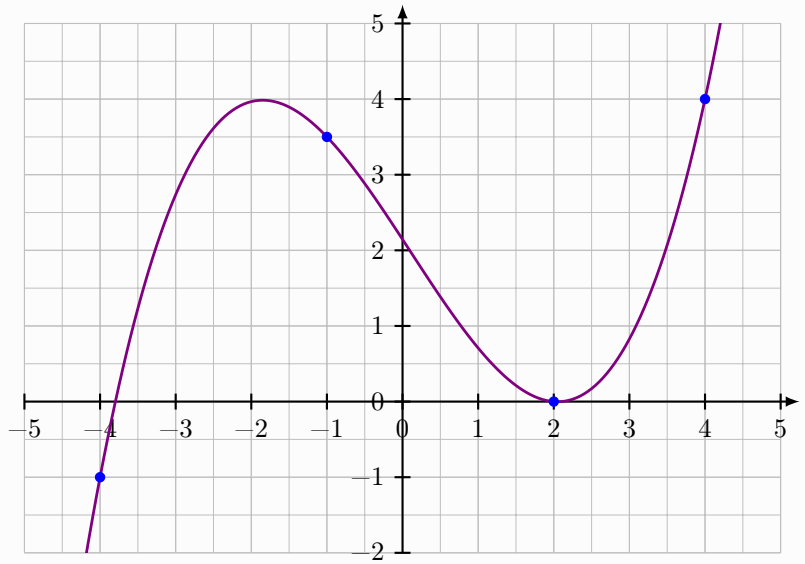
3.5 Définir/tracer une courbe d'interpolation (Lagrange)

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

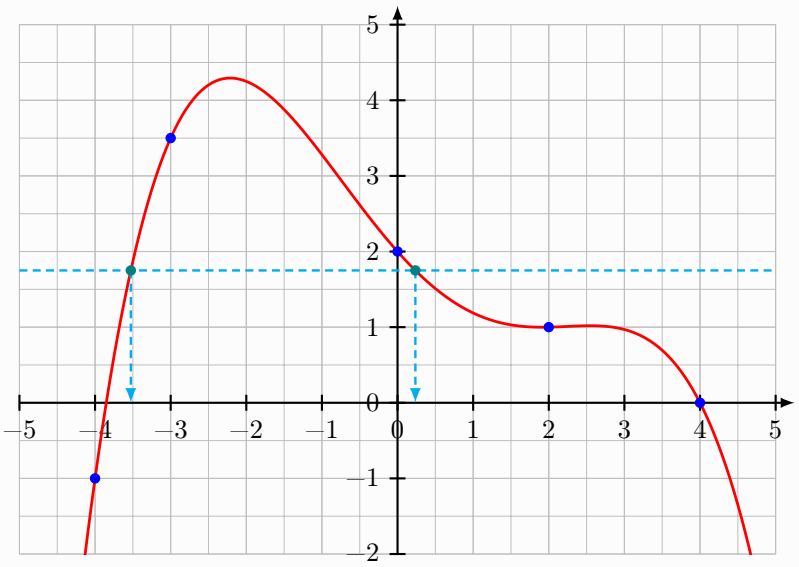
```
%dans l'environnement GraphiqueTikz
\GenererPolynomeLagrange[nom fonction]{liste X}{liste Y}
\TracerCourbe[clés]{f(x)}
```

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.

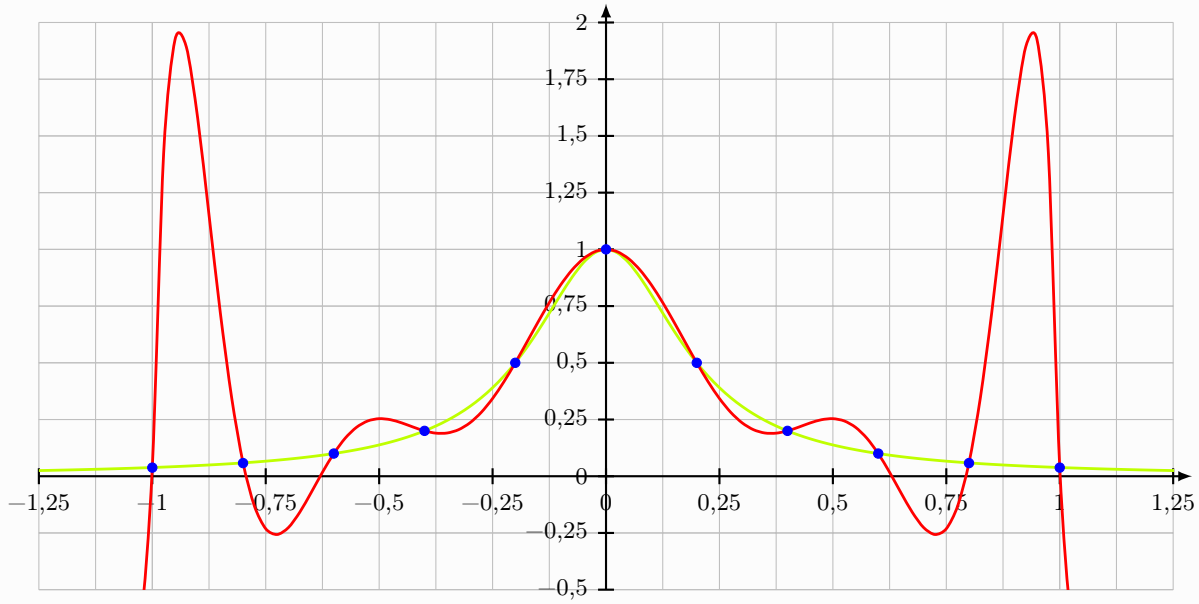
```
\begin{GraphiqueTikz}[Xmin=-5,Xmax=5,Ymin=-2,Ymax=5]
  \TracerAxesGrilles[Elargir=2.5mm,Police=\small]{auto}{auto}
  \GenererPolynomeLagrange{-4,-1,2,4}{-1,3.5,0,4}
  \TracerCourbe[Couleur=violet]{polylagrange(x)}
  \MarquerPtsLagrange*[Couleur=blue]{-4,-1,2,4}{-1,3.5,0,4}
\end{GraphiqueTikz}
```



```
\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}
```



3.6 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}
```

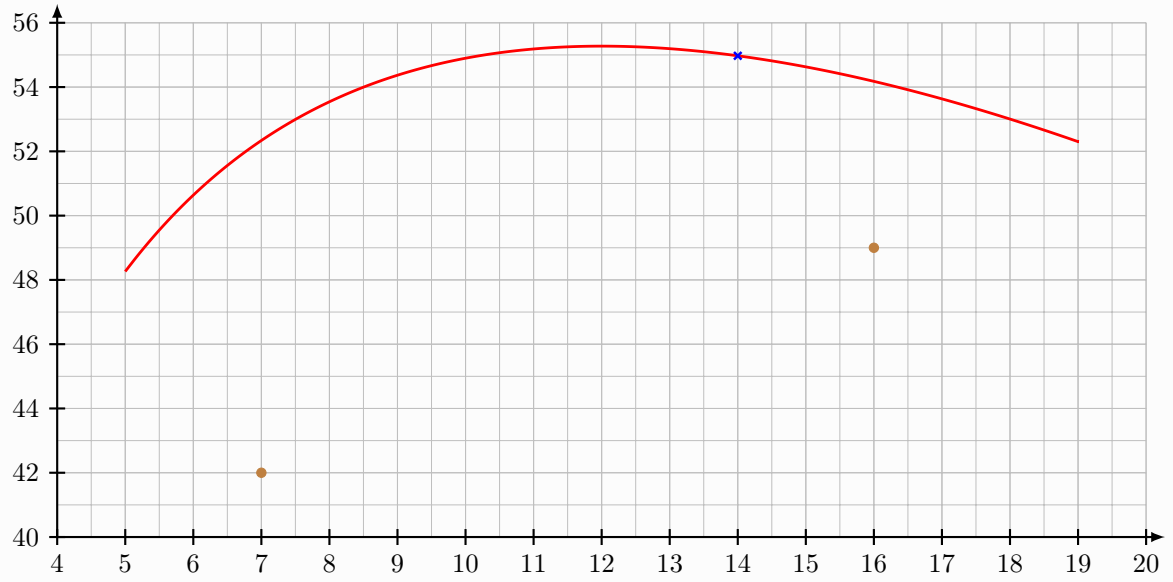
Les `[clés]`, optionnelles, disponibles sont :

— `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}
```



3.7 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),...` ;

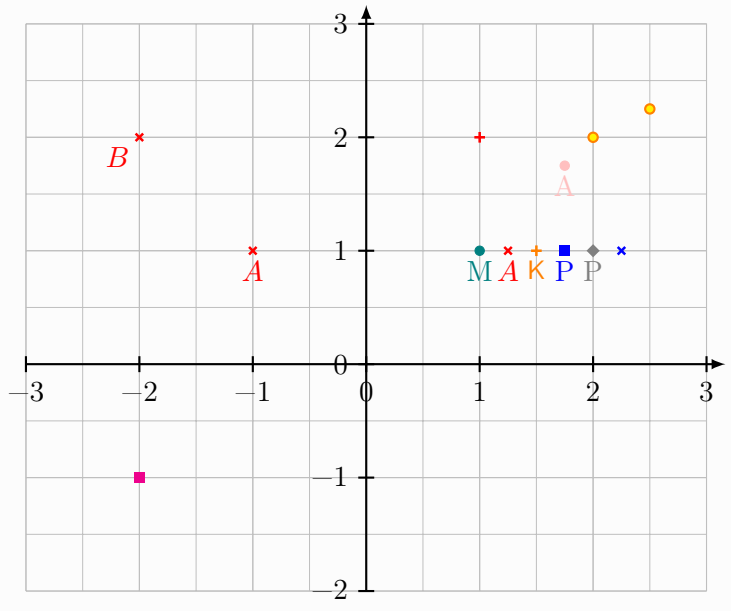
— sinon, la liste est à donner sous la forme `(ptA)/labelA/poslabelA,...`.

