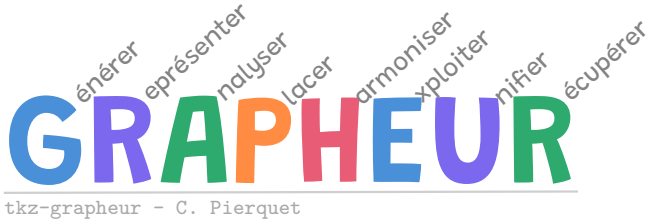
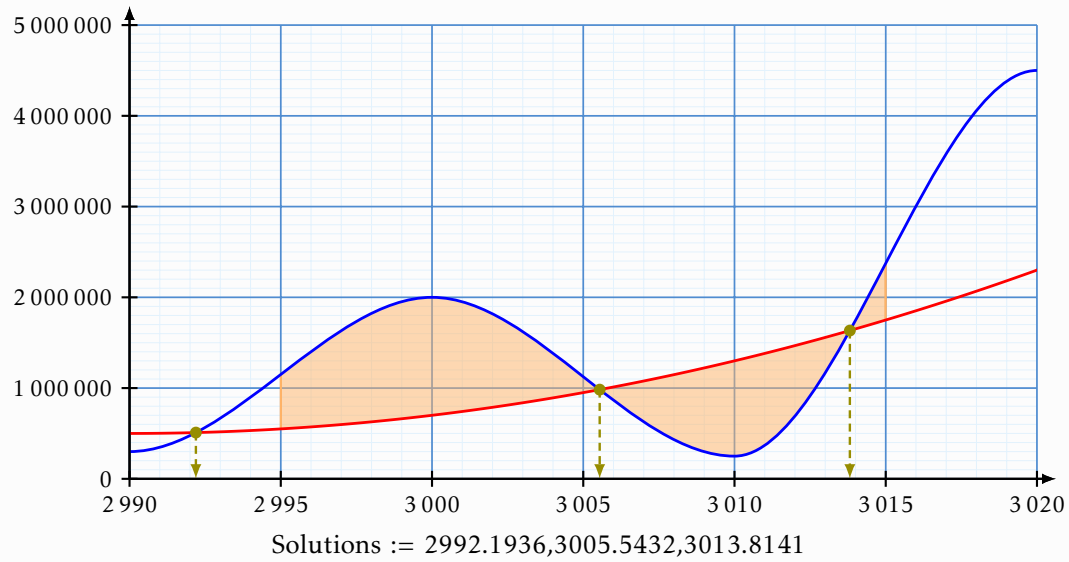
tkz-grapheur-altversion (v0.32c)

Version alternative de l'environnement graphique, basée sur TikZ, xint/calx et la normalisation interne.

Chargé automatiquement avec tkz-grapheur

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 et philosophie **4**
- 1.1 Problématique . . . 4
- 1.2 Principe de la normalisation . . . 4
- 1.3 Philosophie : tout est un path . . . 4
- 1.4 Rétrocompatibilité . . . 4

2 Environnement tkzGrph **5**
- 2.1 Syntaxe . . . 5
- 2.2 Clés disponibles . . . 5
- 2.3 Nœuds créés . . . 5
- 2.4 Axes et grilles . . . 6
- 2.5 Styles disponibles . . . 7
  - 2.5.1 Style des grilles . . . 7
  - 2.5.2 Style des points . . . 8

3 Environnement alternatif tkzgraph **9**
- 3.1 Syntaxe . . . 9
- 3.2 Différences avec tkzGrph . . . 9

4 Définir et tracer des courbes **10**
- 4.1 Courbe de fonction : \tkzGrphDefineCurve . . . 10
- 4.2 Interpolation lisse : \tkzGrphDefineInterpoCurve . . . 12
- 4.3 Spline de Hermite : \tkzGrphDefineSpline . . . 12
- 4.4 Wrapper universel : \tkzGrphCurve . . . 12
- 4.5 Tracer des courbes existantes . . . 13
- 4.6 Droites verticales et horizontales . . . 13

5 Compléments **15**
- 5.1 Récupération de coordonnées réelles . . . 15
- 5.2 Création de nœuds : \tkzGrphCreateNode(s) . . . 16
- 5.3 Marquage de points : \tkzGrphMarkPts . . . 16

6 Analyse(s) et exploitation(s) **17**
- 6.1 Intégrales : \tkzGrphDrawIntg . . . 17
- 6.2 Sommes de Riemann : \tkzGrphDrawRiemannSum . . . 18
- 6.3 Intersections : \tkzGrphFindItsc . . . 19
- 6.4 Images et antécédents . . . 20
  - 6.4.1 Image : \tkzGrphFindY . . . 20
  - 6.4.2 Images : \tkzGrphFindListY . . . 20
  - 6.4.3 Antécédents : \tkzGrphFindX . . . 20
- 6.5 Extremums : \tkzGrphFindMin/Max . . . 21
- 6.6 Points d'intérêt : \tkzGrphInterestPoints . . . 23
- 6.7 Point sur une courbe : \tkzGrphDefPointBy . . . 26

7 Nuage et régression : statistiques **27**
- 7.1 Nuage de points : \tkzGrphScatterPlot . . . 27
- 7.2 Nuage depuis CSV : \tkzGrphNuageCSV . . . 27
- 7.3 Droite de régression : \tkzGrphDrawRegLin . . . 28
- 7.4 Régressions : \tkzGrphDrawFit . . . 29

8 Compléments sur les axes **30**
- 8.1 Valeurs manuelles sur les axes . . . 30
- 8.2 Placer un texte : \tkzGrphPlaceText . . . 31

9 Annotations graphiques **32**
- 9.1 Label sur un path : \tkzGrphLabelPath . . . 32
- 9.2 Point sur un path : \tkzGrphMarkOnPath . . . 32
- 9.3 Intervalle sur un axe : \tkzGrphIntervalX/Y . . . 33

10 Nuage de points de suites **34**
10.1 Clés communes 34
10.2 Suite explicite : \tkzGrphSeqExplicite 35
10.3 Suite récurrente : \tkzGrphSeqRecurrent 35
10.4 Suite récurrente avec rang : \tkzGrphSeqRecurrentN 36
10.5 Double récurrence : \tkzGrphSeqDouble 36
10.6 Double récurrence avec rang : \tkzGrphSeqDoubleN 37
10.7 Récurrence avec toile : \tkzGrphSeqDoubleN 37
10.8 Wrapper universel : \tkzGrphDrawSequence 38

1 Introduction et philosophie

1.1 Problématique

L'environnement `GraphiqueTikz` de `tkz-grapheur` travaille directement en coordonnées Ti*k*Z — des valeurs extrêmes sur les axes (années, populations, distances…) génèrent des erreurs `dimension too large`.

L'environnement alternatif `tkzGrph` résout ce problème par une **normalisation interne** : Ti*k*Z travaille toujours dans $[0, L] \times [0, H]$ (en cm), quelle que soit l'amplitude réelle.

1.2 Principe de la normalisation

Pour une fenêtre $[\text{Xmin}, \text{Xmax}] \times [\text{Ymin}, \text{Ymax}]$ et des dimensions $L \times H$:

$$x_{\text{norm}} = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \times L \qquad y_{\text{norm}} = \frac{y - y_{\min}}{y_{\max} - y_{\min}} \times H$$

Ti*k*Z ne voit jamais que des coordonnées dans $[0, L] \times [0, H]$.

1.3 Philosophie : tout est un path

Toutes les courbes sont des `name path` nommés — fonction xint, interpolation ou spline. Cela permet :

- — de **réutiliser** un tracé avec des styles différents via `\tkzGrphDrawExistingCurve(s)` ;
- — de calculer des **intersections** entre n'importe quelles courbes ;
- — de calculer des **intégrales** entre n'importe quelles courbes ;
- — d'**unifier** la syntaxe indépendamment du type de courbe.

1.4 Rétrocompatibilité

L'environnement `tkzGrph` accepte les mêmes noms de clés que `GraphiqueTikz` (`Xmin`, `Xgrille`…) ainsi que leurs alias anglais (`width`, `xgrid`…).

2 Environnement tkzGrph

2.1 Syntaxe

```
\begin{tkzGrph}[options tikz]<clés>
  ...
\end{tkzGrph}
```

2.2 Clés disponibles

— `Xmin`/`Xmax` : bornes de l'axe Ox, `0`/`10` par défaut ;
— `Ymin`/`Ymax` : bornes de l'axe Oy, `0`/`10` par défaut ;
— `Largeur=...` / `width=...` : largeur (dimension pure), `8cm` par défaut ;
— `Hauteur=...` / `height=...` : hauteur (dimension pure), `6cm` par défaut ;
— `x=...` : unité x éventuelle (dimension pure), `vide` par défaut ;
— `y=...` : unité y éventuelle (dimension pure), `vide` par défaut ;
— `Origx=...` / `Origy=...` : position de l'origine, `{0}` par défaut ;
— `Xgrille=...` / `xgrid=...` : pas de grille principale en x, `1` par défaut ;
— `Xgrilles=...` / `xgrids=...` : pas de grille secondaire en x, `0.5` par défaut ;
— `Xgrillei=...` / `xgridi=...` : pas de grille intermédiaire en x ;
— `Ygrille=...`, `Ygrilles=...`, `Ygrillei=...` : idem en y ;
— `Theme=...` / `theme=...` : thème de couleurs (`standard`, `gris`, `bleu`, `vert`, `chaud`, `contraste`) ;
— `NomFigure=...` / `figurename=...` : préfixe des nœuds pour plusieurs figures ;
— `TailleGrad=...` / `gradsize=...` : taille des graduations (`longueur` ou `dessus/dessous`) ;
— `AffCadre` / `showframe` : affiche un cadre ;
— `Milli` / `milli` : grille millimétrée.

💣 `Largeur` et `Hauteur` attendent des **dimensions pures** (`8cm`, `120mm`...).

💣 Les clés `x` et/ou `y` permettent de déterminer automatiquement la largeur et/ou la hauteur du graphique, et écrasent donc les clés `Largeur` et/ou `Hauteur`.

2.3 Nœuds créés

— `grph-ne`, `grph-nw`, `grph-se`, `grph-sw` : coins de la fenêtre ;
— `grph-n`, `grph-s`, `grph-e`, `grph-w`, `grph-c` : milieux des bords et centre ;
— `grph-axox-w`, `grph-axox-e`, `grph-axoy-s`, `grph-axoy-n` : bords des axes ;
— `grph-orig` : origine du repère.

2.4 Axes et grilles

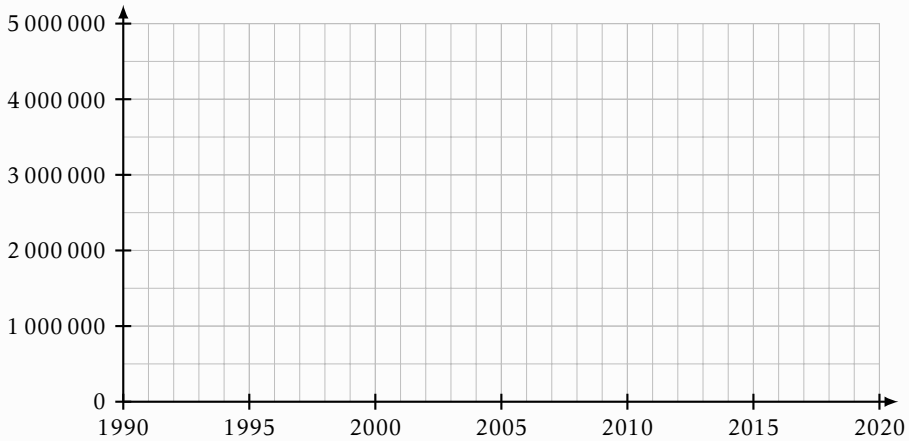
```
\tkzGrphDrawGridAxis[clés]{liste valeurs X}{liste valeurs Y}
% alias FR :
\tkzGrphTracerAxesGrille[clés]{liste valeurs X}{liste valeurs Y}
```

Les listes sont des **valeurs réelles** — la conversion est automatique.

