Université de Limoges

Guide LaTeX – TikZ – pgfplots

Stéphane Vinatier

stephane.vinatier@unilim.fr

Formation doctorale

Université de Limoges

Table des matières

Chapitre 1

Structuration du document

Le traitement de texte LaTeX produit un document `pdf` à partir d'un fichier source (suffixe en `.tex`) dans lequel, en plus du contenu à mettre en page, on place un certain nombre de commandes indiquant notamment le type de document, les extensions de LaTeX (ou *packages*) à charger, ainsi que des indications de mise en page. C'est la *compilation* de ce ficher source qui produit le fichier pdf ; parfois des erreurs sont signalées à la compilation, il est bon de s'assurer qu'elles sont bégnines et, le cas échéant, de les corriger.

Dans la première section ci-dessous, on s'intéresse à la strucure du fichier source, qui commence toujours par la commande **`\documentclass`** ; dans la section suivante, on introduira les commandes permettant de structurer le contenu : des commandes indiquant les chapitres, sections, sous-sections ..., paragraphes ainsi que des commandes pour introduire des environnements spécifiques : listes, tableaux, équations, insertion d'images...

1.1 Structure d'un fichier source

1.1.1 Préambule

Un document LaTeX a nécessairement un *préambule* (c'est souvent le même pour différents fichiers) qui fixe un certain nombre de paramètres importants (par exemple le type ou *classe* du document : livre ou article ?) et charge un certain nombre de `packages`, extensions de LaTeX qu'on ajoute en fonction des besoins, mais dont certaines s'avèrent quasiment indispensables pour un bon usage du logiciel.

```
\documentclass{article}

\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage{lmodern}
\usepackage[a4paper]{geometry}
\usepackage[french]{babel}

\usepackage{amsfonts,amssymb,amsmath,amsthm}
\usepackage{enumerate}
```

```
\usepackage{graphicx}
\usepackage{array}

\usepackage{hyperref}
\hypersetup{pdfstartview=XYZ}

\begin{document}

...

...

...

\end{document}
```

Le préambule est constitué de tout ce qui se trouve avant la commande `\begin{document}`.

Classe. La *classe* du document est l'argument de la **1re commande** du fichier : `\documentclass` . Les principaux choix sont `book`, `report`, `article` pour des documents de type écrit (composés de « pages »), `beamer` pour les présentations numériques (composées de « transparents », aussi appelés « diapositives », « slides » en anglais).

Les associations ou revues scientifiques (notamment mathématiques) définissent souvent leur propre classe (par exemple `amsart`). Celles-ci ne sont pas toujours fournies avec la distribution LaTeX, il faut alors récupérer le fichier de style correspondant (`amsart.cls` dans l'exemple) et le placer dans le même répertoire que le fichier dans lequel la classe est utilisée.

1.1.2 Corps du document

Le corps du document est tout ce qui se trouve entre les *balises* `\begin{document}` et `\end{document}` (ce qui se trouverait après cette dernière est ignoré par LaTeX). L'essentiel du contenu du document se trouve là [1].

En plus du contenu proprement dit, on y placera des commandes indiquant le statut de certains contenus (titre de section, définition, théorème, algorithme...), des commandes de mise en forme portant sur des expressions :

— en *italique* : `\textit{...}` ;
— en **gras** : `\textbf{...}` ;
— expression mathématique : `$ ... $` ou `\( ... \)`

ou sur un contenu plus important :

— mise en valeur de certains passages (par exemple texte centré) ;
— en particulier équations mathématiques centrées hors du texte ;

1. Les commandes `\title` (titre), `\author` (auteur(s), affiliation(s),...), `\date` (date, ou autre information) peuvent être placés avant, mais la commande qui gère ces informations et crée le titre, `\maketitle`, est normalement placée après.
La note de bas de page est obtenue avec la commande `\footnote{...}`

— citations ;
— listes (comme celle-ci, ou un peu différentes) ;
— tableaux ;
— illustrations ;
ou encore la bibliographie, la table des matières, celle des illustrations, l'index,...

Environnements. Pour tous les cas cités dans la liste ci-dessus, LaTeX a prévu des *environnements* spécifiques, que l'on délimite à l'aide de balises, comme pour l'environnement `document` qui forme le corps du texte :

```
\begin{center}       ... \end{center}       % texte centré
\begin{equation}     ... \end{equation}     % équations centrées
\begin{quote}        ... \end{quote}        % citations
\begin{itemize}      ... \end{itemize}      % listes
\begin{description}  ... \end{description}  % définition de termes
\begin{enumerate}    ... \end{enumerate}    % listes
\begin{tabular}      ... \end{tabular}      % tableaux
\begin{figure}       ... \end{figure}       % figures
```

et beaucoup d'autres... Noter que la partie d'une ligne qui suit le symbole % n'est pas lue par LaTeX et peut donc servir à commenter le code. Chaque environnement a ses options, ses arguments, il sera bien utile de trouver des documents de référence pour utiliser chacun d'eux au mieux ! Il en existe beaucoup, de toutes tailles et sur tous supports. Mentionnons le très accessible [Wik] :

https://fr.wikibooks.org/wiki/LaTeX

1.2 Organisation du contenu

1.2.1 Espacement

LaTeX gère extrêmement bien l'espacement entre les mots, entre les lignes, entre les paragraphes et autres découpages (voir ci-dessous). C'est un de ses principaux intérêts et cela permet à l'utilisateur de se concentrer sur le *fond* plutôt que sur la *forme*.

Par exemple, LaTeX ne tient pas compte du nombre de fois où vous avez appuyé sur la touche 'espace' entre deux mots :

```
la                    touche                    `espace'.
```

produit le résultat :

la touche 'espace'.

Il ne tient pas compte non plus du nombre de lignes que vous sautez entre deux paragraphes : c'est comme si vous en sautiez une seule.

```
les
sauts

de


ligne
```

produit le résultat :

les sauts
de
ligne

Comme on le voit, passer à la ligne (sans en sauter) revient à appuyer sur la touche 'espace' ; sauter une ligne ou plusieurs produit un passage à la ligne (avec indentation par défaut).

Principe de base : ne pas interférer (ou alors le moins possible) avec la gestion automatisée de l'espacement, c'est-à-dire *ne pas utiliser* un certain nombre de commandes qui agissent dessus, mais plutôt les très nombreuses possibilités de découpage du texte offertes par LaTeX.

1.2.2 Découpage

Selon la classe de document (donnée par **\documentclass**), on découpe en

- parties : \part,
- chapitres : \chapter,
- sections : **\section**,
- sous-sections : \subsection,
- ... \subsubsection, \paragraph, \subparagraph...

Par exemple, la suite de commandes

```
\subsection[Ex. ssec.]{Exemple de sous-section}
  Nous allons voir dans la suite ...
\subsubsection{Exemple de sous-sous-section}
  ... que \LaTeX{}  gère automatiquement ...
\paragraph{Exemple de paragraphe.}
  ... non seulement l'espacement ...
\subparagraph{Exemple de sous-paragraphe.}
  ... mais aussi la numérotation !
```

produit le résultat :

1.2.3 Exemple de sous-section

Nous allons voir dans la suite ...

Exemple de sous-sous-section

... que LaTeX gère automatiquement ...

Exemple de paragraphe. ... non seulement l'espacement ...

Exemple de sous-paragraphe. ... mais aussi la numérotation !

Titre court. L'option `[Ex. ssec.]` passée à la commande `\subsection` remplace le titre choisi par un autre (ici abrégé) pour les affichages annexes (haut ou bas de page lorsque ceux-ci sont utilisés, table des matières, sommaire du visionneur pdf...). Cette option (valable pour toutes les variantes de commandes de sectionnement) peut également être utilisée lorsqu'une commande LaTeX « fragile » utilisée dans un titre pourrait ne pas apparaître correctement dans un des affichages annexes (c'est le cas de la commande de couleur `Ti\textit{\color{orange}k}Z` du titre du chapitre 4).

