

Figures mathématiques avec TikZ

5.1 Introduction

Dans beaucoup d'ouvrages mathématiques, les figures souffrent de défauts rédhibitoires : traits trop épais, flèches dans un style complètement différent de celles du reste du document, polices de caractères différentes de celles du document, pixellisation, etc.

Il y a plusieurs façon de faire des figures mathématiques avec LaTeX¹, mais celle que nous allons voir aujourd'hui, TikZ, a l'avantage de pallier à tous ces défauts. Un autre avantage de TikZ est de disposer d'un manuel complet (560 pages) décrivant toutes les fonctions disponibles en un seul et unique endroit. On peut trouver ce manuel à l'adresse

<http://www.ctan.org/tex-archive/graphics/pgf/base/doc/generic/pgf/pgfmanual.pdf>

On pourra aussi consulter la galerie d'exemples en ligne :

<http://www.texample.net/tikz/examples/all/>

Il est également possible de faire des animations avec TikZ et le package animate. Voici, à titre d'exemple, la convergence des sommes de Riemann vers l'intégrale d'une fonction :

(Cliquer sur le dessin pour lancer l'animation.)

1. Citons par exemple pstricks, metapost, asymptote ou des logiciels externes comme xfig ou texgraph. Noter que quelques uns d'entre eux ont certains des écueils mentionnés plus haut.

5.2 Traits, flèches, rectangles, cercles et ellipses

Noter que, dans TeXmaker, il y a un menu TikZ dans la barre latérale. Elle permet d'accéder à la plupart des fonctions usuelles. Avant de pouvoir faire une figure avec TikZ, il faut charger le package tikz :


```
\usepackage{tikz}
```

Toute figure TikZ doit être à l'intérieur d'un environnement `{tikzpicture}`, qu'il vaut mieux toujours mettre dans un environnement `{center}` (ou `{figure}` si besoin est) :

```
\begin{center}\begin{tikzpicture}
```


```
\end{tikzpicture}\end{center}
```

Pour tracer une ligne entre deux points, on utilise la commande `\draw` de la façon suivante




```
\begin{tikzpicture}
\draw (0,0) -- (1,1);
\end{tikzpicture}
```

Bien noter le ; à la fin de la ligne. Pour tracer une flèche, il suffit de rajouter une option à `\draw` :




```
\begin{tikzpicture}
\draw[->] (0,0) -- (1,1);
\draw[<->] (0.5,0) -- (0,1);
\end{tikzpicture}
```

On peut vouloir que les lignes soient en pointillé :




```
\begin{tikzpicture}
\draw[dotted] (0,0) -- (1,1);
\draw[<-.,dashed] (0.5,0) -- (0,1);
\end{tikzpicture}
```

Il est possible de tracer plusieurs traits de suite en les enchaînant :




```
\begin{tikzpicture}
\draw (0,0) -- (1,1) -- (0.5,0) -- (0,1);
\end{tikzpicture}
```

Si on veut refermer le tracé, il suffit de terminer la chaîne par un cycle :




```
\begin{tikzpicture}
\draw (0,0) -- (1,1) -- (0.5,0) -- (0,1) -- cycle;
\end{tikzpicture}
```

On peut changer la couleur du trait avec l'option `color` et la couleur de remplissage avec `fill` (les différentes couleurs disponibles sont décrites dans l'aide-mémoire, page 3) :



```
\begin{tikzpicture}
\draw[color=red,fill=blue] (0,0) -- (1,1) -- (0.5,0) -- (0,1) -- cycle;
\end{tikzpicture}
```


Pour rendre un trait plus épais, on peut spécifier une épaisseur, par exemple



```
\begin{tikzpicture}
\draw[ultra thick] (0,0) -- (1,1);
\end{tikzpicture}
```


Les épaisseurs disponibles sont `ultra thin`, `very thin`, `thin` (largeur par défaut), `semithick`, `thick`, `very thick`, `ultra thick`.

Pour tracer un rectangle, on peut utiliser `rectangle` et spécifier le point en bas à gauche et en haut à droite :




```
\begin{tikzpicture}
\draw[fill=green!30] (0,0) rectangle (1,0.75);
\end{tikzpicture}
```

De même, on peut tracer des cercles :




```
\begin{tikzpicture}
\draw[color=violet] (0,0) circle (0.5cm);
\end{tikzpicture}
```

et des ellipses :



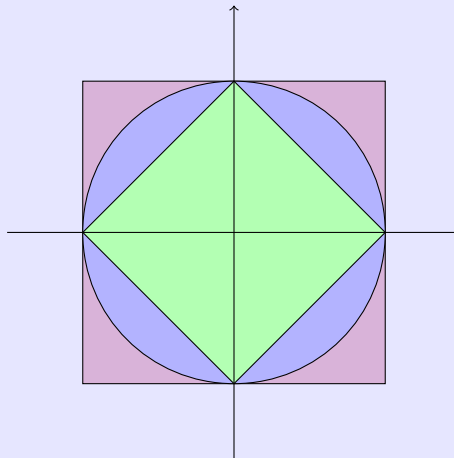
```
\begin{tikzpicture}
\draw[dashed] (0,0) ellipse (0.5cm and 0.33cm);
\end{tikzpicture}
```