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[Couleur=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)/P/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}
```



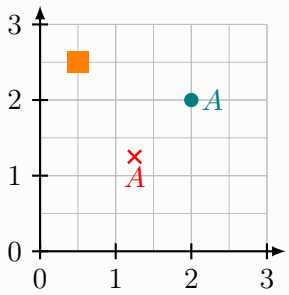
À noter qu'il est également possible de modifier la taille des marques `o/x/+/c` via les `[clés]` :

— `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}
```



3.8 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.

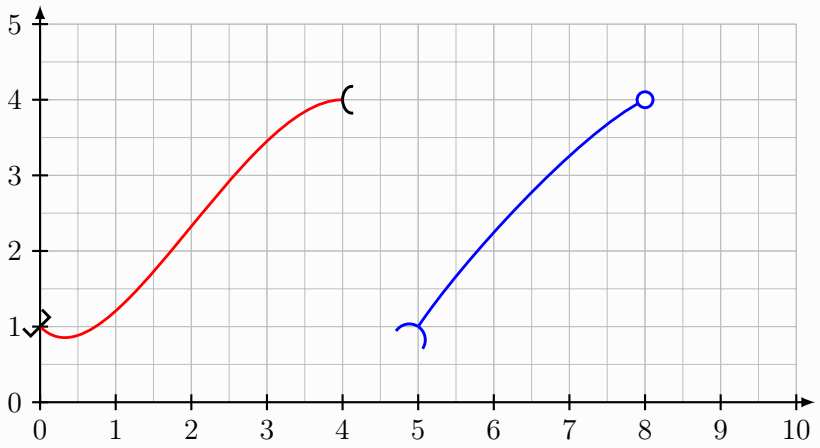
```
%dans l'environnement GraphiqueTikz
\AfficherPtsDiscont[clés]{liste}
```

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}
```



3.9 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]
```

3.10 Placer du texte

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

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

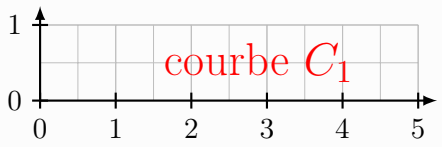
Les `[clés]` disponibles sont :

— `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` ;

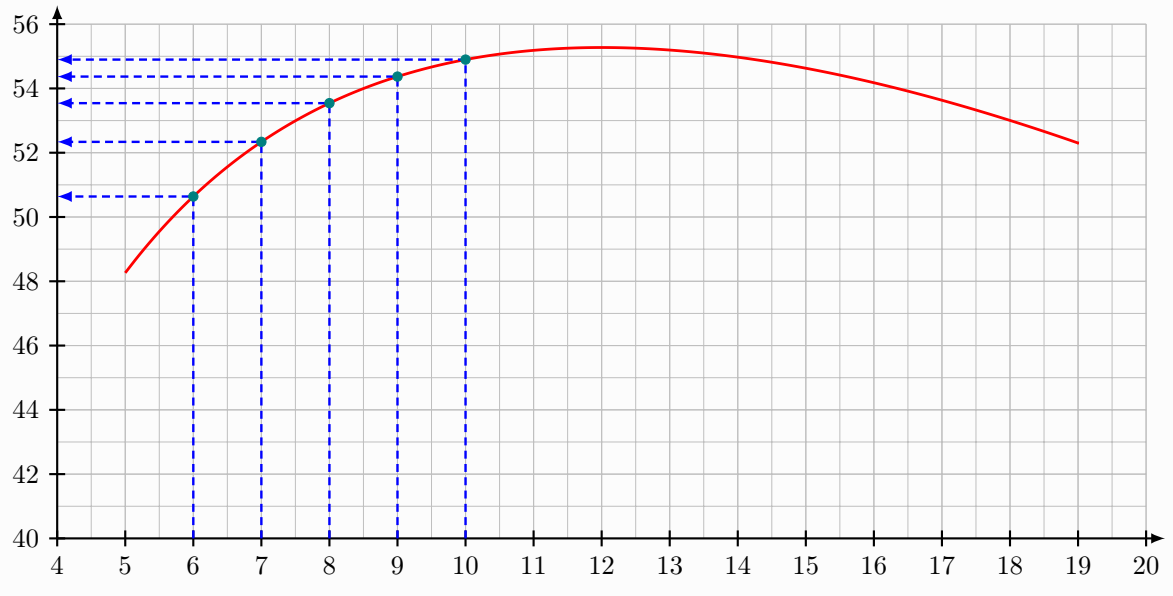
— `Spline` : booléen pour préciser que la courbe utilisée est définie comme un spline (`false` par défaut).

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 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.

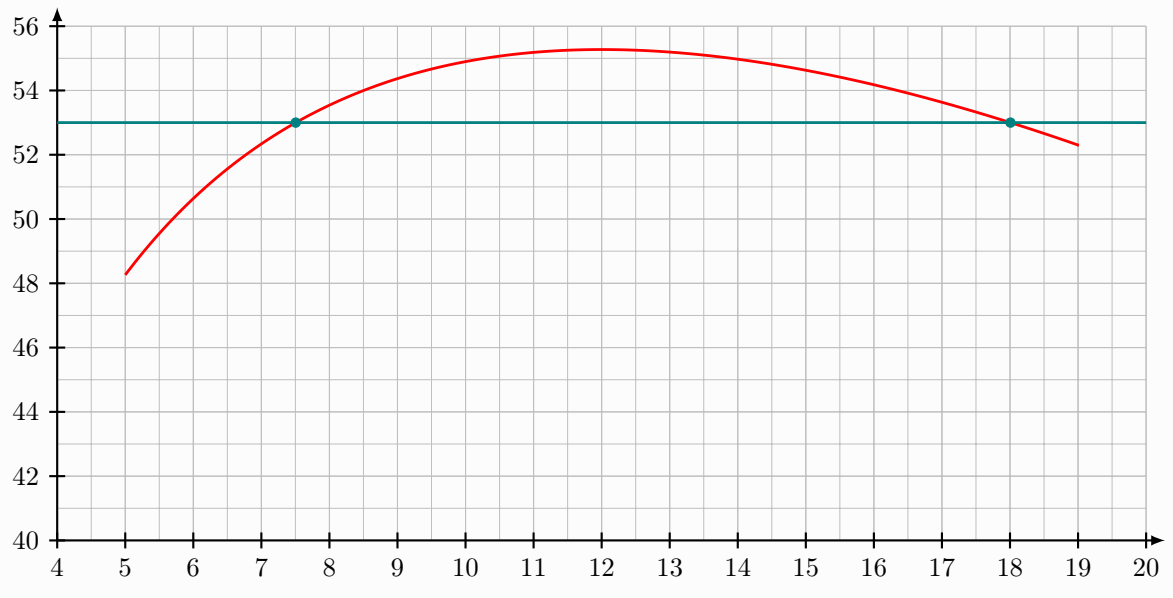
```
%dans l'environnement GraphiqueTikz
\TrouverAntecedents[clés]{courbe}{k}
```

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

— `Nom` : base du nom des **nœuds** intersection (`S` par défaut, ce qui donnera S-1, S-2, etc) ;
— `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.

```
\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
  \TrouverAntecedents[Couleur=teal,AffDroite,Aff]{cf}{53}
  %les deux antécédents sont aux nœuds (S-1) et (S-2)
\end{GraphiqueTikz}
```



4.3 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.