Pour les livres. Avec la classe `book`, on dispose de trois commandes de structuration supplémentaires : `\frontmatter` indique tout ce qui se trouve avant la matière principale (page de titre, dédicace, avant-propos, introduction, préface) ; les parties ne sont pas numérotées mais figurent dans la table des matières et les pages sont numérotées en chiffres romain bas-de-casse ; `\mainmatter` introduit le corps de l'ouvrage ; `\backmatter` introduit les index et tables des matières.

1.2.4 Référencement

La numérotation est automatique et les numéros peuvent être *référencés* dès lors qu'ils ont été *étiquetés*.

Imaginons que dans l'exemple ci-dessus, nous avons (enlevé l'option et) ajouté l'étiquette `\label{ssec:ex}` :

```
\subsection{Exemple de sous-section}\label{ssec:ex}
```

Alors (après compilation), on peut faire référence à cette sous-section ou à sa page avec les commandes `\ref{ssec:ex}` et `\pageref{ssec:ex}`. Ainsi,

```
Comme nous le disons dans la section \ref{ssec:ex} en page \pageref{
ssec:ex}, ...
```

produit le résultat :

> Comme nous le disons dans la section 1.2.3 en page 8, ...

Noter la couleur bleue du numéro dans le visionneur de pdf, qui indique la possibilité de cliquer sur celui-ci pour être amené automatiquement au début de la section référencée (grâce au chargement du package `hyperref`). Il est bien sûr conseillé de choisir des étiquettes suffisamment « parlantes » pour retrouver facilement à quoi elles font référence.

Avantage. En cas d'ajout d'une section ou d'une sous-section avant celle qui est référencée, LaTeX ajuste automatiquement tous les numéros !

Généralisation. Ce procédé étiquetage / référence (`\label{..}` / `\ref{..}` ou `\pageref{..}`) fonctionne pour tout ce qu'on veut numéroter :

- figures,
- tables (en utilisant l'environnement `table`),
- ...
- équations hors-texte (centrées)

avec une variante `\eqref{..}` dans le dernier cas. Voici une manière de présenter la formule d'Euler pour les polyèdres :

```
\begin{equation}\label{eq:euler}
  S-A+F=2
\end{equation}
La Formule \eqref{eq:euler} n'est pas toujours valable !
```

produit le résultat :

$$S - A + F = 2 \tag{1.1}$$

La Formule (1.1) n'est pas toujours valable !

Non numérotation. On peut choisir de ne pas numéroter les sections ou sous-sections, qu'on ne souhaite pas référencer en ajoutant une étoile à la commande :

```
\subsection*{No number}
```

produit le résultat :

No number

Pour une équation hors-texte sans numéro, on utilise `\[ ... \]` [2] :

```
\[
\Z/nm\Z \simeq \Z/n\Z \times \Z/m\Z
\]
```

produit le résultat :

$$\mathbb{Z}/nm\mathbb{Z} \simeq \mathbb{Z}/n\mathbb{Z} \times \mathbb{Z}/m\mathbb{Z}$$

(valable sous l'hypothèse que les entiers n et m sont premiers entre eux). Dans cette formule, une *macro* a été utilisée : dans le préambule (c'est-à-dire avant `\begin{document}`), on a écrit :

```
\newcommand\Z{{\mathbb Z}}
```

La commande `\Z` est donc un raccourci pour `{\mathbb Z}`, qui écrit Z dans la police mathématique *blackboard*.

2. Il est conseillé de *ne plus* utiliser la commande obsolète $$ ... $$ qui fontionne mal dans certains contextes. On peut utiliser $ ... $, ou `\(` ... `\)`, pour les formules mathématiques *en ligne*.

Autres environnements mathématiques. Il existe plusieurs alternatives à l'environnement `equation` : `align`, `gather`, `multline` pour différentes sortes d'alignement, `matrix` et ses nombreuses variantes, `cases`. On peut consulter [Gou10, p. 10] pour la syntaxe et les détails, on y trouvera aussi un certain nombre de commandes très utiles en mode mathématique (voir également les deux pages suivantes).

1.2.5 Environnements d'énoncés

Le package `amsthm` permet de définir des environnements pour des énoncés mathématiques ou autres, avec différents styles ([Gou10, p. 9]). Par exemple, on placera dans le préambule (après avoir chargé le package `amsthm`) les commandes suivantes :

```
\theoremstyle{plain}
\newtheorem{thm}{Théorème}
\newtheorem{prop}[thm]{Propriété}
\newtheorem{coro}[thm]{Corollaire}
%
\theoremstyle{definition}
\newtheorem{defi}[thm]{Définition}
\newtheorem{voca}[thm]{Vocabulaire}
\newtheorem{exo}[thm]{Exercice}
%
\theoremstyle{remark}
\newtheorem{rem}{Remarque}
```

Ces commandes créent les environnement `thm`, `prop` et `coro` dans le style `plain`, `defi`, `voca` et `exo` dans le style `definition`, et l'environnement `rem` dans le style `remark`.

L'option entre crochets `[thm]` qui est mentionnée pour plusieurs de ces environnements indique qu'ils sont tous numérotés en utilisant le même compteur que pour l'environnement `thm`, tandis que les énoncés du style `remark` sont numérotés à part, avec un compteur dédié. On peut choisir de numéroter les théorèmes par section en ajoutant l'option `[section]` à la commande `\newtheorem{thm}{Théorème}`.

Voici ce que donnent les styles `definition`, `plain` et `remark` :

Définition 1. On dit qu'un polyèdre est *convenable* s'il satisfait la formule d'Euler (1.1), où S désigne le nombre de sommets, A le nombre d'arêtes et F le nombre de faces du polyèdre.

Théorème 2. *Pour tout polyèdre* convenable*, on a* $S - A + F = 2$.

Remarque 1. Le texte de la Définition 1 et celui de cette remarque sont par défaut en romain (droit) tandis que celui du Théorème 2 est en italique. Le mot *convenable* a été mis en avant en utilisant la commande `\emph{convenable}`, ce qui produit de l'italique dans l'environnement `definition` où le reste du texte est droit et du texte droit dans l'environnement `plain` où le reste du texte est en italique.

Comme toujours la numérotation est automatique et l'étiquette `\label{rem:ex}` placée après `\begin{rem}` permet de faire référence à cette Remarque 1 en tapant `Remarque \ref{rem:ex}`. Noter la variante non numérotée `\newtheorem*{rem}{Remarque}`.

Preuves. Le package `amsthm` fournit un environnement `proof` qui s'ajuste en fonction de la langue du document (indiquée dans le préambule en option de `\uspackage{babel}`) :

```
\begin{proof}
  C'est tautologique !
\end{proof}
```

produit

Démonstration. C'est tautologique ! □

Pour être plus précis, on aurait pu écrire `\begin{proof}[Démonstration du Théorème \ref{thm:ex}]`.

1.2.6 Insertion d'images

Une fois qu'on s'est assuré d'avoir inclus le package `graphicx` dans le préambule, il est très simple d'inclure une image dans un document LaTeX :

```
\begin{center}
  \includegraphics[width=5cm]{logo-unilim.png}
\end{center}
```

donne le résultat :

Ici on a choisi de centrer l'image dans la page et de régler sa largeur à 5cm. Si le fichier `logo-unilim.png` s'était trouvé dans un autre répertoire que le répertoire courant, on aurait pu indiquer le chemin menant à ce répertoire, par exemple (dans un système de fichiers Linux) :

```
  \includegraphics[width=5cm]{../Images/logo-unilim.png}
```

ou encore `{~/Enseignement/Latex/Images/logo-unilim.png}`, `{/home/vinatier/Enseignement/Latex/Images/logo-unilim.png}`.