Les `[clés]` disponibles sont :

- `Grille` / `grid` : booléen, `true` par défaut ;
- `GrilleIntermediaire` / `intermediategrid` : booléen ;
- `Grads` / `labels` : booléen, `true` par défaut ;
- `AxeOy` / `yaxis` : booléen, `true` par défaut ;
- `Fleches` / `arrows` : booléen, `true` par défaut ;
- `Elargir` / `enlarge` : élargissement des axes ;
- `Police` / `font` : police des labels ;
- `Format` / `format` : format des labels (`num`, `annee`, `frac`…) ;
- `Derriere` / `behind` : grille en arrière-plan ;
- `Devant` / `front` : axes en avant-plan (sans grille).

```
\begin{tkzGrph}%
    <Xmin=1990,Xmax=2020,Ymin=0,Ymax=5000000,
    Largeur=10cm,Hauteur=5cm,Xgrille=5,Xgrilles=1,
    Ygrille=1000000,Ygrilles=500000,Origx=1990>
  \tkzGrphTracerAxesGrille[Elargir=2.5mm,Police=\small,Format=annee/num]%
    {1990,1995,2000,2005,2010,2015,2020}%
    {0,1000000,2000000,3000000,4000000,5000000}
\end{tkzGrph}
```

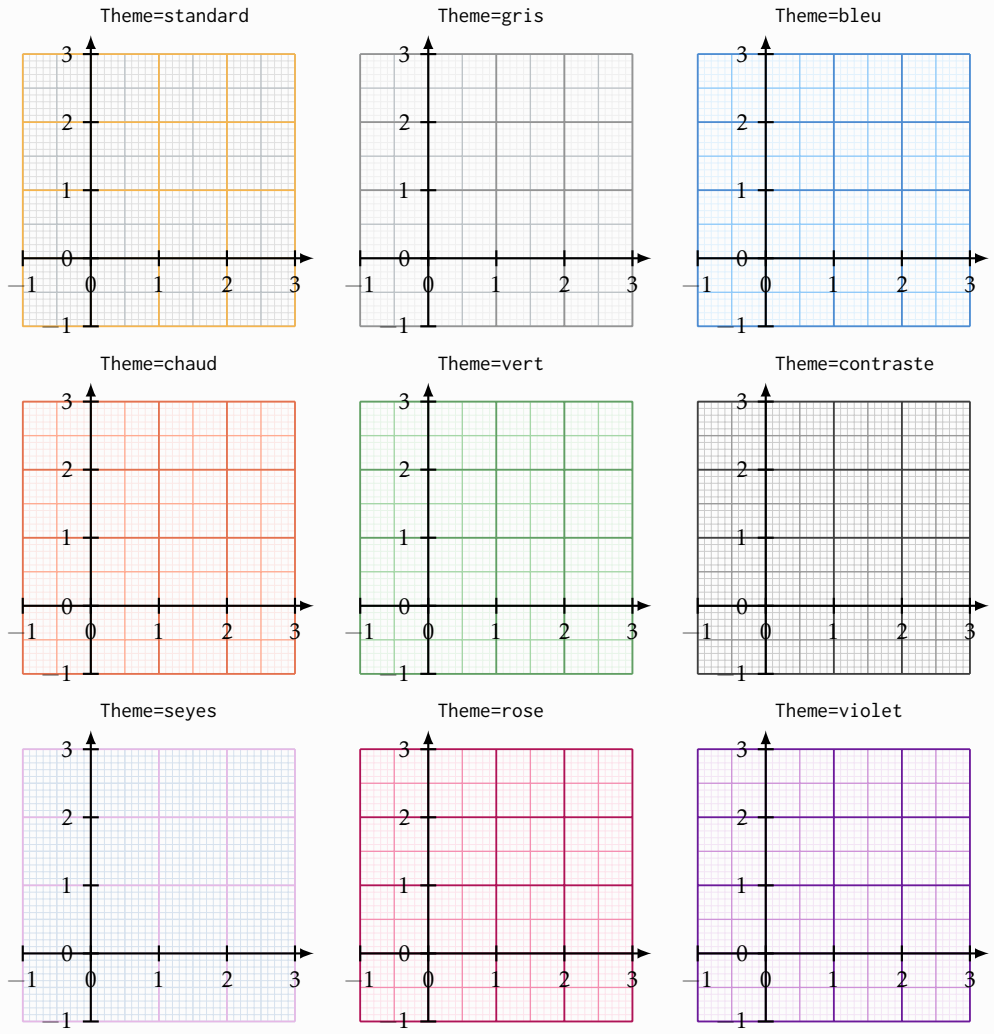


2.5 Styles disponibles

2.5.1 Style des grilles

Il existe de plus 7 thèmes prédéfinis de couleurs pour les grilles, parmi :
`<Theme=standard/gris/bleu/vert/chaud/contraste/seyes/rose/violet>`.

```
\begin{tkzGrph}[options tikz]<clés,Theme=...>
  ...
\end{tkzGrph}
```

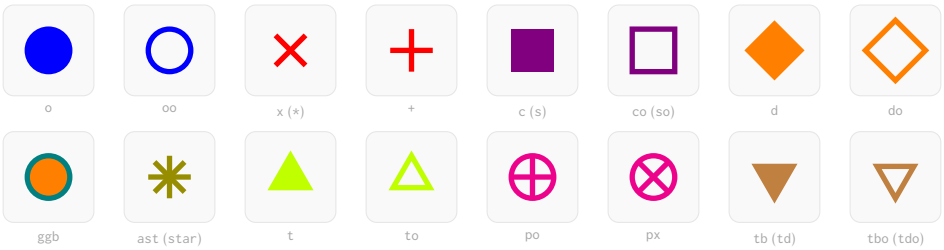


Les épaisseurs des traits pour ce type de grille sont données (et modifiables) via :

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

2.5.2 Style des points

Les styles disponibles pour les points sont : `o` (disque plein), `oo` (cercle vide), `x` (croix diagonale), `+` (croix droite), `c/s` (carré plein), `co/so` (carré vide), `d` (diamant plein), `do` (diamant vide), `ggb` (style GeoGebra). La clé `Couleur=A/B` permet de différencier la couleur du contour et du fond.



Ces styles sont accessibles dans toute commande permettant de marquer des points, via les clés partagées :

— `Style / style` ;
— `Taillec / sizec` ;
— `Taillex / sizex` ;
— `Tailleo / sizeo` ;
— `Epaisseur / thick`.

3 Environnement alternatif tkzgraph

3.1 Syntaxe

```
\begin{tkzgraph}[clés]
  ...
\end{tkzgraph}
```

Cet environnement est un alias fonctionnel de tkzGrph : il produit exactement le même résultat, mais adopte une syntaxe unifiée où **toutes les options** — clés graphiques *et* options TikZ — sont regroupées dans un seul argument optionnel [...].

3.2 Différences avec tkzGrph

— Les clés graphiques (Xmin, Largeur, Theme...) s'écrivent directement dans [...], sans l'argument <...> de tkzGrph.

— Les booléens (AffCadre, showframe, Milli, GradsAuto...) s'utilisent en forme courte sans =true : [AffCadre,Xmin=0,Xmax=5].

```
%% équivalents :
\begin{tkzGrph}[]{Xmin=0,Xmax=5,AffCadre=true}... \end{tkzGrph}

\begin{tkzgraph}[Xmin=0,Xmax=5,AffCadre]...\end{tkzgraph}
```

💣 Les macros internes sont partagées : toute modification des styles ou des macros de tracé s'applique identiquement aux deux environnements.

4 Définir et tracer des courbes

4.1 Courbe de fonction : \tkzGrphDefineCurve

```
\tkzGrphDefineCurve[clés]<nom fct>{expression xint}
% alias FR :
\tkzGrphDefinirCourbe[clés]<nom fct>{expression xint}
```

Les [clés] disponibles sont :

— Couleur / color : couleur du tracé, black par défaut ;
— Nom / name : nom du path créé ;
— Debut / start : abscisse de départ (réelle), Xmin par défaut ;
— Fin / end : abscisse de fin (réelle), Xmax par défaut ;
— Pas / step : pas du tracé, calculé automatiquement par défaut ;
— Trace / draw : booléen, true par défaut ;
— TypeTrace / tracetype : style de tracé, smooth par défaut ;
— Auto / auto : mode *branche automatique*, false par défaut — voir ci-dessous.

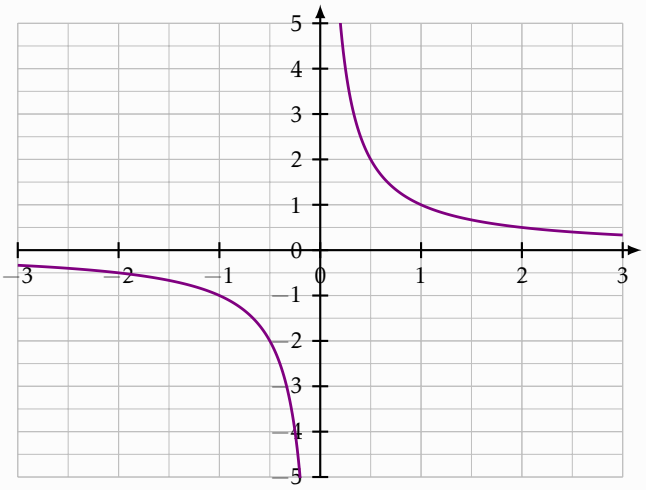
Clé Auto : gestion automatique des branches

La clé booléenne Auto/auto active la **détection automatique des valeurs interdites et/ou discontinuités** lors du tracé d'une fonction. Sans cette clé, les valeurs non finies (pôles, Inf, NaN) provoquent des erreurs ou des artefacts visuels (segments parasites traversant l'axe des abscisses). Avec Auto, le tracé est automatiquement *découpé en branches* à chaque saut détecté.

⚠ Dépendance à calx

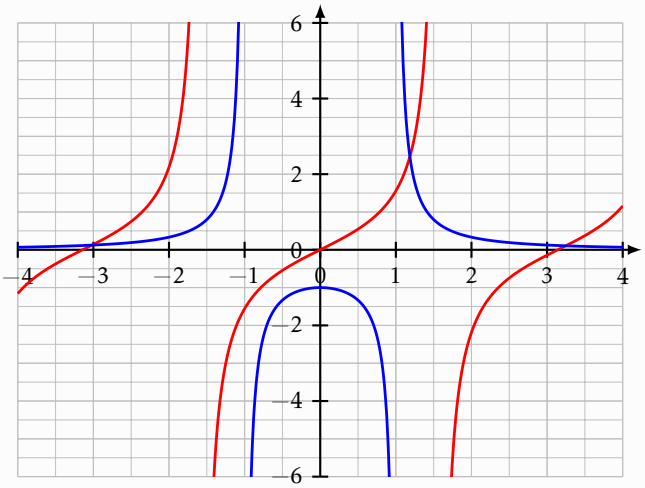
La détection fine des valeurs non finies (NaN, ±Inf) repose sur la macro \xintifNonFinitefloatexpr fournie par le package calx (distribué avec xint ⩾ 2.0). Si calx n'est pas disponible, les asymptotes de type *saut brutal de valeur* restent détectées, mais les pôles donnant Inf/NaN ne seront pas filtrés et donc des erreurs pourront persister.

```
\begin{tkzGrph}<Xmin=-3,Xmax=3,Ymin=-5,Ymax=5,Largeur=8cm,Hauteur=6cm>
  \tkzGrphDrawGridAxis[enlarge=2.5mm,font=\small]{-3,-2,...,3}{-5,-4,...,5}
  \tkzGrphDefineCurve[Trace,Auto,Couleur=violet,Nom=cf]{1/x}
\end{tkzGrph}
```



La clé Auto fonctionne également pour les fonctions à plusieurs singularités :

```
\begin{tkzGrph}<Xmin=-4,Xmax=4,Ymin=-6,Ymax=6,Largeur=8cm,Hauteur=6cm>
  \tkzGrphDrawGridAxis[enlarge=2.5mm,font=\small]{-4,-3,...,4}{-6,-4,...,6}
  \tkzGrphDefineCurve[Trace,Auto,Couleur=red,Nom=ctg]{tan(x)}       %multiples discontinuités
  \tkzGrphDefineCurve[Trace,Auto,Couleur=blue,Nom=cfrac]{1/(x^2-1)} %pôles en ±1
\end{tkzGrph}
```



4.2 Interpolation lisse : \tkzGrphDefineInterpoCurve

```
\tkzGrphDefineInterpoCurve[clés]{x/y § x/y § ...}
% alias FR :
\tkzGrphDefinirCourbeInterpo[clés]{x/y § x/y § ...}
```

Séparateur `§` entre les points, `/` entre les coordonnées. Clé spécifique : `tension` (défaut `0.5`).

4.3 Spline de Hermite : \tkzGrphDefineSpline

```
\tkzGrphDefineSpline[clés]{x/y/f'(x) § x/y/f'(x) § ...}
% alias FR :
\tkzGrphDefinirCourbeSpline[clés]{x/y/f'(x) § x/y/f'(x) § ...}
```

Remarques :

— Pour chaque point : abscisse / ordonnée / dérivée (toutes en valeurs réelles).
— La dérivée est normalisée automatiquement.
— Clé spécifique (Hermite) : `Alt` (mode alternatif Bézier, conseillé pour les courbes oscillantes).

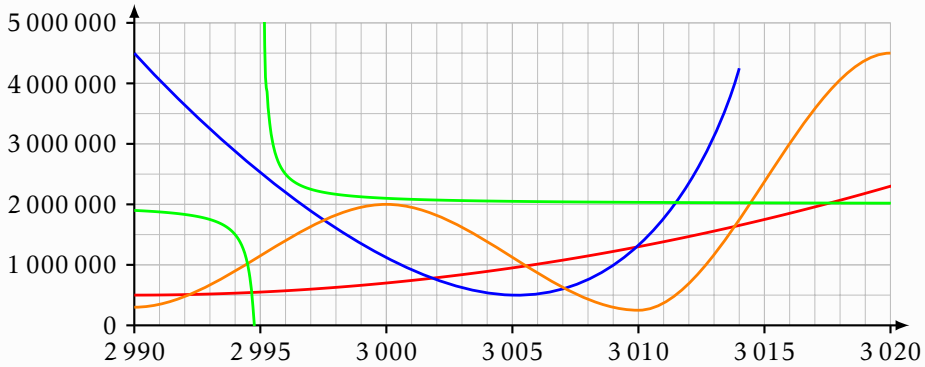
4.4 Wrapper universel : \tkzGrphCurve

La commande `\tkzGrphCurve` unifie les trois types de courbes en une seule syntaxe. Le type est passé entre parenthèses en premier argument (défaut : `fct`).

```
\tkzGrphCurve(type)[clés]<nom>{données}
% type : fct (défaut) | interpo | spl
% alias FR : \tkzGrphCourbe
```

Les `[clés]` et l'argument `<nom>` sont identiques à ceux des commandes individuelles. L'argument `<nom>` est utile uniquement pour les `fct` (nom de la fonction) et `spline` (macro de stockage du spline).

```
\begin{tkzGrph}%
    <Xmin=2990,Xmax=3020,Ymin=0,Ymax=5000000,Largeur=10cm,Hauteur=4cm,Xgrille=5,
    Xgrilles=1,Ygrille=1000000,Ygrilles=500000,Origx=2990,TailleGrad=0pt/3pt>
  \tkzGrphTracerAxesGrille[Elargir=2.5mm]%
    {2990,2995,3000,3005,3010,3015,3020}{0,1000000,2000000,3000000,4000000,5000000}
  % courbe de fonction
  \tkzGrphCurve[Nom=cf]<f>{2000*(x-2990)^2+500000}
  \tkzGrphCurve[Nom=cg,Auto]<g>{2000000 + 500000/(x-2995)}
  \tkzGrphCurve(interpo)[Tension=1,Nom=interpo]{2990/4500000 § 3005/500000 § 3014/4250000}
  \tkzGrphCurve(spl)[Alt,Nom=sp]%
    {2990/300000/0.5 § 3000/2000000/0 § 3010/250000/0 § 3020/4500000/0.8}
  \tkzGrphTracerCourbes{cf/red,interpo/blue,sp/orange,cg/green}
\end{tkzGrph}
```