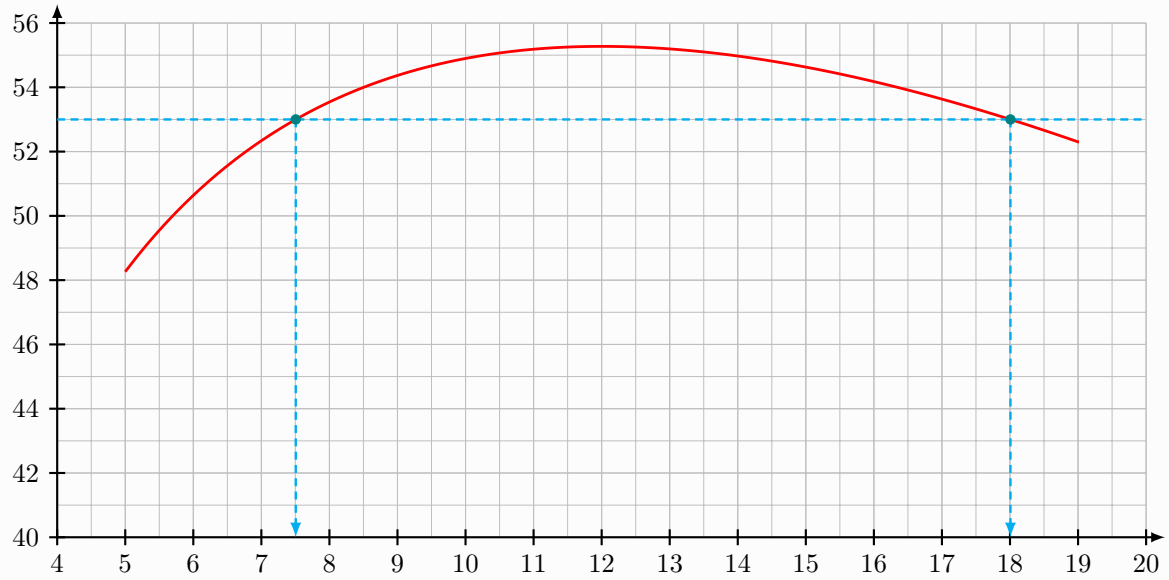
```
%dans l'environnement GraphiqueTikz
\PlacerAntecedents[clés]{courbe}{k}
```

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

4.4 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}
```

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

— `Nom` : base du nom des **nœuds** intersection (`S` par défaut, ce qui donnera S-1, S-2, etc) ;

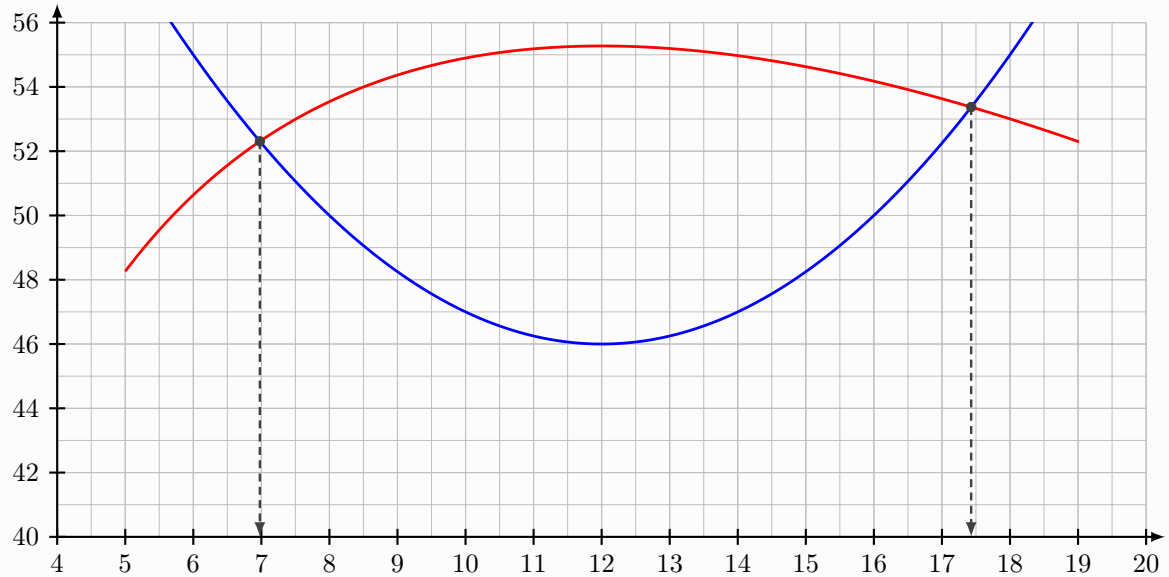
— `Aff` : booléen pour afficher les points (`true` par défaut) ;

— `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.

```
\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 $(17{,}43; 53{,}37)$.

4.5 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 :

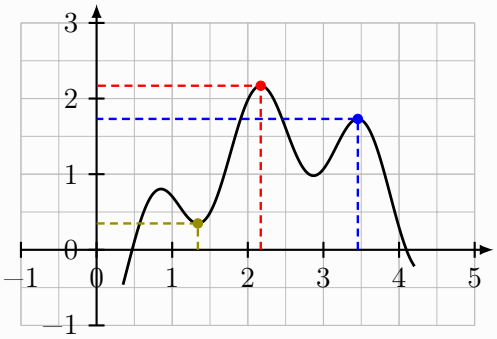
- pas de gestion d'extremums multiples (seul le premier sera traité)...
- pas de gestion d'extremums aux bornes du tracé...
- pas de récupération automatique des paramètres de définition des courbes...
- le temps de compilation peut être plus long...

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

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

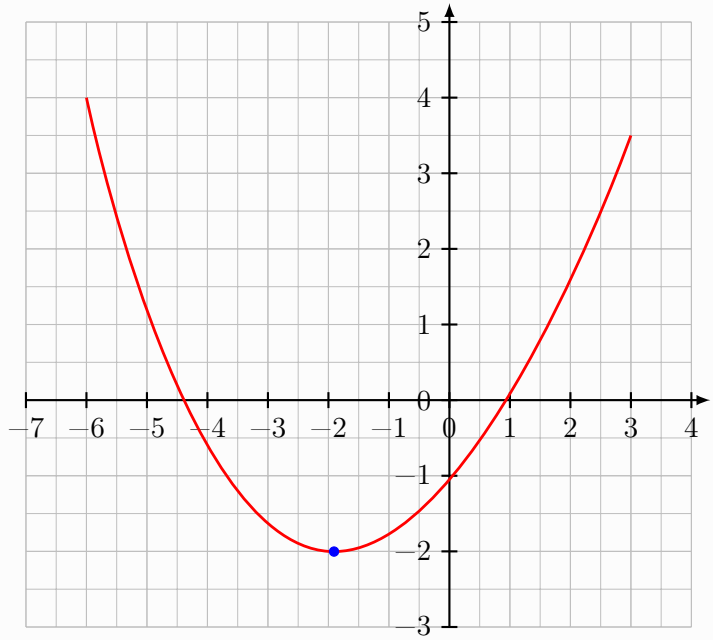
- `Methode` : méthode, parmi `fonction/interpo/spline` pour les calculs (`fonction` par défaut) ;
- `Debut` : début du tracé (`\pflxmin` par défaut) ;
- `Fin` : fin du tracé (`\pflxmax` par défaut) ;
- `Pas` : pas du tracé si `fonction` (il est déterminé *automatiquement* au départ mais peut être modifié) ;
- `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
↪  $x\approx\ArrondirNum{\MonMaxX}$
```



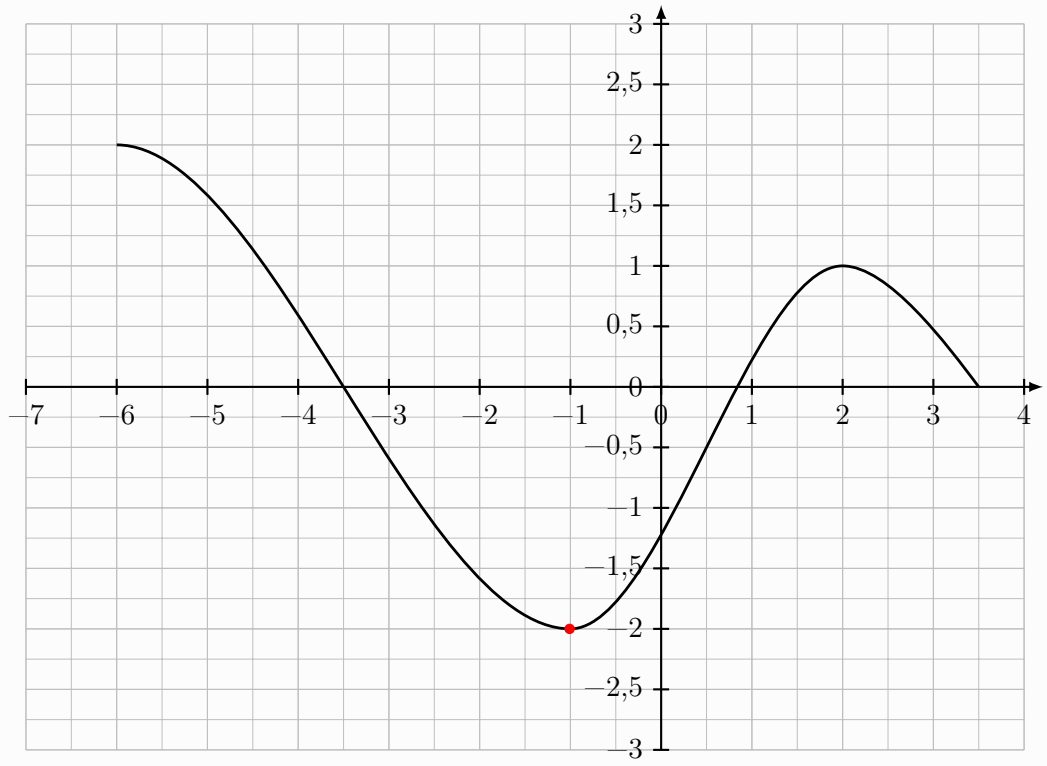
Le maximum est $M \approx 2{,}17$, atteint en $x \approx 2{,}17$