Pour agrandir une figure d'un facteur, disons 2, il suffit d'utiliser l'option `scale=2` :



```
\begin{tikzpicture}[scale=2]
\draw[dashed] (0,0) ellipse (0.5cm and 0.33cm);
\end{tikzpicture}
```

Exercice 1. — Reproduire la figure suivante, qui représente différentes les boules unités de \mathbb{R}^2 pour les trois normes usuelles :



Solution de l'exercice 1. — Voici le code de la figure précédente. Bien faire attention à l'ordre dans lequel on dessine les différents éléments. Comme le texte spécifie qu'il s'agit de boules unités, il paraît préférable d'utiliser un rayon de 1cm pour tous ces objets puis d'utiliser l'option `scale=2`.

```
\begin{center}\begin{tikzpicture}[scale=2]
\draw[fill=violet!30] (-1,-1) rectangle (1,1);
\draw[fill=blue!30] (0,0) circle (1cm);
\draw[fill=green!30] (1,0) -- (0,1) -- (-1,0) -- (0,-1) -- cycle;
\draw[->] (-1.5,0) -- (1.5,0);
\draw[->] (0,-1.5) -- (0,1.5);
\end{tikzpicture}\end{center}
```

5.3 Texte et légendes

Pour placer du texte à un endroit précis, on utilise `\node` de la façon suivante :

```

texte  \begin{tikzpicture}
       \node at (0,0) {texte};
       \end{tikzpicture}

```

Le texte est alors centré à la position indiquée. On peut néanmoins le mettre au-dessus, en-dessous ou à côté en spécifiant l'option adéquate :

```

h      \begin{tikzpicture}
gCr    \node at (0,0) {c};
b      \node[above] at (0,0) {h};
       \node[below] at (0,0) {b};
       \node[left] at (0,0) {g};
       \node[right] at (0,0) {r};
       \end{tikzpicture}

```

On peut aussi utiliser `below right`, `above left`, etc. Il est possible, si besoin, d'éloigner le texte de son point d'ancrage en spécifiant la distance explicitement :

```

hg      hd      \begin{tikzpicture}
c      \node at (0,0) {c};
bg      \node[above left=0.25cm] at (0,0) {hg};
bd      \node[below left=0.25cm] at (0,0) {bg};
       \node[above right=0.25cm] at (0,0) {hd};
       \node[below right=0.25cm] at (0,0) {bd};
       \end{tikzpicture}

```

On peut aussi rapprocher le point en spécifiant une distance négative. Il est possible d'ancrer un texte non pas à un point qu'on vient de spécifier, mais à l'intérieur d'un `\draw`, par exemple :

```

A      B      \begin{tikzpicture}
       \draw (0,0) node[below] {A} -- (1,0) node[below] {B};
       \end{tikzpicture}

```

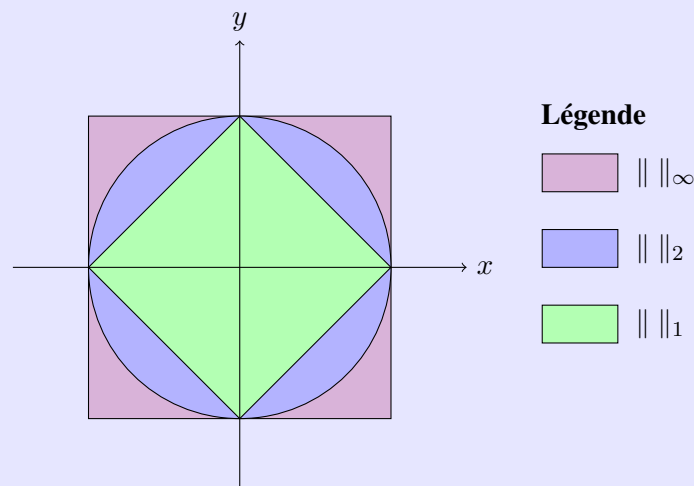
Bien noter que c'est `node` et non `\node` à l'intérieur d'un `\draw`. Il est possible de mettre un `node` au milieu de ce qui vient d'être dessiné. Par exemple :

```

l      \begin{tikzpicture}
       \draw (0,0) -- (1,0) node[pos=0.5,below] {$\ell$};
       \end{tikzpicture}

```

Exercice 2. — Reprendre le code de la figure de l'exercice précédent pour obtenir



Solution de l'exercice 2. — Voici le code de l'image précédente.

```

\begin{center}\begin{tikzpicture}[scale=2]
\draw[fill=violet!30] (-1,-1) rectangle (1,1);
\draw[fill=blue!30] (0,0) circle (1cm);
\draw[fill=green!30] (1,0) -- (0,1) -- (-1,0) -- (0,-1) -- cycle;
\draw[->] (-1.5,0) -- (1.5,0) node[right] {$x$};
\draw[->] (0,-1.5) -- (0,1.5) node[above] {$y$};
\node[right=-0.15cm] at (2,1) {\textbf{Légende}};
\draw[fill=violet!30] (2,0.5) rectangle (2.5,0.75)
  node[pos=0.5,right=0.6cm] {$\lVert\cdot\rVert_{\infty}$};
\draw[fill=blue!30] (2,0) rectangle (2.5,0.25)
  node[pos=0.5,right=0.6cm] {$\lVert\cdot\rVert_2$};
\draw[fill=green!30] (2,-0.25) rectangle (2.5,-0.5)
  node[pos=0.5,right=0.6cm] {$\lVert\cdot\rVert_1$};
\end{tikzpicture}\end{center}