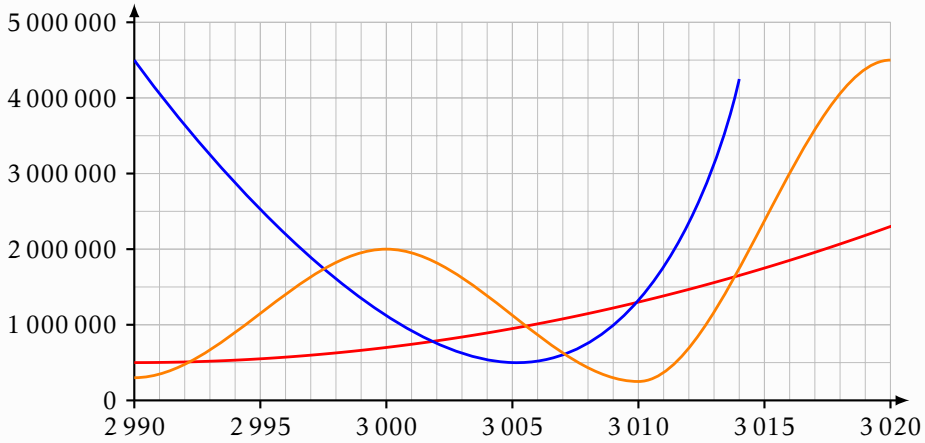


4.5 Tracer des courbes existantes

```
% tracer UN path existant avec un style
\tkzGrphDrawExistingCurve[options draw]{nom path}

% tracer PLUSIEURS paths en une fois : nom/couleur,...
\tkzGrphDrawExistingCurves[options draw]{nom/couleur, nom/couleur, ...}
% alias FR :
\tkzGrphTracerCourbe / \tkzGrphTracerCourbes
```

```
\begin{tkzGrph}%
    <Xmin=2990,Xmax=3020,Ymin=0,Ymax=5000000,
    Largeur=10cm,Hauteur=5cm,Xgrille=5,Xgrilles=1,
    Ygrille=1000000,Ygrilles=500000,Origx=2990,%
    TailleGrad=0pt/3pt>
  \tkzGrphTracerAxesGrille[Elargir=2.5mm]%
     {2990,2995,3000,3005,3010,3015,3020}%
     {0,1000000,2000000,3000000,4000000,5000000}
  \tkzGrphDefinirCourbe[Nom=cf]<f>{2000*(x-2990)^2+500000}
  \tkzGrphDefinirCourbeInterpo[Tension=1,Nom=interpo]%
    {2990/4500000 § 3005/500000 § 3014/4250000}
  \tkzGrphDefinirCourbeSpline[Alt,Nom=sp]%
    {2990/300000/0.5 § 3000/2000000/0 § 3010/250000/0 § 3020/4500000/0.8}
  % tracer avec des couleurs différentes
  \tkzGrphTracerCourbes{cf/red,interpo/blue,sp/orange}
\end{tkzGrph}
```



4.6 Droites verticales et horizontales

La version `alt` fournit deux commandes pour tracer des droites parallèles aux axes, exprimées en **coordonnées réelles**. Elles tracent une ou plusieurs droites en une seule commande (liste de valeurs séparées par des virgules).

```
% Droites verticales x = cte
\tkzGrphDrawVLines[cles]{val1,val2,...}
% alias : \tkzGrphVLine  \tkzgrphvl

% Droites horizontales y = cte
\tkzGrphDrawHLines[cles]{val1,val2,...}
% alias : \tkzGrphHLine  \tkzgrphhl
```

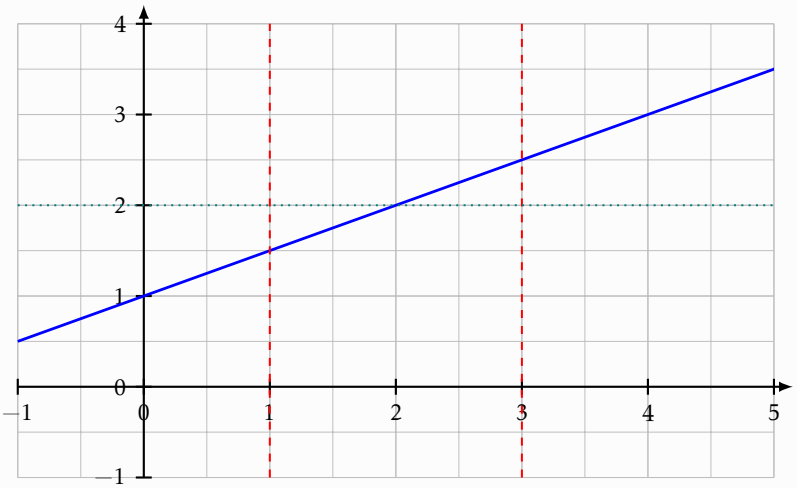
Les `[clés]` disponibles (famille `GraphiqueTikzAltVHL`) :

— `Couleur` / `color` : couleur du trait (`black` par défaut) ;

— `Style` / `style` : style de ligne TikZ (`solid` par défaut) — `dashed`, `dotted`, *etc.*

Les droites sont automatiquement clippées à la fenêtre du graphique.

```
\begin{tkzGrph}<Xmin=-1,Xmax=5,Ymin=-1,Ymax=4,
  Largeur=10cm,Hauteur=6cm>
  \tkzGrphDrawGridAxis[enlarge=2.5mm,font=\small]%
  {-1,0,...,5}{-1,0,...,4}
  \tkzGrphDefineCurve[trace,name=cf,color=blue]{0.5*x+1}
  % droites verticales en x = 1 et x = 3 (rouge, tiretés)
  \tkzGrphDrawVLines[color=red,style=dashed]{1,3}
  % droite horizontale en y = 2 (vert, pointillés)
  \tkzGrphDrawHLines[color=teal,style=dotted]{2}
\end{tkzGrph}
```



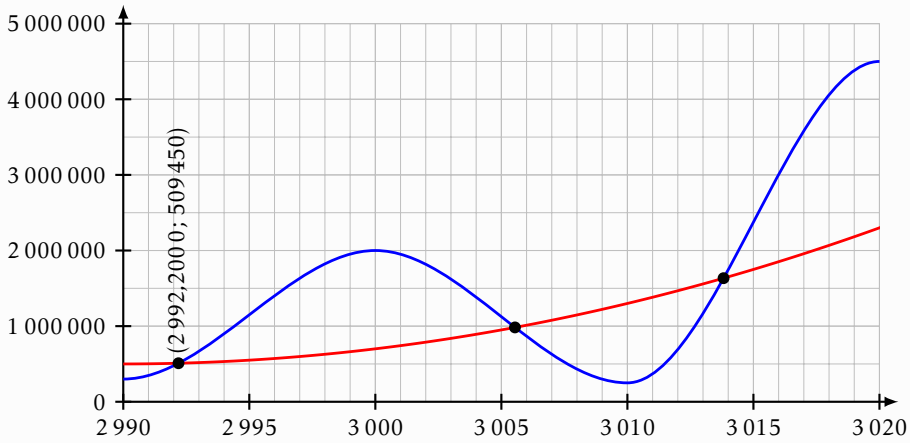
5 Compléments

5.1 Récupération de coordonnées réelles

```
\tkzGrphGetX{(nœud)}[\mavaleur]    % abscisse réelle
\tkzGrphGetY{(nœud)}[\mavaleur]    % ordonnée réelle
\tkzGrphGetXY{(nœud)}[\monx][\mony]
% alias FR :
\tkzGrphRecupX / \tkzGrphRecupY / \tkzGrphRecupXY
```

Dénormalise les coordonnées d'un nœud TikZ existant vers les vraies valeurs.

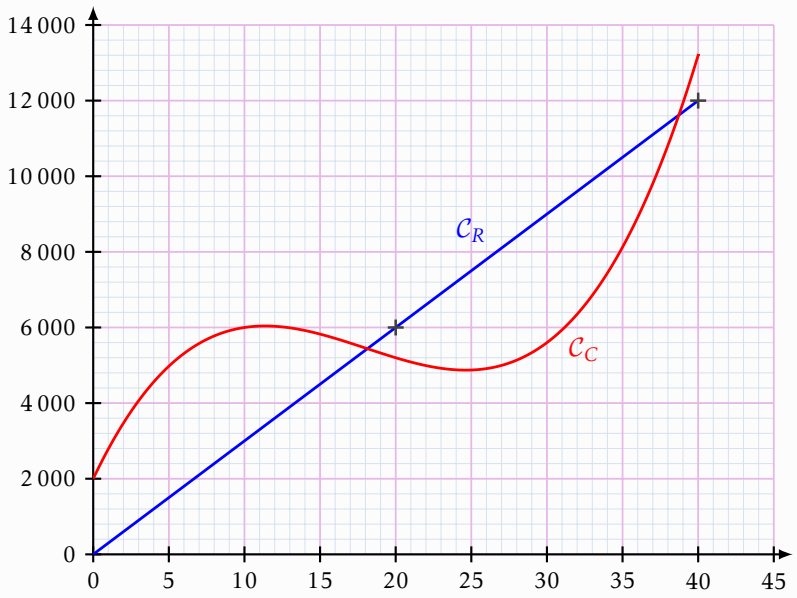
```
\begin{tkzGrph}%
    <Xmin=2990,Xmax=3020,Xgrille=5,Xgrilles=1,
    Ymin=0,Ymax=5000000,Ygrille=1000000,Ygrilles=500000,
    Largeur=10cm,Hauteur=5cm,Origx=2990>
  \tkzGrphDrawGridAxis[enlarge=2.5mm,font=\small]%
    {2990,2995,3000,3005,3010,3015,3020}%
    {0,1000000,2000000,3000000,4000000,5000000}
  \tkzGrphDefineCurve[name=cf]<f>{2000*(x-2990)^2+500000}
  \tkzGrphDefineSpline[alt,name=sp]%
    {2990/300000/0.5 § 3000/2000000/0 § 3010/250000/0 § 3020/4500000/0.8}
  \tkzGrphDrawExistingCurves{cf/red,sp/blue}
  \tkzGrphFindItsc[show,name=K]{cf}{sp}
  % récupérer et afficher les coordonnées réelles
  \tkzGrphGetXY{(K-1)}[\xun][\yun]
  \node[right,font=\small,rotate=90] at (K-1)
    {$(\ArrondirNum[1]{\xun}\,;\,\ArrondirNum[0]{\yun})$} ;
\end{tkzGrph}
```



5.2 Création de nœuds : \tkzGrphCreateNode(s)

```
% version un seul nœud
\tkzGrphCreateNode[nom]{(x,y)}
% version plusieurs nœuds
\tkzGrphCreateNode{(xA,yA) § nomA , (xB,yB) § nomB , ... }
% aliases FR :
\tkzGrphCreerNoeud et \tkzGrphCreerNoeuds
```

```
\begin{tkzGrph}%
  <Xmin=0,Xmax=45,Xgrille=5,Xgrilles=1,%
  Ymin=0,Ymax=14000,Ygrille=2000,Ygrilles=400,%
  Largeur=9cm,Hauteur=7cm,Theme=seyes>
  \tkzGrphDrawGridAxis[enlarge=2.5mm,font=\small]{0,5,...,45}{0,2000,...,14000}
  \tkzGrphDefineCurve[name=CR,end=40]<R>{300*x}
  \tkzGrphDefineCurve[name=CC,end=40]<C>{x^3-54*x^2+840*x+2000}
  \tkzGrphDrawExistingCurves{CR/blue,CC/red}
  \tkzGrphCreateNodes{{(20,6000)} § ptA , {(40,12000)} § ptB}
  \tikzset{pflpoint/.style={line width=1.25pt}}
  \tkzGrphMarkPts*[style=+,sizex=3pt,color=darkgray]{(ptA),(ptB)}
  \tkzGrphPlaceText[pos=below,color=red]{(32.5,6000)}{$\mathcal{C}_C$}
  \tkzGrphPlaceText[pos=above,color=blue]{(25,8000)}{$\mathcal{C}_R$}
\end{tkzGrph}
```



5.3 Marquage de points : \tkzGrphMarkPts

```
% version normale : avec label
\tkzGrphMarkPts[clés]<police>{liste de nœuds TikZ}
% version étoilée : sans label
\tkzGrphMarkPts*[clés]{liste de nœuds TikZ}
% alias FR :
\tkzGrphMarquerPts
```

Les nœuds sont donnés en coordonnées normalisées (à faire en amont) ou via des nœuds TikZ existants. Clés : `Couleur/color`, `Style/style` (`o`, `x`, `+`, `ggb`...).

6 Analyse(s) et exploitation(s)

6.1 Intégrales : \tkzGrphDrawIntg

```
% sous une courbe (axe Ox)
\tkzGrphDrawIntg[clés]{nom path}{borne A}{borne B}
% entre deux courbes
\tkzGrphDrawIntg[clés]{nom path 1}[nom path 2]{borne A}{borne B}
% alias FR :
\tkzGrphIntegrale[clés]{...}
```

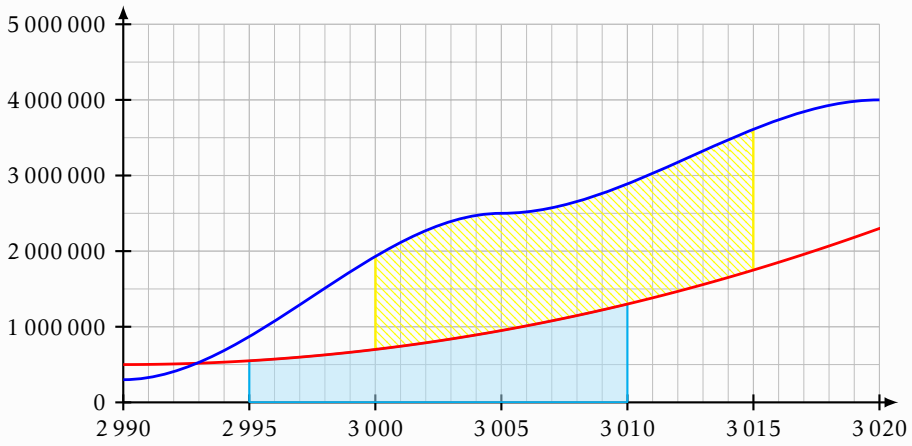
Les bornes sont des **valeurs réelles** ou des **nœuds TikZ** (`(ITSC-1)`). Le paramètre optionnel `[nom path 2]` active le mode *entre deux courbes*.