```
\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}
```



4.6 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.

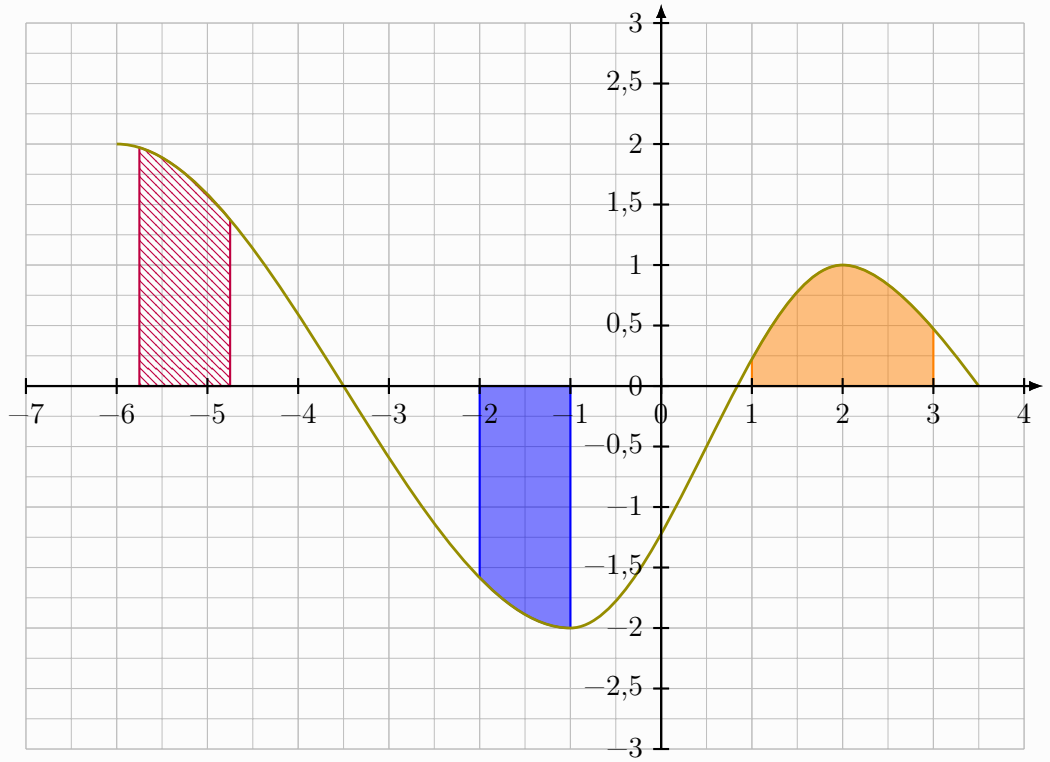
L'argument suivant, optionnel, est la fonction ou la courbe du spline ou la liste de points d'interpolation.

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}
```



4.7 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.

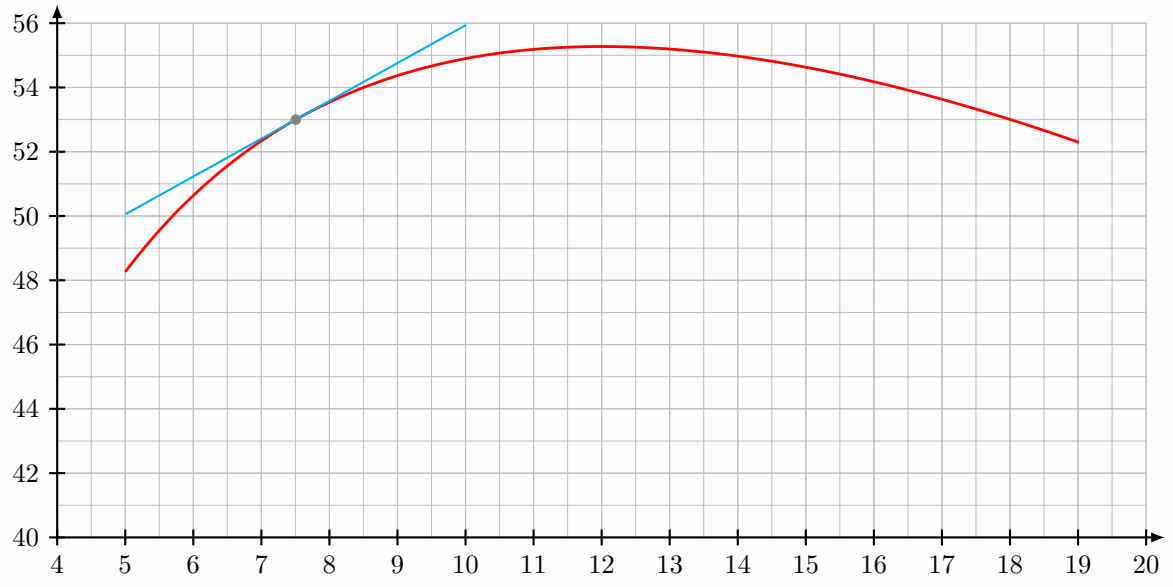
```
%dans l'environnement GraphiqueTikz
\TracerTangente[clés]{fonction ou courbe}{point}<options traits>
```

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

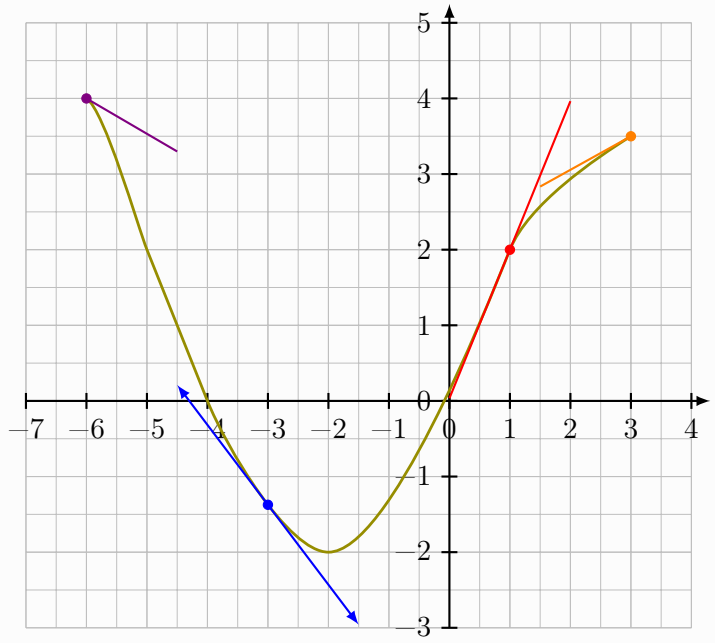
— `Couleurs` = : couleurs des tracés, sous la forme `Couleur` ou `CouleurLigne/CouleurPoint` (`black` par défaut) ;
— `DecG` = : écartement horizontal gauche pour débuter le tracé (`1` par défaut) ;
— `DecD` = : écartement horizontal gauche pour débuter le tracé (`1` par défaut) ;
— `AffPoint` : booléen 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}
```



4.8 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.

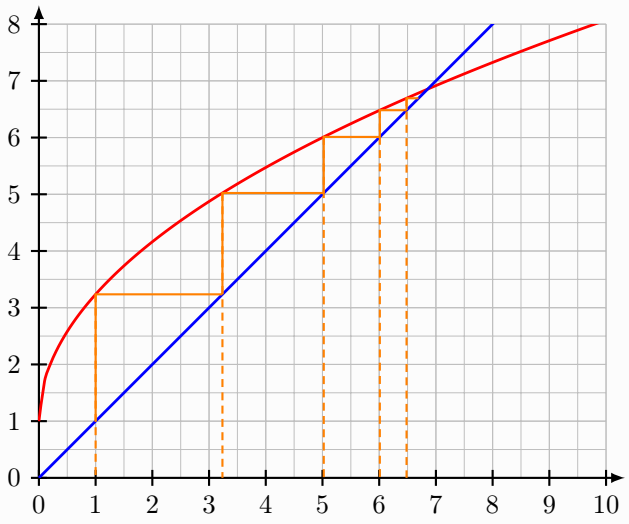
```
%dans l'environnement GraphiqueTikz
\TracerToileRecurrence[clés]{fct ou courbe}
```

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

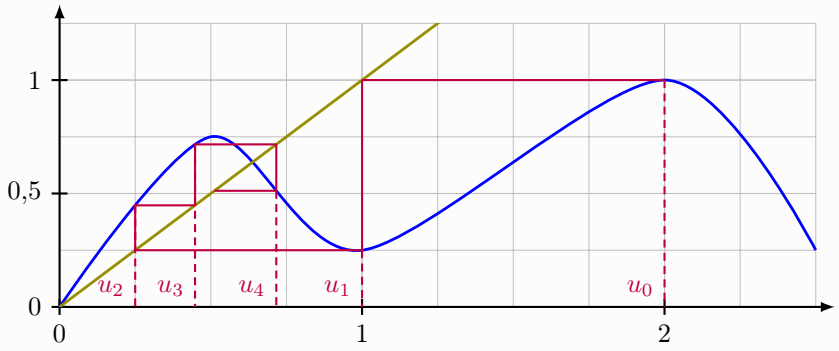
- `Couleur=...` (`black` par défaut) ;
- `Spline=...` (`false` par défaut) pour spécifier qu'une courbe *spline* est utilisée ;
- `No=...` (`0` par défaut) est l'indice initial ;
- `Uno=...` est qui est la valeur du terme initial (à donner obligatoirement !) ;
- `Nom=...` (`u` par défaut) est le nom 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}
```