```

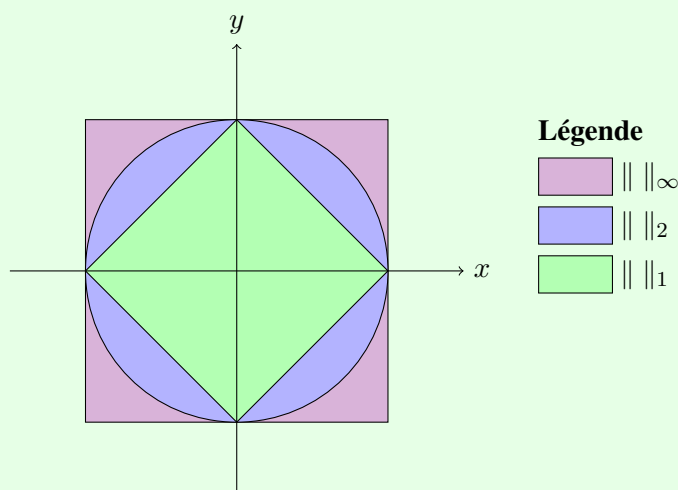
Noter qu'il serait plus astucieux d'utiliser, pour la légende, une minipage et des fcolorbox. À titre indicatif, voici le code correspondant :

```

\begin{center}\begin{tikzpicture}[scale=2]
\draw[fill=violet!30] (-1,-1) rectangle (1,1);
\draw[fill=blue!30] (0,0) circle (1cm);
\draw[fill=green!30] (1,0) -- (0,1) -- (-1,0) -- (0,-1) -- cycle;
\draw[->] (-1.5,0) -- (1.5,0) node[right] {$x$};
\draw[->] (0,-1.5) -- (0,1.5) node[above] {$y$};
\node[below right=-0.15cm] at (2,1) {\begin{minipage}{3cm}\fboxsep=0pt
\textbf{Légende}\par\addvspace{3pt}
\colorbox{black}{violet!30}{\hspace{2.5em}\strut} $\lVert\cdot\rVert_{\infty}$\par\addvspace{3pt}
\colorbox{black}{blue!30}{\hspace{2.5em}\strut} $\lVert\cdot\rVert_2$\par\addvspace{3pt}
\colorbox{black}{green!30}{\hspace{2.5em}\strut} $\lVert\cdot\rVert_1$\par\addvspace{3pt}
\end{minipage}};
\end{tikzpicture}\end{center}

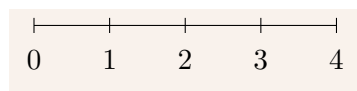
```

et le résultat :



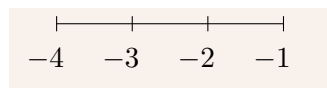
5.4 Boucles

Pour faire des graduations sur un axe, il est possible d'utiliser la boucle `\foreach` de la façon suivante :



```
\begin{tikzpicture}
\draw (0,0) -- (4,0);
\foreach \x in {0,1,...,4} {
  \draw (\x,0.1cm) -- (\x,-0.1cm) node[below] {\x\strut};
}
\end{tikzpicture}
```

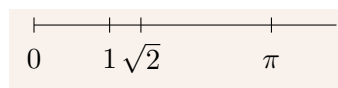
(Le `\strut` est là uniquement pour s'assurer de l'alignement vertical des nombres.) Pour les nombres négatifs, on peut employer la même commande ou alors, si on veut que le signe `-` ne soit pas pris en compte dans l'alignement, utiliser un `\phantom` :



```
\begin{tikzpicture}
\draw (-4,0) -- (-1,0);
\foreach \x in {-4,...,-1} {
  \draw (\x,0.1cm) -- (\x,-0.1cm) node[below] {\x\phantom{-}\strut};
}
\end{tikzpicture}
```