💣 Nécessite la librairie Ti*k*Z `spath3` (`\usetikzlibrary{spath3}`).

Les `[clés]` disponibles sont :

— `Couleurs` / `colors` : couleur(s) (`bord/fond` ou unique), `gray` par défaut ;
— `Style` / `style` : `fill` (défaut) ou `hachures` ;
— `Hachures` / `hatch` : pattern TikZ, `north west lines` par défaut ;
— `Opacite` / `opacity` : opacité du fond, `0.5` par défaut ;
— `Bord` / `border` : booléen, `true` par défaut, traits verticaux aux bornes ;
— `Bornes` / `bound` : `abs` (défaut) ou `noeuds`.

```
\begin{tkzGrph}%
    <Xmin=2990,Xmax=3020,Xgrille=5,Xgrilles=1,
    Ymin=0,Ymax=5000000,Ygrille=1000000,Ygrilles=500000,
    Largeur=10cm,Hauteur=5cm,Origx=2990>
  \tkzGrphDrawGridAxis[enlarge=2.5mm,font=\small]%
    {2990,2995,3000,3005,3010,3015,3020}%
    {0,1000000,2000000,3000000,4000000,5000000}
  \tkzGrphDefineCurve[draw=false,name=cf]<f>{2000*(x-2990)^2+500000}
  \tkzGrphDefineSpline[alt,draw=false,name=sp]%
    {2990/300000/0.5 § 3005/2500000/0 § 3020/4000000/0.8}
  % sous une courbe
  \tkzGrphDrawIntg[colors=cyan/cyan!30]{cf}{2995}{3010}
  % entre deux courbes
  \tkzGrphDrawIntg[colors=yellow,style=hachures]{sp}[cf]{3000}{3015}
  \tkzGrphDrawExistingCurves{cf/red,sp/blue}
\end{tkzGrph}
```



6.2 Sommes de Riemann : `\tkzGrphDrawRiemannSum`

Trace les rectangles ou trapèzes d'une somme de Riemann sur $[a, b]$ avec n subdivisions.

```
\tkzGrphDrawRiemannSum[cles]<nomCourbe>{a}{b}{n}
% alias : \tkzGrphSommesRiemann   \tkzgrsmsum
% raccourcis directs (Type fixé) :
\tkzGrphRiemannGauche[cles]<nomCourbe/fct>{a}{b}{n}
\tkzGrphRiemannDroite[cles]<nomCourbe/fct>{a}{b}{n}
\tkzGrphRiemannMilieu[cles]<nomCourbe/fct>{a}{b}{n}
\tkzGrphRiemannTrapeze[cles]<nomCourbe/fct>{a}{b}{n}
```

— `<nomCourbe>` : nom du `name path` TikZ (ex. `cf`) en mode path (défaut), ou nom de la fonction `xint` si `Formule` ;

— `{a}`, `{b}` : bornes réelles de l'intervalle ;

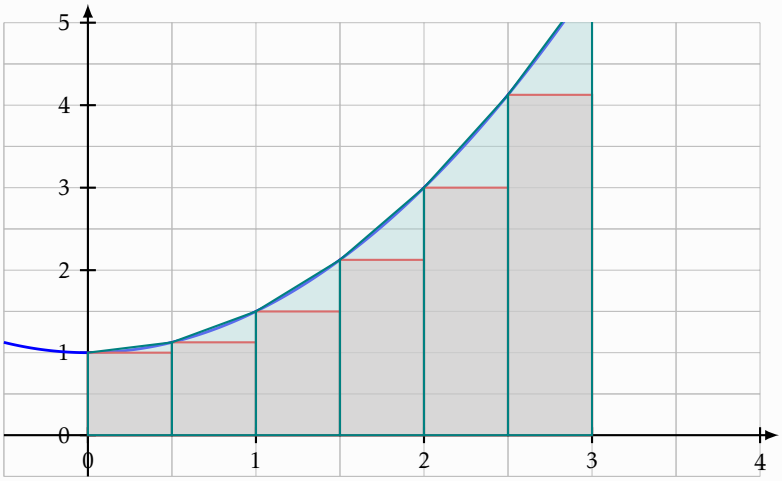
— `{n}` : nombre de subdivisions.

Les `[clés]` disponibles (famille `GraphiqueTikzAltRSM`) :

Clé FR/EN	Valeurs	Défaut
`Type` / `type`	`Gauche`/`left`, `Droite`/`right`, `Milieu`/`mid`, `Trapeze`/`trap`	`Gauche`
`Couleur` / `color`	couleur TikZ	`gray!40`
`CouleurBord` / `bc`	couleur TikZ	`black`
`Opacite` / `opacity`	réel $\in [0,1]$	`0.5`
`Formule` / `formula`	booléen	`false`

Comme pour `\tkzGrphDefPointBy`, la clé `Formule` bascule en mode analytique (calcul xint, plus précis) : `<nomCourbe>` doit alors être le nom d'une fonction déclarée via `\tkzGrphDefineCurve<f>{...}`.

```
\begin{tkzGrph}<Xmin=-0.5,Xmax=4,Ymin=-0.5,Ymax=5,
  Largeur=10cm,Hauteur=6cm>
  \tkzGrphDrawGridAxis[enlarge=2.5mm,font=\small]%
  {0,1,...,4}{0,1,...,5}
  \tkzGrphDefineCurve[trace,name=cf,color=blue]<f>{0.5*x^2+1}
  % rectangles gauche (mode path)
  \tkzGrphDrawRiemannSum[Type=Gauche,color=red!30,bc=red]<cf>{0}{3}{6}
  % trapèzes (mode Formule)
  \tkzGrphDrawRiemannSum[Type=Trapeze,Formule,color=teal!30,bc=teal]<f>{0}{3}{6}
\end{tkzGrph}
```



6.3 Intersections : \tkzGrphFindItsc

```
\tkzGrphFindItsc[clés]{nom path 1}{nom path 2}<nb>
% alias FR :
\tkzGrphIntersections[clés]{nom path 1}{nom path 2}<nb>
```

Les nœuds sont nommés S-1, S-2…(ou le préfixe choisi via Nom/name).

Les [clés] disponibles sont :

— Nom / name : préfixe des nœuds, S par défaut ;

— Couleur / color : couleur des points et traits ;

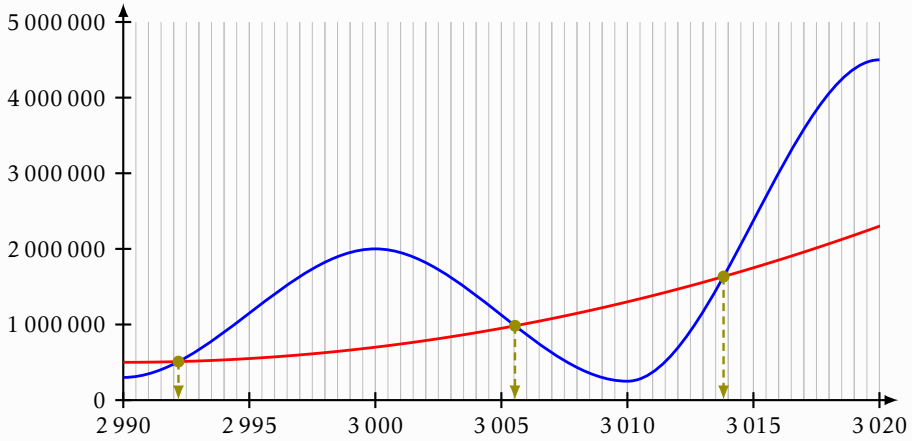
— Aff / show : booléen, true par défaut ;

— Traits / lines : booléen, traits vers les axes ;

— ListeRes / reslist : nom d'une macro pour la liste des abscisses réelles.

```
\begin{tkzGrph}<Xmin=2990,Xmax=3020,Ymin=0,Ymax=5000000,
               Largeur=10cm,Hauteur=5cm,Origx=2990>
  \tkzGrphDrawGridAxis[enlarge=2.5mm,font=\small]%
    {2990,2995,3000,3005,3010,3015,3020}%
    {0,1000000,2000000,3000000,4000000,5000000}
  \tkzGrphDefineCurve[name=cf]<f>{2000*(x-2990)^2+500000}
  \tkzGrphDefineSpline[alt,name=sp]%
    {2990/300000/0.5 § 3000/2000000/0 § 3010/250000/0 § 3020/4500000/0.8}
  \tkzGrphDrawExistingCurves{cf/red,sp/blue}
  \tkzGrphFindItsc[show,lines,reslist=LISTRES,color=olive]{cf}{sp}
\end{tkzGrph}

Solutions : \LISTRES
```



Solutions : 2992.1899,3005.5396,3013.8147

6.4 Images et antécédents

6.4.1 Image : \tkzGrphFindY

```
\tkzGrphFindY[clés]{nom path}{x réel}<nom nœud>
% alias FR :
\tkzGrphTrouverY[clés]{nom path}{x réel}<nom nœud>
```

Intersection avec la droite verticale $x = x_{\text{réel}}$. Le nœud `<nom nœud>` est réutilisable ensuite.

6.4.2 Images : \tkzGrphFindListY

```
\tkzGrphFindListY[clés]{nom path}{x1,x2,...}
% alias FR :
\tkzGrphTrouverListeY[clés]{nom path}{x1,x2,...}
```

6.4.3 Antécédents : \tkzGrphFindX

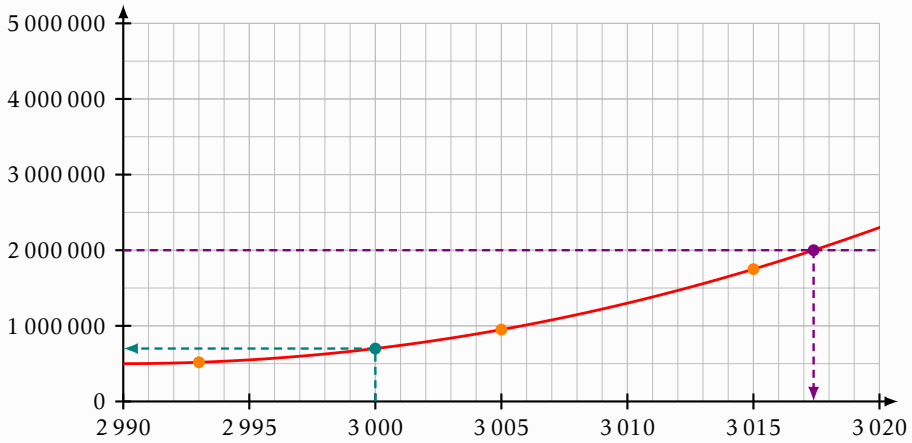
```
\tkzGrphFindX[clés]{nom path}{y réel}<nb>
% alias FR :
\tkzGrphTrouverX[clés]{nom path}{y réel}<nb>
```

`<nb>` reçoit le nombre de solutions. Clé `reslist`/`ListeRes` pour stocker la liste des abscisses.
Les `[clés]` communes à `\tkzGrphFindY` et `\tkzGrphFindX` :

- `Couleur` / `color` : couleur du point et des traits ;
- `Style` / `style` : style du point (`o`, `x`, `ggb`...) ;
- `Aff` / `show` : booléen, `true` par défaut ;
- `Traits` / `lines` : booléen, traits vers les axes ;
- `AffDroite` / `showline` : booléen, affiche la droite horizontale/verticale.

```
\begin{tkzGrph}%
    <Xmin=2990,Xmax=3020,Xgrille=5,Xgrilles=1,
    Ymin=0,Ymax=5000000,Ygrille=1000000,Ygrilles=500000,
    Largeur=10cm,Hauteur=5cm,Origx=2990>
  \tkzGrphDrawGridAxis[enlarge=2.5mm,font=\small]%
    {2990,2995,3000,3005,3010,3015,3020}%
    {0,1000000,2000000,3000000,4000000,5000000}
  \tkzGrphDefineCurve[trace,name=cf,color=red]<f>{2000*(x-2990)^2+500000}
  % image en x=3000
  \tkzGrphFindY[show,lines,color=teal]{cf}{3000}
  % images multiples
  \tkzGrphFindListY[show,color=orange]{cf}{2993,3005,3015}
  % antécédents de y=2000000
  \tkzGrphFindX%
    [show,lines,showline,color=violet,reslist=ANTS]{cf}{2000000}
\end{tkzGrph}

Antécédents : \ANTS
```