L'environnement `figure`. Il est souvent souhaitable d'« habiller » une image en précisant une légende, voire en fournissant un numéro permettant d'y faire référence à d'autres endroits du texte (ou dans d'autres documents). C'est ce que permet l'environnement `figure` :

```
\begin{figure}[h]
\centering
\includegraphics[width=6cm]{etoile-11-branches-couleurs.jpg}
\caption{Six hendécagones imbriqués}\label{11gones}
\end{figure}
```

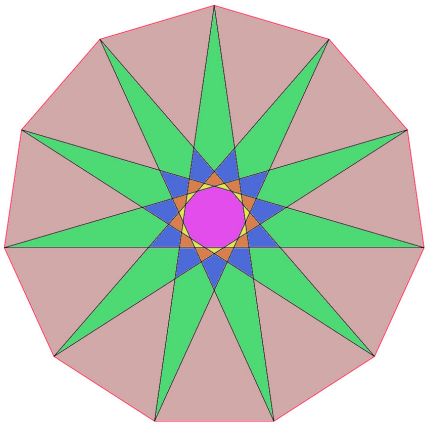


FIGURE 1.1 – Six hendécagones imbriqués

Voir la Figure 1.1 pour le résultat. Plusieurs remarques importantes :

- on *n'utilise pas* `\begin{center} ... \end{center}` à l'intérieur d'un environnement `figure` ; si l'on souhaite centrer la figure dans la page, on utilise la commande `\centering` ;
- la légende est saisie dans la commande `\caption` ;
- la figure est automatiquement numérotée (par défaut dans un compteur réservé aux figures) et on peut y faire référence en évoquant la `Figure \ref{11gones}` (c'est-à-dire la Figure 1.1) ;

Flottants. L'environnement `figure` est *flottant*, c'est-à-dire que LaTeX est susceptible de placer la figure à un autre endroit du document que celui où on l'a incluse dans le code source ; l'option `[h]` (h pour "here") qui suit la commande `\begin{figure}` indique à LaTeX qu'on souhaite qu'elle soit placée à l'endroit où on l'a incluse, mais LaTeX privilégie souvent l'harmonie de la mise en page aux souhaits qu'on lui indique... d'ailleurs la plupart du temps, l'option `[h]` est transformée en `[ht]` à la compilation (t pour "top", donc ici ou en haut), mais même avec cette souplesse supplémentaire LaTeX n'accède pas toujours à nos désirs.

Il est en général préférable de laisser LaTeX faire ce travail de mise en page, quitte à remplacer une formule comme `dans la figure suivante` par `dans la Figure \ref{..}` en tirant avantage du système de référencement de LaTeX. Cependant il arrive qu'on ait besoin de mieux contrôler le comportement flottant des figures. La commande `\clearpage` oblige LaTeX à placer tous les flottants en attente avant de continuer, ce qui permet de s'assurer que les figures apparaissent dans la bonne (sous-(sous-))section, par exemple.

Pour les cas extrêmes, le package `float` (à appeler dans le préambule) permet de renforcer le contrôle des flottants. En particulier, il donne accès à l'option `[H]` pour les environnements flottants qui assure le placement à l'endroit précis où il se trouve dans le code source [3].

3. Voir par exemple https://tex.stackexchange.com/questions/8625/force-figure-placement-in-text ou https://zestedesavoir.com/tutoriels/826/introduction-a-latex/1322_completer-vos-documents/les-flottants/

Figures faites maison. Il est souvent avantageux de produire la figure à l'intérieur du document LaTeX, pour l'inclure au mieux dans le document, harmoniser les fontes, polices, tailles de caractères, obtenir une résolution inégalable, gagner en souplesse en cas de modification des données représentées sur la figure, voire pour faire dessiner automatiquement celle-ci à partir d'un fichier de données externes. Ce sera l'objet des chapitres 4 et 5.

1.2.7 Tables et tableaux

L'environnement `table`. Il sert à numéroter et référencer les tableaux et s'utilise exactement comme l'environnement `figure`, avec le même comportement flottant. Par exemple :

```
\begin{table}[h]
  \centering
  \begin{tabular}{|c|c|c|}
    \hline
    $\circ$ & $\times$ & $\times$ \\
    \hline
    $\times$ & $\circ$ & $\circ$ \\
    \hline
    $\times$ & $\circ$ & $\times$ \\
    \hline
  \end{tabular}
  \caption{Partie de morpion}\label{tab:morpion}
\end{table}
```

produit la Table 1.1.

○	×	×
×	○	○
×	○	×

TABLE 1.1 – Partie de morpion

L'environnement `tabular`. C'est cet environnement qui permet de créer le tableau. Il peut être utilisé indépendamment de l'environnement `table`, auquel cas le tableau n'est ni flottant, ni numéroté. Voici quelques éléments de syntaxe :

- la balise `\begin{tabular}` est suivie du descriptif et de la justification des colonnes, ici `{|c|c|c|}` indique trois colonnes de contenu centré (horizontalement) et bordées de traits verticaux ;
- remplacer `c` par `l` pour justifier à gauche, par `r` pour justifier à droite dans une colonne ;
- on peut supprimer les barres verticales `|` si on veut pas de trait séparateur vertical ou les doubler pour un trait de séparation double ;
- quand le comportement demandé est identique pour n colonnes, on peut le résumer à l'aide de `*{`n`}{`*motif*`}`, ce qui donne sur notre exemple `{|*{3}{c|}}` ;

— le contenu du tableau est donné en lignes, les entrées sont séparées par des esperluètes `&` et chaque ligne se termine par `\\` ; attention à avoir le bon nombre d'esperluètes pour chaque ligne (un de moins que le nombre de colonnes) ;
— les **`\hline`** qui apparaissent entre les lignes sont facultatifs, ils indiquent qu'on souhaite un trait de séparation horizontal entre deux lignes du tableau (doubler la commande pour doubler le trait).

Il est conseillé d'ajouter le package `array` dans le préambule. Pour plus d'options (justification, fusion de lignes ou de colonnes, espacement, couleur, index des tables...), on peut consulter [Gou10, p. 16-17] ; pour plus de détails, voir [Wik, 5. Tableaux].

Chapitre 2

Références bibliographiques

Un procédé spécifique, similaire au référencement évoqué plus haut, existe pour les références bibliographiques, avec la commande **\cite** pour citer (à la place de **\ref**). L'étiquetage se fait dans la bibliographie elle-même, pour laquelle deux possibilités existent :

— bibliographie « à la main », incluse dans le fichier ;

— bibliographie externe dans un fichier `.bib` séparé.

Dans le dernier cas, le fichier externe peut être géré à l'aide de `bibtex` ou de `biblatex`. De plus seules les références du fichier externe qui sont citées dans le document apparaissent dans la bibliographie (si on veut y faire apparaître une référence qui n'est pas citée, on peut le faire avec la commande **\nocite**).

2.1 Bibliographie à la main

En fin de document (du moins à l'endroit où on veut placer la bibliographie), on place l'environnement *thebibliography* :

```
\begin{thebibliography}
  ...
\end{thebibliography}
```

à l'intérieur duquel on placera les références bibliographiques sous la forme :

```
\bibitem[A]{andrews} Andrews G.E., {\it The theory of partitions},
Encyclopedia of Mathematics and its applications {\bf 2}, Addison-
Wesley (1976).
```

Noter l'étiquette qui suit **\bibitem** : `{andrews}`, qui permet de faire référence à ce livre dans le corps du texte avec la commande **\cite**`{andrews}`, que LaTeX remplacera par l'option `[A]` de **\bibitem**. Si rien n'est indiqué en option de **\bibitem**, LaTeX donne un numéro, disons n, à la référence, qui apparaît alors sous la forme `[`n`]` dans le texte.

Voici un exemple complet :

```
\begin{thebibliography}{M}
```

```
\bibitem[A]{andrews} Andrews G.E., {\it The theory of partitions},
Encyclopedia of Mathematics and its applications {\bf 2}, Addison-
Wesley (1976).
\bibitem[DM]{dm} Dixon J.D., Mortimer B., {\it Permutation groups},
Graduate Texts in Mathematics {\bf 163}, Springer-Verlag, New York
(1996).
\end{thebibliography}
```

L'argument qui suit `\begin{thebibliography}`, ici `{M}`, indique à LaTeX l'indentation nécessaire, dans la liste des références, pour faire de la place aux lettres entre crochets (`[A]` et `[DM]`) qui remplacent les noms des ouvrages dans le texte. Vu le résultat (capture d'écran ci-dessous, il s'agit d'une publication en anglais), il aurait sans doute mieux valu ici mettre `{DM}` en argument.