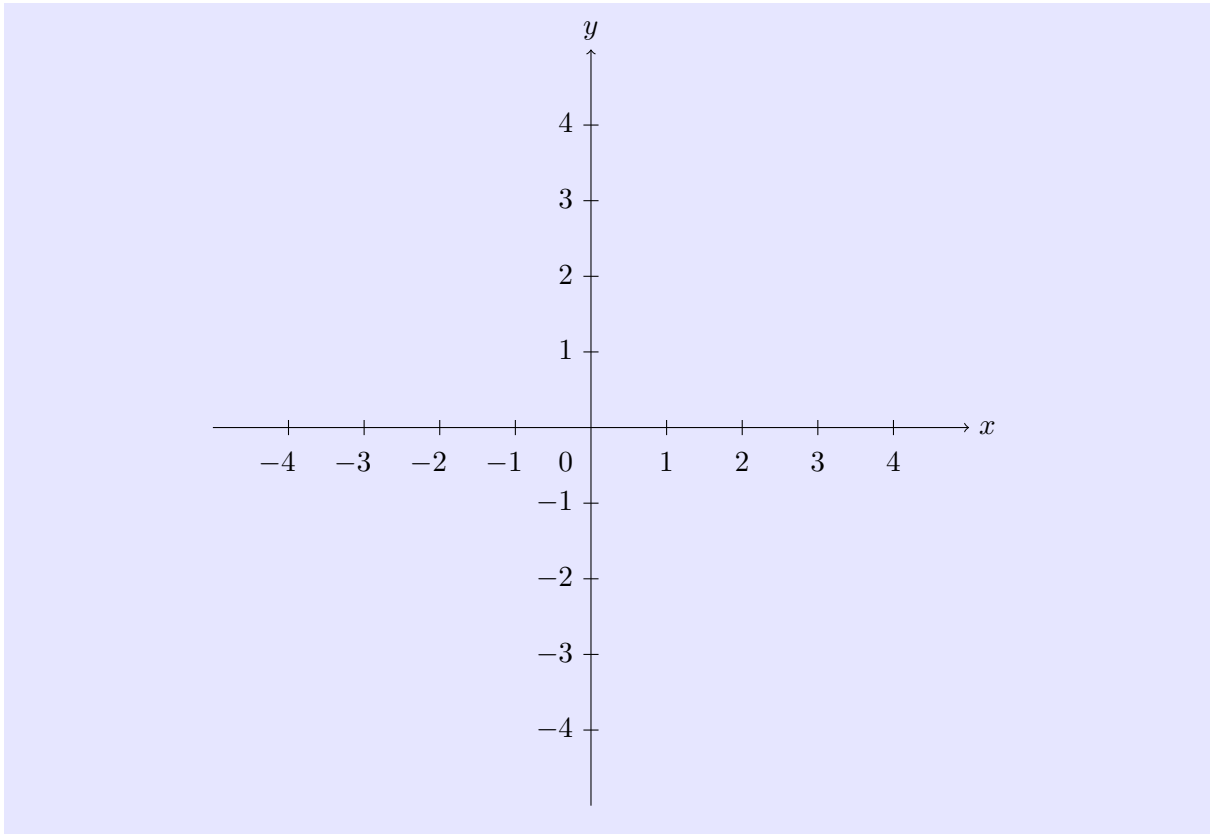
(Ce qu'on vient de faire revient à écrire `-4-` mais avec le deuxième `-` non affiché grâce au `\phantom`.)

Si jamais on a besoin d'afficher un texte différent de la valeur, on peut le faire ainsi :



```
\begin{tikzpicture}
\draw (0,0) -- (4,0);
\foreach \x/\xtext in {0,1,1.414/\sqrt{2},3.14/\pi} {
  \draw (\x,0.1cm) -- (\x,-0.1cm) node[below] {\xtext\strut};
}
\end{tikzpicture}
```

Exercice 3. — Avec les commandes précédentes pour les boucles, tracer les axes gradués suivants :



Solution de l'exercice 3. — Voici le code de l'image précédente.

```
\begin{center}\begin{tikzpicture}
\draw[->] (-5,0) -- (5,0) node[right] {$x$};
\draw[->] (0,-5) -- (0,5) node[above] {$y$};
\foreach \x in {-4,...,-1} {
  \draw (\x,0.1cm) -- (\x,-0.1cm) node[below] {$\x\phantom{-}\strut$};
}
\foreach \x in {1,...,4} {
  \draw (\x,0.1cm) -- (\x,-0.1cm) node[below] {$\x\strut$};
}
\foreach \y in {-4,...,-1,1,2,...,4} {
  \draw (0.1cm,\y) -- (-0.1cm,\y) node[left] {$\y\strut$};
}
\node[below left] at (-0.1cm,-0.1cm) {$0\strut$};
\end{tikzpicture}\end{center}
```

5.5 Tracé de fonctions

Pour des raisons techniques (caractères actifs), à partir de maintenant, toujours rajouter `\shorthandoff{:}` avant `\begin{tikzpicture}` et `\shorthandon{:}` après `\end{tikzpicture}`:

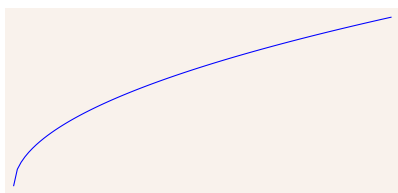
```
\begin{center}\shorthandoff{:}\begin{tikzpicture}
```

```
\end{tikzpicture}\shorthandon{:}\end{center}
```

Utiliser `\shorthandoff{:}` dans la définition d'une macro n'aura aucun effet ; il faut le mettre avant le `\newcommand` et mettre `\shorthandon{:}` après :

```
\shorthandoff{:}
\newcommand{\dessine}[1]{\begin{tikzpicture}...\end{tikzpicture}}
\shorthandon{:}
```

Pour tracer une fonction, on utilise la commande `draw` avec un `plot`. Par exemple, pour tracer la fonction $x \mapsto \sqrt{x}$,



```
\shorthandoff{:}\begin{tikzpicture}
\draw[domain=0:5,samples=100,color=blue] plot ({\x},{sqrt(\x)});
\end{tikzpicture}\shorthandon{:}
```