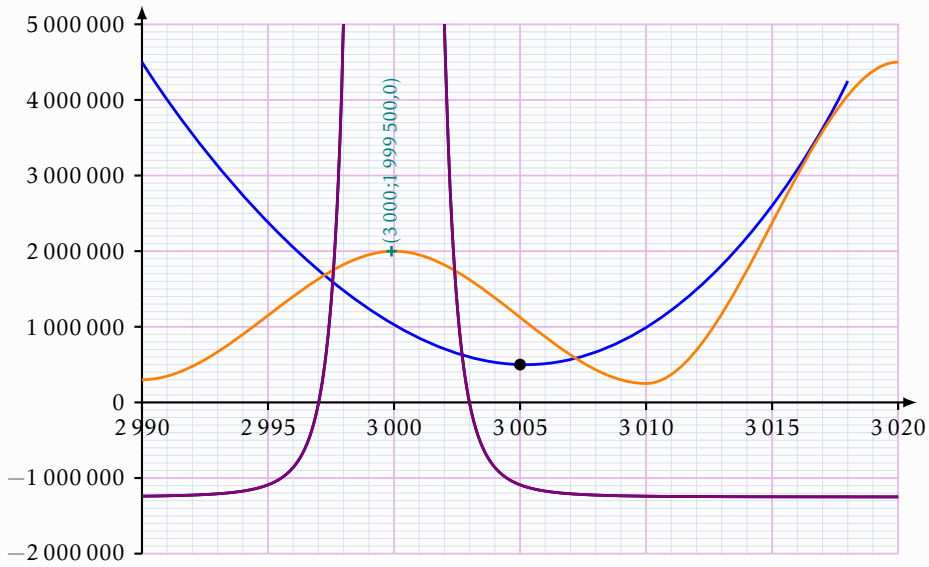


Antécédents : 3017.3875

6.5 Extremums : \tkzGrphFindMin/Max

```
\tkzGrphFindMin[clés]{nom path}[nœud]
\tkzGrphFindMax[clés]{nom path}[nœud]
```

```
\begin{tkzGrph}%
  <xmin=2990,Xmax=3020,Xgrille=5,Xgrilles=1,%
  Ymin=-2000000,Ymax=5000000,Ygrille=1000000,Ygrilles=100000,Origx=2990,%
  Largeur=10cm,Hauteur=70mm,Theme=seyes,TailleGrad=0pt/1mm>
  \tkzGrphDrawGridAxis[enlarge=2.5mm,font=\small]%
  {2990,2995,3000,3005,3010,3015,3020}%
  {-2000000,-1000000,0,1000000,2000000,3000000,4000000,5000000}
  \tkzGrphDefineCurve[name=cf,trace,forbiddenvalues={3000}]%
    <f>{-1250000 + 100000000/(x-3000)^4}
  \tkzGrphDefineInterpoCurve[tension=1,name=interpo]%
    {2990/4500000 § 3005/500000 § 3018/4250000}
  \tkzGrphDefineSpline[alt,name=spline]%
    { 2990/300000/0.5 § 3000/2000000/0 § 3010/250000/0 § 3020/4500000/0.8 }
  \tkzGrphDrawExistingCurves{interpo/blue,spline/orange,cf/violet}
  %  \tkzGrphFindItsc[show,lines,reslist=LISTRES,Couleur=olive]{cf}{spline}
  %max/min
  \tkzGrphFindMin{interpo}
  \tkzGrphFindMax[start=2990,end=3010,color=teal,style=+]{spline}
  \tkzGrphGetXY{(grph-max)}[\mylocalxmax][\mylocalymax]
  \draw[teal] (grph-max) node[right,rotate=90,scale=0.75]
  ↪  {(\ArrondirNum[0]{\mylocalxmax};\ArrondirNum[0]<exponent-mode=fixed>{\mylocalymax})} ;
\end{tkzGrph}
```



À noter que le temps de compilation peut être accru lors de la recherche d'extrema… Dans ce cas, on peut utiliser une technique dichotomique, pus rapide mais moins fiable, en priorisant l'étude sur un intervalle sur lequel la fonction est unimodale, en sachant que le nombre d'itérations peut être spécifié.

```
\tkzGrphFindMinDichoto[clés,Iter=...]{nom path}[nœud]
\tkzGrphFindMaxDichoto[clés,Iter=...]{nom path}[nœud]
```

6.6 Points d'intérêt : `\tkzGrphInterestPoints`

La commande `\tkzGrphInterestPoints` est un *wrapper* qui combine en une seule commande la recherche des **racines**, du **maximum global** et du **minimum global** d'une courbe définie via un `name path`.

Elle est compatible avec les trois types de courbes (`\tkzGrphDefineCurve`, `\tkzGrphDefineSpline`, `\tkzGrphDefineInterpoCurve`).

💣 Les *limitations* sont les mêmes que pour `\tkzGrphFindMin/Max` :

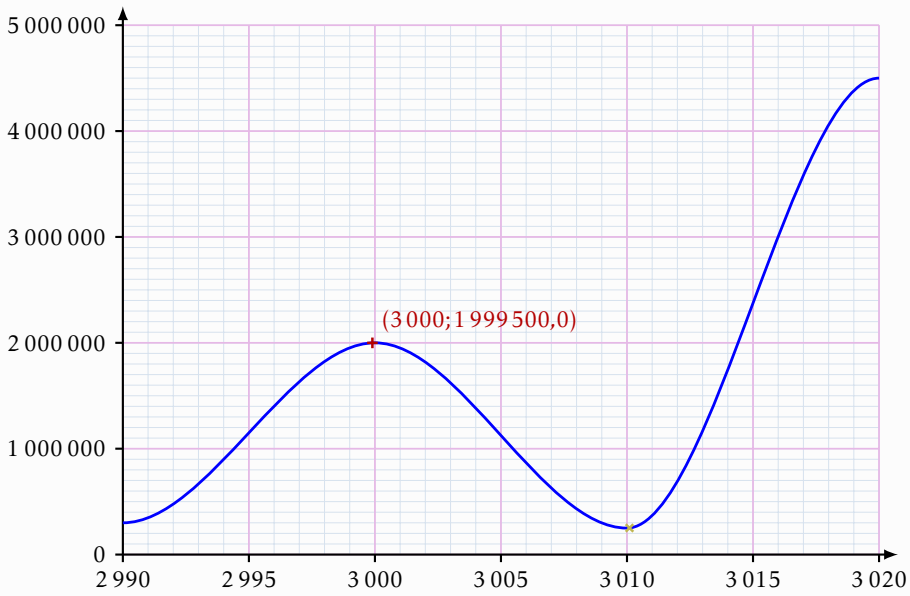
— pas de gestion d'extremums multiples (seul le global sera traité) ;
— pour des courbes sans minimum visible, utiliser `Minimum=false` ;
— le temps de compilation peut être plus long.

```
%dans l'environnement tkzGrph
\tkzGrphInterestPoints[clés]{nom path}
% alias FR :
\tkzGrphPointsInteret[clés]{nom path}
```

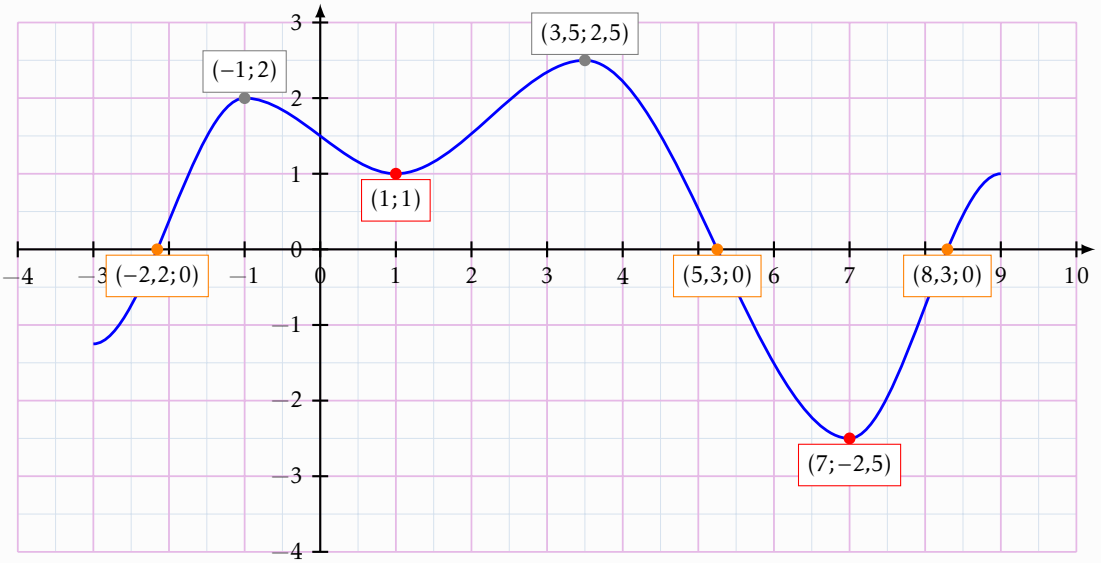
Les `[clés]`, optionnelles, disponibles sont :

— `Couleur` / `color` : couleur globale (`black` par défaut) ;
— `CouleurRacines` / `colorroots` : couleur des racines (défaut → `Couleur`) ;
— `CouleurMax` / `colormax` : couleur du maximum (défaut → `Couleur`) ;
— `CouleurMin` / `colormin` : couleur du minimum (défaut → `Couleur`) ;
— `Style` / `style` : style global des points (`o` par défaut) ;
— `StyleRacines` / `styleroots` : style des racines (défaut → `Style`) ;
— `StyleMax` / `stylemax` : style du maximum (défaut → `Style`) ;
— `StyleMin` / `stylemin` : style du minimum (défaut → `Style`) ;
— `Local` / `local` : booléen pour les extrema locaux, en *test* (`false` par défaut) ;
— `Racines` / `roots` : booléen (`true` par défaut) ;
— `Maximum` / `maximum` : booléen (`true` par défaut) ;
— `Minimum` / `minimum` : booléen (`true` par défaut) ;
— `NomMax` / `namemax` : nom du nœud du maximum (`grph-max` par défaut) ;
— `NomMin` / `namemin` : nom du nœud du minimum (`grph-min` par défaut) ;
— `NomRacines` / `nameroots` : préfixe des nœuds des racines (`grph-rac` par défaut) ;
— `Debut` / `start` : début de la recherche (fenêtre complète par défaut) ;
— `Fin` / `end` : fin de la recherche (fenêtre complète par défaut).

```
\begin{tkzGrph}%
  <Xmin=2990,Xmax=3020,Xgrille=5,Xgrilles=1,%
  Ymin=0,Ymax=5000000,Ygrille=1000000,Ygrilles=100000,Origx=2990,%
  Largeur=10cm,Hauteur=70mm,Theme=seyes,TailleGrad=0pt/1mm>
  \tkzGrphDrawGridAxis[enlarge=2.5mm,font=\small]%
    {2990,2995,3000,3005,3010,3015,3020}%
    {0,1000000,2000000,3000000,4000000,5000000}
  \tkzGrphDefineSpline[alt,name=spline]%
    { 2990/300000/0.5 § 3000/2000000/0 § 3010/250000/0 § 3020/4500000/0.8 }
  \tkzGrphDrawExistingCurves{spline/blue}
  % points d'intérêt de la spline
  \tkzGrphInterestPoints[%
    Debut=2995,Fin=3012,%
    CouleurMax=red!70!black,StyleMax=+,
    CouleurMin=yellow!70!black,StyleMin=x,
    NomMax=monmax,NomMin=monmin%
    ]{spline}
  % récupération des coordonnées du maximum
  \tkzGrphGetXY{(monmax)}[\xmax][\ymax]
  \node[above right,font=\small,red!70!black] at (monmax)
    {$(\ArrondirNum[0]{\xmax};%
      \ArrondirNum[0]<exponent-mode=fixed>{\ymax})$} ;
\end{tkzGrph}
```



```
\begin{tkzGrph}%
  <Xmin=-4,Xmax=10,Xgrille=1,Xgrilles=0.5,%
  Ymin=-4,Ymax=3,Ygrille=1,Ygrilles=0.5,%
  Largeur=14cm,Hauteur=7cm,Theme=seyes>
  \tkzGrphDrawGridAxis[enlarge=2.5mm,font=\small]{-4,-3,...,10}{-4,-3,...,3}
  \tkzGrphDefineSpline[alt,name=spline,Couleur=red]%
    { -3/-1.25/0 § -1/2/0 § 1/1/0 § 3.5/2.5/0 § 7/-2.5/0 § 9/1/0}
  \tkzGrphDrawExistingCurves{spline/blue}  % points d'intérêt de la spline
  %\tkzGrphFindMax[Aff=true]{spline}[\pflpinommax]%
  \tkzGrphPointsInteret[Local,CouleurMax=gray,CouleurMin=red,CouleurRacines=orange]{spline}
  %tests
  \foreach \i in {1,...,\pflstlocalmaxn}{%
    \tkzGrphGetXY{(grph-max-\i)}[\tmpx][\tmpy]%
    \node[draw=gray,fill=white,above=2pt,font=\small] at (grph-max-\i)
    {$(\ArrondirNum[1]{\tmpx};\ArrondirNum[1]{\tmpy})$} ;%
  }%
  \foreach \i in {1,...,\pflstlocalminn}{%
    \tkzGrphGetXY{(grph-min-\i)}[\tmpx][\tmpy]%
    \node[draw=red,fill=white,below=2pt,font=\small] at (grph-min-\i)
    {$(\ArrondirNum[1]{\tmpx};\ArrondirNum[1]{\tmpy})$} ;%
  }%
  \foreach \i in {1,...,\myantec}{%
    \tkzGrphGetXY{(grph-rac-\i)}[\tmpx][\tmpy]%
    \node[draw=orange,fill=white,below=2pt,font=\small] at (grph-rac-\i)
    {$(\ArrondirNum[1]{\tmpx};\ArrondirNum[1]{\tmpy})$} ;%
  }
\end{tkzGrph}
```



6.7 Point sur une courbe : \tkzGrphDefPointBy

Définit un nœud TikZ à l'abscisse réelle x_0 sur une courbe (mode path ou mode analytique) et expose les coordonnées réelles du point calculé.

```
\tkzGrphDefPointBy[cles]<nomCourbe>{x0}[nomNoeud]
% alias : \tkzGrphDefinirPointPar   \tkzgdefptby
```

— `<nomCourbe>` : nom du `name path` TikZ (ex. `cf`) en mode path (défaut), ou nom de la fonction `xint` si `Formule` ;

— `{x0}` : abscisse réelle du point à définir ;

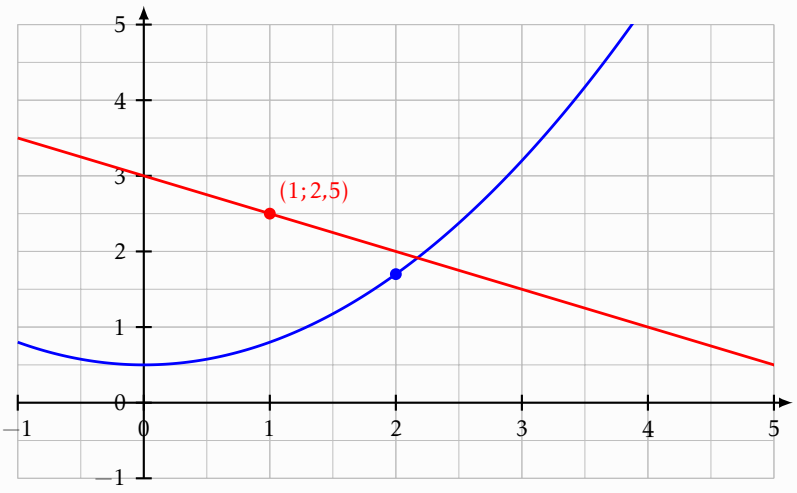
— `[nomNoeud]` : nom du nœud TikZ créé (`tkzgrphpt` par défaut).