References

[A] Andrews G.E., *The theory of partitions*, Encyclopedia of Mathematics and its applications **2**, Addison-Wesley (1976).

[DM] Dixon J.D., Mortimer B., *Permutation groups*, Graduate Texts in Mathematics **163**, Springer-Verlag, New York (1996).

2.2 Bibliographie externe avec `bibtex`

La bonne façon de gérer la bibliographie, dès lors qu'elle prend une certaine importance ou qu'elle est susceptible d'être utilisée dans différentes publications, est de placer la liste des références dans un fichier externe, avec l'extension `.bib` (on peut l'appeler `bib.bib` mais ce n'est sans doute pas le choix le plus heureux, appelons-le plutôt `refs.bib`[1]).

Ce fichier est à éditer avec l'éditeur de texte de son choix (le même que celui du document LaTeX de préférence). On peut y entrer manuellement les références mais c'est un peu fastidieux à la longue (voir l'exemple ci-dessous). Heureusement les bases de données bibliographiques, par exemple le Service commun de documentation de l'Université de Limoges, proposent l'export des références trouvées dans le format `bibtex` qui est celui qui nous intéresse maintenant. Les différentes caractéristiques de l'ouvrage (en premier lieu son type : livre, article, rapport..., également ses titre, auteur, éditeur, année, et bien d'autres) sont enregistrées dans les champs dédiés comme dans l'exemple :

```
@book{Darwin,
Author = {Darwin, Charles},
Publisher = {London: John Murray},
Title = {On the Origin of Species by Means of Natural Selection, or
the Preservation of Favoured Races in the Struggle for Life},
Year = {1859},
}
```

1. Il est bon de rappeler ici qu'il vaut mieux s'en tenir aux caractères ASCII pour les noms de fichiers à utiliser avec LaTeX, sans accents ou cédilles donc, et sans espaces.

Le fichier `refs.bib` est une suite de références écrites dans ce format, dans l'ordre qu'on veut. Comme on le voit, il ne contient aucune mise en forme, c'est à la compilation, en fonction du style choisi, que celle-ci se fera, automatiquement bien sûr. De plus, une entrée du fichier `refs.bib` n'apparaît dans la bibliographie d'un document qui fait appel à ce fichier que si elle est citée dans le corps du texte (du moins c'est le comportement par défaut). Il est donc possible de rassembler toutes les références qu'on est susceptible de citer dans un seul et même fichier, qu'on appelera depuis tous les documents qu'on sera amené à rédiger.

Dans l'exemple ci-dessus, l'étiquette de l'ouvrage cité est `Darwin` (c'est la suite de caractères écrits entre l'accolade qui suit le type de l'ouvrage, ici `@book`, et la virgule, en première ligne). Cette référence apparaîtra donc dans la bibliographie si et seulement si une commande `\cite{Darwin}` se trouve dans le document.

En pratique. Convaincus de l'intérêt d'une bibliographie externe ? Alors la légère complexification de la compilation qu'elle engendre (seulement lorsque la bibliographie est modifiée) ne vous paraîtra pas insurmontable.

Dans le document LaTeX, à l'endroit où on veut voir apparaître la bibliographie, on écrit la commande :

```
\bibliography{refs}
```

dans laquelle l'argument `refs` est le préfixe du nom du fichier de bibliographie externe (ici `refs.bib` comme convenu). N'importe où dans le document, en général juste après `\begin{document}`, on précise le style de bibliographie à adopter, par exemple :

```
\bibliographystyle{plain}
```

Il y a aussi le style `alpha` (voire aussi `smfplain` et `smfalpha` pour des publications en français).

Il n'y a plus alors qu'à exécuter la suite d'instructions suivantes (souvent prises en charge par l'éditeur de texte, s'il est dédié à LaTeX) :

— compiler avec la commande pdflatex (ou latex) ;
— compiler avec la commande bibtex ;
— compiler deux fois avec la commande pdflatex (ou latex).

Et ça marche. Ouf ! Cette série d'instructions n'est nécessaire qu'en cas de changement dans la bibliographie (modification du fichier externe `refs.bib` ou ajout ou suppression de la citation d'une des références de ce fichier). Le reste du temps, la simple compilation avec pdflatex suffit.

2.3 Bibliographie externe avec `biblatex`

Avec le même fichier externe de références bibliographiques `refs.bib` que pour `bibtex`, on dispose d'un outil plus puissant et offrant plus de possibilités appelé `biblatex`, qui permet en particulier d'organiser la bibliographie de façon très souple à l'aide de mots-clefs, à ajouter dans le fichier `refs.bib` et à inclure en option dans la commande d'affichage de la bibliographie.

Tri par mots-clefs. Le fichier modèle de thèse LaTeX pour l'Université de Limoges propose d'organiser sa liste de travaux personnels selon trois rubriques :

```
  \begin{refsection}[mestravaux.bib]
    \nocite{*}
    \printbibliography[keyword={ConfInter},title={Conferences
internationales a comite de lecture}]
    \printbibliography[keyword={ConfNat},title={Conferences
nationales a comite de lecture}]
    \printbibliography[keyword={JourNat},title={Journaux nationaux a
comite de lecture}]
\end{refsection}
```

Noter la commande `\nocite{*}` qui permet de faire apparaître *toutes* les références concernées par les trois rubriques (c'est-à-dire celles auxquelles on a attribué l'un des trois mots-clefs dans le fichier `mestravaux.bib`), même si elles ne sont pas citées dans le texte.

La commande `\printbibliography`, qui affiche la bibliographie à l'endroit où elle est placée, accepte des options (entre crochets : `[ ... ]`). Ici des mots-clefs (un pour chaque instance de `\printbibliography`) permettent d'organiser la bibliographie en plusieurs rubriques — on peut aussi le faire en utilisant le *type* de référence (article, book,...) —, chacune se voyant attribuer un titre *ad-hoc*. Seules les références auxquelles on a attribué le mot-clef correspondant apparaitront dans chaque rubrique.

On peut attribuer plusieurs mots-clefs à une référence (noter le 's' du champ `keywords`, qu'on ajoutera dans le fichier de références bibliographiques aux références pour lesquelles il est utile) :

```
@online{CFEM,
author = {CFEM},
title = {Présentation Nationale de l'enseignement des mathématiques
en France au congrès ICME-14},
year = {2021},
url = {http://www.cfem.asso.fr/icmi/icme-14},
keywords = "livre, cfem"
}
```

Par contre on ne pourra appeler qu'un mot-clef à la fois au moment de faire afficher une bibliographie (noter l'absence de 's' de l'option `keyword` de la commande `\printbibliography` dans l'exemple ci-dessus). Si une référence doit apparaître dans plusieurs bibliographies définies par des mots-clefs, il faudra lui attribuer tous les mots-clefs correspondants.

Fonctionnement. Pour l'essentiel, `biblatex` fonctionne comme `bibtex` :

- dans le préambule, on fait apparaître les commandes :

```
\usepackage{biblatex}
\addbibresource{refs.bib}
```

si l'on a choisi le nom `refs.bib` pour le fichier de références bibliographiques. On peut également ajouter des options quand on appelle le package :

```
\usepackage[backend=biber,style=alphabetic,sorting=nyt]{biblatex}
```

avec ici un tri selon le nom, puis l'année, puis le titre ;

- on insère la commande `\printbibliography` là où l'on souhaite faire afficher la bibliographie, éventuellement avec les options choisies ;
- on compile le document avec `pdflatex` puis avec la commande `biber`, puis à nouveau avec `pdflatex`.

On peut se référer à la page d'aide d'*Overleaf* pour plus de détails : https://fr.overleaf.com/learn/latex/Bibliography_management_with_biblatex et consulter le fichier `source/biblio.tex` du modèle de thèse LaTeX de l'Université de Limoges pour voir comment utiliser plusieurs fichiers de références bibliographiques dans un même document (un fichier standard pour la bibliographie générale et un avec mots-clefs pour la liste des travaux personnels).