5 Commandes spécifiques des fonctions de densité

5.1 Loi normale

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

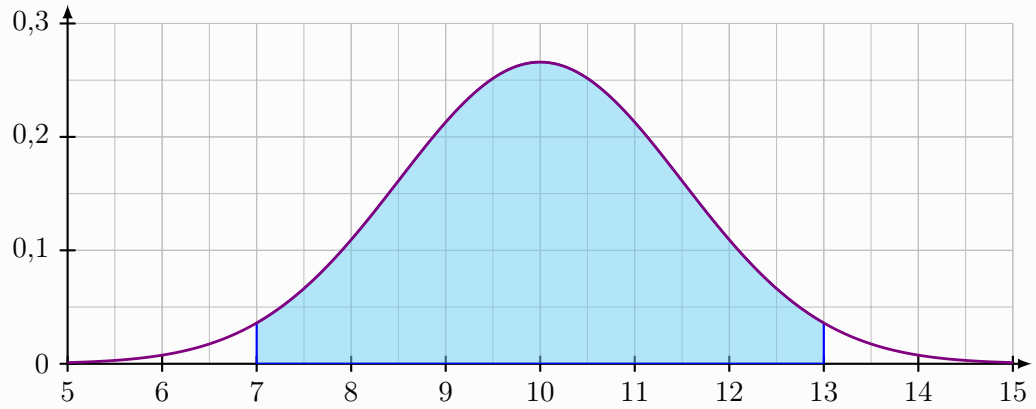
```
%dans l'environnement GraphiqueTikz
\DefinirLoiNormale[clés]<nom fct>{mu}{sigma}
\TracerLoiNormale[clés]{fct(x)}
```

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}
```



5.2 Loi du khi deux

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

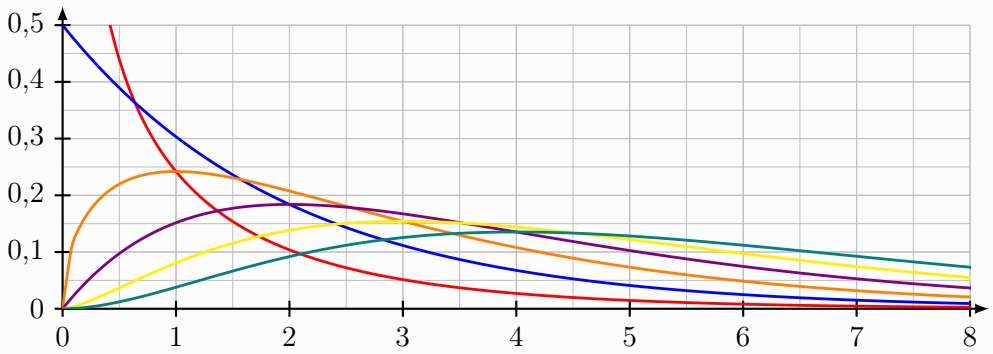
```
%dans l'environnement GraphiqueTikz
\DefinirLoiKhiDeux[clés]<nom fct>{k}
\TracerLoiKhiDeux[clés]{fct(x)}
```

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}
```



5.3 Histogramme pour une 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 :

— `Plage` : plage, sous la forme `a-b` du coloriage éventuel ;

— `CouleurPlage` : couleur de la plage éventuelle ;

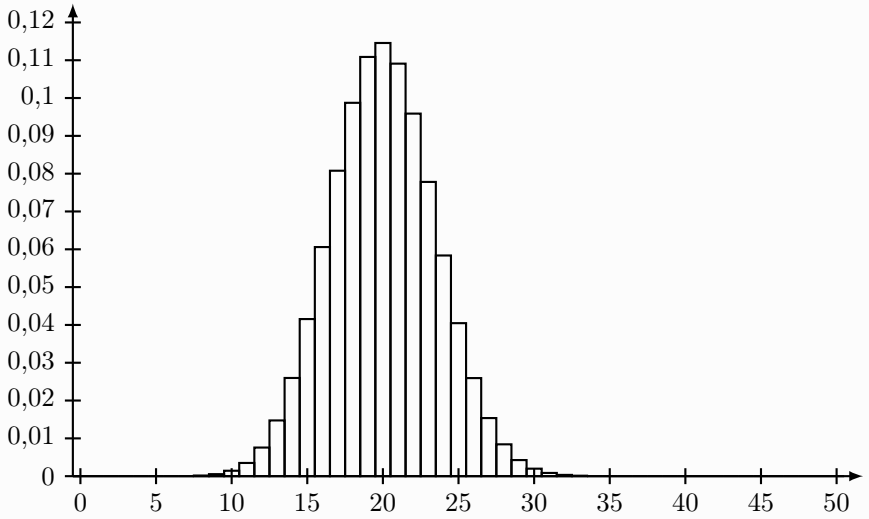
— `ClipX` : restriction de l'axe Ox, sous la forme `a-b` ;

— `AffNormale` : booléen (`true` par défaut) pour rajouter la loi normale ;

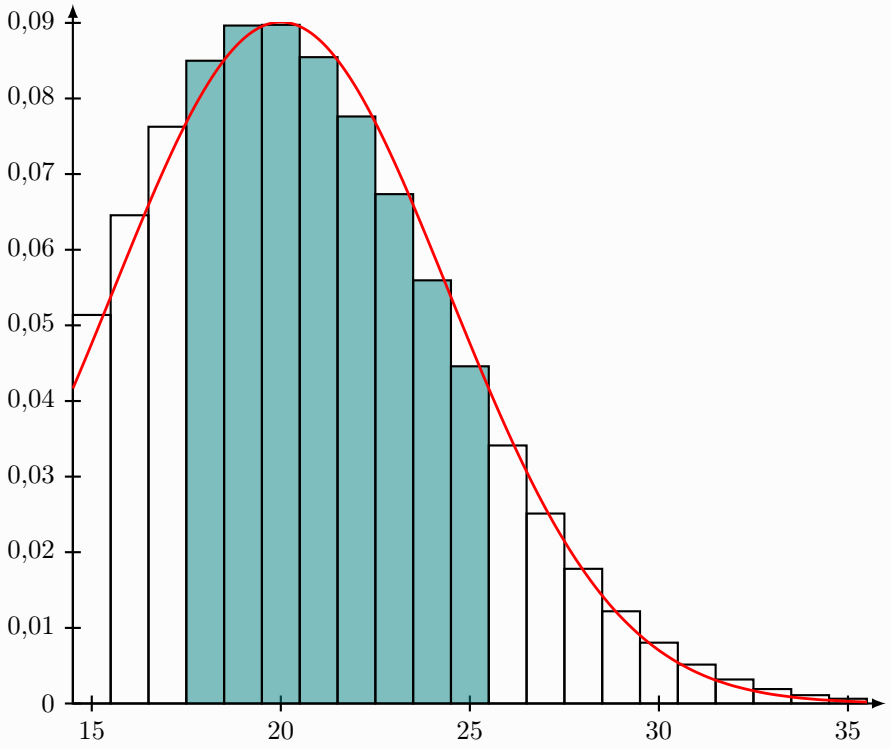
— `CouleurNormale` : couleur pour la loi normale.

