TikZ, dessiner avec LaTeX

Jean-Pierre Franc

Janvier 2014

Table des matières

Vous avez choisi de produire vos publications avec LaTeX pour sa qualité typographique y compris dans le domaine des équations, son ouverture et sa portabilité. Vous souhaitez maintenant inclure dans vos textes des figures, des diagrammes, sans sortir de l'environnement LaTeX.

Deux possibilités s'offrent à nous, il s'agit des « packages » *pstricks* et *TikZ*. Le premier est assez technique et ne produit que du PostCript, ce qui ne correspond guère au confort d'impression du plus grand nombre. Notre choix s'est, dès lors, porté sur *TikZ* qui offre des performances similaires tout en produisant une impression de qualité au format PDF.

Le présent travail ne prétend nullement être exhaustif, il s'agit de présenter un large éventail de commandes permettant de se débrouiller avec les outils offerts par le package *TikZ*. Pour aller plus loin, signalons le manuel *TikZ pour l'impatient* qui peut être obtenu à l'adresse suivante :

`http://www.pedagogicon.be/latex/TikZ%20pour%20l'impatient.pdf`

ou encore, le très riche manuel TikZ and PGF,

`http://mirrors.rit.edu/CTAN/graphics/pgf/base/doc/generic/pgf/pgfmanual.pdf`

Chapitre 1

Figures, grille, axes et graphes de fonctions

1.1 Déclarer et utiliser TikZ

TikZ est un package LaTeX. Il doit être déclaré dans le préambule du document par l'instruction

```
\usepackage{pgf, tikz}
```

Attention, pour éviter le risque d'incompatibilité avec le package *xcolor*, la nouvelle instruction doit impérativement être déclarée après celle de *xcolor*.

Les futures instructions TikZ devront toutes être placées dans un environnement tikzpicture, lui même situé à l'intérieur de l'environnement document, comme cela est illustré à la figure 1 ci-dessous.

```
\begin{document}

\begin{tikzpicture}

\end{tikzpicture}

\end{document}
```

FIGURE 1.1 – L'environnement tikzpicture

La colonne de gauche de *texmaker* fournit une aide précieuse en ce qui concerne un grand nombre de commandes *TikZ*. On y accède par un simple clic de souris sur le bouton **TI**, voir figure 2. Ainsi, le second item de cet ensemble de

commandes ouvre dans le fichier de travail l'environnement *tikzpicture* évoqué plus haut.

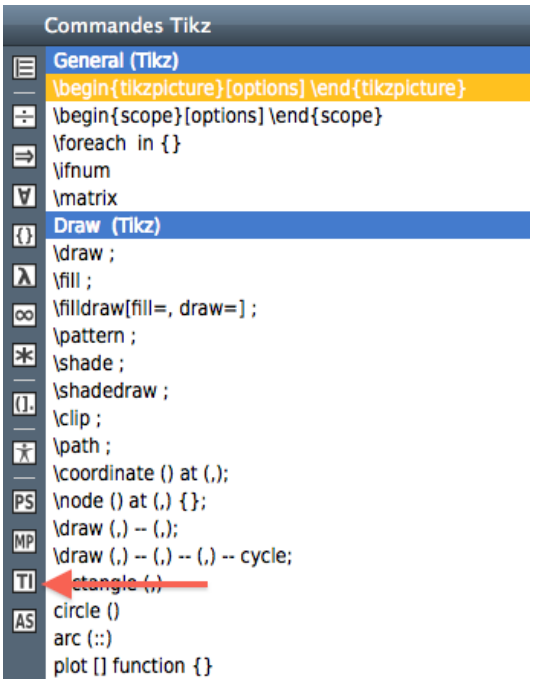


FIGURE 1.2 – Les commandes TikZ dans TexMaker

1.2 Les bases de la construction de figures

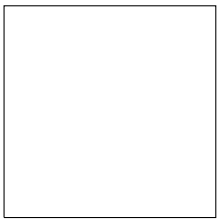
1.2.1 L'instruction \draw

TikZ travaille avec les coordonnées cartésiennes ou polaires. Dans ce manuel, nous avons choisi de travailler essentiellement avec les premières. Ainsi, l'instruction `\draw (0, 0) -- (4, 0);` donnera un segment de longueur 4 débutant au point de coordonnées (0, 0) et se terminant au point de coordonnées (4, 0).

L'origine ne possède pas de position prédéfinie sur la feuille, la figure demandée s'adaptera de telle sorte que l'ensemble des éléments à tracer soient visibles dans votre travail.

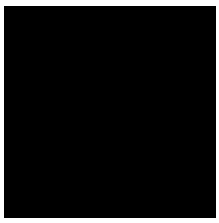
Toutes les instructions TikZ se terminent impérativement par un ;

De la même façon, le carré ci-dessous



s'obtient avec l'instruction
`\draw (0, 0) -- (4, 0) -- (4, 4) -- (0, 4) -- (0, 0);` dans laquelle chaque sommet du carré est donné par ses propres coordonnées.
L'instruction `\draw (0, 0) -- (4, 0) -- (4, 4) -- (0, 4) -- cycle;` donne le même résultat.

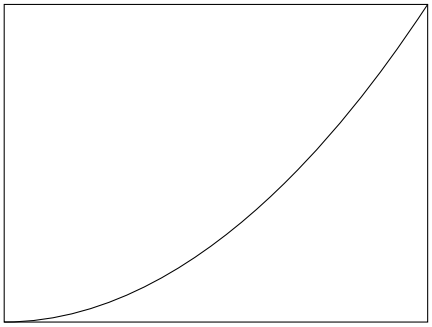
L'intérêt de cycle dans la commande ne réside pas dans une économie d'écriture mais dans la possibilité de définir des chemins fermés que l'on peut remplir avec la couleur de son choix à l'aide de l'instruction \fill.
Ainsi, `\fill (0, 0) -- (4, 0) -- (4, 4) -- (0, 4) -- cycle;` donnera :



Il est possible de représenter un certain nombre de figures géométriques simples à l'aide de commandes assez parlantes. Les commandes :

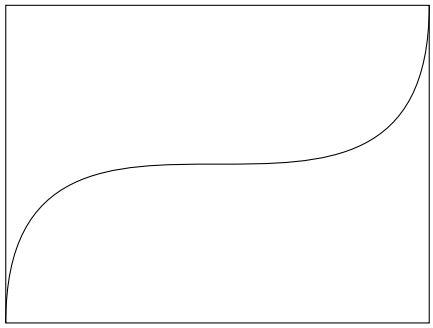
```
\draw (0, 0) rectangle (8, 6);
\draw (0, 0) parabola (8, 6);
```

donnent respectivement un rectangle dont les deux sommets situés de part et d'autre de la diagonale principale sont (0, 0) et (8, 6) ainsi qu'une parabole passant par ces deux mêmes points.



Plus ambitieux, les courbes de Bézier sont représentées à l'aide de la seconde commande ci-dessous :

```
\draw (0, 0) rectangle (8, 6);
\draw (0, 0) .. controls (0, 6) and (8, 0) .. (8, 6);
```
ce qui donne :



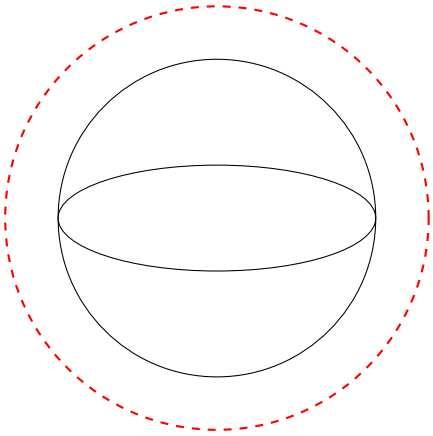
L'instruction *controls* mérite que l'on s'y attarde quelque peu. Cette dernière permet de construire une courbe de Bézier à l'aide de 4 points de contrôle :

- A, ici (0, 0), le point de départ de la courbe,
- S, ici (0, 6), le point tel que AS est la tangente à la courbe au point A,
- T, ici (8, 0), le point tel que BT est la tangente à la courbe au point B,
- B, ici (8, 6), le point d'arrivée de la courbe.

Il est fortement conseillé de jouer avec cette instruction dans le but de se familiariser avec son comportement.

1.2.2 Couleur, épaisseur et style des traits

La compréhension des commandes suivantes est facilement interprétable :
```
\draw (2, 2) circle (3cm);
\draw[red, thick, dashed] (2, 2) circle (4cm);
\draw (2, 2) ellipse (3cm and 1cm);
```
La première dessine un cercle centré en (2, 2) de rayon 3 cm, tandis que la seconde trace un cercle de plus grand rayon dans un trait pointillé, plus épais et de couleur rouge. La dernière, enfin, produit une ellipse centrée également en (2, 2) avec un grand axe de 3 cm et un petit axe de 1 cm de longueur. L'effet de ces commandes est illustré ci-après :



Les couleurs disponibles en LaTeX sont :

red | *green* | *blue* | *cyan* | *yellow* | *magenta* | *black* | *white* | *gray*

L'épaisseur des traits offre les choix suivants :

ultrathin | *verythin* | *thin* | *thick* | *verythick* | *ultrathick*

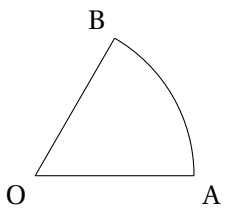
Quant au style du trait, LaTeX donne les possibilités ci-dessous :

dotted	*looselydotted*	*denselydotted*
dashed	*looselydashed*	*denselydashed*

1.2.3 L'instruction arc

L'instruction arc permet de représenter un arc de cercle. La syntaxe de cette commande est simple et efficace mais elle n'est malheureusement pas immédiate. Par exemple, `\draw (3, 0) arc (0: 60: 3cm);` donnera :

Pour expliciter l'instruction arc, le schéma suivant nous sera utile.

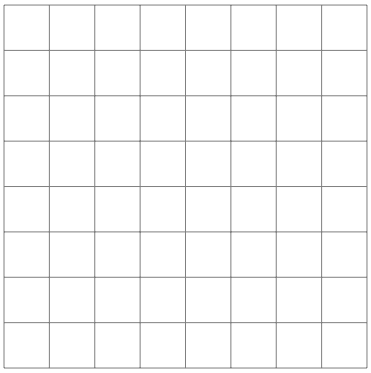


Avec l'instruction *arc (A : angle : rayon)*, TikZ a choisi de s'exprimer de telle sorte que l'arc de cercle commence au point A – le point (3, 0) dans notre exemple – se poursuit jusqu'à un angle polaire de 60 ° – dans l'exemple présenté – avec un rayon qui, dans notre schéma, vaut 3 cm. Le centre du cercle n'est jamais spécifié même si nous disposons de suffisamment d'éléments pour le retrouver.

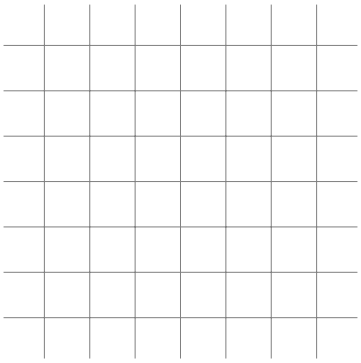
1.3 Grille, axes

1.3.1 Grille

L'instruction `\draw[step=1cm, gray, very thin] (-2, -2) grid (6, 6);` donne une grille dont le coin inférieur gauche a pour coordonnées (-2, -2) tandis que le coin supérieur droit se trouve en (6, 6). Les traits sont espacés de 1cm (step=1cm), de couleur grise (gray) et les traits sont fins (very thin).



Si on remplace (-2, -2) par (-1.9, -1.9) et (6, 6) par (5.9, 5.9), on obtient la même grille mais, cette fois, dépourvue de bordure extérieure.

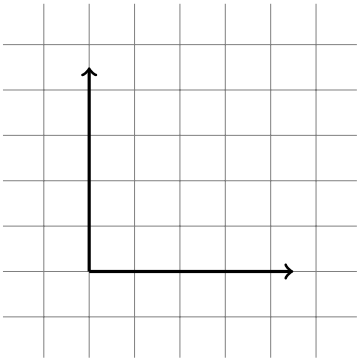


1.3.2 Axes

En ajoutant à l'instruction précédente les commandes suivantes

```
\draw[very thick, ->] (0, 0) -- (4.5, 0);
```

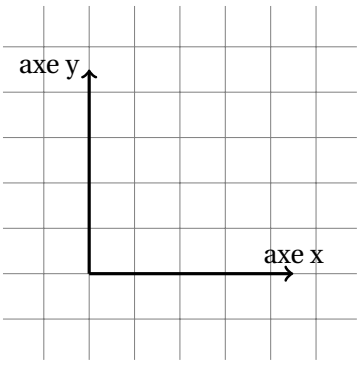
```
\draw[very thick, ->] (0, 0) -- (0, 4.5);
```
on obtient les axes traditionnels de coordonnées. On peut spécifier que l'extrémité d'un segment soit une pointe de flèche avec l'option [->]. Si la flèche doit se trouver à l'origine du trait, on mettra [<-]. Il est possible de combiner les deux, ainsi l'option [<->] placera une pointe de flèche à chaque extrémité.



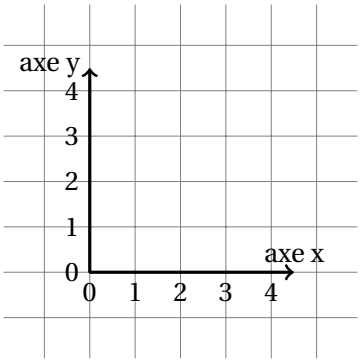
Il est possible également de légender les axes avec *node* qui permet de placer un texte LaTeX entre les accolades selon les options de positionnement suivantes.

above	*below*	*right*	*left*
aboveleft	*aboveright*	*belowleft*	*belowright*

Ainsi, en introduisant
```
\draw[very thick, ->] (0, 0) -- (4.5, 0) node[below]{axe x};
\draw[very thick, ->] (0, 0) -- (0, 4.5) node[left]{axe y};
```
nous obtenons :



Il reste à, éventuellement, graduer nos axes de coordonnées.



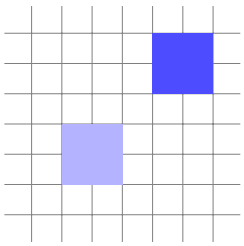
1.4 Gestion des couleurs

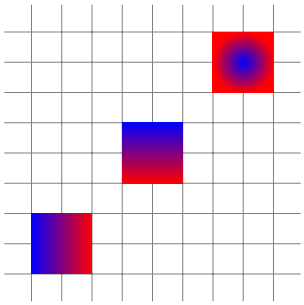
Nous trouvons, dans le code ci-dessous, l'instruction *fill[options]* qui dessine, dans l'exemple de la figure ci-dessous à gauche, un rectangle. L'option, ici *blue !30* définit la couleur de remplissage. Les couleurs disponibles sont les mêmes que précédemment. L'ajout de *!30* indique une possibilité de nuance pour la couleur sélectionnée par le choix d'un nombre entre 0 et 100.

```
\begin{tikzpicture}
\draw[step=1cm, gray, very thin] (-1.9, -1.9) grid (5.9, 5.9);
\fill[blue!30] (0, 0) rectangle (2, 2);
\fill[blue!70] (3, 3) rectangle (5, 5);
\end{tikzpicture}
```

L'instruction \shade joue un rôle similaire à \fill mais elle permet d'obtenir un dégradé de couleurs comme on peut le voir dans le code ci-après qui donne pour résultat la figure ci-dessous à droite.

```
\begin{tikzpicture}
\draw[step=1cm, gray, very thin] (-1.9, -1.9) grid (7.9, 7.9);
\shade[left color = blue, right color = red] (-1, -1) rectangle (1, 1);
\shade[top color = blue, bottom color = red] (2, 2) rectangle (4, 4);
\shade[inner color = blue, outer color = red] (5, 5) rectangle (7, 7);
\end{tikzpicture}
```



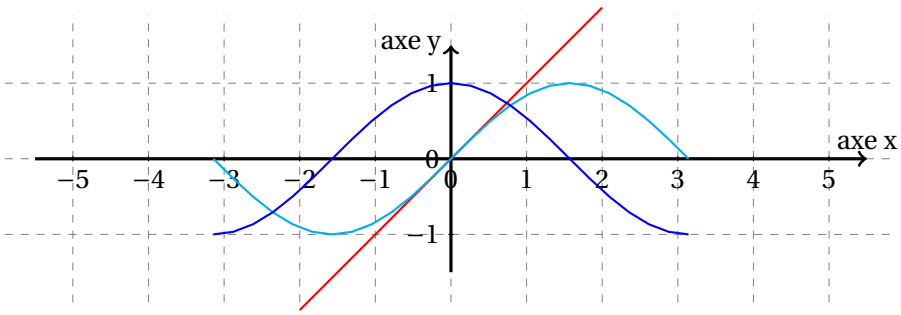


1.5 Graphes de fonctions

Dans cette section, nous présentons les principaux éléments permettant de représenter les fonctions usuelles. Les fonctions données par une équation cartésienne doivent d'abord être réécrites sous forme d'équations paramétriques. Ainsi, les fonctions $f(x) = x$ et $g(x) = sin(x)$ s'écriront respectivement :

$$\begin{cases} x = x \\ y = x \end{cases} \quad \text{et} \quad \begin{cases} x = x \\ y = sin(x) \end{cases}$$

Avec cette écriture, la figure ci-dessous



est obtenue à partir des commandes suivantes :

```
\begin{center}
\begin{tikzpicture}
\draw[step=1cm, gray, very thin] (-5.9, -1.9) grid (5.9, 1.9);
\draw[very thick, ->] (-5.5, 0) -- (5.5, 0) node[above]{axe x};
\draw[very thick, ->] (0, -1.5) -- (0, 1.5) node[left]{axe y};
\foreach \x in {-5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5}
   \draw(\x, 1pt) -- (\x, -1pt) node[below]{$\x$};
\foreach \y in {-1, 0, 1}
   \draw(1pt, \y) -- (-1pt, \y) node[left]{$\y$};
\draw[red, thick] [domain=-2:2] plot (\x, \x);
\draw[cyan, thick] [domain=-pi:pi] plot (\x,{sin(\x r)});
\draw[blue, thick] [domain=-pi:pi] plot (\x,{cos(\x r)});
\end{tikzpicture}
\end{center}
```

FIGURE 1.3 – grille, axes, graphes

Parmi celles-ci, seules les trois lignes

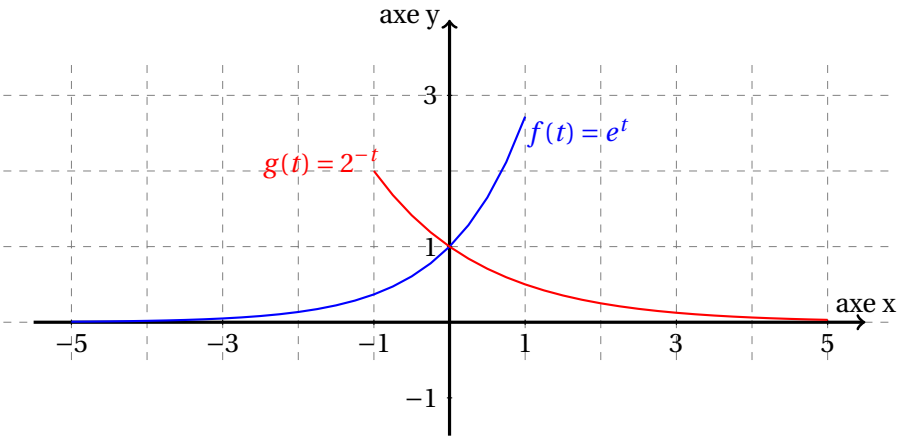
```
\draw[red, thick] [domain=-2:2] plot (\x, \x);
\draw[cyan, thick] [domain=-pi:pi] plot (\x,{sin(\x r)});
\draw[blue, thick] [domain=-pi:pi] plot (\x,{cos(\x r)});
```

sont nouvelles et nécessitent quelques mots d'explication.

Pour la première, nous avons précisé, avec l'option [domain=a :b] l'intervalle [a, b] sur lequel nous souhaitons représenter la fonction $y = x$. Après la commande *plot* la fonction est introduite sous forme d'équations paramétriques (\x, \x) dans laquelle l'abscisse est \x et, bien évidemment, pour cette fonction l'ordonnée est aussi \x (on notera, dans la syntaxe, la présence obligatoire du symbole \ devant la variable).

À la ligne suivante, nous avons `plot (\x,{sin(\x r)});` les fonctions usuelles sont toujours écrites entre accolades {sin(\x r)}. **Nous n'avons pas écrit {sin(\x)} mais {sin(\x r)}. La présence du facteur « r » mérite que l'on s'y attarde. Dans TikZ, les fonctions trigonométriques attendent un argument en degrés. Nous travaillons habituellement en radians. Le facteur r permet de convertir les radians en degrés. Ainsi, sin(π/2 r) renvoie la valeur 1.**

La figure suivante illustre deux fonctions exponentielles. Ces dernières posent également un problème de syntaxe aux habitués des logiciels gérant les mathématiques. TikZ n'est pas un logiciel de calcul. Parmi les problèmes que cela engendre, il ne comprend pas l'exponentiation avec le symbole « ^ ». Ainsi, `\x^2` donne lieu à un graphe incorrect alors que `\x*\x` affiche le graphe attendu. De même, `e^(\x)` pose un problème (essayez, vous serez convaincu) tandis que `exp(\x)` fournit le bon résultat.