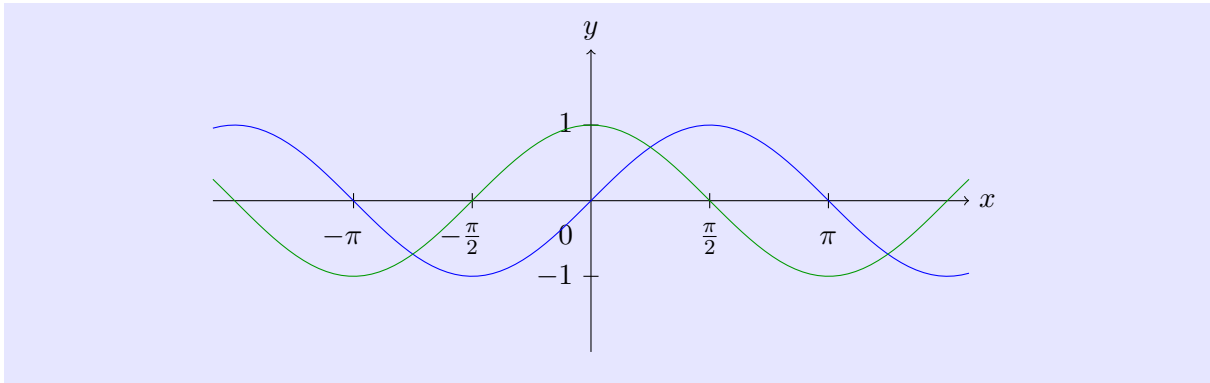
L'option `domain=0:5` permet de se restreindre au domaine allant de $x = 0$ à $x = 5$ (le domaine par défaut est $x = -5$ à $x = 5$), où la racine carrée est définie et l'option `samples=100` permet de prendre plus de points pour tracer la fonction, ce qui améliore la qualité du tracé.

Les fonctions disponibles sont récapitulées dans le tableau suivant :

| NOM | SIGNIFICATION | EXEMPLE |
|--------------------|------------------|----------------------------|
| <code>exp</code> | exponentielle | <code>exp(\x)</code> |
| <code>ln</code> | logarithme | <code>ln(\x)</code> |
| <code>sqrt</code> | racine carrée | <code>sqrt(\x)</code> |
| <code>sin</code> | sinus | <code>sin(\x r)</code> |
| <code>cos</code> | cosinus | <code>cos(\x r)</code> |
| <code>tan</code> | tangente | <code>tan(\x r)</code> |
| <code>cot</code> | cotangente | <code>cot(\x r)</code> |
| <code>abs</code> | value absolue | <code>abs(\x)</code> |
| <code>asin</code> | arcsinus | <code>rad(asin(\x))</code> |
| <code>acos</code> | arccosinus | <code>rad(acos(\x))</code> |
| <code>atan</code> | arctangente | <code>rad(atan(\x))</code> |
| <code>sec</code> | $\frac{1}{\sin}$ | <code>sec(\x r)</code> |
| <code>cosec</code> | $\frac{1}{\cos}$ | <code>cosec(\x r)</code> |

Le `r` après les fonctions trigonométriques est là pour que l'argument soit en radian. Attention, dans la version actuelle (2.0), la fonction `tan` est bugguée pour les valeurs négatives de l'argument. Pour obtenir un carré, ne pas taper `\x^2`, mais utiliser plutôt `\x*\x` ; c'est un peu lourd à taper mais ça évite certaines erreurs difficiles à trouver.

Exercice 4. — Tracer les fonctions cosinus et sinus :

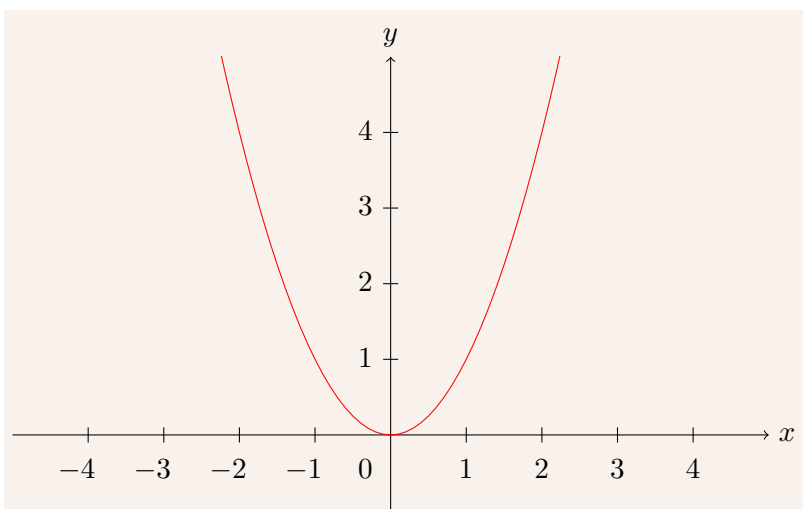


Solution de l'exercice 4. — Voici le code de l'image précédente.

```
\begin{center}\begin{tikzpicture}[samples=100]
\draw[->] (-5,0) -- (5,0) node[right] {$x$};
\draw[->] (0,-2) -- (0,2) node[above] {$y$};
\foreach \x/\xtext in {-3.14/-\pi,-1.57/-\frac{\pi}{2}} {
  \draw (\x,0.1cm) -- (\x,-0.1cm) node[below] {$\xtext\phantom{-}\strut$};
}
\foreach \x/\xtext in {1.57/\frac{\pi}{2},3.14/\pi} {
  \draw (\x,0.1cm) -- (\x,-0.1cm) node[below] {$\xtext\strut$};
}
\foreach \y in {-1,1} {
  \draw (0.1cm,\y) -- (-0.1cm,\y) node[left] {$\y\strut$};
}
\node[below left=0.1cm] at (-0,0) {$0\strut$};
\draw[color=blue] plot ({\x},{sin(\x r)});
\draw[color=green!60!black] plot ({\x},{cos(\x r)});
\end{tikzpicture}\end{center}
```

Noter qu'on a mis `samples=100` en option de `{tikzpicture}` pour ne pas avoir à le répéter à chaque fois. Cela fonctionne avec la plupart des options.