Chapitre 3

Création de diaporamas

C'est la classe `beamer` qui permet de produire des documents à projeter avec LaTeX, que nous appelerons ici des diapos. Elle est à indiquer dès la première ligne du document LaTeX, tout au début du préambule, sous la forme :

```
\documentclass{beamer}
```

Par défaut, les diapos sont en format paysage et les fontes de police utilisées sont beaucoup plus grosses que dans un document classique. La plupart des environnements disponibles pour LaTeX sont utilisables dans la classe `beamer`, pour laquelle existent de plus certains thèmes et environnements spécifiques.

Pour des détails sur les commandes ci-dessous, on pourra consulter avec profit la page [Wik] :

https://en.wikibooks.org/wiki/LaTeX/Presentations

3.1 Définition des diapos

Enivronnement `frame`. Le principal environnement spécifique à `beamer` est celui qui délimite le contenu de chaque diapo. Celui-ci doit être inséré entre les balises :

```
\begin{frame}{Titre de la diapo}
  ...
\end{frame}
```

Le *titre de la diapo* est optionnel, il peut aussi être indiqué via la commande `\frametitle{Titre de la diapo}` à insérer à l'intérieur de l'environnement `frame`.

Options de l'environnement `frame`. Attention à ne pas trop charger les diapos ! Pour une bonne lisibilité, il est en général préférable de répartir le contenu sur plusieurs diapos (ou le réduire au maximum) que de vouloir tout caser dans une seule diapo. On peut utiliser l'option `allowframebreaks` (entre crochets `[...]` après `\begin{frame}`) pour permettre à LaTeX de découper automatiquement un contenu trop long en plusieurs diapos.

On peut aussi, exceptionnellement, utiliser l'option `shrink` pour avoir la possibilité de faire apparaître plus de contenu sur une seule diapo. Enfin l'option `plain` permet d'enlever les contenus ajoutés automatiquement par LaTeX (plan du diaporama, logo...),

Diapo de présentation. L'option `plain` peut servir par exemple pour la page de titre, si on la code sous la forme :

```
\begin{frame}[plain]
  \titlepage
\end{frame}
```

Il y a aussi la possibilité, alternativement, d'utiliser la commande habituelle `\maketitle`, en dehors de l'environnement `frame`. Les informations : titre du diaporama, auteurs, affiliations... sont à insérer dans le préambule (voir [Wik] pour les commandes précises).

Insertion d'un logo. La commande `\logo` permet de définir un logo qui sera automatiquement placé sur toutes les diapos (en général en bas à droite). L'argument de cette commande peut être du texte ou une image, par exemple :

```
\logo{\includegraphics[width=2cm]{logo-unilim.png}}
```

Avec le package TikZ dont nous parlerons plus loin, il est possible de déplacer le logo en bas à gauche des diapos, en plaçant la commande suivante dans le préambule (après avoir chargé TikZ avec `\usepackage{tikz}`) :

```
\logo{
  \begin{tikzpicture}[overlay, remember picture]
    \node[right=0.2cm] at (current page.210){
      \includegraphics[width=2cm]{logo-unilim.png}
    };
  \end{tikzpicture}
}
```

Ce codage fonctionne plus ou moins bien selon le thème choisi (voir ci-dessous).

Table des matières. Les commandes de structuration du texte en (sous-(sous-))sections peuvent être placées à l'extérieur de l'environnement `frame`, elles permettent de créer la table des matières du diaporama. Celle-ci peut être affichée dans une diapo :

```
\begin{frame}
  \frametitle{Table des matières}
  \tableofcontents[currentsection]
\end{frame}
```

L'option `currentsection` fait ressortir la section courante. Cet affichage peut être automatisé à chaque début de section en plaçant le code précédent dans le préambule, entre les accolades de la commande `\AtBeginSection[]{...}`.

Selon le thème choisi, les titres de (sous-(sous-))sections sont aussi affichés dans un menu qui apparaît sur chaque diapo à un endroit fixe (par exemple en haut).

Choix des thèmes. La classe `beamer` vient avec un ensemble de thèmes préconçus qui gèrent la structure des diapos (positionnement et présentation des menus pour la table des matières en particulier), avec différents choix de couleurs. Voici un exemple de choix de thème et de couleur (à insérer dans le préambule) :

```
\usetheme{Singapore}
\usecolortheme{orchid}
```

Les autres thèmes préconçus et couleurs disponibles sont listés dans [Wik].

Il est possible également de définir ses propres thèmes et de choisir les couleurs pour les différents types d'objets à afficher (voir [Wik]).

3.2 Autres commandes spécifiques

Mise en valeur. L'environnement `block` permet de mettre en valeur un contenu important :

```
\begin{block}{À retenir !}
  \textit{Texte à mettre en valeur.}
\end{block}
```

Noter le « titre » du bloc qui vient en option entre accolades juste après la commande `\begin{block}`. Avec les choix de thème et de couleur ci-dessus, on obtient :

Cet environnement a deux variantes : `alertblock` et `exampleblock`.

Mise en colonnes. On peut faire afficher le contenu d'une diapo sur deux colonnes, on copie ici un exemple de [Wik] où on trouvera plus de détails.

```
\begin{frame}
  \frametitle{Exemple de mise en colonnes}
  \begin{columns}[c]
    \column{.45\textwidth}
    Contenu de la première colonne
    \column{.45\textwidth}
    Contentu réparti \newline sur deux lignes
  \end{columns}
\end{frame}
```

Dans cet exemple, chacune des deux colonnes est décrite par la commande `column` suivie d'une option indiquant sa largeur (par rapport à la largeur de la zone de texte de la diapo), suivie du contenu.

Alternativement, on peut placer le contenu de chaque colonne dans un environnement propre :

```
\begin{column}{5cm}
  Contenu réparti \newline sur deux lignes
\end{column}
```

à insérer à l'intérieur de l'environnement `columns` (avec un `s`!). Il est prudent de ne pas mélanger les deux syntaxes...

Animations. Il peut être (parfois) utile de faire apparaître le contenu d'une diapo petit à petit, morceau par morceau. La classe `beamer` offre plusieurs outils pour cela : `\pause`, `\only` , `\uncover`. La page déjà citée de [Wik] donne tous les détails, notons tout de même que l'utilisation de `\pause` produit une diapo à deux versions : dans un premier temps, seul le contenu antérieur à `\pause` est affiché, dans un second temps le reste également (du moins jusqu'à la commande du même type suivante). Concernant les deux autres, voici un exemple :

```
\begin{frame}
  \frametitle{Animations}
  \only<1>{texte 1}
  \uncover<2-3>{texte 2}
  \uncover<4->{texte 3}
\end{frame}
```

Ici la diapo *Animations* sera affichée sous 4 formes différentes : d'abord avec le texte 1, qui disparaît ensuite et est remplacé (au même endroit) par le texte 2 pour les deux affichages suivants (identiques) ; enfin le texte 2 s'efface et le texte 3 apparaît pour le dernier affichage à la suite de l'emplacement occupé précédemment par le texte 2, qui bien qu'absent n'est pas *recouvert*.

Le même fonctionnement peut être obtenu en passant les mêmes arguments `<1>`, `<2-3>`... aux commandes `\item` d'une liste créée avec l'environnement `itemize` ou avec `enumerate`. Si l'on veut simplement que les items de la liste apparaissent l'un après l'autre, on peut plus simplement utiliser `\begin{itemize}[<+->]`, voire `\begin{itemize}[<+-| alert@+>]`.

Enfin, des animations beaucoup plus sophistiquées sont possibles, on pourra consulter le diaporama [Riv16] pour une présentation.