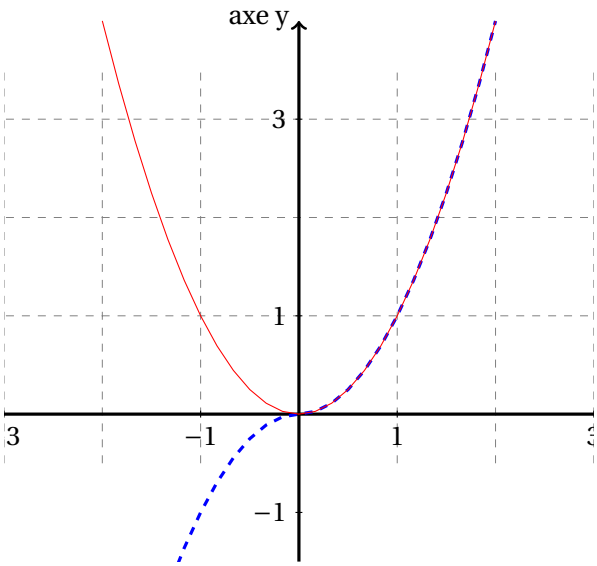
Il est parfaitement possible d'utiliser une autre variable que x. Ainsi, pour utiliser la variable t, nous devons placer l'option *[variable=\t]* dans l'instruction *plot*. De même avec la commande node, on peut introduire des annotations dans la figure obtenue. Le code ci-après, correspond à la figure précédente et illustre notre propos. La fonction $f(t) = 2^{-t}$ est écrite ici $f(t) = e^{-ln(2)\ t} \approx e^{-0.693t}$ ce qui explique la syntaxe utilisée dans l'avant dernière ligne de code.

```
\begin{tikzpicture}[scale=1.3]
\draw[step=1cm, gray, very thin,dashed] (-5.9, -0.5) grid (5.9, 3.5);
\draw[very thick, ->] (-5.5, 0) -- (5.5, 0) node[above]{axe x};
\draw[very thick, ->] (0, -1.5) -- (0, 4) node[left]{axe y};
\foreach \x in {-5, -3, -1, 1, 3, 5}
   \draw(\x, 1pt) -- (\x, -1pt) node[below]{$\x$};
\foreach \y in {-1, 1, 3}
   \draw(1pt, \y) -- (-1pt, \y) node[left]{$\y$};
\draw [domain=-5:1,thick,blue] plot [variable=\t] (\t,{exp(\t)});
\node[blue,very thick] (A) at (1.7,2.5) {$f(t) = e^t$};
\draw [domain=-1:5,thick,red] plot [variable=\t] (\t,{exp(-0.693*\t)});
\node[red,very thick] (B) at (-1.7,2.1) {$g(t) = 2^{- t}$};
\end{tikzpicture}
```

FIGURE 1.4 – graphe de fonctions exponentielles

Les fonctions usuelles implantées dans TikZ sont données ci-dessous. Il s'agit de : *abs(x), exp(x), ln(x), sqrt(x), sin(x), cos(x), tan(x), cot(x), sec(x), cosec(x), asin(x), acos(x), atan(x)*. La liste n'est pas exhaustive, nous renvoyons à la documentation principale pour tous compléments d'informations `http://www.texample.net/media/pgf/builds/pgfmanualCVS2012-11-04.pdf`.

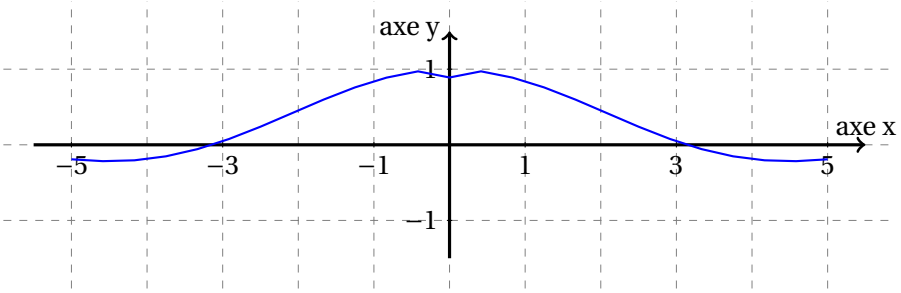
La figure suivante montre, en trait discontinu bleu, ce que l'on obtient avec `\t^2` alors que `\t*\t` donne, en rouge la courbe attendue.



1.5.1 Travailler l'aspect du graphe avec samples

L'option *samples* permet de forcer *TikZ* à prendre davantage de points en considération. Le graphe ci-dessous est celui de la fonction $f(x) = \frac{sin(x)}{x}$. Ce

dernier présente, visiblement, un problème dans le voisinage de $x = 0$. Ceci est dû au fait que quand *TikZ* doit tracer une courbe avec *plot*, il calcule par défaut 25 points de celle-ci qu'il joint ensuite par des segments.



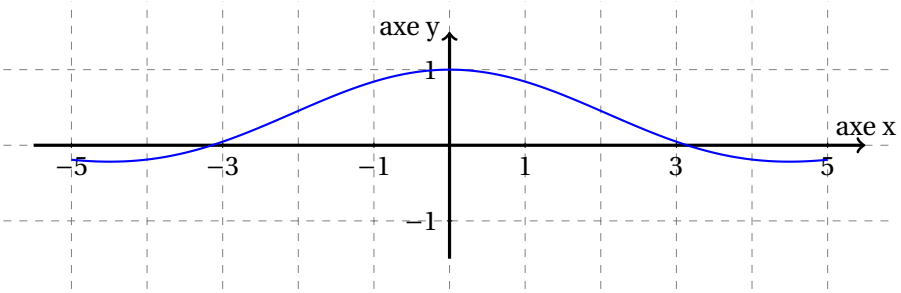
Ce nombre de points calculés peut être modifié par l'option *samples*. Dans la figure ci-dessous, nous avons remplacé la ligne de code

```
\draw [domain=-5:5] plot (\x,{sin(\x r)/\x});
```

par

```
\draw [domain=-5:5,samples=200] plot (\x,{sin(\x r)/\x});
```

TikZ a, cette fois, calculé 200 points et le résultat correspond cette fois à la fonction souhaitée.

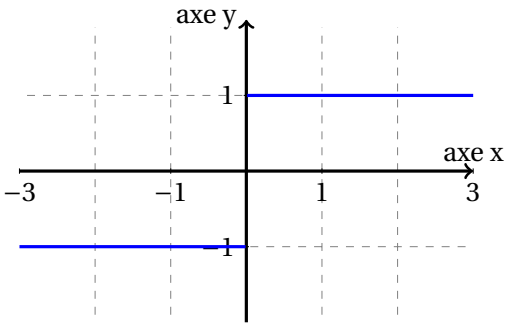


1.5.2 Discontinuités et valeurs arbitrairement grandes

La fonction $f(x) = \dfrac{|x|}{x}$ présente une discontinuité en $x = 0$. Pour éviter un message d'erreur signalant que l'on divise par zéro, il est judicieux de tracer la fonction en deux étapes avec, par exemple,

```
\draw [domain=-3:-0.01] plot (\x,{abs(\x)/\x});
 \draw [domain=0.01:3] plot (\x,{abs(\x)/\x});
```

nous avons alors :

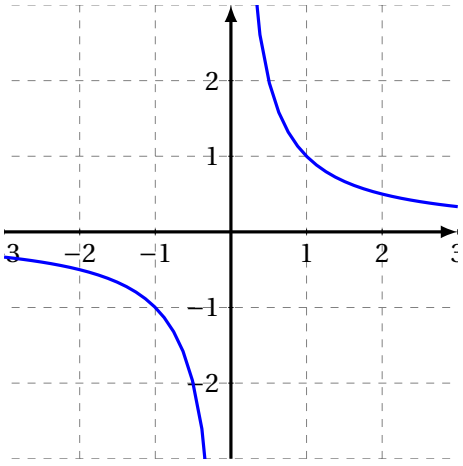


Pour des fonctions telles que $1/x$ qui prennent des valeurs arbitrairement grandes au voisinage de $x = 0$, on peut utiliser l'option *scale* de *tikzpicture* que nous avons déjà rencontrée mais aussi l'instruction *clip* qui limite la figure à l'intérieur d'un rectangle (`\clip (-3,-3) rectangle (3,3);`).

Pour éviter d'être confronté à un message d'erreur indiquant qu'il est impossible de diviser par zéro, on sépare le graphe de la fonction $1/x$ en deux parties.

```
\draw [domain=-3:-0.01,very thick,blue] plot (\x,{1/\x});
\draw [domain=0.01:3,very thick,blue] plot (\x,{1/\x});
```

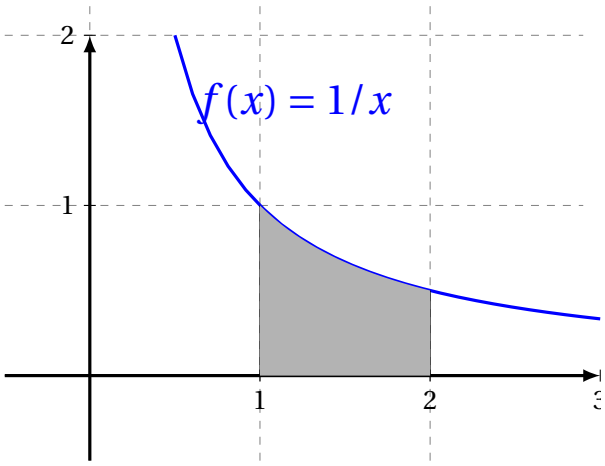
La fonction $f(x) = 1/x$ est alors représentée par :



1.5.3 Surface ombrée limitée par des courbes

La figure ci-dessous illustre la mesure d'une aire par une intégrale définie. Il s'agit, bien entendu, de

$$\int_1^2 \frac{1}{x}\,dx = ln(2) \approx 0.693.$$



L'ensemble de la figure ne pose pas de problème. Attardons-nous sur deux lignes de code :

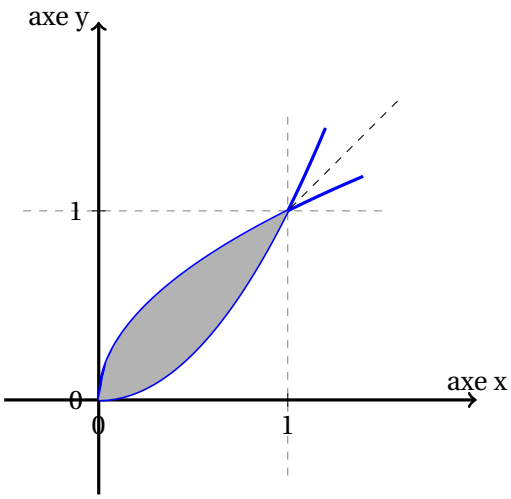
```
\node [color=blue, thick] (T) at (1.2,1.6) {\LARGE $f(x) = 1/x$};
\fill[color=gray!60]
   (1,0) -- (1,1) -- plot [domain=1:2] (\x, 1/\x) -- (2, 0) -- cycle;
```

la première montre qu'il est possible d'insérer dans un noeud une formule mathématique précédée d'une instruction sur la taille de la police utilisée, la seconde utilise la commande *fill* pour remplir la surface désirée. On remarque la manière dont l'instruction *plot* est intégrée dans le cycle.

Nous voulons maintenant ombrer la région comprise entre la courbe $y = \sqrt{x}$ et la courbe $y = x^2$ sur l'intervalle $[0,1]$. Nous avons utilisé l'option samples=100 pour le tracé des courbes. Quant à la région ombrée, elle est construite par la ligne de code suivante :

```
\fill[color=gray!60]
   (0,0) -- plot [domain=0:1] (\x, {sqrt(\x)}) --
   plot [domain=1:0] (\x, {\x*\x}) -- cycle;
```

Il s'agit, dans un cas semblable, d'être attentif au sens des tracés des courbes. La première est parcourue de $x = 0 \longrightarrow x = 1$, la seconde est parcourue de $x = 1 \longrightarrow x = 0$. Nous obtenons alors :



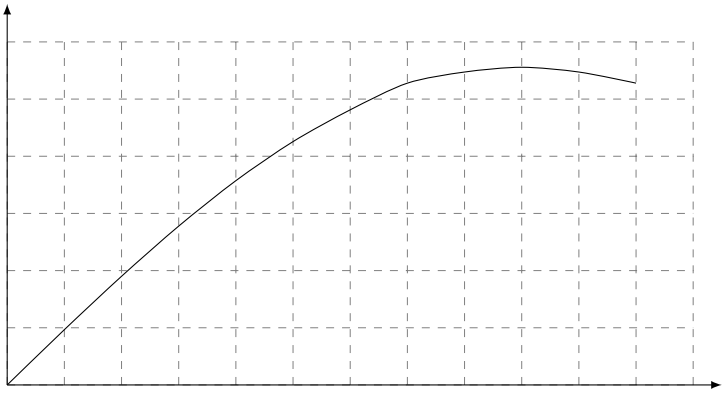
1.5.4 Fonction donnée par un ensemble de points

Soit une fonction donnée par un ensemble de points. Dans l'exemple qui suit, l'abscisse des points est comprise entre 0 et 11, quant à l'ordonnée des points elle est située entre 0 et 1000. Dans les options de l'environnement tikzpicture, nous définissons l'unité des variables x et y, ici le mm. À la dernière ligne de code, nous utilisons l'instruction *plot[options]coordinates* qui contient les coordonnées des différents points. Parmi les options, nous trouvons *smooth* qui permet un lissage de la courbe obtenue ainsi que des instructions quant aux échelles utilisées *xscale=3.6* (une abscisse de 11 est représentée par 39,6mm) et *yscale=0.02* (une ordonnée de 1000 est représentée par 20mm). N'oublions pas que l'option scale=2.5 de tikzpicture multiplie sur la feuille toutes ces valeurs par 2.5.

```
\begin{center}
\begin{tikzpicture}[x=1mm,y=1mm,scale=2.5]
\draw[step=3.6mm, gray, very thin,dashed] (0, 0) grid (43.2, 22);
\draw[->,>=latex] (0,0) -- (45,0);
\draw[->,>=latex] (0,0) -- (0,24);
\draw plot[smooth,very thick,xscale=3.6,yscale=0.02]coordinates{(0,0) (1,174)
(2,342) (3,500) (4,643) (5,766) (6,866) (7,950) (8,985) (9,1000) (10,985) (11,950)};
\end{tikzpicture}
\end{center}
```

La grille va de (0, 0) à (43.2, 22). Ici, 43.2 = (12)(3.6) et 22 = (1100)(0.02). Quant aux axes, ils sont construits sur base des mêmes valeurs.

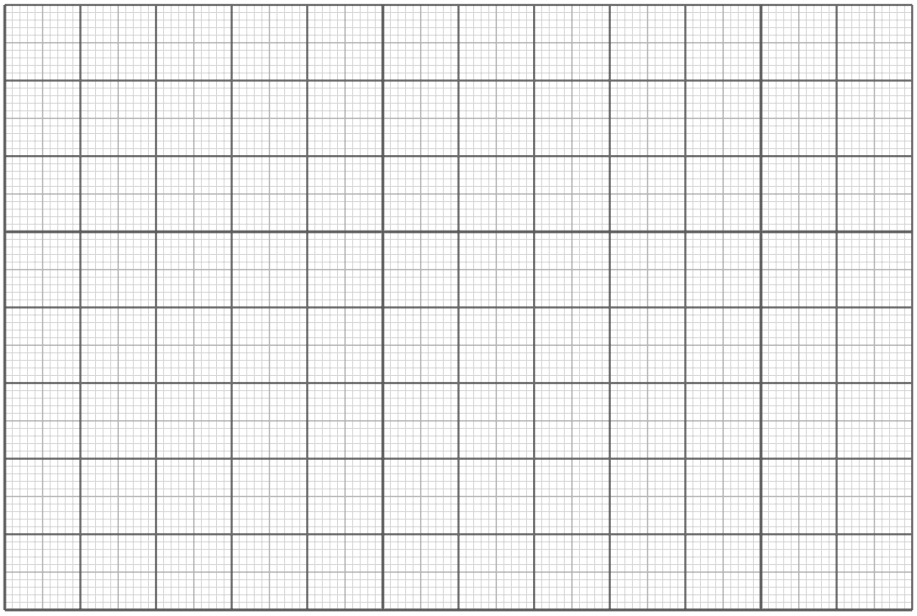
Le résultat obtenu est présenté ci-après.



1.5.5 Insérer une feuille de papier millimétré dans un document

Pour clore ce premier regard sur l'outil TikZ, nous donnons le code permettant d'obtenir l'équivalent d'une portion de papier millimétré. Cette dernière peut facilement être insérée dans des travaux d'élèves.

```
\begin{center}
% Pour une présentation correcte, on se limitera à une largeur de 12 cm et à une
hauteur de 20 cm
\def\width{12}
\def\hauteur{8}
\begin{tikzpicture}[x=1cm, y=1cm, semitransparent]
\draw[step=1mm, line width=0.1mm, black!30!white] (0,0) grid (\width,\hauteur);
\draw[step=5mm, line width=0.2mm, black!40!white] (0,0) grid (\width,\hauteur);
\draw[step=5cm, line width=0.5mm, black!50!white] (0,0) grid (\width,\hauteur);
\draw[step=1cm, line width=0.3mm, black!90!white] (0,0) grid (\width,\hauteur);
\end{tikzpicture}
\end{center}
```



Chapitre 2

Transformations géométriques avec *scope*

2.1 Transformation par translation

Ce chapitre porte sur les transformations géométriques d'une figure avec un nouvel environnement *scope* et des options de translation, de rotation et de dilatation.

Pour ce faire, on place la figure de base dans un environnement *tikzpicture* à l'intérieur duquel nous dupliquons la figure au sein d'un environnement scope comme dans la figure suivante.

```
\begin{tikzpicture}
   % code de la figure
   \begin{scope}[options de la transformation]
      % code de la figure
   \end{scope}
\end{tikzpicture}
```

Afin d'illustrer notre propos, nous avons choisi pour figure de base un triangle quelconque, celui de gauche dans la figure ci-dessous. La figure de droite est obtenue en répétant le code de la figure de base dans l'environnement *scope* avec, ici, l'option de translation *xshift* qui déplace la figure parallèlement à l'axe des x.

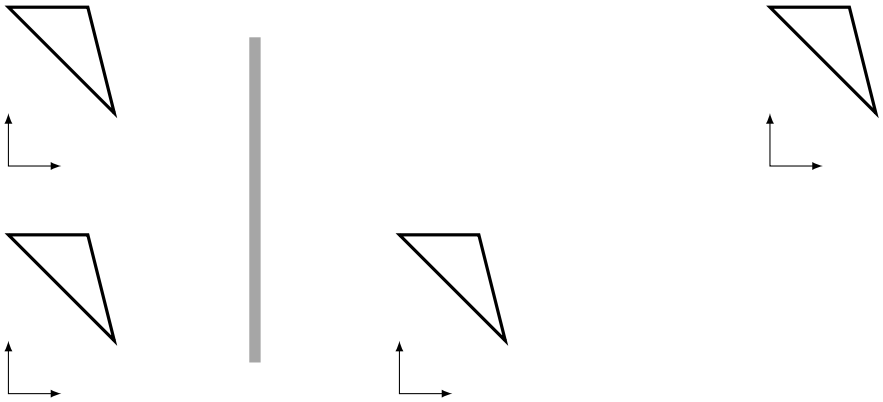
Ainsi, pour obtenir la figure ci-après,

nous avons utilisé le code suivant.

```
\begin{tikzpicture}
   \draw[<->,>=latex] (0, 1) -- (0, 0) -- (1, 0);
   \draw[very thick] (2,1) -- (0,3) -- (1.5,3) -- cycle;
   \begin{scope}[xshift=7cm]
      \draw[<->,>=latex] (0, 1) -- (0, 0) -- (1, 0);
      \draw[very thick] (2,1) -- (0,3) -- (1.5,3) -- cycle;
   \end{scope}
```

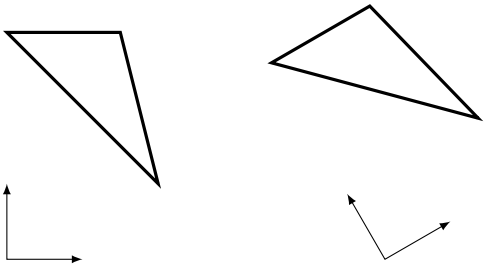
Les figures qui suivent illustrent une translation verticale (à gauche) et une autre définie par un vecteur translation(à droite). La première s'obtient simplement à partir de l'option ***[yshift=43mm]*** tandis que la seconde est construite avec ***[shift=(7cm,43mm)]***. Dans ce dernier cas, les coordonnées du vecteur translation sont placées entre accolades afin d'éviter que la virgule ne soit interprétée par LaTeX comme un passage à l'option suivante.