Il peut arriver, dans certains cas, que la courbe monte trop haut pour certaines valeurs de x , au point de ne plus tenir sur la page ; c'est le cas avec $x \mapsto x^2$ par exemple qui vaut 25 pour $x = 5$. Il faut dans ce cas *clipper* le dessin



```

\shorthandoff{:}\begin{tikzpicture}
\draw[->] (-5,0) -- (5,0) node[right] {$x$};
\draw[->] (0,-1) -- (0,5) node[above] {$y$};
\foreach \x in {-4,...,-1} {
  \draw (\x,0.1cm) -- (\x,-0.1cm) node[below] {$\x\phantom{-}\strut$};
}
\foreach \x in {1,...,4} {
  \draw (\x,0.1cm) -- (\x,-0.1cm) node[below] {$\x\strut$};
}
\foreach \y in {1,2,...,4} {
  \draw (0.1cm,\y) -- (-0.1cm,\y) node[left] {$\y\strut$};
}
\node[below left=0.1cm] at (-0,0) {$0\strut$};
\begin{scope}
\clip (-5,-1) rectangle (5,5);
\draw[color=red,samples=100] plot ({\x},{\x*\x});
\end{scope}
\end{tikzpicture}\shorthandon{:}

```

Dans le code précédent le `\begin{scope}... \end{scope}` sert à délimiter l'action du `\clip` (pour mettre, par exemple, du code après qui pourra dépasser du rectangle de *clippage*).

5.6 Animations

Pour pouvoir faire des animations, il suffit de rajouter au préambule le package `animate` :

```
\usepackage{animate}
```

On utilise ensuite l'environnement `{animateinline}` avec le paramètre obligatoire 1 (ce paramètre contrôle la vitesse de défilement). On met ensuite l'image ou la figure qu'on veut pour première image de l'animation puis un `\newframe` puis la seconde image/figure puis encore `\newframe`, etc. La dernière image sera celle qui sera affichée à la fin de l'animation (souvent, on remet la première image, mais ce n'est pas obligatoire).

Voici un exemple, où on trace un rectangle pas à pas. Pour des raisons techniques (interaction entre les bounding box et `animate`), on rajoute dans toutes les images la ligne

```
\useasboundingbox (-0.1,-0.1) rectangle (1.1,1.1);
```

qui permet de s'assurer que toutes les images auront la même taille, taille qui doit être suffisante pour afficher toutes les images. Voici le code :

```

\begin{center}\shorthandoff{:}
\newcommand{\dessiner}[1]{
\begin{tikzpicture}
\useasboundingbox (-0.1,-0.1) rectangle (1.1,1.1);
#1
\end{tikzpicture}
}
\begin{animateinline}{1}%
\dessiner{\draw (0,0) -- (1,0);}
\newframe
\dessiner{\draw (0,0) -- (1,0) -- (1,1);}
\newframe
\dessiner{\draw (0,0) -- (1,0) -- (1,1) -- (0,1);}
\newframe
\dessiner{\draw (0,0) -- (1,0) -- (1,1) -- (0,1) -- cycle;}
\newframe

```

```
\dessiner{\draw (0,0) -- (1,0);}
\end{animateinline}
\shorthandon{:}\end{center}
```

(exceptionnellement, on a mis un `\newcommand` ailleurs que dans le préambule, car la commande `\dessiner` n'est utile nulle part ailleurs) et voici le résultat :

(Cliquer sur l'image pour lancer l'animation.)

Exercice 5. — Reproduire l'animation suivante, qui dessine la courbe représentative de cosinus avec les polynômes de Taylor, successivement $x \mapsto 1$, $x \mapsto 1 - \frac{x^2}{2}$, $x \mapsto 1 - \frac{x^2}{2} + \frac{x^4}{24}$ et $x \mapsto 1 - \frac{x^2}{2} + \frac{x^4}{24} + \frac{x^6}{720}$:

Que se passe-t-il si on essaie de dessiner $x \mapsto 1 - \frac{x^2}{2} + \frac{x^4}{24} - \frac{x^6}{720} + \frac{x^8}{40320}$? Pourquoi ? Comment contourner ce problème ?