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=0.5cm,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}
```



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 ;

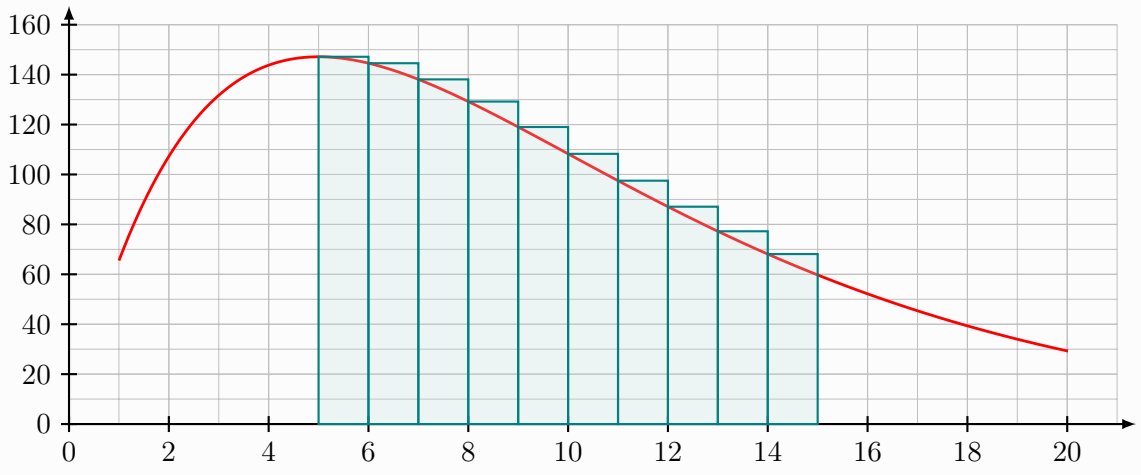
— `CouleurRemplissage` : couleur de remplissage, définie par rapport à la couleur principale par défaut ;

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

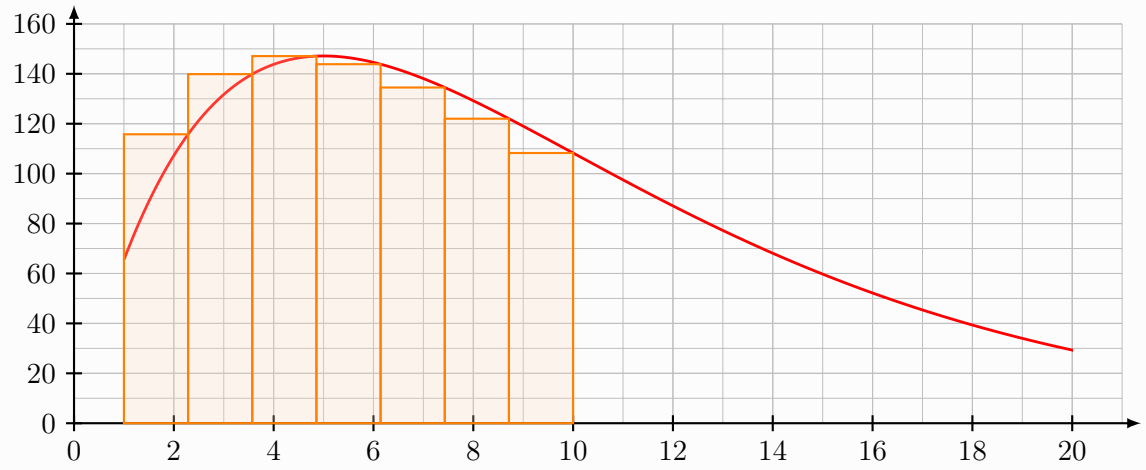
Le deuxième argument, optionnel et entre `<...>`, correspond à la fonction ou le spline **précédemment définie** !

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}
```

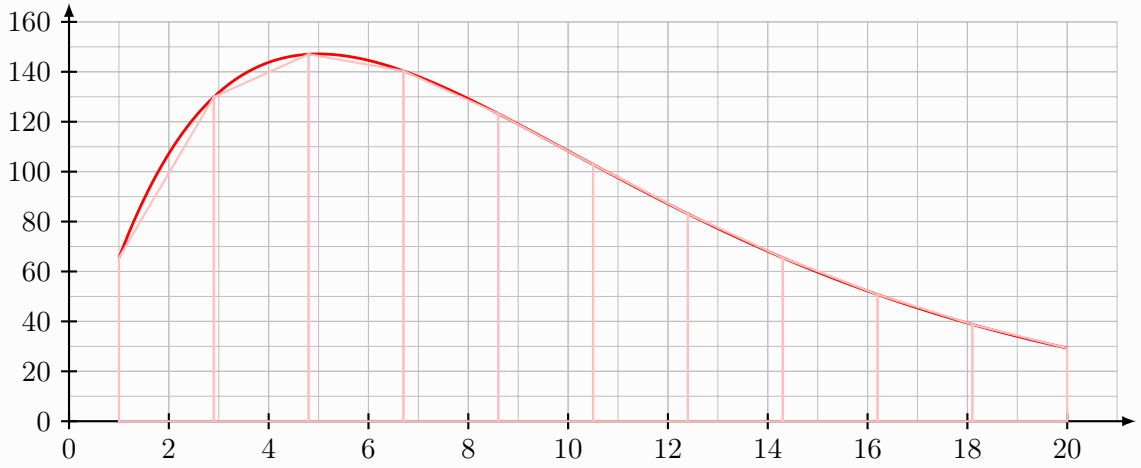


```
\begin{GraphiqueTikz}
  [x=0.66cm,y=0.033cm,Xmin=0,Xmax=21,Xgrille=2,Xgrilles=1,
  Ymin=0,Ymax=160,Ygrille=20,Ygrilles=10]
  \TracerAxesGrilles[Elargir=2.5mm]{auto}{auto}
  \DefinirCourbe[Couleur=red,Nom=cf,Debut=1,Fin=20,Trace]<f>{80*x*exp(-0.2*x)}
  \RepresenterMethodeIntegrale
    [Methode=RectanglesDroite,Couleur=orange,NbSubDiv=7]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

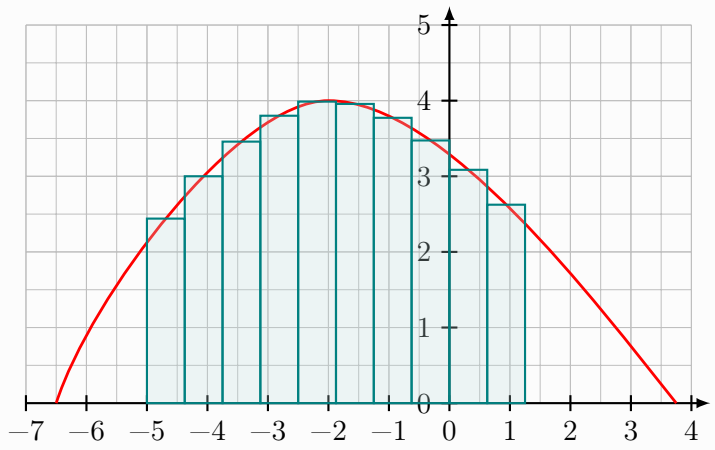



```
\begin{GraphiqueTikz}
  [x=0.66cm,y=0.033cm,Xmin=0,Xmax=21,Xgrille=2,Xgrilles=1,
  Ymin=0,Ymax=160,Ygrille=20,Ygrilles=10]
  \TracerAxesGrilles[Elargir=2.5mm]{auto}{auto}
  \DefinirCourbe[Couleur=red,Nom=cf,Debut=1,Fin=20,Trace]<f>{80*x*exp(-0.2*x)}
  \RepresenterMethodeIntegrale
  [Methode=RectanglesMilieu,Couleur=yellow,NbSubDiv=25]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