Les distances, visiblement, peuvent être données en cm, en mm ou encore en pt (ce qui, on en conviendra, est plus ardu à gérer).

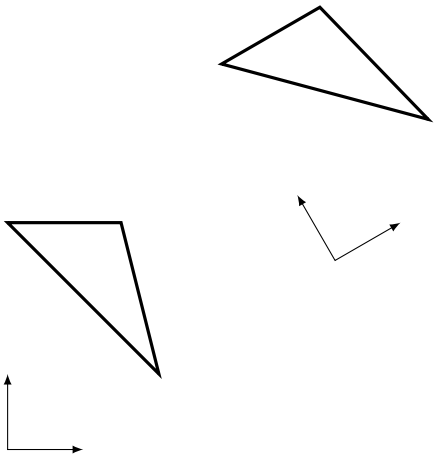


2.2 Transformation par rotation

L'option *rotate* attend une indication d'angle en degrés. Ainsi, avec la même figure qu'auparavant mais avec l'option ***[xshift=5cm,rotate=30]***, on obtient :

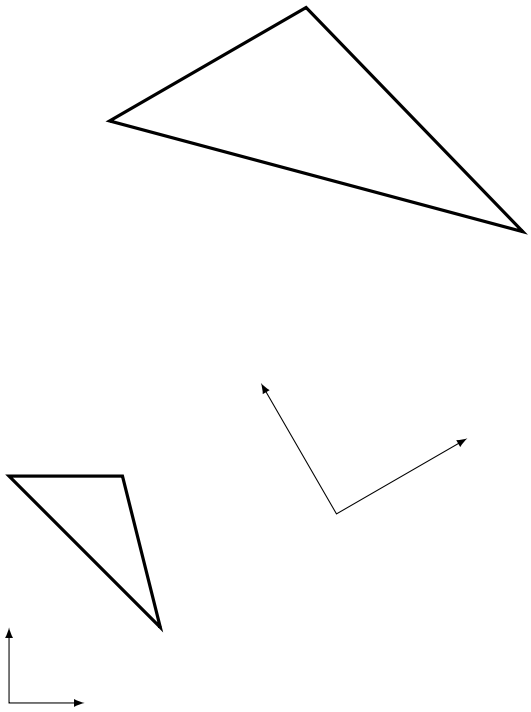


Attention , l'ordre des options a, ici, son importance. En inversant l'ordre des options précédentes, c'est-à-dire avec ***[rotate=30,xshift=5cm]*** , nous aurons, assez logiquement :



2.3 Transformation par dilatation

Une transformation par dilatation est une homothétie qui est gérée par l'option *scale* de l'environnement *scope*. Ainsi, l'option ***[rotate=30,xshift=5cm,scale=2]*** engendre la figure :



Signalons, pour clore le sujet des transformations qu'il est permis de combiner une translation, une rotation et une dilatation dans la même série d'options.

Chapitre 3

Diagrammes – éléments de construction

Commençons par charger quelques librairies utiles pour l'élaboration de nos diagrammes.
Pour ce faire, l'instruction `\usetikzlibrary{arrows,shapes,positioning}` doit être placée, dans le préambule, juste en-dessous de la déclaration du package *tikz*.

```
\usepackage{pgf, tikz}
  \usetikzlibrary{arrows,shapes,positioning}
```

FIGURE 3.1 – Librairie TikZ dans le préambule

3.1 Noeuds contenant de l'information et leurs liens

3.1.1 les instructions node et \node

En mathématique, une primitive est une famille de fonctions. On peut utiliser la syntaxe suivante pour relier ces deux informations par un trait.

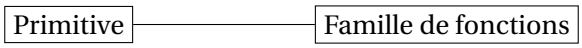
```
\begin{tikzpicture}
    \draw (0,0)node[draw]{Primitive} -- (5,0)node[draw]{Famille de fonctions} ;
\end{tikzpicture}
```

Ici, l'instruction *node* est exécutée sur un chemin. La commande *draw* trace un trait entre deux points qui sont les noeuds où l'on place l'information. Ceci donne :

Il existe aussi une commande **node** qui permet de définir un noeud, de le nommer et de le dessiner. Les noeuds sont ensuite reliés avec *draw*. On écrit alors :

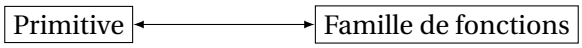
```
\begin{tikzpicture}
    \node[draw] (P) at (0,0) {Primitive} ;
    \node[draw] (F) at (5,0) {Famille de fonctions} ;
    \draw (P) -- (F) ;
\end{tikzpicture}
```

pour obtenir :

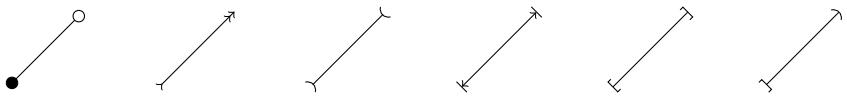


Par défaut, un noeud a la forme d'un rectangle, sans l'option draw, le bord de celui-ci n'est pas tracé.

Dans la dernière commande draw, on peut imposer l'allure des extrémités des traits.
En remplaçant `\draw (P) -- (F);` par `\draw[<->,>=latex] (P) -- (F);` . Le trait se termine alors par une pointe de flèche à chacune de ses extrémités.



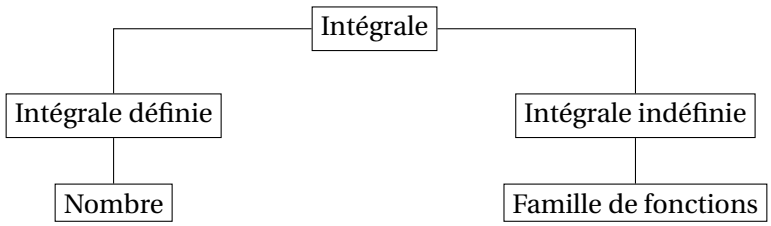
Quelques exemples de terminaisons des traits sont donnés ci-dessous :



Celles-ci sont obtenues à partir des commandes suivantes :

```
\begin{tikzpicture}
\draw[*-o] (0, 0) -- (1, 1); \draw[>->>] (2, 0) -- (3, 1); \draw[]-(] (4, 0) -- (5, 1);
\draw[|<->|] (6, 0) -- (7, 1); \draw[{[-]}] (8, 0) -- (9, 1); \draw[{]-)}] (10, 0) -- (11, 1);
\end{tikzpicture}
```

Pour clore ce premier paragraphe, les notions vues jusqu'ici permettent déjà d'élaborer bon nombre de diagrammes. Ainsi, le graphe suivant :



se construit à partir du code ci-dessous :

```
\begin{tikzpicture}[scale=1.15]
\node[draw] (I) at (0,0) {Intégrale};
\node[draw] (ID) at (-3,-1) {Intégrale définie};
\node[draw] (P) at (3,-1) {Intégrale indéfinie};
\node[draw] (N) at (-3,-2) {Nombre};
\node[draw] (F) at (3,-2) {Famille de fonctions};
\draw (I) -| (ID); \draw (I) -| (P);
\draw (ID) -- (N); \draw (P) -- (F);
\end{tikzpicture}
```

Attention l'angle droit que présente le trait entre *intégrale* et *intégrale définie* s'obtient en remplaçant dans la commande *draw* le double trait (- -) habituel par un trait suivi d'un second mais vertical (-|) comme on peut le voir dans la troisième ligne de code en partant du bas.
La liaison entre deux noeuds s'écrit - - ou -| ou encore |- ce qui correspond respectivement à un trait direct entre les noeuds ou un trait d'abord horizontal puis vertical ou encore un trait initial vertical suivi d'une partie horizontale.

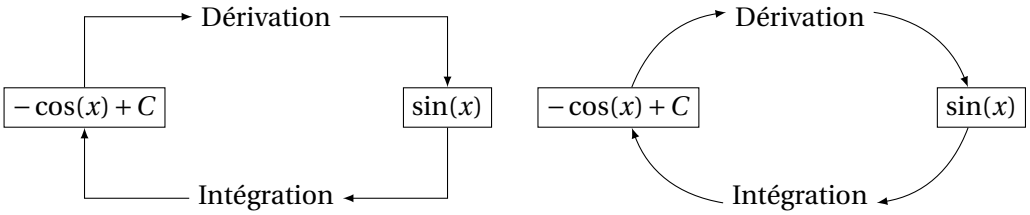
Notons à la première ligne de ce code l'instruction ***scale*** qui offre la possibilité de modifier l'échelle de l'ensemble de la figure.

3.2 Agrémenter la présentation

3.2.1 Des liens courbes entre les noeuds

Dans les schémas présentés ci-dessous, celui de gauche ne devrait plus poser de problème si ce n'est que les formules mathématiques sont, comme toujours en LaTeX, placées entre deux symboles $.

Pour relier deux noeuds, on peut aussi utiliser la commande *to* suivie d'une option de courbure. Ainsi, le schéma de droite est obtenu à partir du code

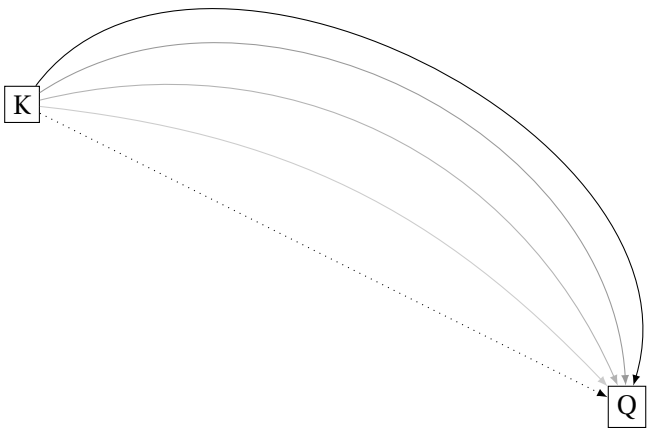



```
\begin{tikzpicture}[scale=0.6]
    \node[draw] (P) at (0,0) {$- \cos(x) + C$} ;
    \node[draw] (F) at (8,0) {$\sin(x)$} ;
    \node (D) at (4,2) {Dérivation} ;
    \node (I) at (4,-2) {Intégration} ;
    \draw[->,>=latex] (P) to[bend left] (D) ; \draw[->,>=latex] (D) to[bend left] (F) ;
    \draw[->,>=latex] (F) to[bend left] (I) ; \draw[->,>=latex] (I) to[bend left] (P) ;
\end{tikzpicture}
```

au sein duquel nous trouvons les instructions **to[bend left]**. Celles-ci indiquent à TikZ l'intention de tracer une courbe allant du premier noeud au second de telle sorte que la courbure soit à la gauche du trait rectiligne allant du noeud 1 au noeud 2. Pour éclairer notre propos, nous donnons ci-après quelques exemples de traits tracés avec **to[bend left]**, le chemin direct étant illustré en gris.

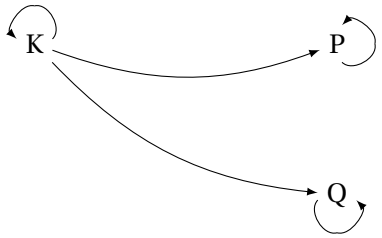
L'instruction **to[bend right]** place la courbure du côté droit de la ligne directe allant du premier noeud au second.

L'importance de la courbure de l'option bend peut être réglée en indiquant un angle en degrés (0˚correspond à la ligne droite), par exemple **to[bend left=20]**.



Le code ci-dessous donne la figure située à sa droite. Notons la façon de décrire une boucle. Pour aller de K à K, nous le faisons en deux étapes avec un point intermédiaire (3^e et 4^e ligne d'instructions \draw).

```
\begin{center}
\begin{tikzpicture}[scale=0.5]
      \node[] (K) at (0,0) {K} ;
      \node[] (Q) at (8,-4) {Q} ;
      \node[] (P) at (8,0) {P} ;
      \draw[->,>=latex] (K) to[bend right=20] (Q) ;
      \draw[->,>=latex] (K) to[bend right=20] (P) ;
      \draw (K) to[bend right=75] (0,1) ;
      \draw[->,>=latex] (0,1) to[bend right=75] (K) ;
      \draw (Q) to[bend right=75] (8,-5) ;
      \draw[->,>=latex] (8,-5) to[bend right=75] (Q) ;
      \draw (P) to[bend right=75] (9,0) ;
      \draw[->,>=latex] (9,0) to[bend right=75] (P) ;
\end{tikzpicture}
\end{center}
```

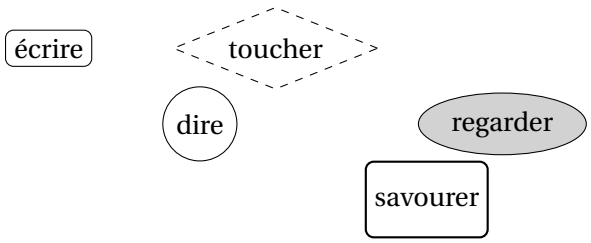


3.2.2 Style des noeuds, placement d'annotations

Les noeuds ont une forme rectangulaire par défaut. Il est tout à fait possible de modifier ceci avec des options

- de forme : **rectangle** (par défaut), **circle, diamond, ellipse**
- de style : **dashed, dotted, thick, fill**
- d'aspect : **rounded corners, minimum width, minimum height**

Ainsi, les contours suivants

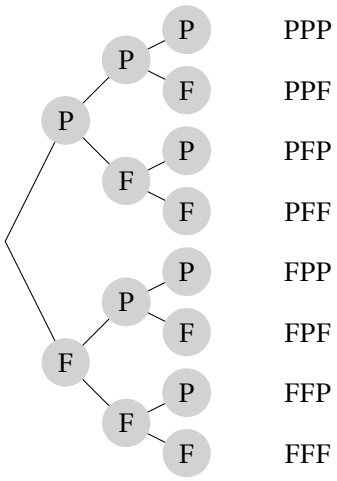


s'obtiennent avec les lignes de code suivantes :

```
\node[draw,rounded corners=3pt] (E) at (0,0) {écrire};
\node[draw,ellipse,fill=gray!35] (R) at (6,-1) {regarder};
\node[draw,circle] (E) at (2,-1) {dire};
\node[draw,diamond,aspect=2.5,dashed] (E) at (3,0) {toucher};
\node[draw,rounded corners=3pt,thick, minimum height=1cm] (E) at (5,-2) {savourer};
```

Remarquons que l'option *diamond* nécessite une option supplémentaire *aspect*= qui impose le rapport entre la largeur et la hauteur de ce contour.

La commande *tikzstyle* évite de devoir répéter l'allure d'un contour lorsque ce dernier est utilisé plusieurs fois dans un même graphe. Ainsi, dans le graphe suivant qui illustre les résultats possibles de trois lancés successifs d'une pièce de monnaie,

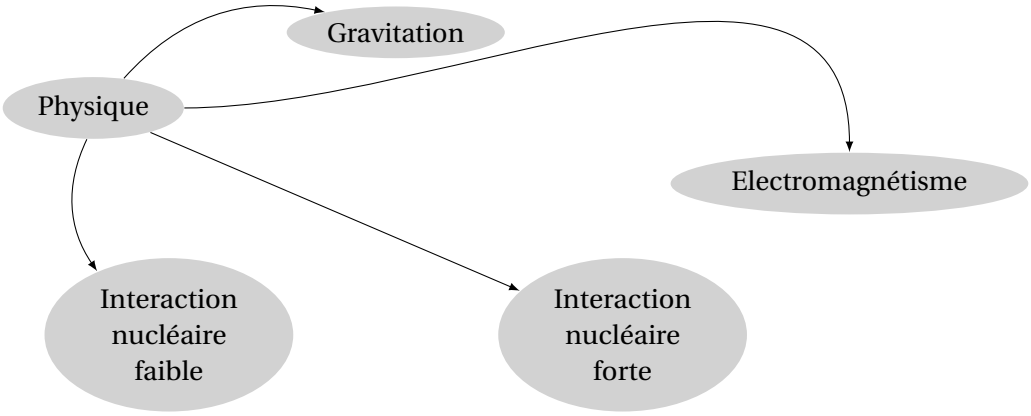


on écrira le code suivant :

```
\tikzstyle{cg}=[circle,fill=gray!35] % cg=cerclegris
\begin{tikzpicture} [scale=0.8]
      \draw (1,2) -- ( 0,0) -- (1,-2);
      \draw (2,3) -- (1, 2)node[cg] {P} -- (2,1) ;
      \draw (2,-1) -- (1,-2)node[cg] {F} -- (2,-3) ;
      \draw (3,3.5)node[cg] {P} -- (2,3)node[cg] {P} -- (3,2.5)node[cg] {F} ;
      \draw (3,1.5)node[cg] {P} -- (2,1)node[cg] {F} -- (3,0.5)node[cg] {F} ;
      \draw (3,-0.5)node[cg] {P} -- (2,-1)node[cg] {P} -- (3,-1.5)node[cg] {F} ;
      \draw (3,-2.5)node[cg] {P} -- (2,-3)node[cg] {F} -- (3,-3.5)node[cg] {F} ;
      \node (a) at (5,3.5) {PPP}; \node (a) at (5,2.5) {PPF};
      \node (a) at (5,1.5) {PFP}; \node (a) at (5,0.5) {PFF};
      \node (a) at (5,-0.5) {FPP}; \node (a) at (5,-1.5) {FPF};
      \node (a) at (5,-2.5) {FFP}; \node (a) at (5,-3.5) {FFF};
\end{tikzpicture}
```

dont la première ligne illustre notre propos.
L'instruction `\tikzstyle{cg}=[circle,fill=gray!35]` définit un style que nous avons choisi de nommer *cg* pour lequel chaque noeud sera un cercle rempli avec un gris à 35%. Dans les lignes suivantes, il suffit de rappeler ce nom *cg* pour obtenir les contours désirés.

L'instruction *edge* peut avantageusement remplacer la commande *to*. Elle présente la particularité de dessiner une ligne jusqu'au point suivant tout en conservant le crayon au point de départ pour construire d'autres liaisons issues de ce même point. Ainsi, le graphe suivant :



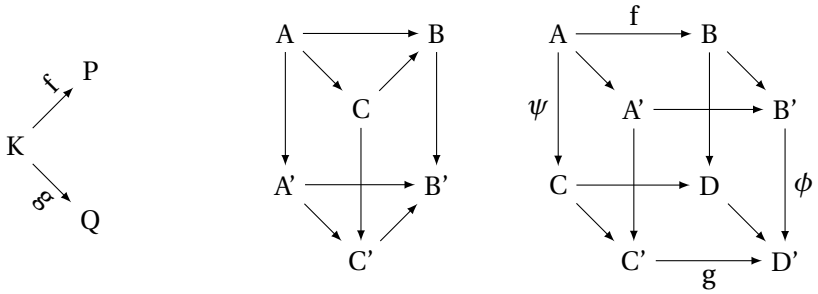
s'obtient à partir du code ci-après.

```
\tikzstyle{phys}=[ellipse,fill=gray!35, minimum width, text centered]
\tikzstyle{phys2}=[ellipse,fill=gray!35, text width=2.1cm, text centered]
\begin{tikzpicture}
   [scale=1]
      \node[phys] (P) at (0,0) {Physique} ;
      \node[phys] (G) at (4,1) {Gravitation} ;
      \node[phys] (E) at (10,-1) {Electromagnétisme} ;
      \node[phys2] (F) at (7,-3) {Interaction nucléaire forte} ;
      \node[phys2] (f) at (1, -3) {Interaction nucléaire faible} ;
      \draw[->,>=latex] (P) edge[bend left] (G)
      edge [out=0,in=90] (E)
      edge (F)
      edge [bend right] (f) ;
\end{tikzpicture}
```

La seule instruction *draw* qu'il contient montre que l'on trace une ligne de (P) vers (G) avec la commande *edge[bend left]*. Ensuite, le point de départ étant toujours (P), il suffit d'un *edge[out=0, in=90]* pour partir vers (E). Le *out=0* indique que l'on sort du noeud (P) avec un trait qui présente un angle polaire de 0°, tandis que le *in=90* force la ligne à entrer dans le noeud (E) avec un angle de 90°(regarder la figure correspondante). Le *edge* suivant permet d'aller, en ligne droite, de (P) vers (F), alors que le dernier *edge[bend right]* lie le noeud (P) au noeud (f) avec la courbure souhaitée.

Le dernier schéma de ce chapitre montre, comme seule nouveauté, la méthode permettant d'écrire un symbole le long d'un chemin. C'est le cas des figures de gauche et de droite ci-dessous.