Solution de l'exercice 5. — Voici le code de la figure précédente. Noter qu'on a défini une commande personnelle (à l'intérieur du `{center}` car elle spécifique à cette figure)

```
\begin{center}\shorthandoff{:}
\newcommand{\tracecosinus}[1]{
\begin{tikzpicture}[samples=100]
\draw[->] (-5,0) -- (5,0) node[right] {$x$};
\draw[->] (0,-2) -- (0,2) node[above] {$y$};
\foreach \x/\xtext in {-3.14/-\pi,-1.57/-\frac{\pi}{2}} {
\draw (\x,0.1cm) -- (\x,-0.1cm) node[below] {$\xtext\phantom{-}\strut$};
}
\foreach \x/\xtext in {1.57/\frac{\pi}{2},3.14/\pi} {
\draw (\x,0.1cm) -- (\x,-0.1cm) node[below] {$\xtext\strut$};
}
\foreach \y in {-1,1} {
\draw (0.1cm,\y) -- (-0.1cm,\y) node[left] {$\y\strut$};
}
\node[below left=0.1cm] at (-0,0) {$0\strut$};
\begin{scope}
\clip (-5,-2) rectangle (5,2);
\draw[color=green!60!black] plot ({\x},{cos(\x r)});
#1
\end{scope}
\end{tikzpicture}
```

```

}
\begin{animateinline}{1}
\tracecosinus{}
\newframe
\tracecosinus{\draw[color=blue] plot ({\x},{1});}
\newframe
\tracecosinus{\draw[color=blue] plot ({\x},{1-\x*\x/2});}
\newframe
\tracecosinus{\draw[color=blue] plot ({\x},{1-\x*\x/2+\x*\x*\x/24});}
\newframe
\tracecosinus{\draw[color=blue] plot ({\x},{1-\x*\x/2+\x*\x*\x/24-\x*\x*\x*\x/720});}
\end{animateinline}
\shorthandoff{:}
\end{center}

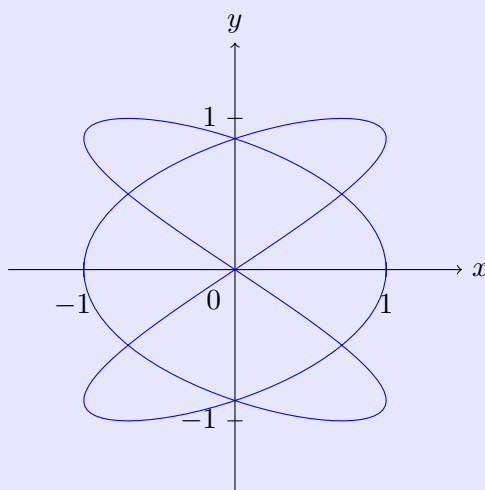
```

Si on essaie de tracer le polynôme de Taylor d'ordre 8, il y a une erreur de « dimension too large ». Cela se produit car TikZ est relativement limité dans les nombres qu'il peut manipuler et tous les calculs intermédiaires ne doivent pas dépasser cette valeur. Ici, pour contourner ce problème, on pourrait essayer de réduire le domaine de tracé du polynôme ou alors utiliser gnuplot pour tracer la fonction (TikZ peut travailler avec gnuplot de manière relativement automatique, mais cela requiert d'activer le `-shell-escape`, ce qu'on ne fera pas sur les ordinateurs de la faculté ; les plus courageux d'entre vous peuvent essayer chez eux, en lisant le manuel de TikZ sur le sujet et en faisant attention à la syntaxe légèrement différente).

5.7 Pour aller plus loin : courbes paramétriques, courbes polaires

Le fait d'écrire `plot ({\x},{f(\x)})` pour tracer une fonction suggère que l'on peut aussi faire des tracés de courbes paramétriques en utilisant `plot ({f(\x)},{g(\x)})`.

Exercice 6. — Tracer la courbe paramétrée $t \mapsto (\cos(3t), \sin(2t))$ pour $t \in [-\pi; \pi]$:



Solution de l'exercice 6. — Voici le code traçant la courbe paramétrée précédente :

```

\begin{center}\shorthandoff{:}
\begin{tikzpicture}[samples=200,scale=2]
\draw[->] (-1.5,0) -- (1.5,0) node[right] {\$x\$};

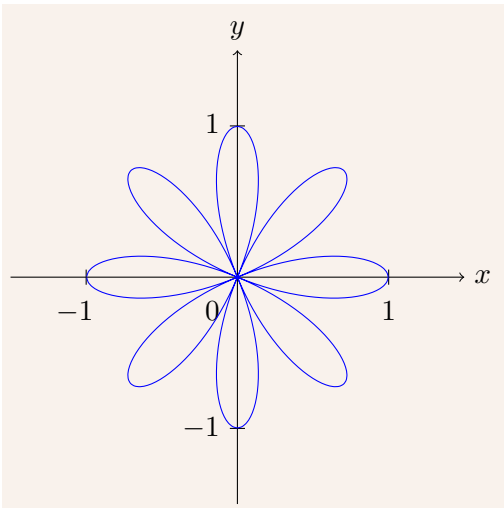
```

```

\draw[->] (0,-1.5) -- (0,1.5) node[above] {$y$};
\foreach \x/\xtext in {-1} {
  \draw (\x,0.05cm) -- (\x,-0.05cm) node[below] {$\xtext\phantom{-}\strut$};
}
\foreach \x/\xtext in {1} {
  \draw (\x,0.05cm) -- (\x,-0.05cm) node[below] {$\xtext\strut$};
}
\foreach \y in {-1,1} {
  \draw (0.05cm,\y) -- (-0.05cm,\y) node[left] {$\y\strut$};
}
\node[below left=0.05cm] at (-0,0) {$0\strut$};
\begin{scope}
\clip (-1.5,-1.5) rectangle (1.5,1.5);
\draw[color=blue,domain=-3.14:3.14] plot ({cos(3*\x r)},{sin(2*\x r)});
\end{scope}
\end{tikzpicture}\shorthandon{:}\end{center}