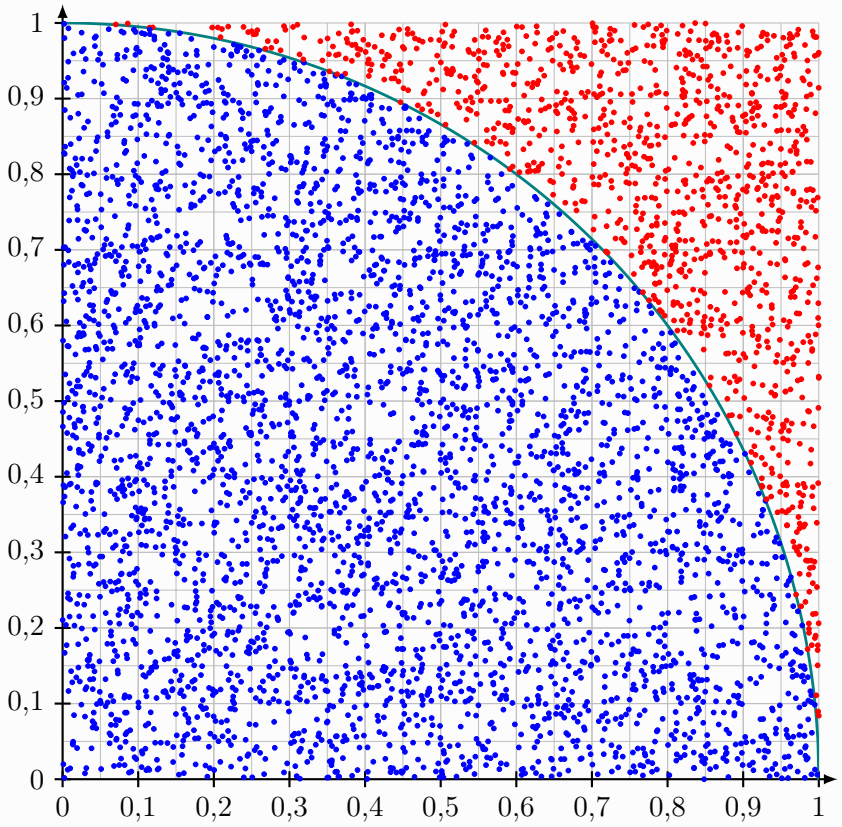
— `BornesX` : bornes *horizontales* pour la simulation, valant `\pflxmin,\pflxmax` par défaut ;

— `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 3 895, le nombre de points rouges est de 1 105.
La proportion de points bleus est de $\frac{3\,895}{5\,000} \approx 0{,}779$ 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 :

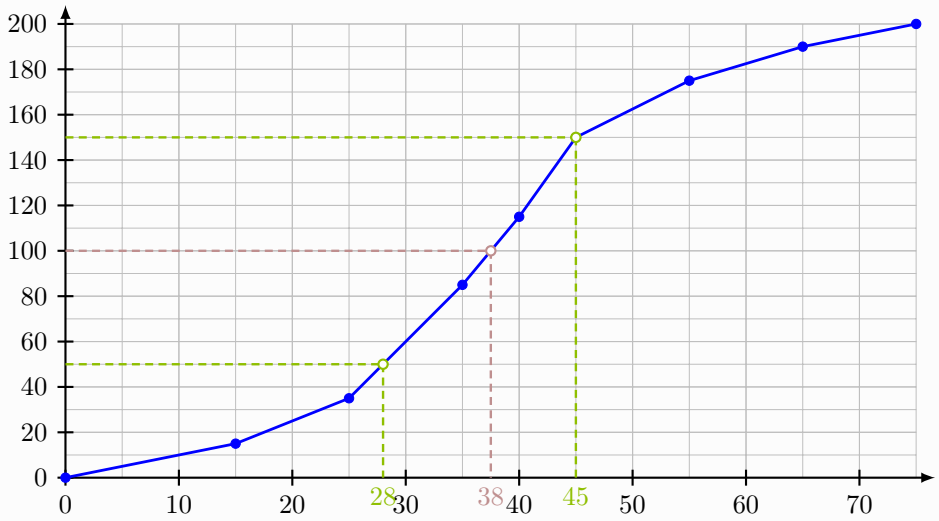
— `Couleur=...` : couleur du tracé, `black` par défaut ;

— `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 :

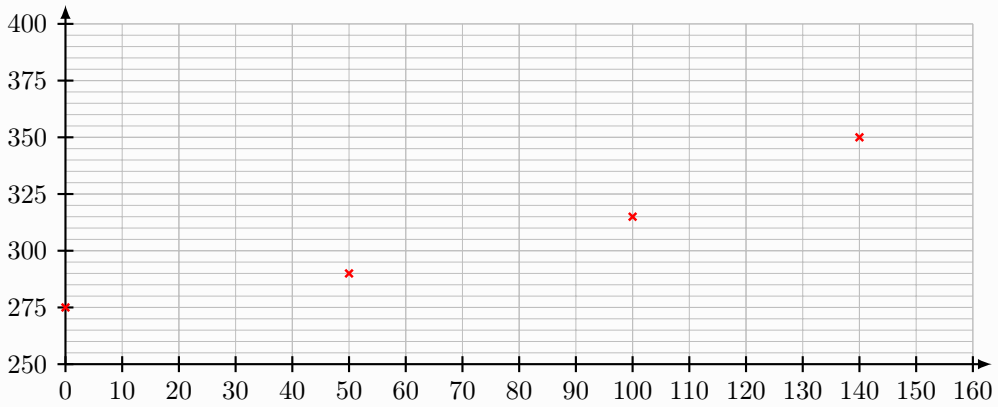
— `CouleurNuage` : couleur des points du nuage (`black` par défaut).

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}
```



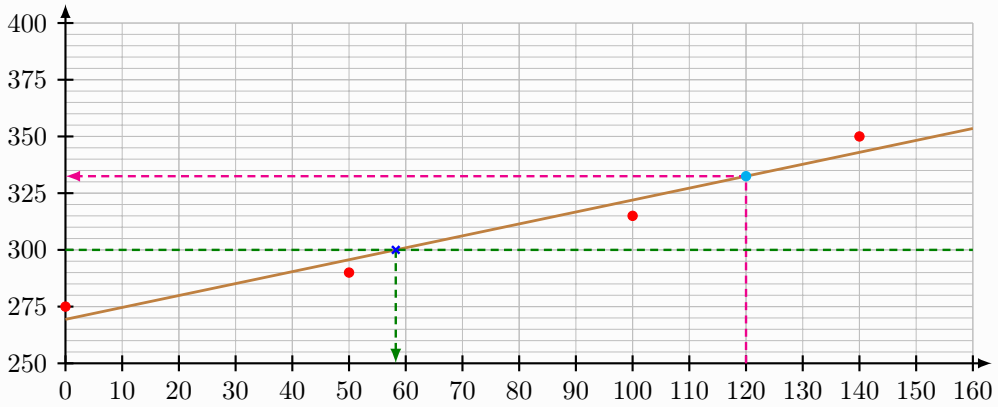
7.4 La droite de régression (2 variables)

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 $\boxed{e^{ax+b}}$;

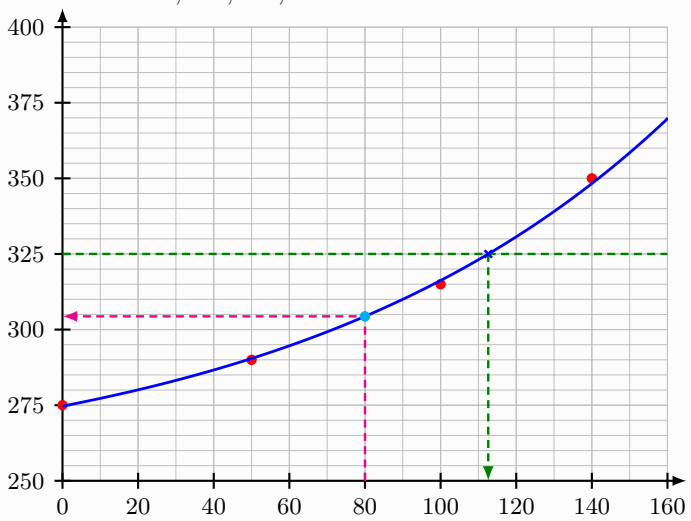
— `expalt` : exponentielle $\boxed{be^{ax}}$;

— `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 $y=250+\num{\expregoffb}\text{e}^{\num{\expregoffa}x}$
```

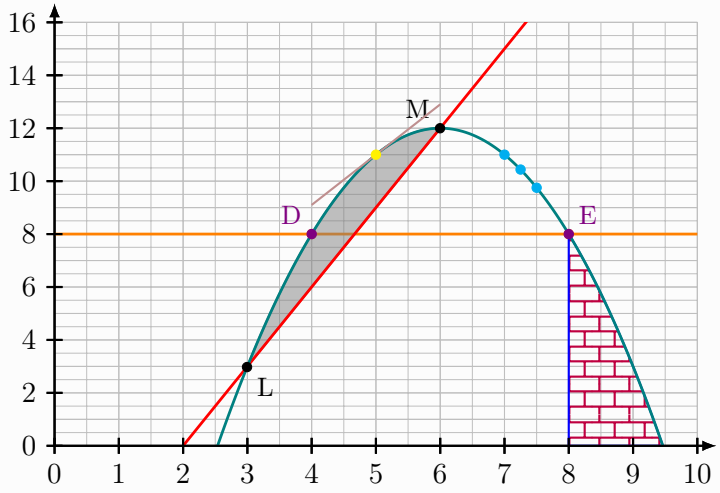
ListeX := 0,50,100,140
ListeY := 275,290,315,350