Inclusion de vidéos. Il existe (au moins) deux manières d'inclure une vidéo dans un diaporama produit avec `beamer`. On peut faire lire la vidéo par le lecteur vidéo installé sur l'ordinateur ou, si celui-ci est compatible, directement par le visionneur pdf qui affiche le diaporama. Dans ce dernier cas, il faut ajouter la commande `\usepackage{multimedia}` dans le préambule.

```
\begin{frame}{Video qui s'ouvre dans le pdf}
  \centering
```

```
    \movie[width=9cm,height=6cm,poster]{image-video.png}{exemple-
video.mp4}
  \end{frame}

  \begin{frame}{Video qui s'ouvre dans le lecteur de l'ordinateur}
    \begin{center}
      \href{run:exemple-video.mp4}{
        \includegraphics[scale=0.25]{image-video.png}}
    \end{center}
  \end{frame}
```

Dans les deux exemples, le nom du fichier vidéo est `exemple-video.mp4` et la vidéo est représentée dans le diaporama, avant son ouverture, par une image nommée `image-video.png` (ici, elle est extraite de la vidéo). Noter la syntaxe différente avec l'utilisation de la commande **`\centering`** dans un cas, de l'environnement **`\begin{centering}`** dans l'autre.

Chapitre 4

Création d'images avec TikZ

Même si LaTeX n'est pas conçu initialement pour créer des dessins et illustrations, il existe de nombreuses solutions pour lui en faire produire. La solution offerte par TikZ est à la fois facile à utiliser et extrêmement riche et puissante, nous allons en découvrir les bases ci-dessous. Ce package est une surcouche d'un ensemble de commandes « de bas niveau » (c'est-à-dire peu accessibles pour l'utilisateur lambda) nommé `pgf`. Pour certains besoins spécifiques, comme le traitement de données issues d'expériences, nous verrons plus loin l'apport d'un autre package basé lui aussi sur `pgf` et appelé `pgfplots`.

4.1 Premiers pas

Le package TikZ est appelé dans le préambule par la commande `\usepackage{tikz}`, qu'il faut faire précéder du chargement du package `xcolor` :

```
\usepackage[usenames,dvipsnames]{xcolor}
\usepackage{tikz}
```

Cela permet d'utiliser des couleurs dans les figures produites par TikZ. Dans le corps du document, l'environnement est introduit classiquement par les balises :

```
\begin{tikzpicture}
   ...
\end{tikzpicture}
```

(on peut également utiliser le raccourci `\tikz{...}`). Cet environnement crée dans la page un espace où le dessin va être inséré, de taille adaptée à celui-ci. Il crée également des systèmes de coordonnées (cartésiennes et polaires) qui permettent de se repérer au sein de cet espace.

4.1.1 Relier des points

Les coordonnées cartésiennes s'expriment sous la forme d'un couple de nombres (*abscisse*,*ordonnée*) séparés par une virgule (les valeurs décimales sont à entrer avec un point). La commande permettant de tracer un trait entre deux ou plusieurs points est `\draw` avec le séparateur `--` entre les points, le tout terminé par un point-virgule `;` :

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

produit le dessin :  , qui ici est inséré dans le texte. Une façon simple de le sortir du texte est d'inclure l'ensemble dans l'environnement `\begin{center} ... \end{center}`, on en verra des exemples plus bas.

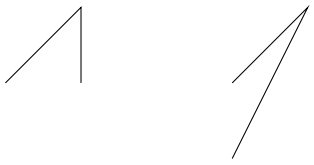
Ne pas oublier le point-virgule `;` qui termine la commande `\draw`, ainsi que (quasiment) toutes les commandes spécifiques à TikZ, en particulier lorsqu'il y a plusieurs commandes TikZ à la suite. Par contre, si on insère des commandes LaTeX dans l'environnement `tikzpicture`, on écrit celles-ci comme dans n'importe quelle partie du document, c'est-à-dire sans point-virgule (on en verra des exemples plus loin).

Unité. Par défaut l'unité de longueur est le centimètre (`cm`) ; TikZ reconnaît les unités de longueur de LaTeX, par exemple le millimètre (`mm`), le point (`pt`), la « hauteur de x » (`ex`), le cadratin (`em`) :

$$1\,\texttt{pt} = 0{,}35114\,\texttt{mm} : \blacksquare,\ 1\,\texttt{ex} \approx 4{,}2 \text{ à } 5{,}5\,\texttt{pt} : \blacksquare,\ 1\,\texttt{em} = 10\,\texttt{pt} : \blacksquare,\ 1\,\texttt{cm} : \blacksquare$$

où la hauteur de x et le cadratin dépendent de la police utilisée (et sont donnés pour une police de 10 `pt`).

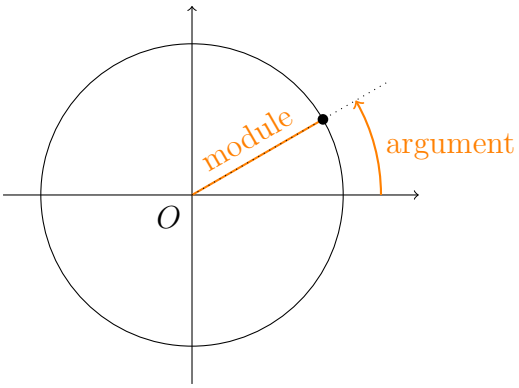
Coordonnées relatives. Les coordonnées peuvent aussi être exprimées de manière relative : `\draw (0,0) -- ++(1,1) -- ++(0,-1);` donne le même résultat que ci-dessus (figure de gauche) :



Le double `+` indique que le nouveau point sert de références pour les coordonnées relatives suivantes, ce qui ne serait pas le cas avec un seul `+` (figure de droite, encodée par `\draw (0,0) -- +(1,1) -- +(0,-1);`).

Nommage des points. Pour des figures plus complexes dans lesquelles un même point est utilisé plusieurs fois, par exemple l'origine $(0,0)$, on peut le nommer (en début de codage) avec la commande `\coordinate (O) at (0,0);` et l'appeler ensuite par `\draw (O) -- ++(1,1) -- ++(0,-1);`.

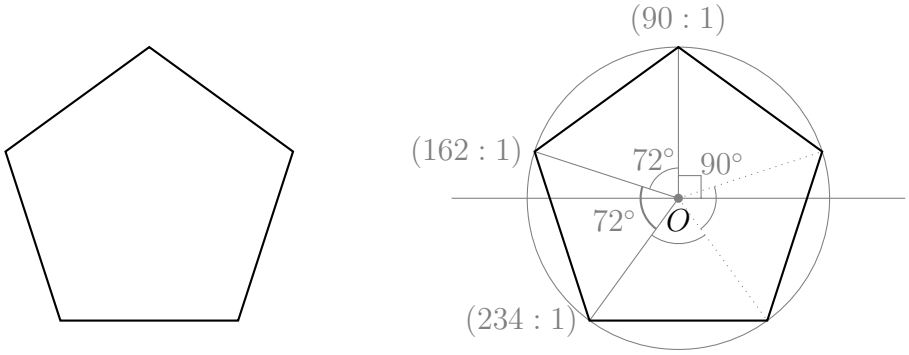
Coordonnées polaires. Elles s'écrivent sous la forme d'un couple de nombres (*argument*:*module*) séparés par « : », où le *module* est la distance du point à l'origine et l'*argument* est la mesure de l'angle (en degrés) que fait la droite qui le relie à l'origine avec l'axe horizontal :



Le point sur le cercle est tracé par l'instruction `\fill (30:2) circle (2pt);` qui trace un cercle de rayon 2 `pt` centré au point de coordonnées polaires (30 : 2) et le remplit.

Les coordonnées polaires peuvent être très pratiques dans certains cas, par exemple si on veut tracer un pentagone régulier :

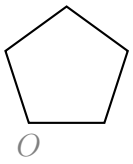
```
\begin{tikzpicture}[scale=2]
  \draw[thick] (90:1) -- (162:1) -- (234:1) -- (306:1) -- (378:1) --
cycle;
\end{tikzpicture}
```



Noter que le chemin se termine par le mot `cycle` qui indique qu'il faut « boucler la boucle », c'est-à-dire revenir au point de départ du chemin, ici (90 : 1).