```

Pour tracer des courbes polaires, il n'y a actuellement pas d'interface utilisateur simple, donc il faut utiliser des commandes internes un peu compliquées. Par exemple, voici le code pour tracer la courbe $r = \cos(4\theta)$:



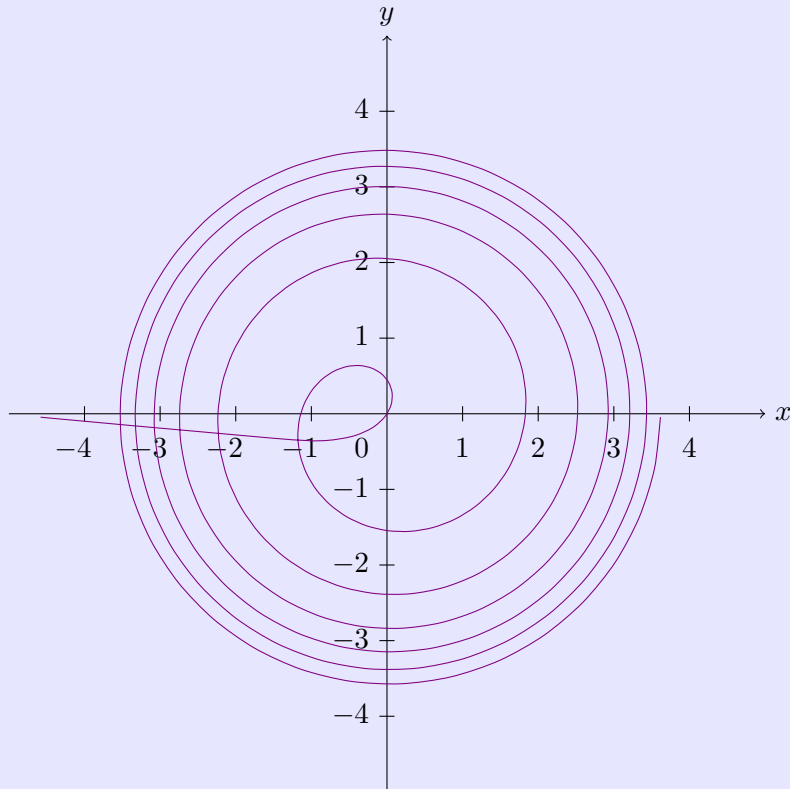
```

\shorthandoff{:}\begin{tikzpicture}[scale=2]
\draw[->] (-1.5,0) -- (1.5,0) node[right] {$x$};
\draw[->] (0,-1.5) -- (0,1.5) node[above] {$y$};
\foreach \x/\xtext in {-1} {
  \draw (\x,0.05cm) -- (\x,-0.05cm) node[below] {$\xtext\phantom{-}\strut$};
}
\foreach \x/\xtext in {1} {
  \draw (\x,0.05cm) -- (\x,-0.05cm) node[below] {$\xtext\strut$};
}
\foreach \y in {-1,1} {
  \draw (0.05cm,\y) -- (-0.05cm,\y) node[left] {$\y\strut$};
}
\node[below left=0.1cm] at (-0,0) {$0\strut$};
\begin{scope}
\clip (-1.5,-1.5) rectangle (1.5,1.5);
\draw[color=blue,domain=-3.14:3.14,samples=200,smooth]
  plot (canvas polar cs:angle=\x r,radius={cos(4*\x r)*28.35});
\end{scope}
\end{tikzpicture}\shorthandon{:}

```

(Le 28.35 est là pour convertir les points (qui sont l'unité de mesure interne par défaut en centimètres vu que $1 \text{ cm} = 28.35 \text{ pt.}$) Noter l'utilisation de l'option `smooth` pour éviter d'avoir à trop augmenter la valeur de samples.

Exercice 7. — Tracer la courbe polaire $r = \ln \theta$ pour $\theta \in [0,01 ; 37,69]$:



Solution de l'exercice 7. — Voici le code de la figure précédente.

```
\begin{center}\shorthandoff{:}\begin{tikzpicture}
\draw[->] (-5,0) -- (5,0) node[right] {$x$};
\draw[->] (0,-5) -- (0,5) node[above] {$y$};
\foreach \x in {-4,...,-1} {
  \draw (\x,0.1cm) -- (\x,-0.1cm) node[below] {$\x\phantom{-}\strut$};
}
\foreach \x in {1,...,4} {
  \draw (\x,0.1cm) -- (\x,-0.1cm) node[below] {$\x\strut$};
}
\foreach \y in {-4,...,-1,1,2,...,4} {
  \draw (0.1cm,\y) -- (-0.1cm,\y) node[left] {$\y\strut$};
}
\node[below left=0.1cm] at (-0,0) {$0\strut$};
\begin{scope}
\clip (-5,-5) rectangle (5,5);
\draw[color=violet,samples=200,smooth] plot[domain=0.01:37.69]
(canvas polar cs:angle=\x r,radius={ln(\x)*28.35});
\end{scope}
\end{tikzpicture}\shorthandon{:}\end{center}
```