La seule clé disponible (famille `GraphiqueTikzAltDefPtBy`) est `Formule` / `formula` (booléen, `false` par défaut). En mode `Formule`, `<nomCourbe>` est le nom d'une fonction déclarée par `\tkzGrphDefineCurve<f>{...}` et le calcul est analytique (plus précis). En mode path (défaut), l'ordonnée est obtenue par intersection géométrique. Après l'appel, deux macros sont renseignées :

— `\tkzGrphLastX` : abscisse réelle (= x_0) ;

— `\tkzGrphLastY` : ordonnée réelle calculée $f(x_0)$.

```
\begin{tkzGrph}<Xmin=-1,Xmax=5,Ymin=-1,Ymax=5,
  Largeur=10cm,Hauteur=6cm>
  \tkzGrphDrawGridAxis[enlarge=2.5mm,font=\small]%
  {-1,0,...,5}{-1,0,...,5}
  % mode path (defaut) : ordonnee par intersection geometrique
  \tkzGrphDefineCurve[trace,name=cf,color=blue]{0.3*x^2+0.5}
  \tkzGrphDefPointBy<cf>{2}[ptA]
  \tkzGrphMarkPts*[Style=ggb,Couleur=blue]{(ptA)}
  % mode Formule : calcul analytique via xint
  \tkzGrphDefineCurve[trace,name=cg,color=red]<g>{-0.5*x+3}
  \tkzGrphDefPointBy[Formule]<g>{1}[ptB]
  \tkzGrphMarkPts*[Style=ggb,Couleur=red]{(ptB)}
  % utilisation de \tkzGrphLastX / \tkzGrphLastY
  \node[above right,font=\small,red] at (ptB) {$(\tkzGrphLastX;\ArrondirNum[1]{\tkzGrphLastY})$} ;
\end{tkzGrph}
```



7 Nuage et régression : statistiques

7.1 Nuage de points : \tkzGrphScatterPlot

```
\tkzGrphScatterPlot[clés]{liste X}{liste Y}
% alias FR :
\tkzGrphNuage[clés]{liste X}{liste Y}
```

7.2 Nuage depuis CSV : \tkzGrphNuageCSV

Cette commande trace directement un nuage de points à partir d'un fichier CSV, sans avoir à construire les listes X/Y à la main. Elle nécessite le package `csvsimple`.

```
\tkzGrphNuageCSV[clés](séparateur){fichier.csv}{colonneX}{colonneY}
% alias :
\tkzgNuageCSV    \tkzGrphScatterCSV
```

Le séparateur est passé entre parenthèses (défaut : `semicolon`) ; les valeurs possibles sont celles de `csvsimple` : `semicolon`, `comma`, `pipe`, `tab`, `bar`.
Les `[clés]` disponibles sont :

- `Couleur` / `color` : couleur des points (`black` par défaut) ;
- `Style` / `style` : style de point — `o`, `x`, `+`, `c`, `*`, `ggb`, `do` (`o` par défaut) ;
- `Epaisseur` / `thick` : épaisseur du tracé (`0.95pt` par défaut) ;
- `MaxLignes` : nombre maximum de lignes du CSV à lire (`100000` par défaut).

```
% fichier donnees.csv (séparateur ;) :
%   x;y
%   -4;17.2
%   -3;9.8
%   ...

\begin{tkzGrph}[x=0.65cm,y=0.25cm]%
    <Xmin=-5,Xmax=6,Ymin=-1,Ymax=26,Xgrille=1,Ygrille=5>
  \tkzgaxisgrid[enlarge=2mm,font=\small]{auto}{auto}
  % nuage brut depuis CSV
  \tkzGrphNuageCSV[Couleur=blue,Style=o]{donnees.csv}{x}{y}
  % séparateur virgule, style croix, couleur rouge
  \tkzGrphNuageCSV[Couleur=red,Style=x](comma){autres.csv}{taille}{masse}
  % deux séries depuis le même CSV (colonnes Y1 et Y2)
  \tkzGrphNuageCSV[Couleur=teal,Style=+]{deux-series.csv}{X}{Y1}
  \tkzGrphNuageCSV[Couleur=violet,Style=o]{deux-series.csv}{X}{Y2}
\end{tkzGrph}
```

La commande `\tkzGrphListesCSV` (alias `\tkzgListesCSV`) charge les colonnes dans deux macros-listes réutilisables, par exemple pour combiner avec `\tkzGrphDrawRegLin` ou `\tkzGrphDrawFit` :

```
\tkzGrphListesCSV[clés](séparateur){fichier.csv}{colonneX}{colonneY}%
  {nomMacroX}{nomMacroY}
% alias : \tkzgListesCSV
```

```
% chargement des listes hors environnement
\tkzGrphListesCSV(semicolon){donnees.csv}{x}{y}{MesX}{MesY}

\begin{tkzGrph}[x=0.65cm,y=0.25cm]%
    <Xmin=-5,Xmax=6,Ymin=-1,Ymax=26,Xgrille=1,Ygrille=5>
  \tkzgaxisgrid[enlarge=2mm,font=\small]{auto}{auto}
  \tkzGrphNuageCSV[Couleur=blue]{donnees.csv}{x}{y}
  \tkzGrphDrawRegLin[red,thick]{\MesX}{\MesY}
\end{tkzGrph}
```

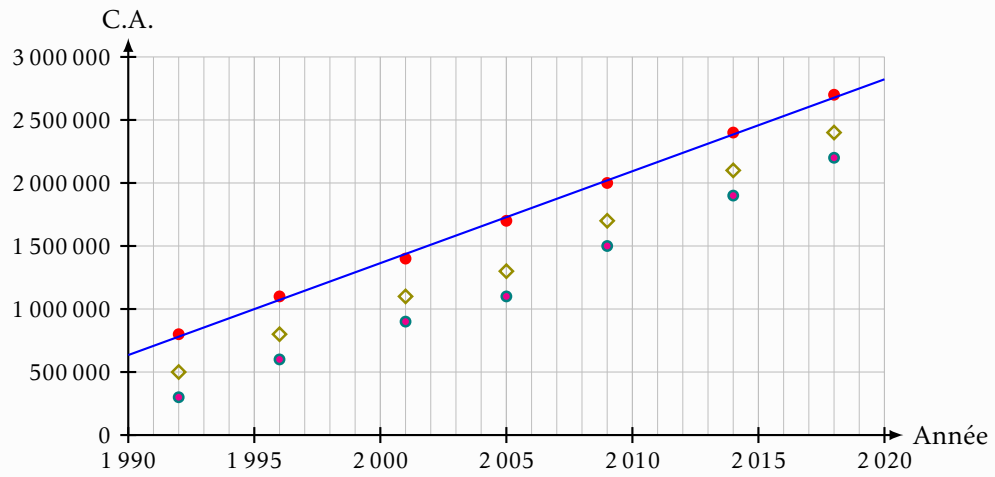
7.3 Droite de régression : \tkzGrphDrawRegLin

```
\tkzGrphDrawRegLin[clés tikz]{liste X}{liste Y}
% alias FR :
\tkzGrphRegLin[clés tikz]{liste X}{liste Y}
```

Calcule et trace la droite de régression par la méthode des moindres carrés. Les coefficients sont stockés dans `\pflstrega` (a) et `\pflstregb` (b).

```
\begin{tkzGrph}
    <Xmin=1990,Xmax=2020,Ymin=0,Ymax=3000000,Largeur=10cm,Hauteur=5cm,%
    Xgrille=5,Xgrilles=1,Ygrille=500000,Origx=1990>
  \tkzGrphDrawGridAxis[enlarge=2.5mm,font=\small]{1990,1995,2000,2005,2010,2015,2020}%
    {0,500000,1000000,1500000,2000000,2500000,3000000}
  \tkzGrphScatterPlot[color=red]{1992,1996,2001,2005,2009,2014,2018}%
    {800000,1100000,1400000,1700000,2000000,2400000,2700000}
  \tkzGrphScatterPlot[color=olive,style=do]{1992,1996,2001,2005,2009,2014,2018}%
    {500000,800000,1100000,1300000,1700000,2100000,2400000}
  \tkzGrphScatterPlot[color=teal/magenta,style=ggb]{1992,1996,2001,2005,2009,2014,2018}%
    {300000,600000,900000,1100000,1500000,1900000,2200000}
  \tkzGrphDrawRegLin[blue,thick]{1992,1996,2001,2005,2009,2014,2018}%
    {800000,1100000,1400000,1700000,2000000,2400000,2700000}
  %labels
  \tkzGrphPlaceTexte[Position=right]{(grph-axox-ee)}{Année}
  \tkzGrphPlaceTexte[Position=above]{(grph-axoy-nn)}{C.A.}
\end{tkzGrph}

Droite : $y = \ArrondirNum[1]{\pflstrega} x \ArrondirNum[0]<exponent-mode=fixed>{\pflstregb}$
```

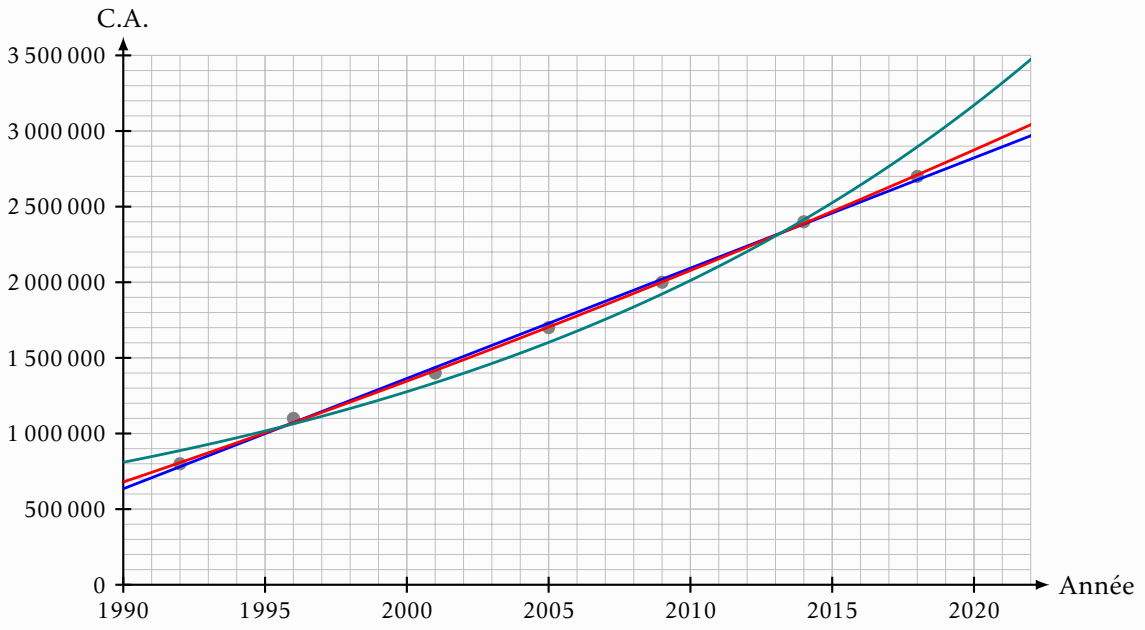


Droite : $y = 72\,932{,}300x - 144\,500\,750$

7.4 Régressions : \tkzGrphDrawFit

```
\tkzGrphDrawFit%
  [clés courbe]<nom courbe>{type}<arrondis>{listeX}{listeY}
% alias FR :
\tkzGrphTracerAjustement
```

```
\begin{tkzGrph}%
    <Xmin=1990,Xmax=2022,Ymin=0,Ymax=3500000,
    Largeur=12cm,Hauteur=7cm,
    Xgrille=5,Xgrilles=1,
    Ygrille=500000,Ygrilles=100000,
    Origx=1990>
  \tkzGrphDrawGridAxis[enlarge=2.5mm,font=\small,format=year/num]%
    {1990,1995,2000,2005,2010,2015,2020}%
    {0,500000,1000000,1500000,2000000,2500000,3000000,3500000}
  %data
  \def\LX{1992,1996,2001,2005,2009,2014,2018}
  \def\LY{800000,1100000,1400000,1700000,2000000,2400000,2700000}
  % nuage de points
  \tkzGrphScatterPlot[color=gray,style=o,size=2pt]{\LX}{\LY}
  % ajustement linéaire
  \tkzGrphDrawFit[color=blue,trace]<ajlin>{lin}{\LX}{\LY}
  % ajustement quadratique
  \tkzGrphDrawFit[color=red,trace]<ajquad>{quad}{\LX}{\LY}
  % ajustement exponentiel alt
  \tkzGrphDrawFit[color=teal,trace]<ajexp>{expalt}{\LX}{\LY}
  % label axe
  \tkzGrphPlaceTexte[Position=right]{(grph-axox-ee)}{Année}
  \tkzGrphPlaceTexte[Position=above]{(grph-axoy-nn)}{C.A.}
\end{tkzGrph}
```



8 Compléments sur les axes

8.1 Valeurs manuelles sur les axes

Ces commandes permettent d'ajouter manuellement des valeurs ou des labels qualitatifs sur les axes, en complément de `\tkzGrphDrawGridAxis`.

```
% valeurs numériques formatées
\tkzGrphAddValX[clés]{liste valeurs X réelles}{liste labels}
\tkzGrphAddValY[clés]{liste valeurs Y réelles}{liste labels}
% alias FR :
\tkzGrphAjouterValeursX / \tkzGrphAjouterValeursY

% labels qualitatifs (avec rotation possible)
\tkzGrphAddLabelsX[clés]{liste valeurs X réelles}{liste labels}
\tkzGrphAddLabelsY[clés]{liste valeurs Y réelles}{liste labels}
% alias FR :
\tkzGrphAjouterLabelsX / \tkzGrphAjouterLabelsY
```