Le code permettant de construire la figure de gauche est :




```
\begin{tikzpicture}
   \node (K) at (0,0) {K};
   \node (P) at (1,1) {P};
   \node (Q) at (1,-1) {Q};
   \draw[->,>=latex,] (K) -- (P) node[near end,left,above,sloped] {f} ;
   \draw[->,>=latex,] (K) -- (Q) node[pos=0.6,right,below,sloped] {g};
\end{tikzpicture}
```

Après affectation dans une commande *draw* d'un chemin (K) - - (P), l'instruction *node* place, par défaut, un noeud au point (B). Si l'on ajoute à cette instruction l'option *[near end]* le noeud sera placé à proximité du point (P) (près du point final de la ligne).

En plus de l'option *near end*, il existe aussi les options *very near start, near start, midway, near end, very near end* pour les placements approximatifs et l'option *pos*= qui permet une plus grande précision. Le nombre qui suit *pos*= est le pourcentage du chemin à parcourir pour placer le noeud : 0 en (K), 1 en (P), 0.5 au milieu de (K) - - (P).

Les options left, right, below, above placent le symbole souhaité respectivement à la gauche, à la droite, en dessous, au-dessus de la ligne.

L'option *sloped* permet d'incliner le symbole en fonction de la pente du chemin.

Le code qui construit la figure de droite est, quant à lui :

```
\begin{tikzpicture}
   \node (C') at (0,0) {C'}; \node (D') at (2,0) {D'}; \node (A') at (0,2) {A'}; \node (B') at (2,2) {B'};
   \node (C) at (-1,1) {C}; \node (D) at (1,1) {D}; \node (A) at (-1,3) {A}; \node (B) at (1,3) {B};
   \draw[->,>=latex,] (A) -- (A') ; \draw[->,>=latex,] (A') -- (C') ;
   \draw[->,>=latex,] (A) -- (B) node[midway,above] {f} ; \draw[->,>=latex,] (B) -- (B') ;
   \draw[->,>=latex,] (A) -- (C) node[midway,left] {$\psi$} ; \draw[->,>=latex,] (C) -- (C') ;
   \draw[->,>=latex,] (A') -- (B') ; \draw[->,>=latex,] (B') -- (D') node[midway,right] {$\phi$} ;
   \draw[->,>=latex,] (C) -- (D) ; \draw[->,>=latex,] (D) -- (D') ;
   \draw[->,>=latex,] (B) -- (D) ; \draw[->,>=latex,] (C') -- (D') node[midway,below] {g} ;
\end{tikzpicture}
```

dans lequel la totalité de la syntaxe est maintenant connue.

Chapitre 4

Créer des cartes mentales – Mind Maps

4.1 Dessiner un arbre

Dans ce chapitre, nous allons montrer comment créer des cartes mentales à partir de l'environnement TikZ. Pour ce faire, nous allons simplifier et adapter un exemple que l'on peut trouver dans l'imposant manuel TikZ et PGF cité dans l'introduction de ce manuel.

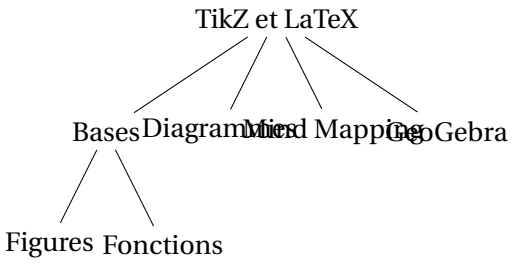
Il importe, avant tout, de vérifier la présence, dans le préambule du document, des lignes suivantes :

- `\usepackage{tikz}`
- `\usetikzlibrary{mindmap}`
- `\pagestyle{empty}`

Nous nous proposons de construire progressivement une carte mentale présentant les différents points traités dans le présent manuel.

```
\begin{center}
\begin{tikzpicture}
  \node {TikZ et LaTeX}
    child {node {Bases}
      child {node {Figures}}
      child {node {Fonctions}}}
    child {node {Diagrammes}}
    child {node {Mind Mapping}}
    child {node {GeoGebra}} ;
\end{tikzpicture}
\end{center}
```

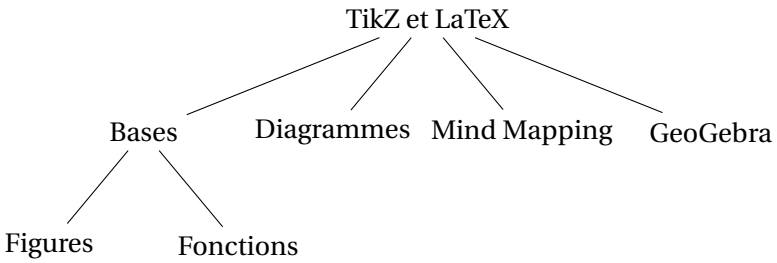
Nous commençons par créer un premier noeud, avec *node*, qui contient le titre ou le concept central de la carte, ici *TikZ et LaTeX*. Ensuite, à ce noeud de base nous pouvons ajouter des « enfants » avec l'instruction *child* qui est suivie entre accolades par un noeud. Ces derniers peuvent eux-mêmes avoir d'autres noeuds enfants et ainsi de suite. Ainsi, le noeud *Bases* a comme enfants les noeuds *Figures* et *Fonctions*. Le code ci-contre illustre la construction de la première figure de ce chapitre. Notez la gestion des accolades pour passer d'un niveau de la carte au suivant et la position du point virgule pour terminer la commande.



Le résultat est largement insatisfaisant du fait de la superposition du premier niveau des noeuds enfants. Pour éviter cela, nous disposons des options *level distance* qui permet de préciser l'écart entre chaque niveau et *sibling distance* qui elle impose l'écart entre les enfants d'un même niveau. Avec ces options, nous reprenons le code précédent en complétant la ligne qui ouvre l'environnement tikzpicture de la façon suivante :

```
\begin{tikzpicture}[level distance=15mm, sibling distance=25mm]
```

Nous obtenons alors :



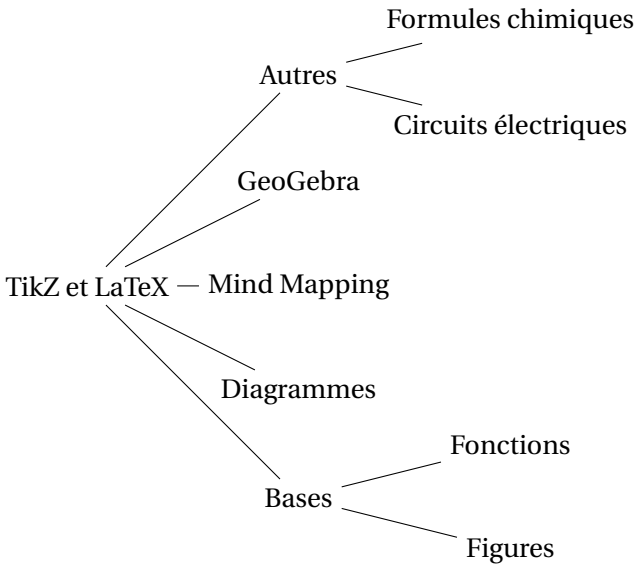
On peut se rendre compte, tout de suite, qu'il est pratiquement impossible de placer les intitulés de tous les chapitres sur la largeur d'une page. Pour contourner le problème, nous disposons de l'option *grow* qui permet de modifier l'orientation de l'arbre. Cette option peut prendre les valeurs suivantes :

- *up* pour obtenir un arbre qui va du bas vers le haut,
- *down* pour un arbre qui va du haut vers le bas,
- *left* pour un arbre qui va de la droite vers la gauche,
- *right* pour un arbre qui va de la gauche vers la droite.

Ainsi, le code ci-dessous

```
\begin{center}
\begin{tikzpicture}[grow=right, level distance=40mm, sibling distance=20mm,scale=0.7]
  \node {TikZ et LaTeX}
    child {node {Bases}
      child {node {Figures}}
      child {node {Fonctions}}}
    child {node {Diagrammes}}
    child {node {Mind Mapping}}
    child {node {GeoGebra}}
    child {node {Autres}
      child {node {Circuits électriques}}
      child {node {Formules chimiques}} }  ;
\end{tikzpicture}
\end{center}
```

donne le résultat ci-après.



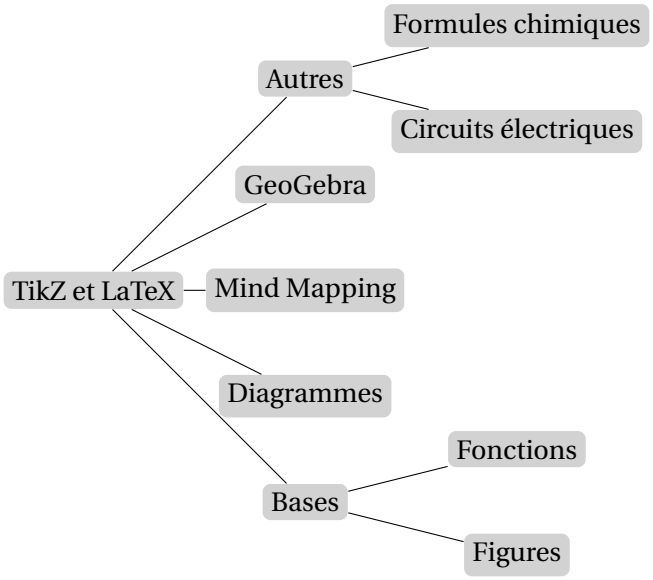
En ajoutant un style pour les noeuds avec la commande tikzstyle, il est possible de leur donner une forme et une couleur à notre convenance. Ainsi, avec

```
\tikzstyle{n}=[rounded corners=3pt,fill=gray!35, minimum width, text centered]
```

nous définissons un noeud du nom de notre choix, ici *n*, qui a la forme d'un rectangle aux coins arrondis dont le fond est de couleur grise avec un remplissage de 35 %. La largeur du noeud sera minimale pour contenir un texte centré. Il suffit alors de nommer le noeud *n* dans chaque instruction *node* comme dans le code qui suit.

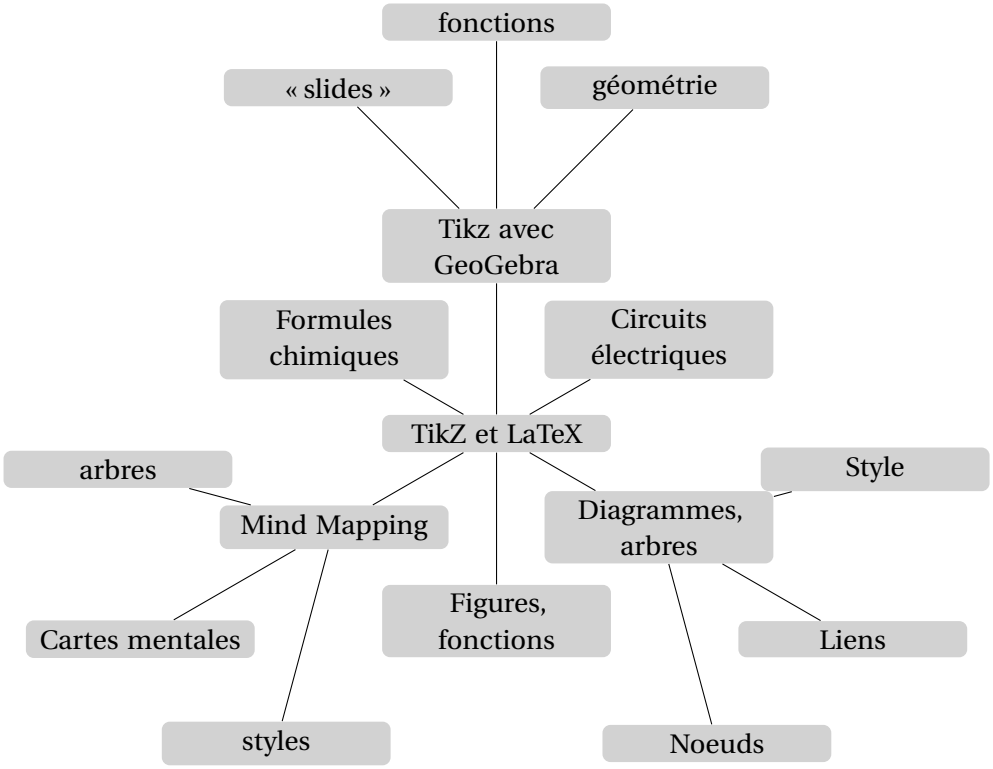
```
\begin{center}
\tikzstyle{n}=[rounded corners=3pt,fill=gray!35, minimum width, text centered]
\begin{tikzpicture}[grow=right, level distance=40mm, sibling distance=20mm,scale=0.7]
   \node[n] {TikZ et LaTeX}
      child {node[n] {Bases}
         child {node[n] {Figures}}
         child {node[n] {Fonctions}}}
      child {node[n] {Diagrammes}}
      child {node[n] {Mind Mapping}}
      child {node[n] {GeoGebra}}
      child {node[n] {Autres}
         child {node[n] {Circuits électriques}}
         child {node[n] {Formules chimiques}} }  ;
\end{tikzpicture}
\end{center}
```

Le résultat est alors :



En utilisant l'option grow cyclic, la carte mentale entourera le noeud central ce qui permet de placer davantage d'éléments. C'est le cas du code suivant dans lequel on peut voir qu'il est possible d'utiliser des valeurs différentes de level distance et de sibling angle pour des niveaux différents.

```
\begin{center}
\tikzstyle{n}=[rounded corners=3pt,fill=gray!35, minimum width, text centered]
\tikzstyle{level 1}=[level distance=3.1cm, sibling angle=60]
\tikzstyle{level 2}=[level distance=3.7cm, sibling angle=45]
\begin{tikzpicture}[scale=0.85,grow cyclic, text width=2.8cm, align=flush center]
  \node[n] {TikZ et LaTeX}
    child {node[n] {Mind Mapping}
      child {node[n] {arbres}}
      child {node[n] {Cartes mentales}}
      child {node[n] {styles}}  }
    child {node[n] {Figures, fonctions}}
    child {node[n] {Diagrammes, arbres}
      child {node[n] {Noeuds}}
      child {node[n] {Liens}}
      child {node[n] {Style}}}
    child {node[n] {Circuits électriques}}
    child {node[n] {Tikz avec GeoGebra}
      child {node[n] {géométrie}}
      child {node[n] {fonctions}}
      child {node[n] {\og slides \fg}}}
    child {node[n] {Formules chimiques}} ;
\end{tikzpicture}
\end{center}
```



En ajoutant l'option *mindmap* dans l'ouverture de l'environnement *tikzpicture*, nous pouvons utiliser le style *concept* avec `every node/.style=concept` et choisir la couleur des noeuds dans une instruction *tikzstyle* comme dans le code ci-après.

```
\tikzstyle{noeuds}=[every node/.style=concept, concept color=orange!40]
\tikzstyle{level 1}=[level distance=5cm, sibling angle=60]
\tikzstyle{level 2}=[level distance=4cm, sibling angle=45]
\begin{tikzpicture}[scale=0.8, mindmap, grow cyclic, text width=2.5cm, align=flush center, noeuds]
  \node {TikZ et LaTeX}
    child {node {Mind Mapping}
      child {node {arbres}}
      child {node {Cartes mentales}}
      child {node {styles}}  }
    child {node {Figures, fonctions}}
    child {node {Diagrammes, arbres}
      child {node {Noeuds}}
      child {node {Liens}}
      child {node {Style}}}
    child {node {Circuits électriques}}
    child {node {Tikz avec GeoGebra}
      child {node {géométrie}}
      child {node {fonctions}}
      child {node {\og slides \fg}}}
    child {node {Formules chimiques}} ;
\end{tikzpicture}
```

ce qui donne la carte mentale qui clôture ce chapitre

Chapitre 5

Utiliser GeoGebra pour produire du code TikZ

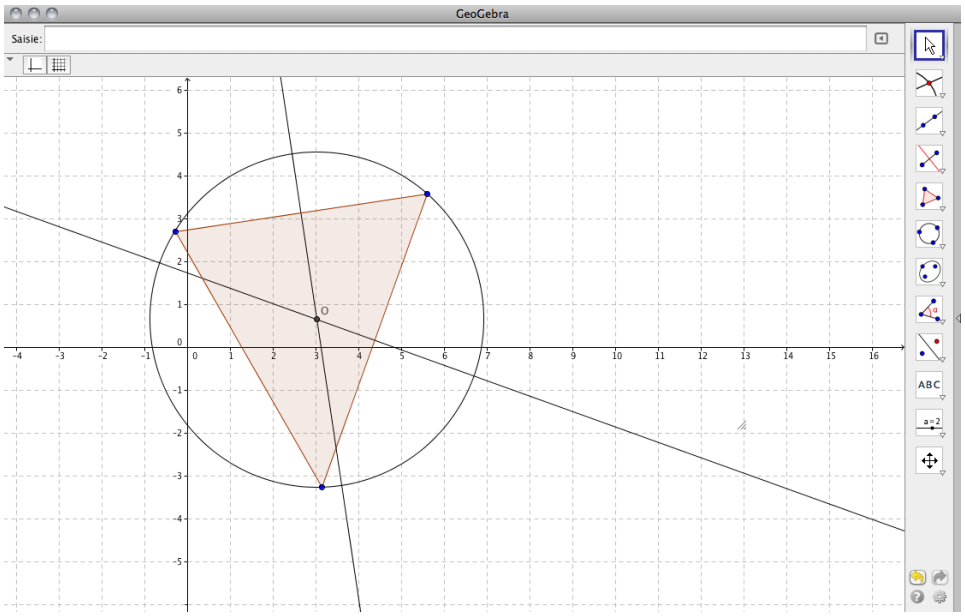
GeoGebra est un excellent outil pour créer et afficher une grande diversité de figures mathématiques. Le lecteur est, ici, supposé connaître GeoGebra et capable de l'utiliser. Si ce n'est pas le cas, il existe des tutoriels et des vidéos produits par l'auteur et que l'on trouve librement sur la toile, aux adresses suivantes :

- pour les tutoriels
  `http://www.pedagogicon.be/noeud/geogebra-2/nos-tutoriels`
- pour les vidéos
  `http://www.pedagogicon.be/noeud/geogebra-2/assistance-video`

Quant au logiciel GeoGebra, il est téléchargeable directement, quelque soit la plateforme utilisée (Linux, Mac ou Windows) à l'adresse :
`http://www.geogebra.org/cms/fr/download/`.

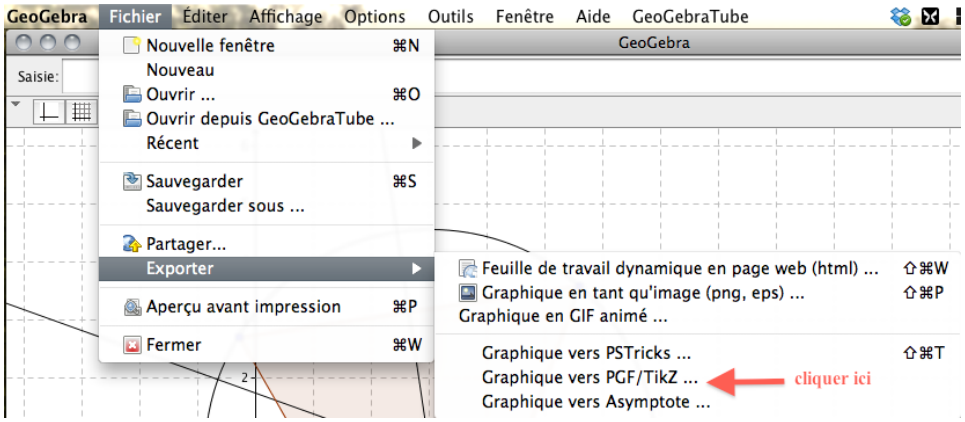
5.1 Insérer le code TikZ produit dans un document LaTeX

Commençons par créer une figure avec GeoGebra. Ici, un cercle passant par trois points qui, eux-mêmes sont les sommets d'un triangle inscrit. Nous avons tracé la médiatrice de deux des côtés de ce triangle. Ces dernières se coupent en un point O qui est le centre du cercle. Dans GeoGebra, cela donne :

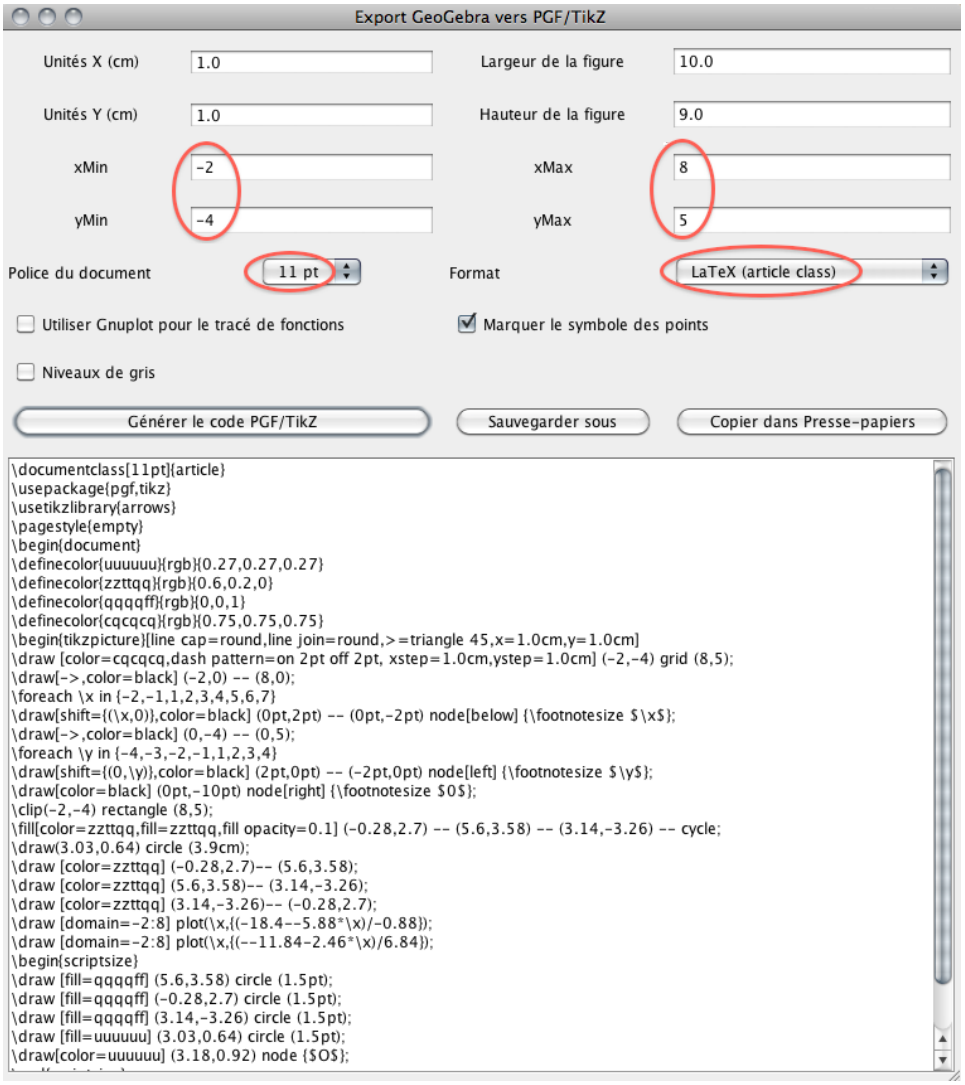


La figure que nous souhaitons intégrer à notre document sera complète si nous choisissons une fenêtre graphique telle que $x \in [-2,8]$ et $y \in [-4,5]$. Ces valeurs nous serons utiles à l'étape suivante.