Il est facile également de réaliser un pentagone étoilé : `\draw (90:1) -- (234:1) -- (18:1) -- (162:1) -- (306:1) -- cycle;`

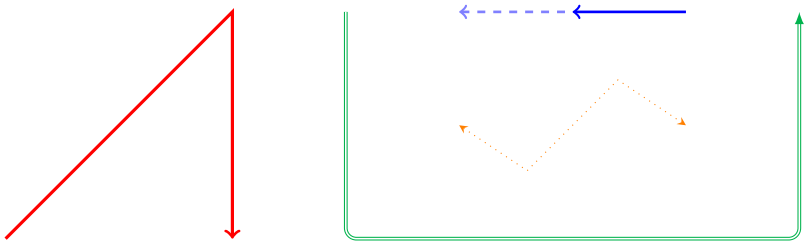
Les coordonnées polaires peuvent aussi s'exprimer de manière relative et se mélanger avec des coordonnées cartésiennes : `\draw (0) -- ++(1,0) -- ++(72:1) -- ++(144:1) -- ++(216:1) -- cycle;`



4.1.2 Un peu de style

Les commandes TikZ, en particulier la commande `\draw`, peuvent recevoir de très nombreuses options (entre crochets `[ ... ]`) pour enrichir les dessins en leur ajoutant des flèches, en modifiant l'épaisseur des traits, leur style, leur couleur...

```
\begin{center}
\begin{tikzpicture}[scale=3]
  \coordinate (O) at (0,0);
  \draw[->,very thick, red] (O) -- (1,1)  -- (1,0);
  \draw[<-,line width=1pt,dashed,blue!50] (2,1)  -- (2.5,1);
  \draw[<-,line width=1pt,blue] (2.5,1) -- (3,1);
  \draw[<->,>=stealth,dotted,orange] (2,0.5) -- (2.3,0.3) -- (2.7,.7)
 -- (3,0.5);
  \draw[->,>=latex,rounded corners,double,green!70!blue] (1.5,1) --
(1.5,0) -- (3.5,0) -- (3.5,1);
\end{tikzpicture}
\end{center}
```



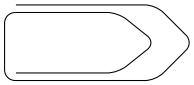
Échelle. L'option `scale=3` de `\begin{tikzpicture}` agrandit l'*ensemble* du dessin d'un facteur 3. Elle peut également être attribuée à la commande `\draw`, auquel cas seul le dessin concerné par l'instruction est agrandi.

Flèches. Les options `->`, `<-` et `<->` de `\draw` ajoutent des flèches en début et/ou fin de chemin (dans le sens du parcours) et les options `>=stealth` et `>=latex` modifient la ou les pointe(s) de la flèche.

Épaisseur. Les options `very thick` et `line width=1pt` augmentent l'épaisseur des traits des deux premiers chemins, de façon qualitative ou quantitative. Le contraire de `thick` est `thin` et on dispose aussi de `ultra` (suivi de `thick` ou `thin`).

Styles de traits. Les options `dashed`, `dotted` et `double` créent des traits en « traitillés », en pointillés ou des traits doubles (cela ne se voit pas sur l'exemple mais l'option `double` laisse un blanc opaque entre les deux traits parallèles).

Coins arrondis. L'option `rounded corners` crée des coins arrondis :



On peut régler l'arrondi en spécifiant une valeur : `rounded corners=10pt`.

Couleurs. Le package `xcolor` utilisé par TikZ permet d'utiliser les couleurs prédéfinies, comme `red` (c'est un raccourci pour `color=red`), `blue`, `orange`... et de les modifier : `blue!50` utilise le bleu à 50% (dilué avec du blanc) et `green!70!blue` crée un mélange de vert (à 70%) et de bleu (à 30%).

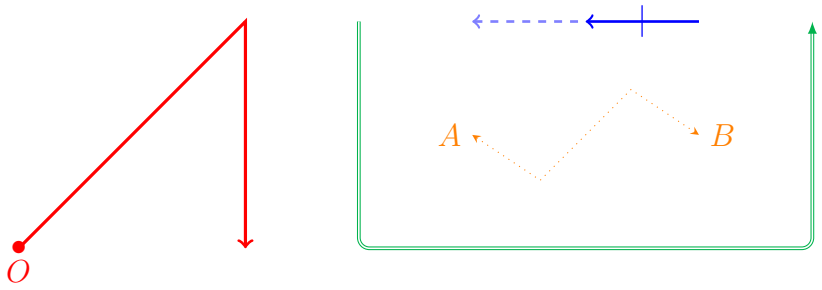
Pour les couleurs prédéfinies, on dispose de 19 couleurs de base et, en ajoutant les options correspondantes à la commande `\usepackage{xcolor}` dans le préambule, de 68 couleurs supplémentaires avec `dvipsnames`, 150 avec `svgnames` et 317 avec `x11names` (voir le manuel de `xcolor` pp. 42-44 [1]). Si cela ne suffit pas, on peut définir ses propres couleurs dans le système rgb (voir l'aide-mémoire LaTeX, p. 18).

Opacité. On peut également régler l'opacité (et donc la transparence), en donnant une valeur entre 0 et 1 à l'option `opacity` (qui vaut 1 par défaut : opacité totale). Comparons une figure comportant un trait vertical par dessus lequel on trace un rectangle rouge avec opacité totale et opacité à 50% (avec la commande `\fill[red,opacity=0.5]`) :

On peut spécifier séparément l'opacité pour les traits et pour les pleins avec les options `draw opacity` et `fill opacity`.

4.1.3 Un peu de texte : les nœuds

Ajoutons quelques annotations au dessin précédent : un point à l'origine, avec la lettre O en dessous, une barre verticale au milieu de la flèche bleue, des lettres A et B aux extrêmités de la flèche orange.



1. Accessible en ligne : https://texlive.mycozy.space/macros/latex/contrib/xcolor/xcolor.pdf

Pour cela voici comment les instructions précédentes ont été modifiées (on a enlevé les options de style pour alléger le codage) :

```
  \draw (O) node{$\bullet$} node[below]{$O$} -- (1,1)  -- (1,0);
  \draw (2.5,1) -- (3,1) node[midway]{$\mid$};
  \draw (2,0.5) node[left]{$A$} -- (2.3,0.3) -- (2.7,.7) -- (3,0.5)
node[right]{$B$};
```

Le mot-clé `node` crée un *nœud* relatif à un chemin :

```
(2.5,1) -- (3,1) node[midway]{$\mid$}
```

ou à un point :

```
                (O) node{$\bullet$} node[below]{$O$}
(2,0.5) node[left]{$A$} ,    (3,0.5) node[right]{$B$}
```

selon l'endroit où il apparaît et l'option qui lui donnée.

Nœud de chemin. Dans le premier cas, il apparaît à la fin d'un chemin avec l'option `midway` , l'argument `$\mid$` (une barre verticale) est écrit à mi-chemin de celui-ci. Les autres options possibles sont (`very`) `near start` ou `near end`, on peut les combiner (en séparant à l'aide d'une virgule) avec `below`, `above`... ainsi qu'avec `sloped`, qui indique que le contenu du nœud suit l'orientation du trait. Ainsi `\draw (0,0) -- (1,1) node[near start,sloped] {$\mid $};` produit :

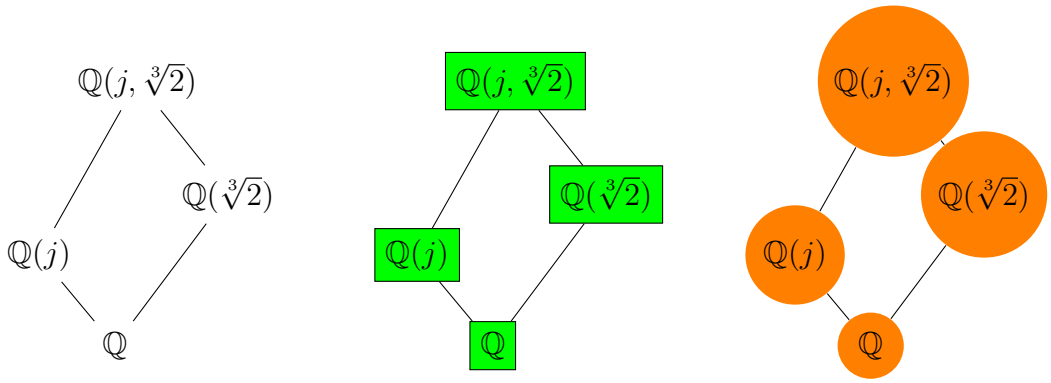
Nœud de point. Dans le deuxième cas, deux nœuds sont créés pour le point `(O)` : l'un, sans option, pour placer une boule (avec la commande mathématique `\bullet`) à la position du point O, l'autre avec l'option `below` écrit l'argument `$O$` en dessous du point. De même pour le point `(2,0.5)` avec l'option `left`, donc l'argument `$A$` est écrit à gauche du point. Le dernier nœud créé apparaît à la fin d'un chemin ; cependant son option `right` indique qu'il porte sur le dernier point de ce chemin, et l'argument `$B$` est placé à droite de celui-ci.