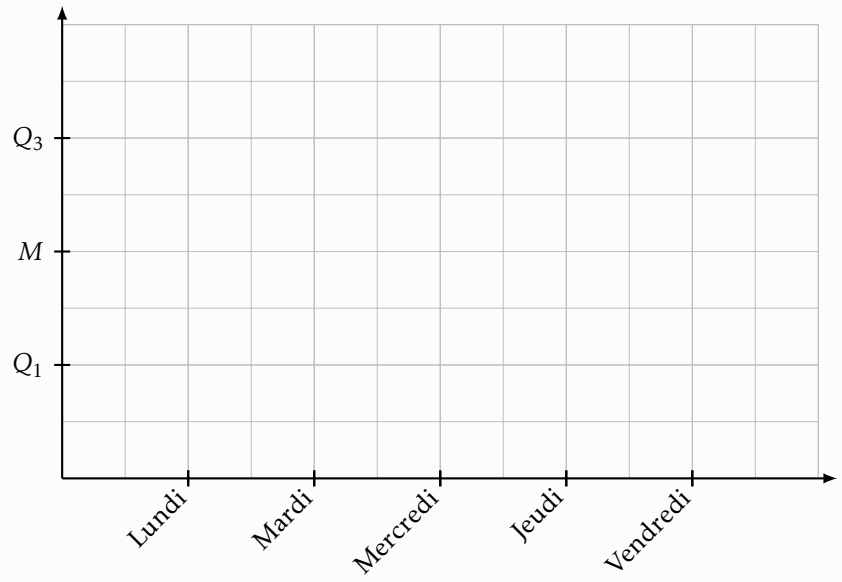
Les `[clés]` disponibles sont :

— `Traits` / `thicks` : booléen, affiche les tirets de graduation ;

— `Rotation` / `angle` : angle de rotation des labels (`0` par défaut) ;

— `Police` / `font` : police des labels.

```
\begin{tkzGrph}<Xmin=0,Xmax=6,Ymin=0,Ymax=4,
               Largeur=10cm,Hauteur=6cm>
  \tkzGrphDrawGridAxis[Grads=false,enlarge=2.5mm]{}{}
  % valeurs numériques personnalisées
  \tkzGrphAddValX[Traits]{1,2,3,4,5}{,,,,}
  \tkzGrphAddValY[Traits]{1,2,3}{$Q_1$,$M$,$Q_3$}
  % labels qualitatifs avec rotation
  \tkzGrphAddLabelsX[Rotation=45]{1,2,3,4,5}%
    {Lundi,Mardi,Mercredi,Jeudi,Vendredi}
\end{tkzGrph}
```



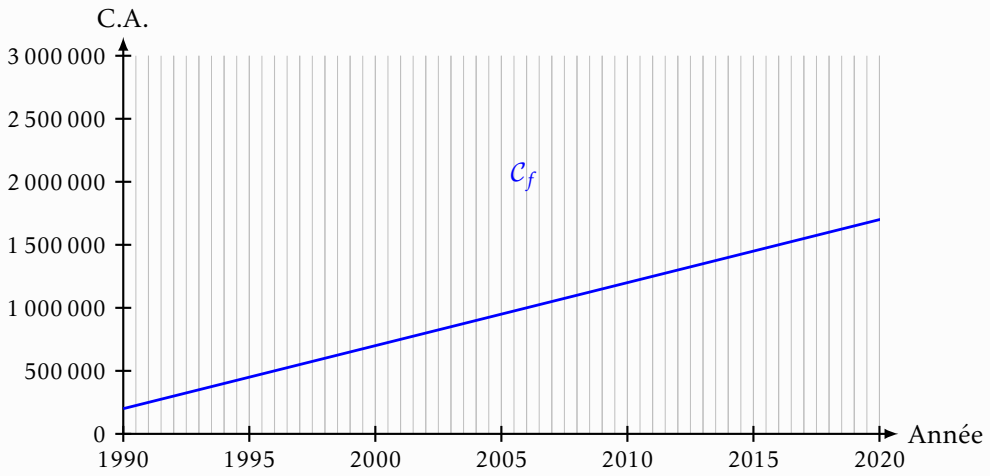
8.2 Placer un texte : \tkzGrphPlaceText

```
% coordonnées réelles
\tkzGrphPlaceText[clés]{({x},{y})}{texte}
% nœud TikZ existant
\tkzGrphPlaceText[clés]{(nom-noeud)}{texte}
% alias FR :
\tkzGrphPlaceTexte[clés]{...}{texte}
```

Les coordonnées réelles doivent être protégées avec `({x},{y})` pour éviter les conflits de parsing. Les nœuds existants (`grph-axox-ee`, `grph-axoy-nn`...) sont utilisables directement.

Les `[clés]` disponibles sont : `Couleur`/`color`, `Police`/`font`, `Position`/`anchor`, `DecalX`/`shiftx`, `DecalY`/`shifty`.

```
\begin{tkzGrph}<Xmin=1990,Xmax=2020,Ymin=0,Ymax=3000000,
                Largeur=10cm,Hauteur=5cm,Origx=1990>
  \tkzGrphDrawGridAxis[enlarge=2.5mm,font=\small,Format=annee/num]%
    {1990,1995,2000,2005,2010,2015,2020}%
    {0,500000,1000000,1500000,2000000,2500000,3000000}
  \tkzGrphDefineCurve[trace,name=cf,color=blue]<f>{50000*(x-1990)+200000}
  % label sur un nœud existant
  \tkzGrphPlaceTexte[Position=right]{(grph-axox-ee)}{Année}
  \tkzGrphPlaceTexte[Position=above]{(grph-axoy-nn)}{C.A.}
  % label en coordonnées réelles
  \tkzGrphPlaceTexte[Position=above right,color=blue]%
    {({2005},{1850000})}{$\mathcal{C}_f$}
\end{tkzGrph}
```



9 Annotations graphiques

9.1 Label sur un path : \tkzGrphLabelPath

Permet de placer un label directement sur un `name path` à une position donnée en **longueur d'arc** ($t \in [0,1]$).

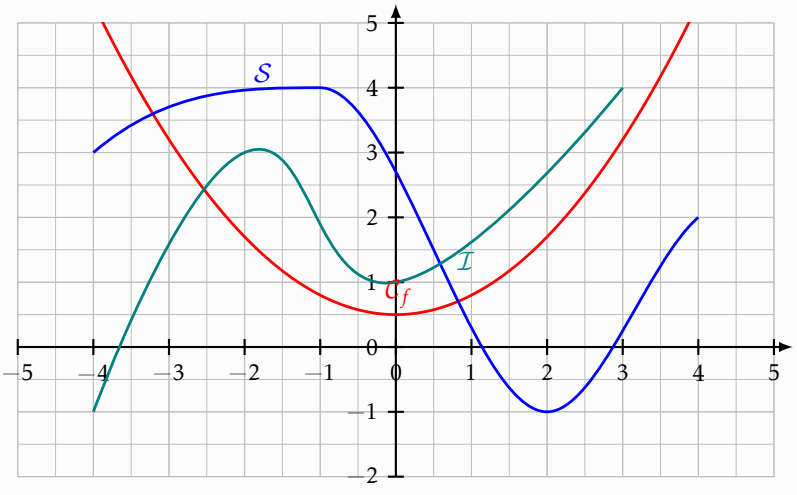
💣 La position `t` est mesurée en longueur d'arc le long du path — `t=0.5` correspond au milieu *géométrique* du tracé, pas au milieu des abscisses.

```
\tkzGrphLabelPath[clés]<options tikz>{nom path}{label}
% alias FR :
\tkzGrphEtiquettePath[clés]<options tikz>{nom path}{label}
```

Les `[clés]` disponibles sont :

— `t` : position en longueur d'arc (`0.5` par défaut) ;
— `Couleur` / `color` : couleur du label (`black` par défaut) ;
— `Police` / `font` : police du label ;
— `Position` / `pos` : positionnement du label (`above` par défaut) ;
— `DecalX` / `shiftx` et `DecalY` / `shifty` : décalages en `pt`.

```
\begin{tkzGrph}<Xmin=-5,Xmax=5,Ymin=-2,Ymax=5,
                Largeur=10cm,Hauteur=6cm>
  \tkzGrphDrawGridAxis[enlarge=2.5mm,font=\small]%
    {-5,-4,...,5}{-2,-1,...,5}
  \tkzGrphDefineCurve[trace,name=cf,color=red]<f>{0.3*x^2+0.5}
  \tkzGrphDefineSpline[trace,alt,name=sp,color=blue]%
    {-4/3/1 § -1/4/0 § 2/-1/0 § 4/2/1}
  \tkzGrphDefineInterpoCurve[trace,name=interpo,color=teal,tension=0.8]%
    {-4/-1 § -2/3 § 0/1 § 3/4}
  % labels au milieu (t=0.5 par défaut)
  \tkzGrphLabelPath[color=red,pos=above]{cf}{$\mathcal{C}_f$}
  % à t=0.25
  \tkzGrphLabelPath[color=blue,pos=above,t=0.25]{sp}{$\mathcal{S}$}
  % à t=0.75
  \tkzGrphLabelPath[color=teal,pos=below,t=0.75]{interpo}{$\mathcal{I}$}
\end{tkzGrph}
```



9.2 Point sur un path : \tkzGrphMarkOnPath

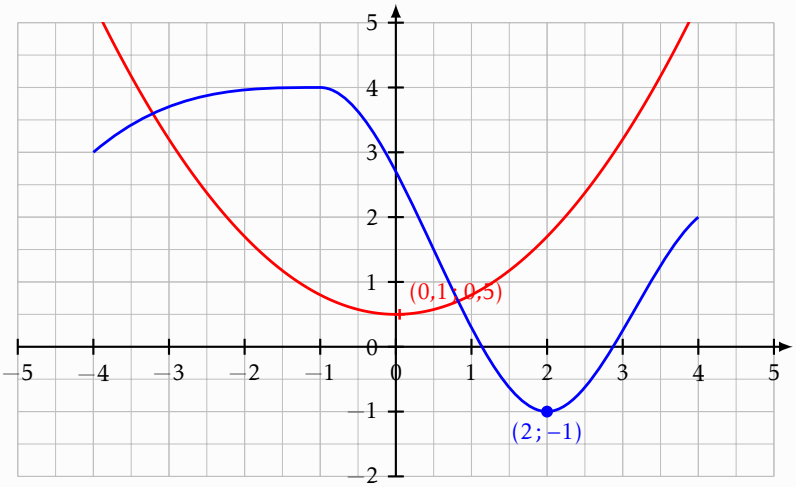
Permet de placer et marquer un point sur un `name path`, à une position en longueur d'arc ou à une abscisse réelle donnée.

```
\tkzGrphMarkOnPath[clés]{nom path}
% alias FR :
\tkzGrphMarquerSurPath[clés]{nom path}
```

Les `[clés]` disponibles sont :

— `t` : position en longueur d'arc (`0.5` par défaut) — si `x` est renseigné, `t` est ignoré ;
— `x` : abscisse réelle — le point est l'intersection avec la droite verticale $x = x_{\text{réel}}$;
— `Couleur` / `color` : couleur du point ;
— `Style` / `style` : style du point (`o`, `x`, `+`, `ggb`...) ;
— `Nom` / `name` : nom du nœud créé (`grph-mop` par défaut) ;
— `Aff` / `show` : booléen, `true` par défaut.

```
\begin{tkzGrph}<Xmin=-5,Xmax=5,Ymin=-2,Ymax=5,
              Largeur=10cm,Hauteur=6cm>
  \tkzGrphDrawGridAxis[enlarge=2.5mm,font=\small]%
    {-5,-4,...,5}{-2,-1,...,5}
  \tkzGrphDefineCurve[trace,name=cf,color=red]<f>{0.3*x^2+0.5}
  \tkzGrphDefineSpline[trace,alt,name=sp,color=blue]%
    {-4/3/1 § -1/4/0 § 2/-1/0 § 4/2/1}
  % mode t : milieu du path
  \tkzGrphMarkOnPath[t=0.5,color=red,style=+,name=ptA]{cf}
  % mode x : abscisse réelle
  \tkzGrphMarkOnPath[x=2,color=blue,style=ggb,name=ptB]{sp}
  % récupérer les coordonnées réelles
  \tkzGrphGetXY{(ptA)}[\xa][\ya]
  \tkzGrphGetXY{(ptB)}[\xb][\yb]
  \node[above right,font=\small,red] at (ptA)
    {$(\ArrondirNum[1]{\xa}\,;\,\ArrondirNum[1]{\ya})$} ;
  \node[below,font=\small,blue] at (ptB)
    {$(\ArrondirNum[1]{\xb}\,;\,\ArrondirNum[1]{\yb})$} ;
\end{tkzGrph}
```



9.3 Intervalle sur un axe : `\tkzGrphIntervalX/Y`

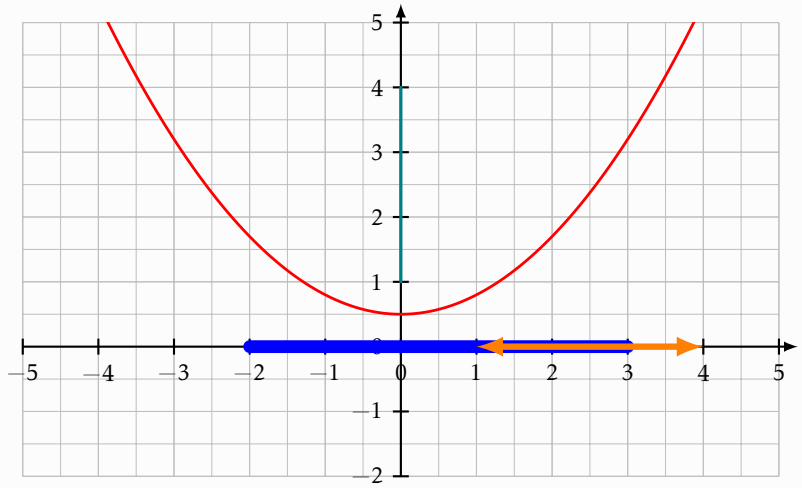
Permet de mettre en évidence un intervalle sur l'axe Ox ou Oy.

```
\tkzGrphIntervalX[clés]{borne a}{borne b}
\tkzGrphIntervalY[clés]{borne a}{borne b}
% alias FR :
\tkzGrphIntervalleSurOx / \tkzGrphIntervalleSurOy
```

Les bornes sont des **valeurs réelles**. Les `[clés]` disponibles sont :

— `Couleur` / `color` : couleur (`black` par défaut) ;
— `Style` / `style` : `segment` (défaut) ou `arrow`/`fleche` ;
— `Epaisseur` / `thick` : `fin`/`thin`, `standard`/`normal` (défaut), `epais`/`thick`.

```
\begin{tkzGrph}<Xmin=-5,Xmax=5,Ymin=-2,Ymax=5,
               Largeur=10cm,Hauteur=6cm>
  \tkzGrphDrawGridAxis[enlarge=2.5mm,font=\small]%
    {-5,-4,...,5}{-2,-1,...,5}
  \tkzGrphDefineCurve[trace,name=cf,color=red]<f>{0.3*x^2+0.5}
  % intervalle sur Ox — style segment épais
  \tkzGrphIntervalX[color=blue,thick=epais]{-2}{3}
  % intervalle sur Ox — style flèche
  \tkzGrphIntervalX[color=orange,style=arrow]{1}{4}
  % intervalle sur Oy — fin
  \tkzGrphIntervalY[color=teal,thick=fin]{1}{4}
\end{tkzGrph}
```



10 Nuage de points de suites

La version alt permet de représenter le **nuage de points** (n, u_n) d'une suite directement dans l'environnement `tkzGrph`, sans calcul préalable. Cinq types de suites sont pris en charge, plus un accès à la toile de récurrence.