Dans le menu *Fichier*, cliquer sur *Exporter* et choisir l'option *Graphique vers PGF/TikZ*...



En cliquant sur cette dernière, vous ouvrez la boîte de dialogue suivante :



dans laquelle il suffit de remplir les champs entourés. Les xMin, xMax, yMin et yMax sont les valeurs tirées de notre propre choix de fenêtre graphique. La police du document, ici 11pt, doit être identique à celle déclarée dans le préambule de votre travail `\documentclass[a4paper, 11pt,...`. Quant au format, pour intégrer la figure à un document LaTeX, on choisit l'option **LaTeX (article class)**.

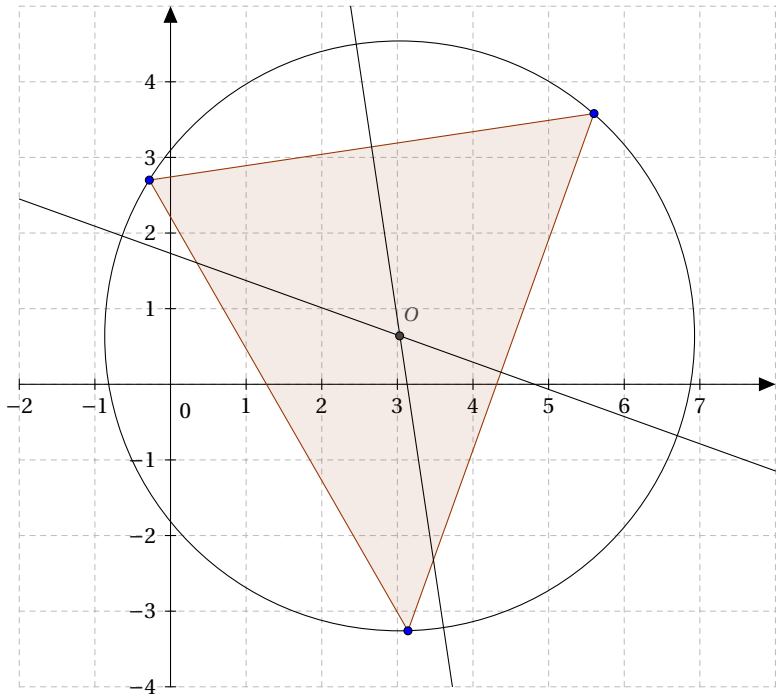
Il ne reste plus qu'à cliquer sur le bouton **Générer le code PGF/TikZ** pour obtenir, dans la fenêtre du dessous, les lignes de code produites par GeoGebra.

Il est impératif de vérifier si le préambule de votre document contient bien les lignes nécessaires au bon fonctionnement du code proposé. Dans notre exemple, votre préambule doit contenir les lignes : //

```
\usepackage{pgf, tikz}
\usetikzlibrary{arrows}
```

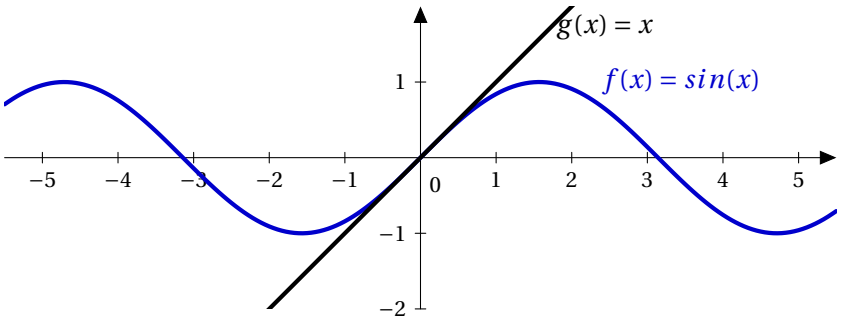
```
\pagestyle{empty}
```

Copier maintenant le code situé entre `\begin{document}` et `\end{document}`. Collez celui-ci dans votre travail à l'endroit souhaité et compilez. Vous obtenez la figure ci-après.



Il arrive que l'on ne souhaite pas conserver la totalité des éléments d'une figure, en particulier les indications de couleurs, la grille, les axes. Deux solutions s'offrent à nous. La plus simple est de cacher sous GeoGebra les éléments en question avant d'engendrer le code. L'autre, plus ardue, consiste à analyser le code proposé et de supprimer les lignes de code inutiles. En général, ces dernières sont identifiables et correctement groupées.

Comme pour les figures géométriques, l'exportation vers TikZ fournit la possibilité de représenter des fonctions.



Ainsi, le graphe ci-dessus a été obtenu à partir du code suivant engendré par GeoGebra.

```
\definecolor{qqqqcc}{rgb}{0,0,0.8}
\begin{tikzpicture}[line cap=round,line join=round,>=triangle 45,x=1.0cm,y=1.0cm]
\draw[->,color=black] (-5.5,0) -- (5.5,0);
\foreach \x in {-5,-4,-3,-2,-1,1,2,3,4,5}
\draw[shift={(\x,0)},color=black] (0pt,2pt) -- (0pt,-2pt) node[below] {\footnotesize $\x$};
\draw[->,color=black] (0,-2) -- (0,2);
\foreach \y in {-2,-1,1}
\draw[shift={(0,\y)},color=black] (2pt,0pt) -- (-2pt,0pt) node[left] {\footnotesize $\y$};
\draw[color=black] (0pt,-10pt) node[right] {\footnotesize $0$};
\clip(-5.5,-2) rectangle (5.5,2);
\draw[line width=1.6pt,color=qqqqcc,smooth,samples=100,domain=-5.5:5.5] plot(\x,{sin(((\x))*180/pi)});
\draw[line width=1.6pt,smooth,samples=100,domain=-5.5:5.5] plot(\x,{(\x)});
\draw [color=qqqqcc](2.2687420261,1.3017722376) node[anchor=north west] {$f(x) = sin(x)$};
\draw (1.6683558021,2.0193069932) node[anchor=north west] {$g(x) = x$};
\begin{scriptsize}
\draw[color=qqqqcc] (-6.2391700759,-0.0307923085) node {$f$};
\draw[color=black] (-6.2391700759,-6.3128823115) node {$g$};
\end{scriptsize}
\end{tikzpicture}
```

FIGURE 5.1 – code TikZ pour graphes de fonctions

Il peut arriver que GeoGebra produise un code qui génère un ou plusieurs messages d'erreur notamment lorsqu'il s'agit des graphes de fonctions. Ainsi, graduer l'axe des abscisses avec une échelle $\pi/2$ ou π engendre une incompréhension de la part de LaTeX. Les fonctions exponentielles posent aussi des problèmes. Dans ce cas, il est possible de corriger le code produit au vu des erreurs signalées lors de la compilation. Ce type de correction est parfois fastidieux, il est alors plus simple d'utiliser les méthodes vues au chapitre 1 ou encore d'effectuer un copier/coller et d'insérer le graphe comme une image dans le document.

5.2 Insérer le code TikZ produit dans un document Beamer

GeoGebra offre également l'opportunité de créer des slides afin de projeter les différentes étapes de la construction d'une figure. Pour ce faire, il suffit, au terme d'une construction, de choisir dans GeoGebra l'option *Exporter* suivie de *Graphique vers PGF/TikZ*...

Les réglages sont les mêmes que pour les figures ou les fonctions si ce n'est le fait que nous devons choisir comme format *LaTeX (beamer class)*.

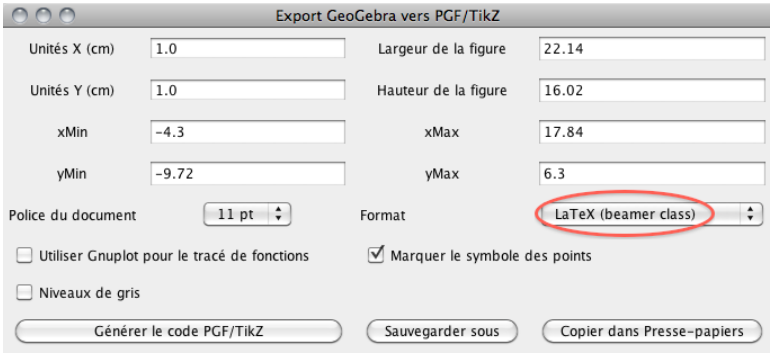
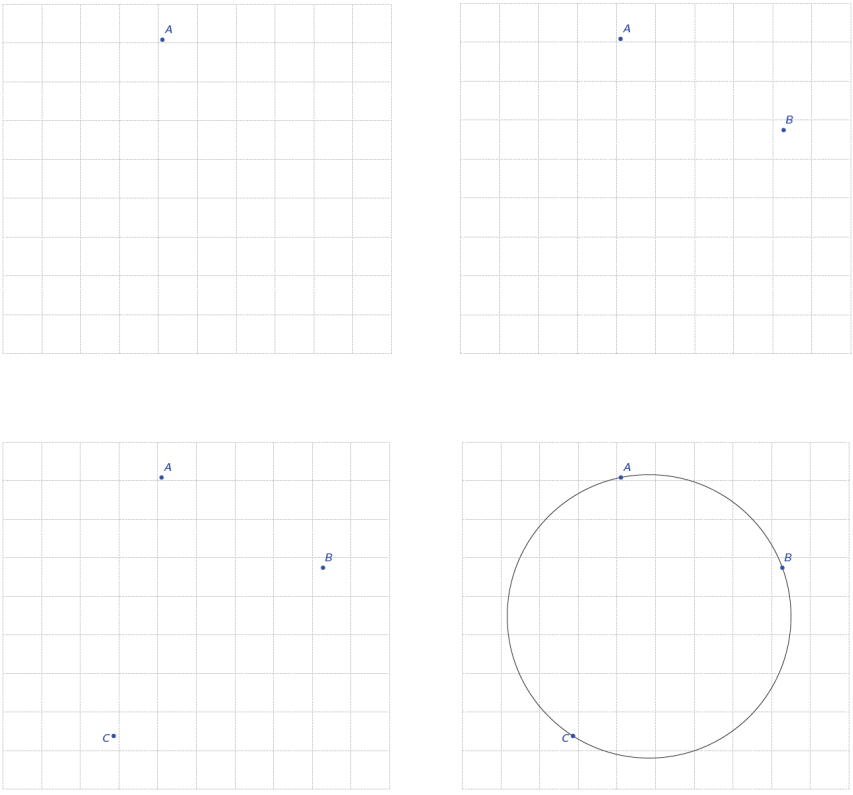
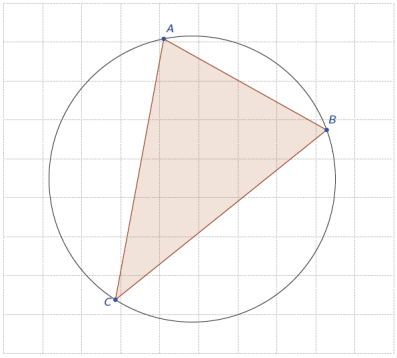


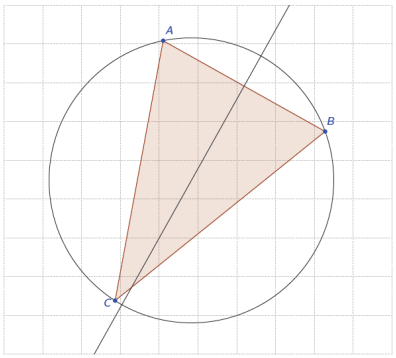
FIGURE 5.2 – exportation vers la classe Beamer de LaTeX

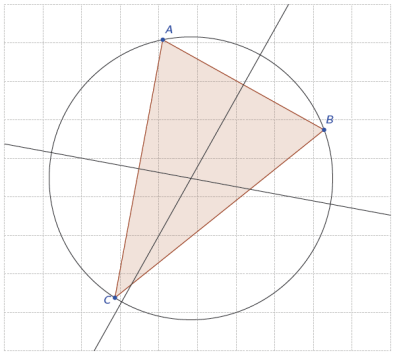
Il suffit ensuite de cliquer sur le bouton *Générer le code PGF/TikZ* puis de copier/coller le code obtenu dans un nouveau document TexMaker pour obtenir, après compilation, les slides montrant les différentes étapes de la construction de votre figure.

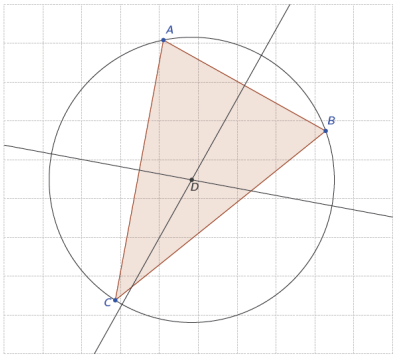
Dans le cas présent, on obtiendra la suite de slides illustrées ci-après.











Chapitre 6

Circuits électriques avec le package circuitikz

Ce chapitre a pour objectif de montrer comment dessiner des circuits électriques simples dans un document LaTeX en utilisant le package *circuitikz* qui, lui même, est basé sur *TikZ*.

Dans le préambule, on commence par introduire ce package avec

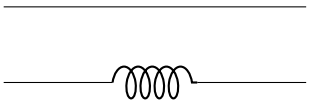
```
\usepackage{circuitikz}.
```

À l'endroit souhaité de votre document, le code s'écrit à l'intérieur d'un environnement circuitikz.

```
\begin{circuitikz}
   le code correspondant au schéma désiré
\end{circuitikz}
```

6.1 Les bases de la construction de circuits

Les différentes branches d'un circuit sont obtenues de façon semblable à celle de la construction de traits avec \draw.
Ainsi, les instructions `\draw (0, 0) -- (4, 0);`
et `\draw (0, -1) to[inductor] (4, -1);` donnent les deux éléments de figure suivant :

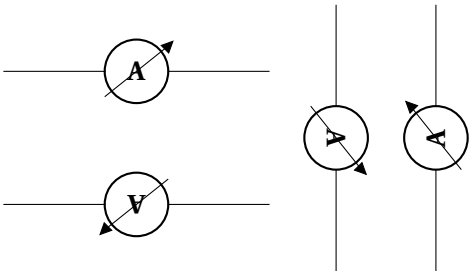


La première dessine un segment de longueur 4, tandis que la seconde utilise la commande *to[inductor]* pour placer sur un trait identique une bobine à mi-

chemin. Nous verrons ultérieurement les différentes commandes diponibles dans *circuitikz*.

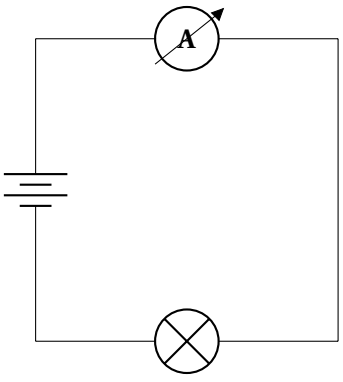
Certaines précautions d'écriture sont nécessaires. Les appareils de mesure tels que les ampèremètres et les voltmètres se comportent curieusement en fonction de la syntaxe. Pour un trait horizontal allant de la gauche vers la droite, le symbole de l'ampèremètre se dessine correctement. Lorsque le trait débute à droite pour se terminer plus à gauche, le dessin est inversé. La figure suivante et le code correspondant illustrent ce propos pour quatre traits différents.

```
\begin{circuitikz}
  \draw (0, 1) to[ammeter] (4, 1);
  \draw (4, -1) to[ammeter] (0, -1);
  \draw (6, 2) to[ammeter] (6, -2);
  \draw (8, -2) to[ammeter] (8, 2);
\end{circuitikz}
```



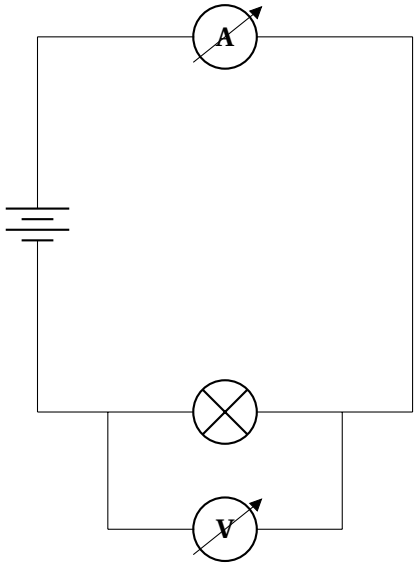
Élaborons un premier circuit à partir de ces idées. Rappelons qu'une instruction \draw se termine par un ;. Le code ci-dessous utilise une seule instruction pour dessiner le circuit qui lui fait face. Le chemin commence au point de coordonnées (0, 0) et va vers le point (0, 4) en plaçant une alimentation continue au milieu du trait. De ce point (0, 4), on va vers (4, 4) en mettant un ampèremètre à mi-chemin. On repart ensuite vers (4, 0) par un simple trait et on revient ensuite en (0, 0) en positionnant une lampe au centre du dernier segment.

```
\begin{circuitikz}
  \draw (0, 0) to[battery] (0, 4)
               to[ammeter] (4, 4) -- (4, 0)
               to[lamp] (0, 0);
\end{circuitikz}
```



Ajoutons un voltmètre à notre premier circuit. Nous avons simplement complété l'instruction *draw* avec **(0.75, 0) – (0.75, -1.25) to[voltmeter] (3.25, -1.25) – (3.25, 0)**. Ce chemin supplémentaire dessine un voltmètre à l'endroit souhaité.