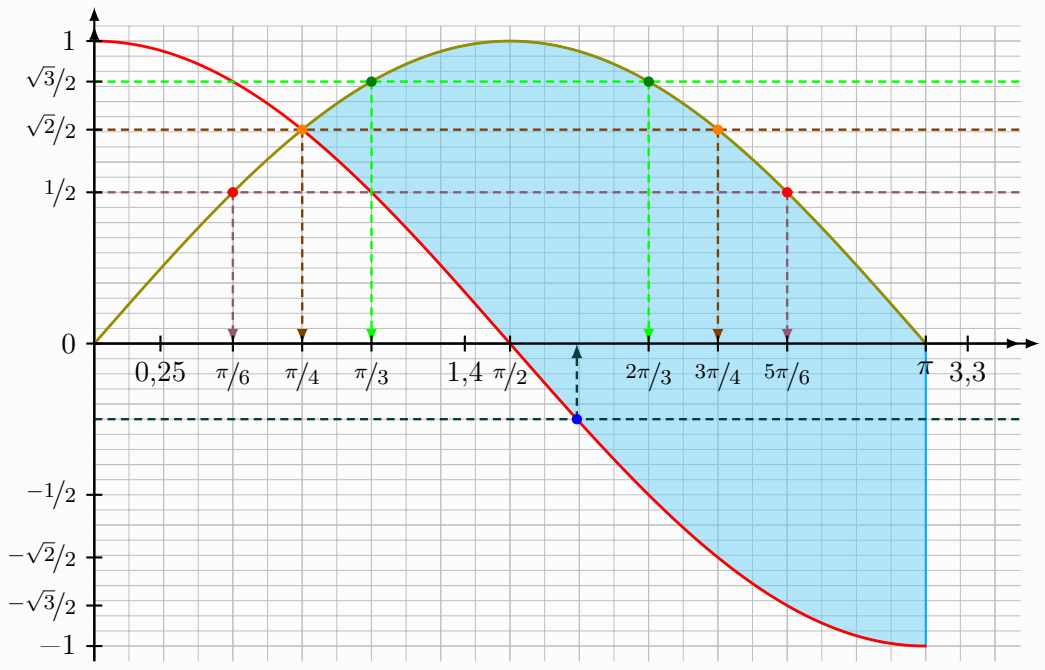
On obtient $y = 250 + 24{,}7\text{e}^{0{,}01x}$

8 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}
```



9 Commandes auxiliaires

9.1 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.

9.2 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

9.3 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 !

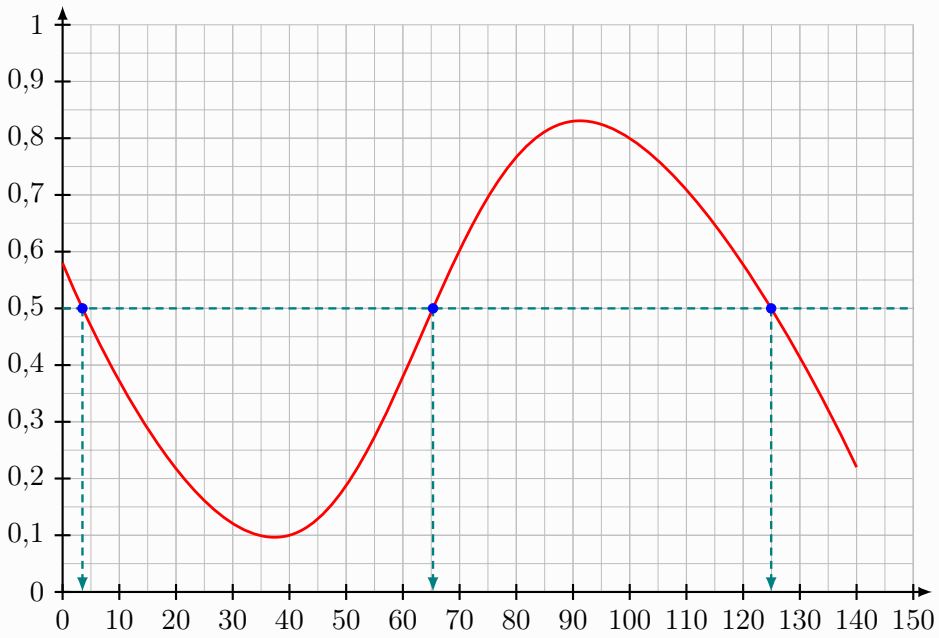
```
%un nombre (2 chiffres après la virgule) entre 0.75 et 0.95
%un nombre (2 chiffres après la virgule) entre 0.05 et 0.25
%un nombre (2 chiffres après la virgule) entre 0.55 et \YrandMax
%un nombre (2 chiffres après la virgule) entre \YrandMin et 0.45
\ChoisirNbAlea[2]{0.75}{0.95}[\YrandMax]%
\ChoisirNbAlea[2]{0.05}{0.25}[\YrandMin]%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

0,81 & 0,17 & 0,66 & 0,23

```
%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}
```

0,95 & 0,11 & 0,83 & 0,23

```
%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 : $\begin{cases} \alpha \approx 4 \\ \beta \approx 65 \\ \gamma \approx 125 \end{cases}$.

10 Liste des commandes

Les commandes disponibles sont :

11 Quelques commandes liées à pgfplots

11.1 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`...

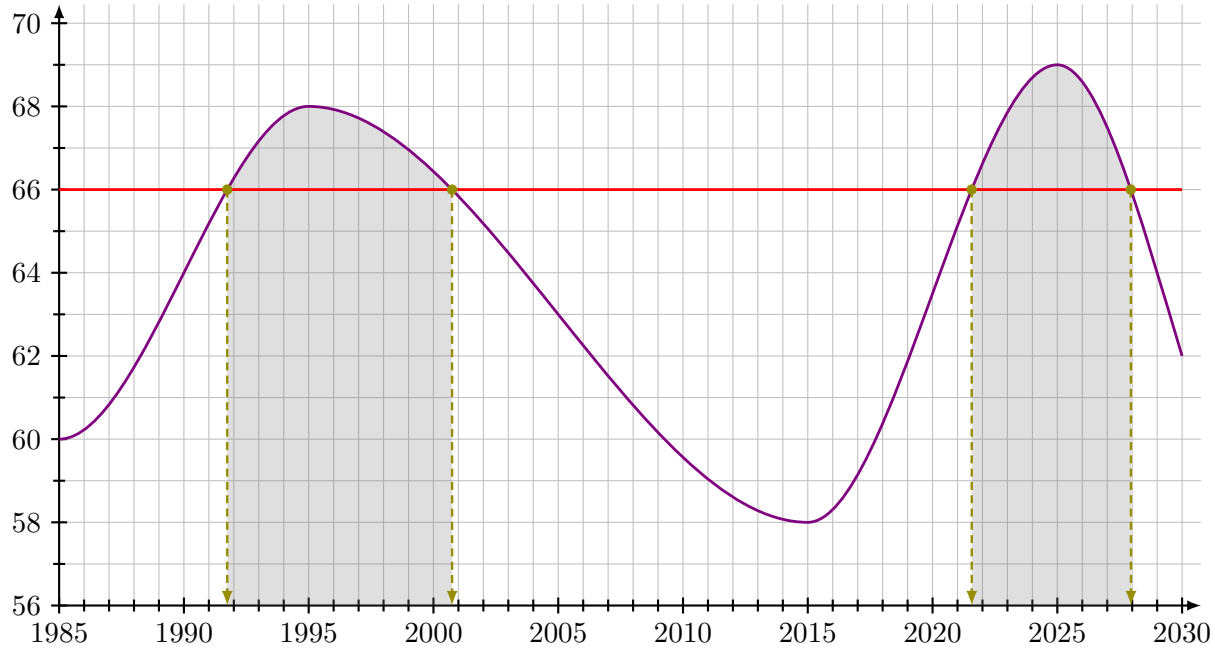
Dans ce cas, il est possible d'utiliser plutôt l'environnement `axis` de `pgfplots`, qui, de plus, permet *souvent* de pallier ce problème *interne*...
`tkz-grapheur` ne fournit pas d'environnement dédié pour la création de la fenêtre, mais quelques commandes spécifiques ont été intégrées pour certains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

11.2 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]
```

11.3 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.

12 Historique

```
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
```