10.1 Clés communes

Toutes les commandes de cette section partagent la même famille de clés `GraphiqueTikzAltSuiteNuage` :

Clé FR	Clé EN	Défaut
`Type`	`type`	`Recurrent`
`Debut`	`from`	`0`
`Fin`	`to`	`10`
`Nom`	`name`	`u`
`Couleur`	`color`	`black`
`Style`	`style`	`o`
`Taille`	`size`	`1.75pt`
`Labels`	`labels`	`true`
`TailleLabel`	`labelsize`	`\small`
`PosLabel`	`labelpos`	`below`

Les valeurs possibles pour `Style` sont `o` (disque creux), `x` (croix diagonale), `plus` (croix droite), `ggb` (style GeoGebra), `*` (alias de `x`), `+` (alias de `plus`).
La clé `Labels=false` / `labels=false` supprime les étiquettes u_n sous chaque point.

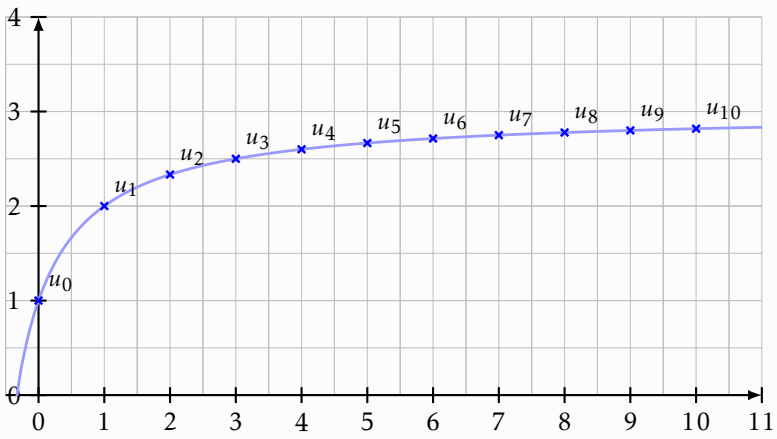
10.2 Suite explicite : \tkzGrphSeqExplicite

Trace le nuage (n, u_n) d'une suite **explicite** $u_n = f(n)$.

```
\tkzGrphSeqExplicite[cles]{fct}
```

`fct` est le nom d'une fonction déclarée au préalable avec `\xintdefloatfunc fct(x) := ... ;`

```
\xintdefloatfunc fexp(x) := (3*x+1)/(x+1) ;
\begin{tkzGrph}<Xmin=-0.5,Xmax=11,Ymin=0,Ymax=4,
               Largeur=10cm,Hauteur=5cm>
  \tkzGrphDrawGridAxis{auto}{auto}
  \tkzGrphDefineCurve[Nom=cf,Trace,Couleur=blue!40]{(3*x+1)/(x+1)}
  \tkzGrphSeqExplicite[Debut=0,Fin=10,Couleur=blue,
    Style=*,Taille=2pt,Labels=true,PosLabel=above right]{fexp}
\end{tkzGrph}
```

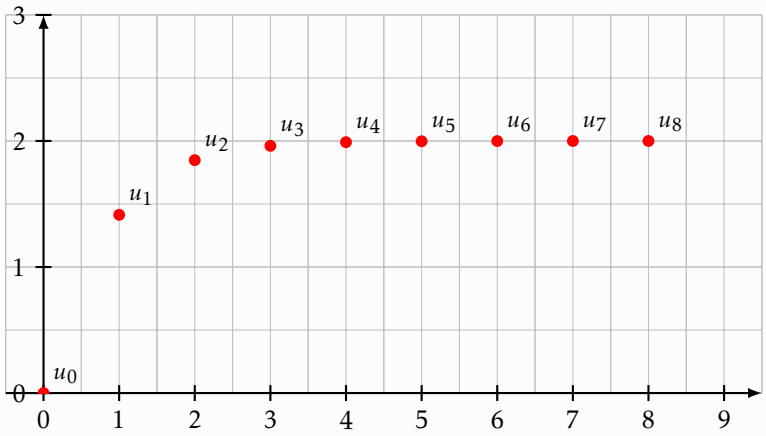


10.3 Suite récurrente : \tkzGrphSeqRecurrent

Trace le nuage (n, u_n) d'une suite **récurrente** $u_{n+1} = f(u_n)$, avec valeur initiale u_0 fournie en troisième argument.

```
\tkzGrphSeqRecurrent[cles]{fct}{u0}
```

```
\xintdefloatfunc frec(x) := sqrt(x+2) ;
\begin{tkzGrph}<Xmin=-0.5,Xmax=9.5,Ymin=0,Ymax=3,
                Largeur=10cm,Hauteur=5cm>
  \tkzGrphDrawGridAxis{auto}{auto}
  \tkzGrphSeqRecurrent[Fin=8,Couleur=red,Style=o,
    Labels=true,PosLabel=above right]{frec}{0}
\end{tkzGrph}
```

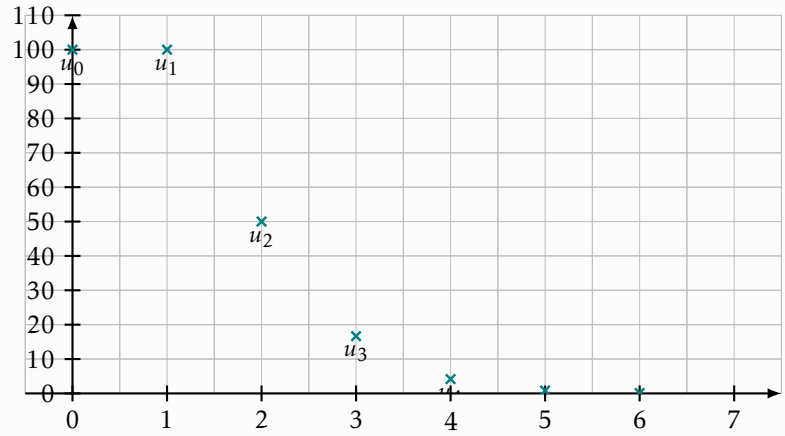


10.4 Suite récurrente avec rang : \tkzGrphSeqRecurrentN

Trace le nuage (n, u_n) d'une suite de la forme $u_{n+1} = f(u_n, n)$, où n est le rang courant. La fonction doit être déclarée avec deux paramètres.

```
\tkzGrphSeqRecurrentN[cles]{fct}{u0}
% fct(x,n) : x = u_n, n = rang courant
```

```
\xintdefloatfunc frecn(x,n) := x/(n+1) ;
\begin{tkzGrph}<Xmin=-0.5,Xmax=7.5,Ymin=0,Ymax=110,Ygrille=10,Ygrilles=10,
                Largeur=10cm,Hauteur=5cm>
  \tkzGrphDrawGridAxis{auto}{auto}
  \tkzGrphSeqRecurrentN[Fin=6,Couleur=teal,Style=*,
    Taille=2.5pt,Labels=true]{frecn}{100}
\end{tkzGrph}
```

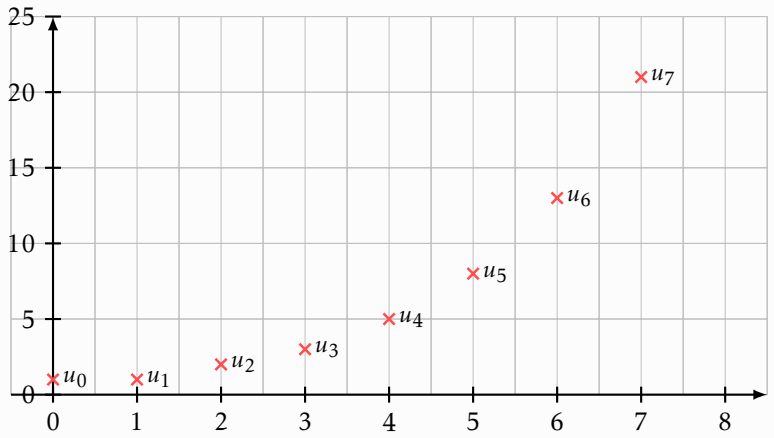


10.5 Double récurrence : \tkzGrphSeqDouble

Trace le nuage (n, u_n) d'une suite doublement récurrente $u_{n+2} = f(u_n, u_{n+1})$. Les deux valeurs initiales u_0 et u_1 sont fournies en troisième et quatrième arguments.

```
\tkzGrphSeqDouble[cles]{fct}{u0}{u1}
% fct(x,y) : x = u_n, y = u_{n+1}
```

```
% Fibonacci : u_{n+2} = u_n + u_{n+1}, u_0 = 1, u_1 = 1
\xintdefloatfunc ffib(x,y) := x+y ;
\begin{tkzGrph}<Xmin=-0.5,Xmax=8.5,Ymin=0,Ymax=25,Ygrille=5,Ygrilles=5,
               Largeur=10cm,Hauteur=5cm>
  \tkzGrphDrawGridAxis{auto}{auto}
  \tkzGrphSeqDouble[Fin=7,Couleur=red!70,Style=x,
    Taille=3pt,Labels=true,PosLabel=right]{ffib}{1}{1}
\end{tkzGrph}
```



10.6 Double récurrence avec rang : \tkzGrphSeqDoubleN

Variante pour les suites de la forme $u_{n+2} = f(u_n, u_{n+1}, n)$. La fonction doit être déclarée avec trois paramètres.

```
\tkzGrphSeqDoubleN[cles]{fct}{u0}{u1}
% fct(x,y,n) : x = u_n, y = u_{n+1}, n = rang de u_n
```

10.7 Récurrence avec toile : \tkzGrphSeqDoubleN

Variante pour les suites de la forme $u_{n+2} = f(u_n)$ via la technique de la *toile*.

```
\tkzGrphDrawCobweb[clés]{fct ou path}

%=== clés régulières FR
% Couleur       = black
% Formule       = false   (pour passerr en mode formule xint)
% No            = 0       (rang initial)
% Nom           = u       (nom de la suite)
% Nb            = 5       (nombre d'itérations)
% AffTermes     = false   (afficher les termes)
% AffPointilles = true    (pointillés de construction)
% TailleLabel   = \small
% PosLabel      = below
```

10.8 Wrapper universel : \tkzGrphDrawSequence

La commande \tkzGrphDrawSequence (alias \tkzGrphNuageSuite, \tkzgrphseq) unifie toutes les variantes grâce à la clé Type.

```
\tkzGrphDrawSequence[cles,Type=...]{fct}[u0][u1]
% alias : \tkzGrphNuageSuite   \tkzgrphseq
```

Les valeurs acceptées pour Type :

Valeur	Commande appelée
Explicite / explicit	\tkzGrphSeqExplicite
Recurrent / recurrent	\tkzGrphSeqRecurrent (défaut)
RecurrentN / recurrentn	\tkzGrphSeqRecurrentN
Double / double	\tkzGrphSeqDouble
DoubleN / doublen	\tkzGrphSeqDoubleN
Toile / cobweb	\tkzGrphDrawCobweb

Les arguments [u0] et [u1] sont optionnels (défaut 0) ; ils sont ignorés pour les types qui n'en ont pas besoin. Pour le type Toile/Cobweb, [u0] fournit la valeur initiale u_0 (injectée comme Uno=u0 dans \tkzGrphDrawCobweb).

```
% Suite récurrente (Type par défaut) — u_0 passé en [u0]
\xintdefloatfunc fwrec(x) := sqrt(x+2) ;
\tkzGrphDrawSequence[Fin=8,Couleur=red,Style=o,Labels=true]{fwrec}[0]

% Suite explicite
\xintdefloatfunc fwexp(x) := (3*x+1)/(x+1) ;
\tkzGrphDrawSequence[Type=Explicite,Debut=0,Fin=10,
  Couleur=blue,Style=*,Taille=2pt]{fwexp}

% Double récurrence — u_0 et u_1 en [u0][u1]
\xintdefloatfunc fwfib(x,y) := x+y ;
\tkzGrphDrawSequence[Type=Double,Fin=7,Couleur=red!70,
  Style=x,PosLabel=right]{fwfib}[1][1]

% Toile de récurrence via le wrapper (Formule = calcul xint direct)
\xintdefloatfunc fwrec(x) := sqrt(x+2) ;
\tkzGrphDrawSequence[Type=Toile,Nb=8,Couleur=red,
  Formule,Style=o]{fwrec}[0]
```