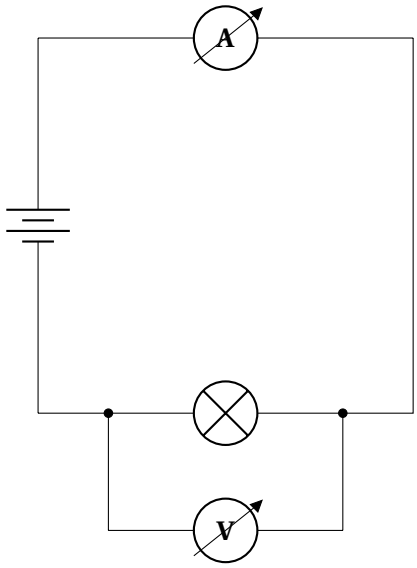
```
\begin{circuitikz}
  \draw (0, 0) to[battery] (0, 4)
               to[ammeter] (4, 4) -- (4, 0)
               to[lamp] (0, 0)
        (0.75, 0) -- (0.75, -1.25) to[voltmeter] (3.25, -1.25) -- (3.25, 0);
\end{circuitikz}
```



Bien souvent, nous cherchons à mettre en évidence les noeuds du circuit (les points ou, dans notre exemple, la branche en parallèle est attachée au circuit principal. Le code précédent est alors enrichi de la façon suivante :

```
\begin{circuitikz}
  \draw (0, 0) to[battery] (0, 4)
               to[ammeter] (4, 4) -- (4, 0) -- (3.25, 0)
               to[lamp, *-*] (0.75, 0) -- (0, 0)
        (0.75, 0) -- (0.75, -1.25) to[voltmeter] (3.25, -1.25) -- (3.25, 0);
\end{circuitikz}
```

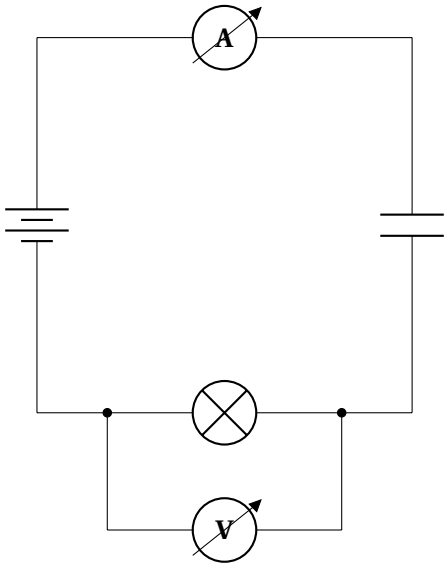
Nous avons ajouté *-* dans le crochet contenant *lamp* (le composant sur lequel est branché le voltmètre). Autour de *to[lamp]* nous devons introduire les coordonnées des noeuds, (3.25, 0) avant et (0.75, 0) juste après avant de fermer le circuit (dans notre exemple en (0, 0)). La ligne dessinant le voltmètre est inchangée. Nous obtenons de la sorte :



Nous pouvons ajouter un condensateur entre l'ampèremètre et la lampe en ajoutant to[C] sur le trait descendant. Ainsi, le code suivant :

```
\begin{circuitikz}[scale=1.24]
  \draw (0, 0) to[battery] (0, 4)
                to[ammeter] (4, 4)
                to[C] (4, 0) -- (3.25, 0)
                to[lamp, *-*] (0.75, 0) -- (0, 0)
        (0.75, 0) -- (0.75, -1.25) to[voltmeter] (3.25, -1.25) -- (3.25, 0);
\end{circuitikz}
```

permet d'obtenir :

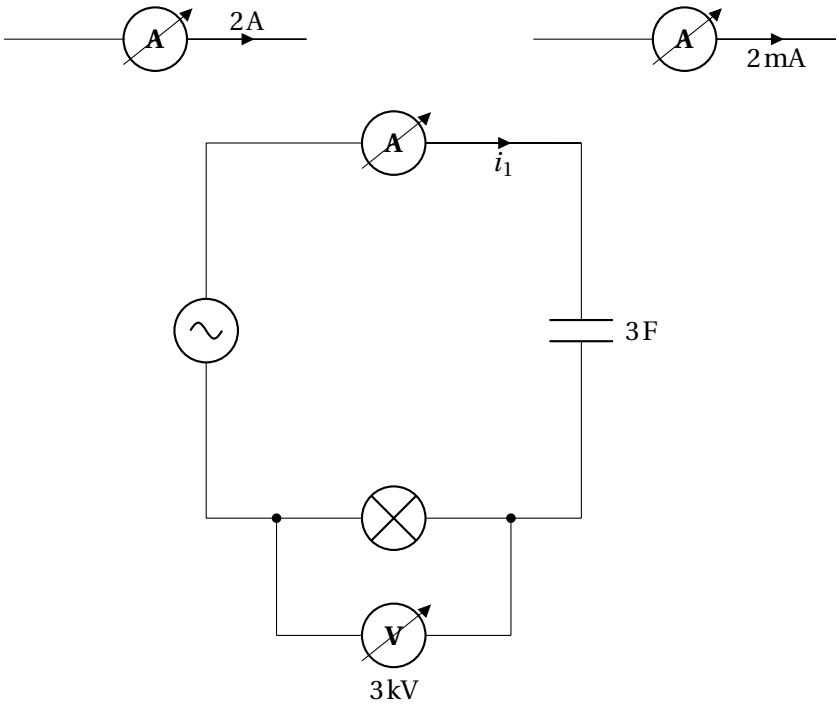


6.2 Annoter le circuit

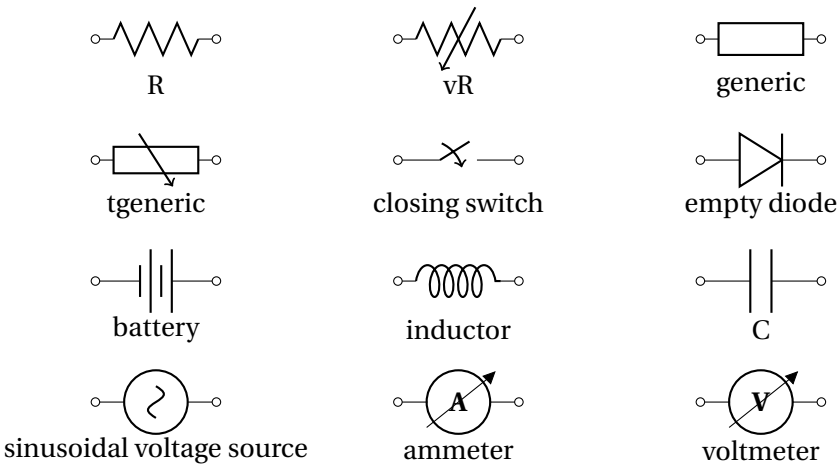
Pour placer des annotations avec des unités électriques correctes nous devons mettre l'option *siunitx* en déclarant le package *circuitikz*. Dans le préambule de notre document, nous remplaçons `\usepackage{circuitikz}` par `\usepackage[siunitx]{circuitikz}`.

Nous pouvons alors annoter l'ampèremètre en écrivant `to[ammeter, l=2<\ampere>]` ou `to[ammeter, l_=2<\milli\ampere>]`, (la lettre l pour label). Nous attirons l'attention sur le underscore qui suit le second l, ce dernier a pour effet de placer l'annotation sous l'ampèremètre plutôt qu'au dessus. Nous avons ainsi :

Nous pouvons préférer placer l'indication de courant avec une flèche sur le trait sortant de l'ampèremètre, il suffit pour cela de remplacer le l (label) par un i. Ainsi, les instructions `to[ammeter, i=2<\ampere>]` ou `to[ammeter, i_=2<\milli\ampere>]` donnent :

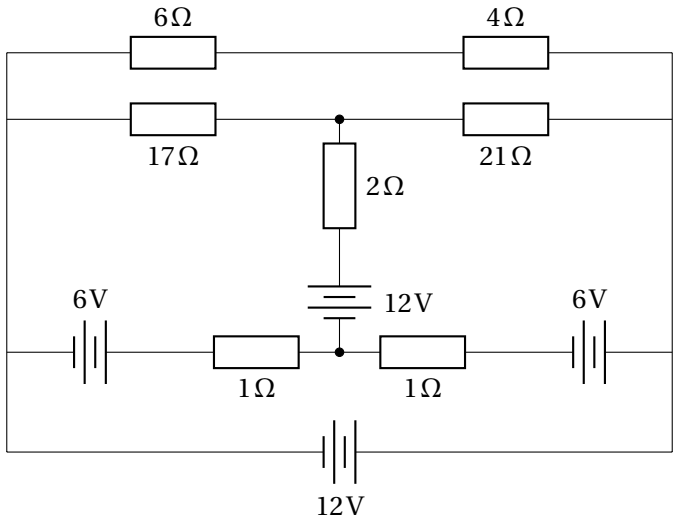


L'image qui suit montre douze des composants les plus couramment utilisés avec, en dessous de chacun d'eux, le nom qui permet de l'appeler à l'intérieur de l'environnement circuitikz.



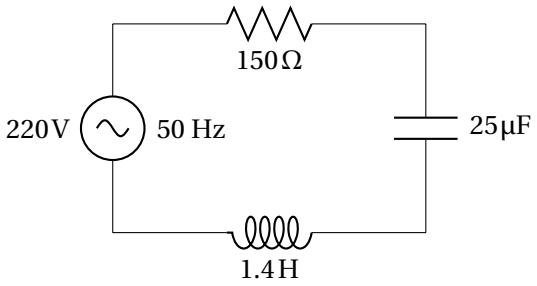
Pour les trois derniers schémas de cette section, nous donnons simplement le code permettant de les obtenir, les connaissances acquises jusqu'ici suffisent à la compréhension de ces constructions. Ces dernières sont de simples illustrations de ce qu'il est possible de représenter.

6.2.1 Exemple 1



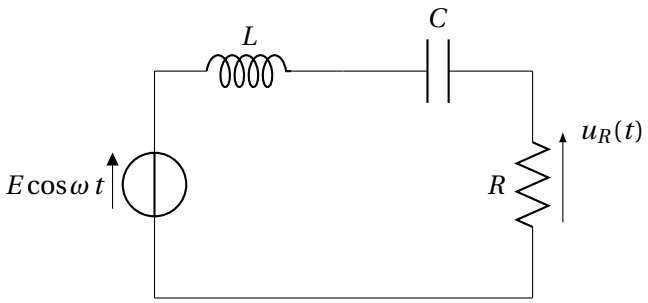

```
\begin{circuitikz}[scale=0.88]
  \draw (0, 0) to[battery, l_=12<\volt>] (10, 0) -- (10, 6) to[generic, l_=4<\ohm>] (5, 6) to[generic, l_=6<\ohm>]
(0, 6) -- (0, 0);
  \draw (0, 1.5) to[battery, l=6<\volt>] (2.5, 1.5) to[generic, l_=1<\ohm>] (5, 1.5) to[generic, l_=1<\ohm>] (7.5,
1.5) to[battery, l=6<\volt>] (10, 1.5);
  \draw (0, 5) to[generic, l_=17<\ohm>] (5, 5) to[generic, l_=21<\ohm>] (10, 5);
  \draw (5, 1.5) to[battery, l_=12<\volt>,*-] (5, 3) to[generic, l_=2<\ohm>,-*] (5, 5);
\end{circuitikz}
```

6.2.2 Exemple 2




```
\begin{circuitikz}[scale=0.69]
  \draw (0, 0) to[sinusoidal voltage source, l=220<\volt>] (0, 4)
               to[R, l_=150<\ohm>] (6, 4)
               to[C, l=25<\micro\farad>] (6, 0)
               to[inductor, l=1.4<\henry>] (0, 0);
  \node (f) at (1.5,2) {50 Hz};
\end{circuitikz}
```

6.2.3 Exemple 3




```
\begin{circuitikz}
\draw (0,0) to[V, v=$E\cos \omega \,t$] (0,3) to [L, l=$L$] (2.5,3);
\draw (2.5,3) to [C, l=$C$] (5,3);
\draw (5,3) to [R, l_=$R$] (5,0);
\draw (5,0) -- (0,0);
\draw[->,>=latex] (5.4,1) -- (5.4,2.2) node[below=0.7cm,right=0.1cm] {$u_R(t)$};
\end{circuitikz}
```

6.3 Les composants avec plus de deux terminaisons

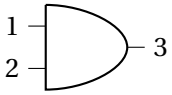
Les composants qui présentent trois terminaisons ou davantage, comme les transistors, les portes logiques, les transformateurs, doivent impérativement être placés dans des noeuds au sens de TikZ.

En vue de permettre des connections avec d'autres composants, ces composants définissent des ancres.

Ainsi une porte logique *and* sera donnée par le code suivant :

```
\begin{circuitikz}
   \draw (0,0) node[and port] (maporteand) {}
     (maporteand.in 1) node[anchor=east] {1}
     (maporteand.in 2) node[anchor=east] {2}
     (maporteand.out) node[anchor=west] {3}
   ;
\end{circuitikz}
```

Nous obtenons de la sorte le résultat ci-dessous.



Quelques mots d'explications sont, sans doute, les bienvenus. La première ligne de code commence par une instruction *draw* qui, dans un noeud situé en (0, 0), dessine la porte logique nommée *and port* par l'environnement *circuitikz*. Le contenu de la parenthèse est un nom de votre choix pour cette porte logique. Ici, nous lui avons donné le nom de *maporteand* mais c'est uniquement un choix.

Une porte logique possède deux entrées et une sortie respectivement désignées, ici, par *in 1*, *in 2* et *out*. Les trois lignes de code qui complètent la première définissent les ancres associées à ces trois terminaisons. Comme pour tous les autres noeuds, on place entre accolades ce que l'on souhaite voir apparaître sur la figure. Dans le cas présent, il s'agit des nombres 1, 2 et 3.

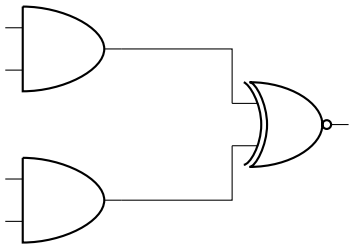
Le nouveau code, ci-après, montre la manière de relier ces composants à d'autres éléments du circuit.

```
\begin{circuitikz}
   \draw
     (0,2) node[and port] (andgate1) {}
     (0,0) node[and port] (andgate2) {}
     (3,1) node[xnor port] (xnorgate) {}
       (andgate1.out) -| (xnorgate.in 1)
       (andgate2.out) -| (xnorgate.in 2)
   ;
\end{circuitikz}
```

Les trois premières lignes de code, situées à l'intérieur de l'instruction *draw*, placent trois portes logiques en trois endroits différents définis par leurs coordonnées. Les deux premières sont des portes *and* obtenues comme précédemment avec *and port*, tandis que la troisième est une porte *xnor* obtenue avec *xnor port*. Les trois portes ont reçu un nom choisi par l'utilisateur, c'est le contenu des parenthèses.

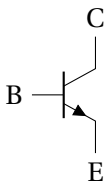
La sortie de la première porte *and* (andgate1.out) est reliée avec les traits -| à la première entrée de la porte *xnor* (xnorgate.in 1). De la même façon, la sortie

de la deuxième porte *and* (andgate2.out) est reliée avec la seconde entrée de la porte *xnor* (xnorgate.in 2). Nous obtenons ainsi le résultat ci-dessous.



La manière d'obtenir un transistor, par exemple *npn*, est semblable. Ainsi, le code suivant, à gauche, donne le résultat ci-dessous, à droite.

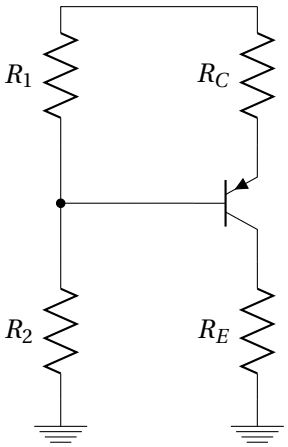
```
\begin{circuitikz}
   \draw
     (0,0) node[npn] (npn) {}
        (npn.base) node[anchor=east] {B}
        (npn.collector) node[anchor=south] {C}
        (npn.emitter) node[anchor=north] {E}
   ;
\end{circuitikz}
```



Sur base de l'exemple ci-dessous, regardons comment intégrer un transistor dans un circuit. Ainsi, le code ci-dessous :

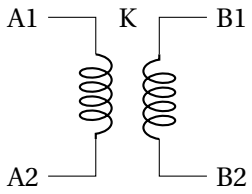
```
\begin{circuitikz}[scale=1.4]
   \draw (0,0) node[ground] {};
   \draw (0,0) to[R=$R_2$] (0,1.4) -- (0,2);
   \draw (0,2) -- (0,2.6) to[R=$R_1$] (0,4) -- (2,4);
   \draw (2,2) node[pnp] (pnp) {}
      (pnp.B) [short,-*] to (0,2); % "short" représente un fil de connexion
   \draw (2,4) to[R,l_=$R_C$] (2,2.6) -- (pnp.E); % l_ modifie la position de l'étiquette
   \draw (2,0) node[ground] {};
   \draw (2,0) to[R=$R_E$] (2,1.4) -- (pnp.C);
\end{circuitikz}
```

donnera la figure suivante.

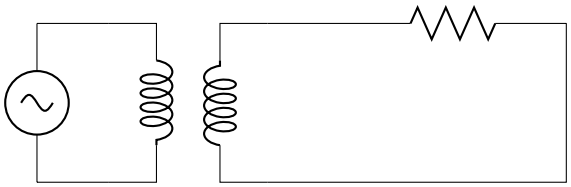


Traitons encore du cas du transformateur en donnant le code, le schéma obtenu et, pour terminer, un exemple simple d'utilisation dans un circuit.

```
\begin{circuitikz}
  \draw (0,0) node[transformer] (T) {}
    (T.A1) node[anchor=east] {A1}
    (T.A2) node[anchor=east] {A2}
    (T.B1) node[anchor=west] {B1}
    (T.B2) node[anchor=west] {B2}
    (T.base) node{K};
\end{circuitikz}
```



Pour relier le transformateur aux circuits primaire et secondaire, nous prenons l'exemple du circuit suivant



dont le code est donné ci-après.

```
\begin{circuitikz}
  \draw (0,0) node[transformer] (T) {}
    (T.A1) -- (-2,0) to[sinusoidal voltage source] (-2,-2.1) -- (T.A2)
    (T.B1) -- (2,0) to[R] (5,0) -- (5,-2.1) -- (T.B2);
\end{circuitikz}
```

Chapitre 7

Formules chimiques avec le package chemfig

L'écriture des formules chimiques nécessite le placement d'un nouveau package, à la suite de *pgf,tikz*, dans le préambule avec

```
\usepackage{chemfig}
```

Chemfig a été élaboré à partir de *TikZ* mais, à la différence de celui-ci, il n'y a pas lieu d'introduire un environnement comme *tikzpicture*. La commande principale permettant de dessiner les molécules est simplement

```
\chemfig{<code>}
```

Ainsi, pour représenter la molécule d'eau, on peut écrire `\chemfig{H-O-H}` pour obtenir : H—O—H.

Chemfig offre la possibilité de dessiner un grand nombre de configurations de molécules chimiques à partir d'une syntaxe simple, souple et intuitive. Il est cependant évident que le <code> qui engendre le dessin de la molécule voit sa complexité croître proportionnellement à celui de la molécule elle-même.

La commande chemfig obéit à la syntaxe suivante :

```
\chemfig{<atome-1><type de liaison>[arguments optionnels]<atome-2>...}
```

Les arguments optionnels sont toujours écrits entre crochets et ils sont séparés par une virgule.

- Le premier indique l'angle de liaison entre 2 atomes,
- le second est un nombre permettant de régler la longueur de la liaison,
- le troisième et le quatrième sont respectivement les numéros de l'atome de départ et de l'atome d'arrivée de la liaison,
- le cinquième est une indication de couleur ou de forme de la liaison.

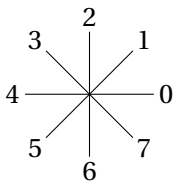
Nous verrons progressivement comment honorer les différents points de cette syntaxe.

7.1 Les différents types de liaisons

code	résultat	type de liaison
`\chemfig{C-H}`	C—H	simple
`\chemfig{C=H}`	C=H	double
`\chemfig{C~H}`	C≡H	triple
`\chemfig{C>H}`	C▶H	cram pleine droite
`\chemfig{C<H}`	C◀H	cram pleine gauche
`\chemfig{C>:H}`	C⁞H	cram pointillé droite
`\chemfig{C<:H}`	C⁞H	cram pointillé gauche
`\chemfig{C>|H}`	C▷H	cram évidée droite
`\chemfig{C<|H}`	C◁H	cram évidée gauche

7.2 Angles de liaisons prédéfinis

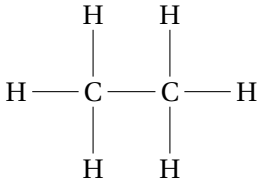
Le premier des arguments optionnels indique l'angle de la liaison. Intéressons-nous, en premier lieu, aux angles prédéfinis représentés par un nombre entier de 0 à 7, noté [chiffre], qui indique l'angle que fait la liaison avec l'horizontale, en multiples de 45°selon



Cette notation complétée par une règle selon laquelle *les parenthèses permettent d'autoriser plusieurs liaisons à partir d'un même atome*[1] suffit pour écrire les formules développées et semi-développées d'un grand nombre de molécules. Ainsi, la molécule d'éthane C_2H_6 s'écrira :

```
\chemfig{C(-[2]H)(-[4]H)(-[6]H)-C(-[2]H)(-[6]H)(-[0]H)}
```

Le premier atome de carbone est suivi de 3 parenthèses ce qui indique 3 liaisons avec, ici, des atomes d'hydrogène. Le premier atome d'hydrogène est en position 2, le suivant en 4 et le troisième en 6. Ensuite, un simple trait lie les deux atomes de carbone ensemble. Le second carbone est lui aussi suivi de 3 parenthèses. Ce code engendre alors le dessin suivant :



[1] Les parenthèses permettent que le dernier atome écrit ne soit pas considéré comme origine pour la liaison suivante.

Quant aux formules semi-développées, elles sont plus faciles à écrire. La molécule d'hex-1-ène, par exemple, s'écrit :

```
\chemfig{H_3C-{{(CH_2)}_3}-CH=CH_2}
```

$$H_3C-(CH_2)_3-CH=CH_2$$

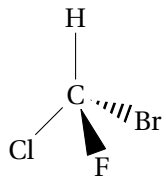
Il est parfois utile de pouvoir modifier la longueur d'une liaison, c'est le rôle du second argument optionnel. Ainsi, dans l'exemple précédent, la longueur de la double liaison est manifestement trop grande par rapport aux liaisons simples.

```
\chemfig{H_3C-{{(CH_2)}_3}-CH=[,0.7]CH_2}
```

La virgule, à l'intérieur du crochet, signale à chemfig que l'argument est placé en deuxième position. Les accolades

$$H_3C-(CH_2)_3-CH=CH_2$$

Ainsi, la molécule chirale $CHFClBr$ s'écrira :



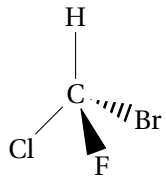
7.3 Représentation de Cram, angles de liaisons arbitraires

L'argument donnant l'angle de liaison peut ne pas être prédéfini. Il est alors exprimé en degré selon les règles habituelles des mathématiques. Il suffit de le préciser en plaçant : devant la valeur de l'angle à l'intérieur des crochets, *[:angle]*.

Ainsi, la molécule chirale $CHFClBr$ s'écrira :

```
\chemfig{C(-[2]H)(-[5]Cl)(<[:-70]F)(<:[:-20]Br)}
```

dans laquelle l'angle de liaison entre le carbone et le fluor a pour valeur -70°alors que le lien entre le carbone et le brome fait un angle de -20°.

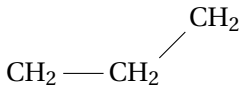


7.4 Angles de liaisons relatifs

La syntaxe de ceux-ci est [: :angle], l'angle étant relatif à la direction de la dernière liaison tracée. Ainsi l'écriture de

```
\chemfig{CH_2-CH_2-[::45]CH_2}
```

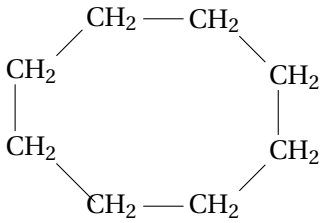
a pour résultat la figure ci-dessous.

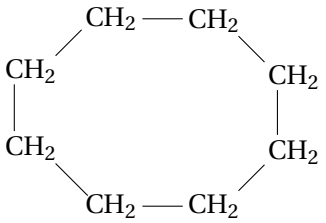


Ces angles sont essentiellement utilisés pour construire certaines molécules cycliques. Par exemple, le cyclooctane qui se construit comme suit :

```
\chemfig{CH_2-CH_2-[::45]CH_2-[::45]CH_2-[::45]CH_2
-[::45]CH_2-[::45]CH_2-[::45]CH_2-[::45]\mbox{ }}
```

La boîte vide `\mbox{ }` est utilisée pour raccourcir la dernière liaison pour éviter que celle-ci empiète sur le premier carbone. Pour visualiser l'utilité de cette `\mbox{ }`, le dessin de cette molécule apparaît ci-dessous, à gauche, sans la boîte vide et, à droite, avec elle.





7.5 Les cycles

ChemFig dessine facilement des polygones réguliers avec la syntaxe

```
\chemfig{<atome>*<n>(<code>)
```

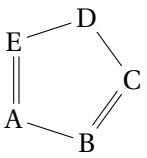
dans laquelle *n* est le nombre de côtés du polygone, tandis que le *code* représente les liaisons et les groupes d'atomes qui constituent les arêtes et les sommets. Ce code doit commencer par une liaison puisque l'atome de départ se trouve, dans l'instruction, à l'extérieur du cycle.

L'atome initial « A » se trouve toujours au sud ouest du cycle. Ce dernier est toujours construit dans le sens trigonométrique et la dernière liaison tombe verticalement sur l'atome de départ.

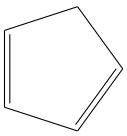
Donnons, par exemple, le code et le dessin d'un 5-cycle attaché à un atome A.

Il est ainsi aisé de dessiner les formules topologiques d'un grand nombre de molécules cycliques. Ainsi, pour le pentadiène

```
\chemfig{A*5(-B=C-D-E=)}
```

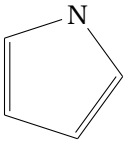



```
\chemfig{*5(-=--=)}
```



ou encore, pour la molécule de pyrolle de formule brute C_4H_5N

```
\chemfig{*5(-=-N-=)}
```

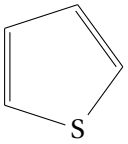


Les molécules de thiophène et de benzène montrent la possibilité d'écrire la molécule sous forme cyclique ou sous forme de cycle aromatique. Pour cette dernière écriture, il suffit de remplacer l'* par une double ** et les traits des doubles liaisons par de simples traits.

Pour le thiophène,

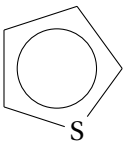
Cycle

```
\chemfig{*5(-S-=-=)}
```



Cycle aromatique

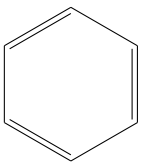
```
\chemfig{**5(-S----)}
```



Pour le benzène,

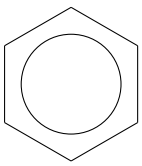
Cycle

```
\chemfig{*6(=-=-=-)}
```



Cycle aromatique

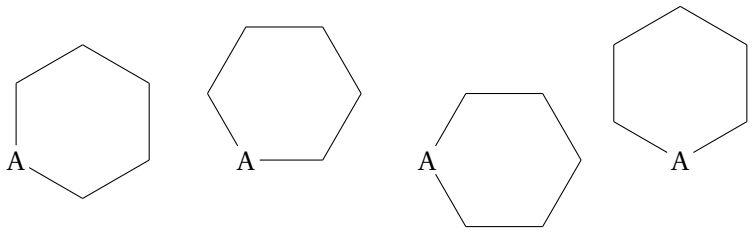
```
\chemfig{**6(------)}
```



Si la position angulaire par défaut ne convient pas, il est possible de la modifier en plaçant un argument optionnel au début de l'écriture de la molécule.

Voici un 6-cycle dessiné par défaut suivi par trois autres qui ont pivoté respectivement de +30°, de -30° puis de +60° :

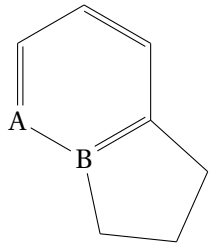
```
\chemfig{A*6(------)} \qquad \chemfig{[:30]A*6(------)} \qquad \chemfig{[:-30]A*6(--
```



7.6 Les cycles imbriqués

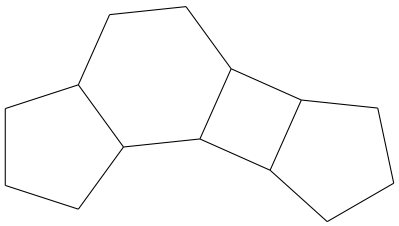
Pour coller deux cycles l'un à l'autre, nous devons identifier le sommet où commence le second cycle. Il suffit, alors, de faire suivre ce sommet par la syntaxe habituelle d'un cycle. Ainsi, un 5-cycle partant du second sommet occupé par l'atome B d'un 6-cycle débutant avec l'atome A s'écrira :

```
\chemfig{A*6(-B*5(----)=-=-=)}
```



Il est évidemment possible de coller plusieurs cycles entre-eux.

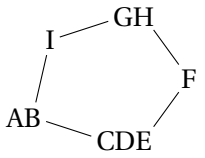
```
\chemfig{*5(--*6(-*4(-*5(----)--)----)---)}
```



Groupements d'atomes dans un cycle

Certaines précautions sont à prendre avec les cycles lorsque un ou plusieurs sommets sont constitués de plusieurs atomes. Par exemple,

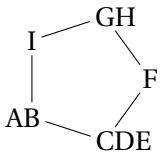
```
\chemfig{AB*5(-CDE-F-GH-I-)}
```



donne un 5-cycle déformé ce qui n'est pas ce que l'on souhaite.

Pour obtenir un cycle de forme régulière, il faut contourner le mécanisme de chemfig qui calcule automatiquement les atomes de départ et d'arrivée des diverses liaisons. Dans notre exemple, nous voulons relier les atomes C et F ainsi que F et G. Pour ce faire, il est bon de rappeler que, entre crochets, les arguments optionnels sont séparés par des virgules. Ceux placés respectivement en 3^e et 4^e positions (après 2 ou 3 virgules) sont les numéros de l'atome de départ et de l'atome d'arrivée de la liaison.

```
\chemfig{AB*5(-CDE-[,,1]F-[,,,1]GH-I-)}
```



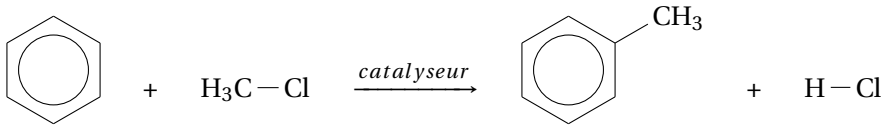
Avec cette écriture, nous aurions pu indiquer comme instruction

```
\chemfig{AB*5(-CDE-[,,1,1]F-[,,1,1]GH-I-)}
```

Celle-ci est correcte et donne le bon résultat. Cependant, l'atome F étant seul sur son sommet, il n'est pas nécessaire d'indiquer dans le premier crochet qu'il est l'atome d'arrivée (le 1 en 4^e position est superflu) et, de même, il va de soi qu'il est l'atome de départ dans le second crochet (le 1 en 3^e position n'est pas nécessaire).

7.7 Écriture des réactions chimiques

Soit à écrire la réaction chimique ci-dessous.



Le code pour y parvenir est celui-ci :

```
\chemfig{**6(------)} \hspace{0.4cm} + \hspace{0.4cm} \chemfig{H_3C-Cl}
\hspace{0.4cm} $\xrightarrow{catalyseur}$ \hspace{0.4cm}
\chemfig{**6(---(-CH_3)---)} \hspace{0.4cm} + \hspace{0.4cm} \chemfig{H-Cl}
```

Seule l'instruction `$\xrightarrow{catalyseur}$` est réellement nouvelle. Elle dessine une flèche orientée vers la droite avec la possibilité de placer une indication au-dessus et une autre en dessous.
La syntaxe étant `$\xrightarrow[en-dessous]{au dessus}$`. Dans notre exemple, nous avons déjà utilisé cette option pour marquer la présence d'un catalyseur pour cette réaction au-desssus de la flèche.

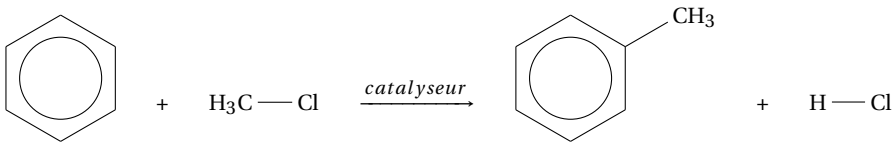
Les commandes `\hspace{0.4cm}$` permettent d'imposer un espace horizontal de largeur réglable aux endroits souhaités.

Nous avons, jusqu'ici, passé sous silence le principal problème, la taille de la figure produite par *chemfig*. Essayez le code proposé et vous verrez que LaTeX est obligé de placer la figure sur deux lignes car elle déborde largement de la largeur totale impartie à notre texte. Ce n'est évidemment pas le but souhaité et nous voulons corriger cela.

Réduction de la taille d'une figure chemfig

Une première solution consiste à faire précéder le code précédent de l'instruction `\setatomsep{2.1em}` qui gère la taille de l'ensemble de la figure[2]. C'est la méthode utilisée dans la construction de la figure précédente.

Une deuxième solution consiste à placer une instruction `\footnotesize` juste avant le code de la figure et d'ajouter l'instruction `\normalsize` à la fin de celui-ci. Avec cette méthode et le même code, nous obtenons :



Une syntaxe différente pour les réactions

Il y a également la possibilité d'utiliser une autre syntaxe pour écrire la réaction. Cette dernière est, au dire de certains, plus parlante que l'autre. Pour la même équation chimique, on écrit :

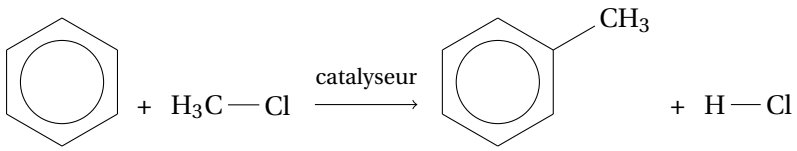
```
\setatomsep{2.5em}
\chemfig{**6(------)} \chemsign{+} \chemfig{H_3C-Cl}
\chemrel[\footnotesize catalyseur]{->}
\chemfig{**6(---(-CH_3)---)} \chemsign{+} \chemfig{H-Cl}
```

On y trouve deux nouvelles instructions `\chemsign{+}` qui ne nécessitent plus de contrôler les espacements et une nouvelle façon de dessiner la flèche `\chemrel{->}` qui permet aussi de mettre une indication au-dessus et en dessous avec comme écriture

```
\chemrel[au dessus][en-dessous]{->}
```

ce qui donne :

[2] L'unité em se rapporte à la taille de la police. Avec elle on peut affecter une mesure relative à la taille de police de l'élément parent. Elle permet d'avoir des feuilles de style plus facilement adaptables d'un média à un autre. Les nombres décimaux sont autorisés, mais il faut tout simplement remplacer la virgule par un point. Cette valeur em est utilisable pour d'autres propriétés acceptant la mention de longueur.

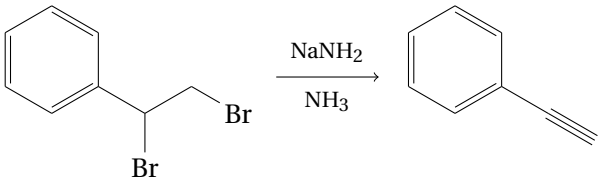


Ces nouvelles instructions permettent, dans le cas de l'équation utilisée dans nos exemples de ne pas avoir à gérer la taille de celle-ci. Elles ne résolvent cependant pas toujours le problème de la taille de l'équation obtenue, il est alors nécessaire de recourir aux deux méthodes ci-dessus.

Quelle que soit la syntaxe utilisée, il est toujours possible de placer des indications « chimiques » sur et sous la flèche grâce à *chemfig*. Ainsi, pour la synthèse du phényléthyne, nous écrivons

```
\setatomsep{2.1em}
\chemfig{*6(-=*6(-(-Br)-(-Br))-=-=)}
\chemrel[\footnotesize \chemfig{NaNH_2}][\footnotesize \chemfig{NH_3}]{->}
\chemfig{*6(-=(-~)-=-=)}
```

Ce code donne :



Nous serions parvenus au même résultat avec l'autre syntaxe en écrivant :

```
\setatomsep{2.1em}
\chemfig{*6(-=*6(-(-Br)-(-Br))-=-=)}
\hspace{0.4cm} $\xrightarrow[\footnotesize \chemfig{NH_3}]{\footnotesize
\chemfig{NaNH_2}}$ \hspace{0.4cm}
\chemfig{*6(-=(-~)-=-=)}
```

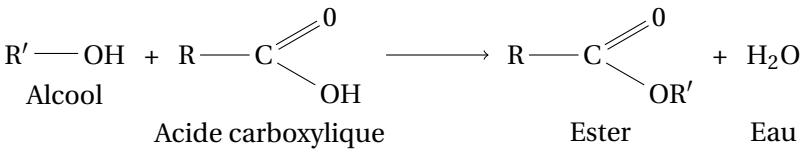
Signalons, à toutes fins utiles, l'existence de différentes flèches de liaisons

code	résultat
`\chemrel{->}`	
`\chemrel{<-}`	
`\chemrel{<->}`	
`\chemrel{<>}`	

Écrire un nom sous une molécule

Il existe une commande *chemname* qui permet d'écrire le nom d'une molécule juste sous sa formule.

Sa syntaxe est `\chemname{\chemfig{code de la molécule}}{Nom}` la distance entre la formule et son nom étant gérée, par défaut, par la commande. Le nom sera centré par rapport à la formule. La réaction ci-dessous



a, ainsi, été obtenue à partir du code suivant :

```
\chemname{\chemfig{R'-OH}}{Alcool}
\chemsign{+}
\chemname{\chemfig{R-C(-[:-30]OH)=[:30]0}}{Acide carboxylique}
\chemrel{->}
\chemname{\chemfig{R-C(-[:-30]OR')=[:30]0}}{Ester}
\chemsign{+}
\chemname{\chemfig{H_2O}}{Eau}
```

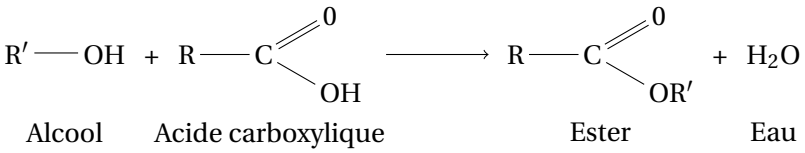
On constate, de suite, le problème de l'alignement des noms qui enlève une part de l'esthétique que nous serions en droit d'attendre. Le problème vient du fait que la seconde molécule, celle de l'acide, présente un dessin qui descend plus bas que celui de la molécule d'alcool. La molécule d'acide est designée, dans la suite, par l'expression *molécule la plus profonde*. En inversant les deux premières molécules dans le code, on résoud le problème. Il est aussi possible de le résoudre avec la commande *chemnameinit* que l'on utilise comme suit :

- on écrit, à la première ligne du code `\chemnameinit{\chemfig{code de la molécule la plus profonde}}`,
- on termine le code en plaçant à la dernière ligne `\chemnameinit{}` afin d'éviter que la plus grande profondeur trouvée dans cette réaction n'interfère dans une réaction future.

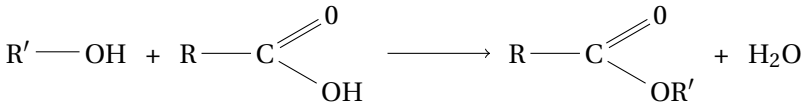
Notre nouveau code est ainsi

```
\chemnameinit{\chemfig{R-C(-[:-30]OH)=[:30]0}}
\chemname{\chemfig{R'-OH}}{Alcool}
\chemsign{+}
\chemname{\chemfig{R-C(-[:-30]OH)=[:30]0}}{Acide carboxylique}
\chemrel{->}
\chemname{\chemfig{R-C(-[:-30]OR')=[:30]0}}{Ester}
\chemsign{+}
\chemname{\chemfig{H_2O}}{Eau}
\chemnameinit{}
```

ce qui donne :



Pour terminer cette section, nous indiquons la possibilité de scinder le nom d'une molécule. Si dans le code précédent, nous remplaçons la ligne `\chemname{\chemfig{R-C(-[:-30]OH)=[:30]O}}{Acide carboxylique}` par `\chemname{\chemfig{R-C(-[:-30]OH)=[:30]O}}{Acide\\carboxylique}`, nous obtenons :



Alcool — Acide carboxylique — Ester — Eau

7.8 Quelques mots sur l'écriture des ions

La commande *chemfig* communique avec l'environnement mathématique de LaTeX. Il est ainsi facile d'écrire un ion. Il faut néanmoins placer le signe « - » entre accolades pour éviter que *chemfig* ne le confonde avec un symbole de liaison. Ainsi, le code suivant

```
\chemname{\chemfig{Na^+}}{ion sodium} \hspace{3cm}
\chemname{\chemfig{Cl^{-}}}{ion chlorure} \hspace{3cm}
\chemname{\chemfig{H_3C-C(-[1]O^{-})=[7]O}}{ion acétate}
```

permet de dessiner :

ion sodium — ion chlorure — ion acétate

Il est également possible d'entourer la charge de l'ion en utilisant les commandes `\ominus` et `\oplus` . De façon à contrôler davantage la présentation, il existe aussi des commandes `\chemabove` et `\chembelow` qui permettent de placer les charges au-dessus et en dessous de l'atome considéré. Le code ci-dessous

```
\chemfig{-(-[1]O^{\ominus})=[7]O} \hspace{3cm}
\chemfig{-\chemabove{N}{\scriptstyle \oplus}(=[1]O)-[7]O^{\ominus}}
\end{center}
```

dessine les ions suivants :

7.9 La représentation de Lewis

La commande *Lewis* permet de placer des paires d'électrons, de simples électrons ou une lacune. Elle peut être utilisée seule ou à l'intérieur d'une commande *chemfig*. La syntaxe utilisée étant :

\Lewis{$n_1 n_2 n_3 ... n_i$,atome}

dans laquelle les $n_1 n_2 n_3 ... n_i$ sont des nombres entiers compris entre 0 et 7 représentant les positions désirées, en multiples de 45°, autour de l'atome (comme pour les angles de liaisons prédéfinis). L'absence de symbole après un nombre indique à Lewis qu'elle doit placer une paire d'électrons représentée par un trait. Si, après un nombre, nous mettons un double point « : », un simple point « . », Lewis fera de même à la position donnée autour de l'atome. Pour placer une lacune, on place après un nombre un trait vertical « | ». Ainsi, le code suivant

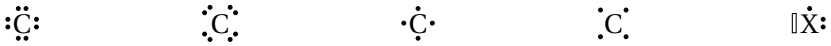
```
\Lewis{0246,A} \hspace{3cm} \Lewis{1357,B}
```

donne la représentation ci-dessous

tandis que

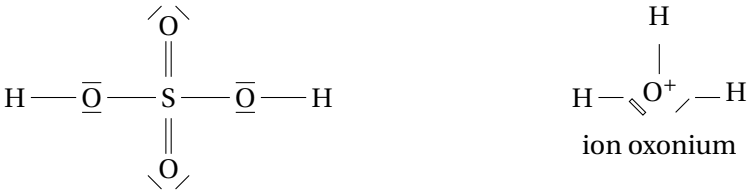
```
\Lewis{0:2:4:6:,C} \hspace{2cm} \Lewis{1:3:5:7:,C}
\hspace{2cm}
\Lewis{0.2.4.6.,C} \hspace{2cm} \Lewis{1.3.5.7.,C}
\hspace{2cm}
\Lewis{0:2.4|,X}
```

dessine ce qui suit



Nous proposons, en guise d'exemples, les codes permettant d'obtenir les deux représentations de Lewis ci-après :

```
\chemfig{H-\Lewis{26,O}-S(=[2]\Lewis{13,O})(=[6]\Lewis{57,O})-\Lewis{26,O}-H}
\hspace{3cm}
\chemname{\chemfig{H-\Lewis{5|7,O^+}(-[2]H)-H}}{ion oxonium}
```



ion oxonium

7.10 Intégrer *chemfig* dans un environnement *tikzpicture*

7.10.1 Une équation d'oxydoréduction

Soit à décrire l'oxydation des ions ferreux par le permanganate. C'est une équation d'oxydoréduction bien connue qui s'écrit :

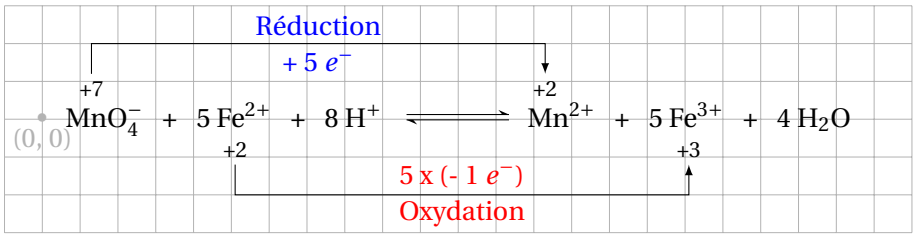
$$MnO_4^- + 5\,Fe^{2+} + 8\,H^+ \rightleftharpoons Mn^{2+} + 5\,Fe^{3+} + 4\,H_2O$$

Le code étant celui-ci :

```
\chemfig{MnO_4^{-}} \chemsign{+} \chemfig{5\ Fe^{2+}} \chemsign{+} \chemfig{8\ H^+}
\chemrel{<>}
\chemfig{Mn^{2+}} \chemsign{+} \chemfig{5\ Fe^{3+}} \chemsign{+} \chemfig{4\ H_2O}
```

Dans le cas présent, nous souhaitons préciser les états d'oxydation des atomes qui participent aux échanges électroniques, indiquer les électrons échangés et situer l'oxydation ainsi que la réduction. L'enjeu est difficile mais avec les connaissances acquises précédemment, nous allons relever le défit.

Commençons par nous placer dans un environnement *tikzpicture*, lui-même contenu dans l'environnement *center*. Nous donnons successivement le résultat recherché et le code pour y parvenir. Ce dernier sera ensuite décrit point par point. Il n'est pas inutile, si le besoin s'en fait sentir, de consulter les éléments étudiés aux sections 1.2, 1.3, 3.1 et 3.2 du présent manuel.




```
\begin{tikzpicture}
% Dessiner une grille de référence pour repérer les divers éléments par leurs coordonnées
\draw[step=0.5cm, gray!30, very thin] (-0.5, -1.5) grid (11.5, 1.5);
% Placer le centre de l'équation au milieu de la grille, ici le point (5.5, 0)
\node at (5.5,0) {
                \chemfig{MnO_4^{-}} \chemsign{+} \chemfig{5\ Fe^{2+}} \chemsign{+} \chemfig{8\ H^+}
                \chemrel{<>}
                \chemfig{Mn^{2+}} \chemsign{+} \chemfig{5\ Fe^{3+}} \chemsign{+} \chemfig{4\ H_2O}
                };
% La réduction
\node at (0.65,0.4) {\footnotesize +7}; \node at (6.65,0.4) {\footnotesize +2};
\draw[->,>=latex] (0.65,0.6) -- (0.65,1) -- (6.65,1) -- (6.65,0.6);
\node[blue] at (3.65,1.25) {Réduction}; \node[blue] at (3.65,0.75) {+ 5 $e^{-}$};
% L'oxydation
\node at (2.55,-0.4) {\footnotesize +2}; \node at (8.55,-0.4) {\footnotesize +3};
\draw[->,>=latex] -- (6.65,0.6) (2.55,-0.6) -- (2.55,-1) -- (8.55,-1) -- (8.55,-0.6);
\node[red] at (5.55,-1.25) {Oxydation}; \node[red] at (5.55,-0.75) {5 x (- 1 $e^{-}$)};
\end{tikzpicture}
```

La première ligne de code dessine la grille de référence. C'est un rectangle dont les sommets inférieur gauche et supérieur droit ont pour coordonnées

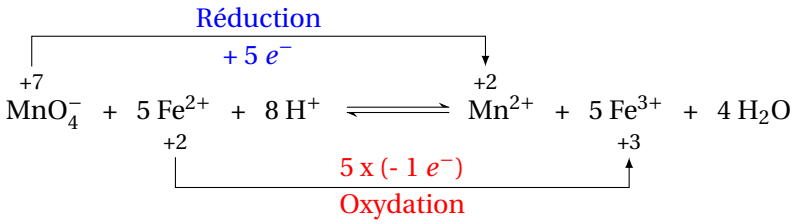
respectives (-0.5, -1.5) et (11.5, 1.5), ces valeurs étant exprimées en cm. Les côtés de chaque carré mesurent 0.5 cm.

À la seconde ligne de code, nous plaçons un noeud au centre de cette grille, le point de coordonnées (5.5, 0). Ce noeud contiendra, entre accolades la totalité du code générant l'équation de la réaction.

Ces deux étapes franchies, nous pouvons apprécier, à partir de la grille, les coordonnées des noeuds qui contiendront les états d'oxydation, le départ et l'arrivée de chacune des deux flèches ainsi que les inscriptions que nous souhaitons ajouter. Ainsi, l'état d'oxydation du manganèse, +7 dans l'ion permanganate, se positionnera avec *node* au point de coordonnées (0.65,0.4). Remarquons, au passage, la présence de l'instruction *footnotesize*, entre les accolades et avant +7, qui permet de redimensionner la taille des caractères.

De la même façon, nous repérons sur la grille les points de départ des flèches que nous dessinons avec l'instruction *draw*. Nous positionnons également les textes et, on voit qu'il est possible d'utiliser des couleurs au sein des noeuds si cette option est souhaitée.

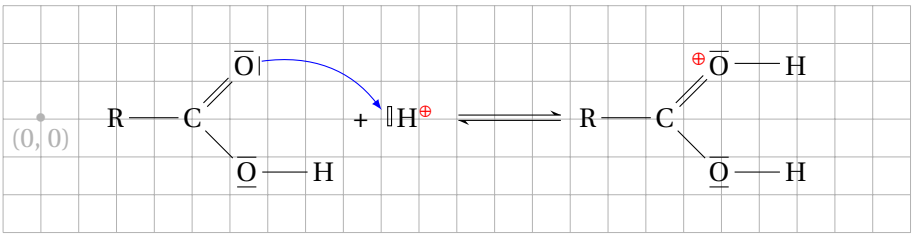
Dernière étape, la construction étant terminée, il suffit d'enlever la première ligne de code pour supprimer la grille et obtenir l'équation ci-dessous.



7.10.2 Mécanismes réactionnels

Qu'il s'agisse de mécanismes réactionnels ou d'effets mésomères, le processus de représentation du phénomène est semblabe à celui utilisé pour les équations d'oxydoréduction.

Nous montrons, ici, la réaction décrivant la première étape du processus d'estérification, à savoir, la protonation du groupe carboxyle.

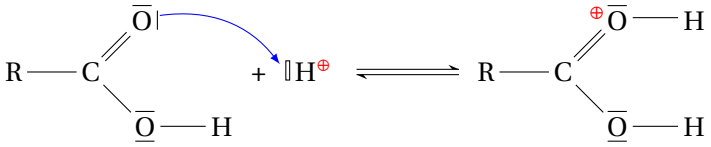


```
\begin{tikzpicture}
% Dessiner une grille de référence pour repérer les divers éléments par leurs coordonnées
\draw[step=0.5cm, gray!30, very thin] (-0.5, -1.5) grid (11.5, 1.5);
% Placer le centre de l'équation au milieu de la grille, ici le point (5.5, 0)
\node at (5.5,0) {
                  \chemfig{R-C(=[1]\Lewis{02,O})(-[7]\Lewis{26,O}-H)} \chemsign{+}
                  \chemfig{\Lewis{4|,H^{\textcolor{red}\oplus}}}
                  \chemrel{<>}
                  \chemfig{R-C(=[1]\Lewis{2,O}-H)(-[7]\Lewis{26,O}-H)}
                  };
\node (D) at (2.8,0.75) {}; \node (A) at (4.6,0) {}; \draw[blue,->,>=latex] (D) to[bend left] (A);
\node[red] at (8.7,0.8) {\footnotesize $\oplus$};
\end{tikzpicture}
```

Il n'y a pas beaucoup d'éléments nouveaux par rapport à ce qui a été dit à la section précédente. Notons l'usage des représentations de Lewis et le fait que la charge positive qui apparaît dans le membre de droite n'a pas été introduite avec *chemfig*. Ce dernier, en effet, positionne mal le doublet supérieur sur l'oxygène si nous tentons de le faire avec cet outil. Nous avons contourné la difficulté en introduisant la charge à l'intérieur d'une instruction *node* à la dernière ligne de code.

Quant à la flèche qui apparaît dans le membre de gauche, elle a été représentée avec l'instruction to[bend left] comme cela a été vu à la section 3.2.1.

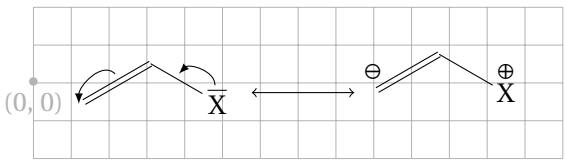
Il suffit maintenant d'éliminer la ligne dessinant la grille pour obtenir la représentation souhaitée.



7.10.3 Effets mésomères

L'effet mésomère en chimie organique est lié à la délocalisation des électrons. Il caractérise la propriété d'un substituant ou d'un groupe fonctionnel de céder ou d'accepter un doublet électronique permettant la délocalisation de celui-ci et, par là même, d'abaisser l'énergie du composé augmentant ainsi sa stabilité.

La représentation de ces effets repose sur les mêmes idées que celles évoquées au point précédent.

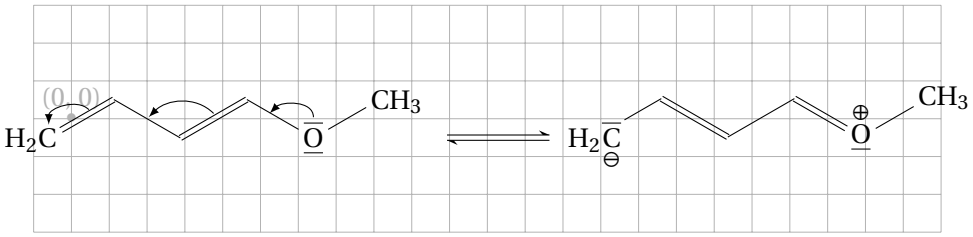


La figure ci-dessus s'obtient à partir du code suivant :

```
\begin{tikzpicture}
% Dessiner une grille de référence pour repérer les divers éléments par leurs coordonnées
\draw[step=0.5cm, gray!60, very thin] (0, -1) grid (7, 1);
% Indiquer l'origine des coordonnées comme repère supplémentaire
\node[gray!60] at (0,0) {$\bullet$}; \node[gray!60,below] at (0,0) {(0, 0)};
% Placer le centre de l'équation au milieu de la grille, ici le point (5.5, 0)
\node at (3.5,0) {
                    \chemfig{=[:30]-[:-30]\Lewis{2,X}}
                    \chemrel{<->}
                    \chemfig{\chemabove{\vphantom{X}}{\ominus}=[:30]-[:-30]\chemabove{X}
{\oplus}}
                    };
\node (D) at (1.2,0) {}; \node (A) at (0.6,-0.4) {}; \draw[->,>=latex] (D) edge[out=135,in=90] (A);
\node (d) at (2.4,-0.15) {}; \node (a) at (1.8,0) {}; \draw[->,>=latex] (d) edge[out=90,in=45] (a);
\end{tikzpicture}
```

dans laquelle nous avons utilisé l'instruction *edge[out= ,in=]* vue au paragraphe 3.2.2. Nous trouvons également l'instruction *vphantom* dont l'usage est général en LaTeX et n'est certainement pas réservé à la chimie[3]. En éliminant la grille, nous obtenons alors :

Pour l'effet mésomère qui suit, nous donnons d'abord la formule au sein du repère constitué par la grille et l'origine des coordonnées.

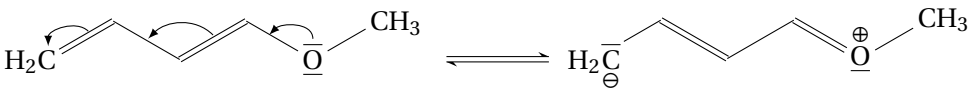


Quant au code permettant d'écrire l'équation, il est donné immédiatement après celle-ci.

[3]La notion de fantômes en LaTeX est simple : il s'agit de réserver un emplacement pour un objet, sans pour autant le faire apparaître. Les fantômes sont souvent utilisés pour ajuster l'alignement. Il y a trois types de fantômes obtenus respectivement par les commandes *phantom*, *hphantom*, et *vphantom*. Une illustration peut être trouvée dans l'alignement des radicaux. Ainsi,

$$\sqrt{x}+\sqrt{X}+\sqrt{x}.$$

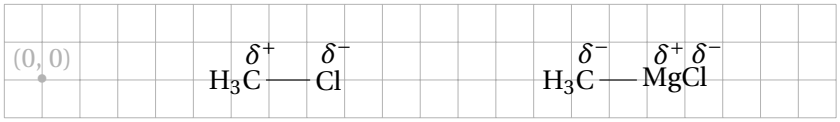
La différence est subtile et sera peut-être uniquement observée sur une version imprimée du document. Quoi qu'il en soit, la composition des deux radicaux de droite est plus homogène (ils sont alignés en haut) et donc visuellement plus satisfaisante.



```
\begin{tikzpicture}
\node at (5.5,0) {
                  \chemfig{H_2C=[:30]-[:-30]=[:30]-[:-30]\Lewis{26,O}-[:30]CH_3}
                  \chemrel{<>}
                  \chemfig{H_2\chembelow{\Lewis{2,C}}{\ominus}-[:30]=[:-30]-[:30]=[:-30]\chemabove{\Lewis{6,O}}
{\oplus}-[:30]CH_3}
                  };
\node (D1) at (0.35,0) {}; \node (A1) at (-0.3,-0.2) {}; \draw[->,>=latex] (D1) edge[out=135,in=90] (A1);
\node (D2) at (2,-0.05) {}; \node (A2) at (0.9,-0.1) {}; \draw[->,>=latex] (D2) edge[out=135,in=45] (A2);
\node (D3) at (3.2,-0.1) {}; \node (A3) at (2.5,-0.05) {}; \draw[->,>=latex] (D3) edge[out=90,in=45] (A3);
\end{tikzpicture}
```

7.10.4 Molécules polaires

Représenter les charges partielles, telles qu'elles se répartissent au sein d'une molécule, peut se faire en travaillant de la même façon.



Ainsi le code ci-dessous, construit à partir de la grille de référence ci dessus,

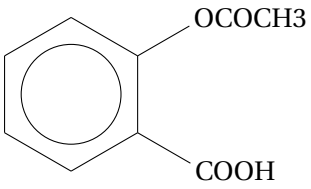
```
\begin{tikzpicture}
\node at (5.5,0) {
                  \chemfig{H_3C-Cl}
                  \hspace{2.5cm}
                  \chemfig{H_3C-MgCl}
                  };
\node at (2.9, 0.35) {$\delta^+$}; \node at (3.9, 0.35) {$\delta^-$};
\node at (7.3, 0.35) {$\delta^-$}; \node at (8.3, 0.35) {$\delta^+$}; \node at (8.8, 0.35) {$\delta^-$};
\end{tikzpicture}
```

permet d'obtenir les formules suivantes

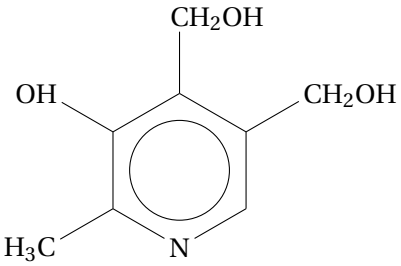
7.11 Exemples de molécules

Comme exercices, et en guise de conclusion, nous proposons cinq constructions de molécules dont le code est à la fois logique et difficile pour un profane. Elles devraient permettre de tester la compréhension de divers points décrits dans ce chapitre. Il est vivement conseillé, pour débuter, de travailler en plusieurs

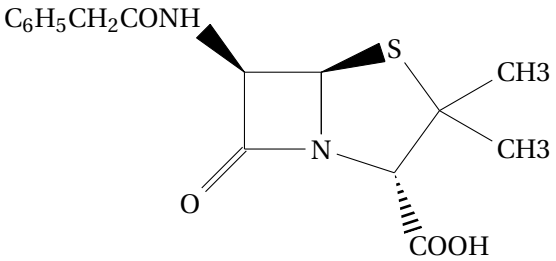
étapes en analysant, à chaque pas, le code utilisé et son effet sur la molécule en construction. Nous préconisons de dessiner, dans un premier temps, les cycles, les cycles imbriqués s'il y en a, ensuite de greffer sur ceux-ci les divers atomes et liaisons nécessaires.



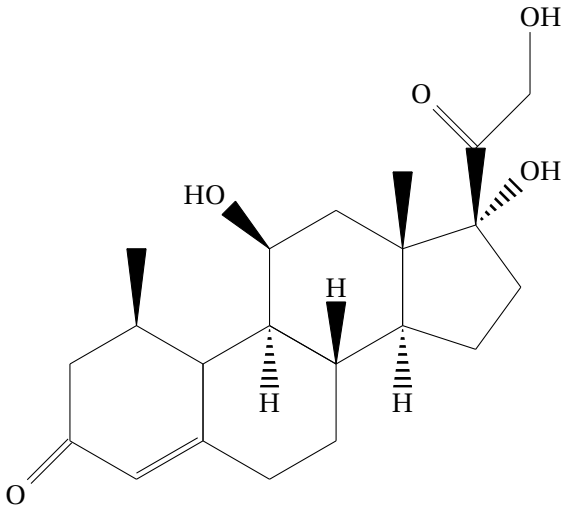
Aspirine



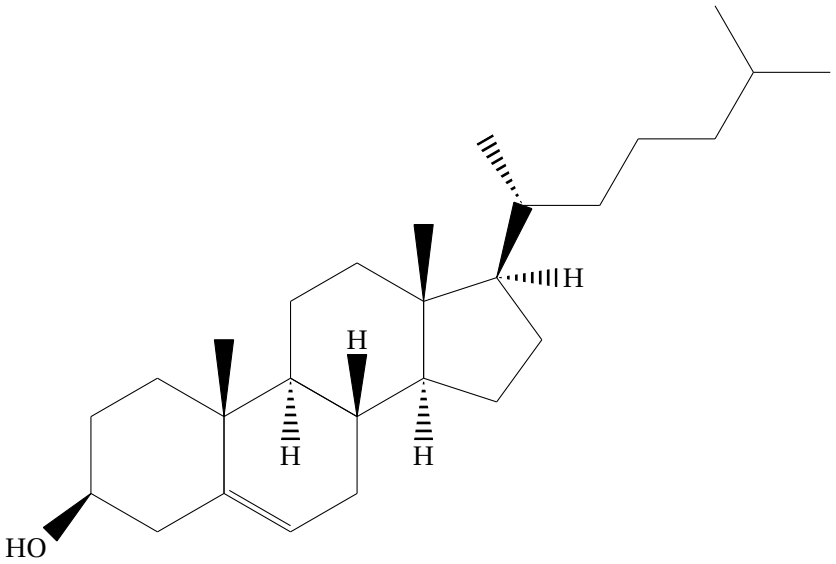
Vitamine B6



Pénicilline



cortisol



cholestérol