On peut utiliser la commande `\draw` pour écrire du texte sans dessiner de trait : `\draw ( O) node{$\bullet$};` place une boule à la position du point O.

Chemin de nœuds. Il est également possible de définir des nœuds pour eux-mêmes avec la commande `\node` (noter l'antislash qui précède, contrairement à ci-dessus), puis de les relier par des traits qui, ainsi, ne passent pas sur les textes écrits dans les nœuds :

```
\begin{tikzpicture}
  \node (Q1) at (0,0) {$\mathbb Q$};
  \node (Q2) at (-1,1.2) {$\mathbb Q(j)$};
  \node (Q3) at (1.5,2) {$\mathbb Q(\sqrt[3]{2})$};
  \node (Q6) at (0.3,3.5) {$\mathbb Q(j,\sqrt[3]{2})$};
```

```
  \draw (Q1) -- (Q2) -- (Q6) -- (Q3) -- (Q1);
\end{tikzpicture}
```



Chaque nœud produit une petite boîte autour de son texte, qu'on peut faire apparaître (figure au centre) et qu'on peut colorer ; elle est de forme rectangulaire par défaut, mais on peut choisir une forme circulaire comme sur la figure de droite. À chaque fois, on a ajouté une instruction en début de figure :

```
  \tikzstyle{every node}=[draw,fill=green];
```

```
  \tikzstyle{every node}=[shape=circle,fill=orange];
```

respectivement pour la figure du centre et pour celle de droite. On aurait pu à la place indiquer ces choix en option des nœuds si on avait voulu qu'ils ne s'appliquent qu'à certains d'entre eux.

Un nœud particulier : repérage dans la page courante. On peut se repérer dans la page courante grâce au nœud nommé `current page` par TikZ, en utilisant les options `remember picture` et `overlay` de l'environnement `tikzpicture`, qui permettent de garder en mémoire les positions calculées par TikZ après la réalisation de la figure (elles sont normalement effacées).

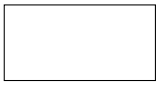
```
\begin{tikzpicture}[remember picture,overlay]
  \node [rotate=60,scale=10,text opacity=0.3]
  at (current page.center) {Ti\textit{\color{orange}k}Z};
\end{tikzpicture}
```

Il faut effectuer deux compilations pour que LaTeX détermine l'emplacement exact.

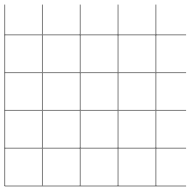
4.1.4 D'autres chemins

À la place de `\draw (0,0) -- (1,1);` qui trace un trait, on peut dessiner :

- un rectangle : `\draw (1,0) rectangle (3,1);`



- une grille : `\draw[step=.5,gray,very thin] (-1,-1) grid (1.4,1.4);`



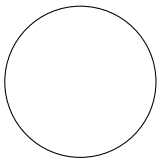
- un chemin fermé : `\draw[line width=3pt] (0,0) -- (2,0) -- (1,1) -- cycle;`

Noter la différence avec une fermeture « à la main » (figure de droite, codée `\draw[line width=3pt] (0,0) -- (2,0) -- (1,1) -- (0,0);`)

- dont on peut colorier l'intérieur : `\fill[color=gray!20] (0,0) -- (2,0) -- (1,1) -- cycle;`

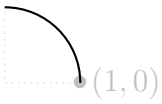
ou même les deux ensembles (figure de droite) : `\filldraw [fill=gray!20,draw=orange, thick] (0,0) -- (2,0) -- (1,1) -- cycle;`

- le cercle centré en $(0,0)$ de rayon 1 : `\draw (0,0) circle (1);`

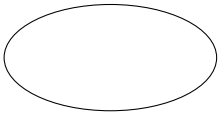


- un disque : `\fill[color=blue] (0,0) circle (0.5);`

- un arc de cercle : `\draw[thick] (1,0) arc (0:90:1);` trace l'arc de cercle commençant en $(1,0)$ et se trouvant sur le cercle de rayon 1 tel que l'arc est délimité par les rayons faisant des angles de mesures 0° et 90° avec l'horizontale passant par le centre du cercle :



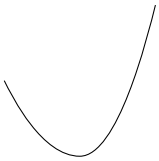
- une ellipse : `\draw (0,0) ellipse [x radius=40pt, y radius=20pt];`



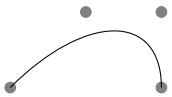
(On obtiendrait le même résultat en écrivant `circle` à la place de `ellipse`.)

- une parabole avec tangente horizontale au point de départ : `\draw (0,0) parabola (2,1);`

ou avec tangente horizontale à un point spécifié : `\draw (0,0) parabola bend (1,-1) (2,1);`

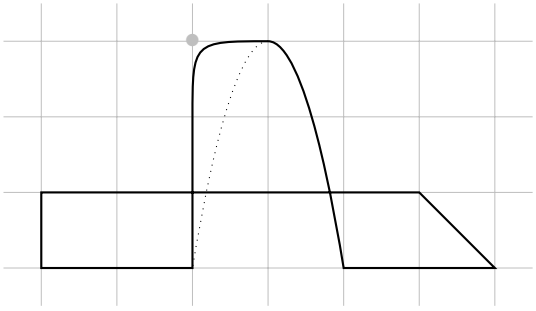


- ...
- Plus généralement, on peut créer des courbes à un ou deux points de contrôle (qui sont ici représentés en gris) :



```
\begin{tikzpicture}
  \filldraw [gray] (0,0) circle [radius=2pt]
                   (1,1) circle [radius=2pt]
                   (2,1) circle [radius=2pt]
                   (2,0) circle [radius=2pt];
  \draw (0,0) .. controls (1,1) and (2,1) .. (2,0);
\end{tikzpicture}
```

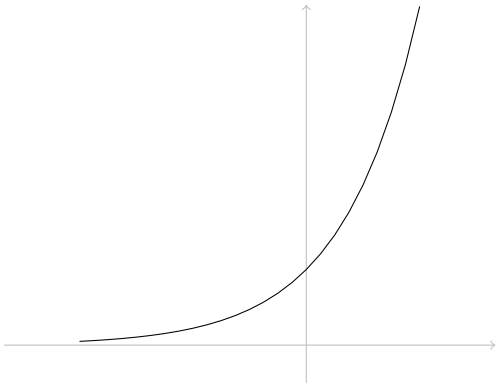
Enfin, on peut combiner des chemins entre eux :



La grille et le point de contrôle sont représentés en gris léger (à 50%) et on a ajouté au chemin principal la partie « montante » de la parabole en pointillés :

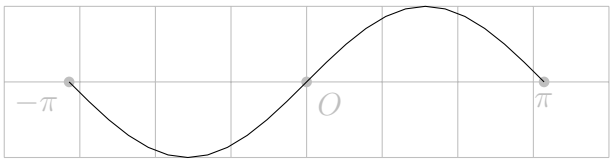
```
\begin{tikzpicture}
  \draw[very thin,gray!50] (-.5,-.5) grid (6.5,3.5);
  \draw[thick] (0,0) rectangle (2,1) -- (2,2) .. controls (2,3) ..
(3,3) parabola (4,0) -- ++(2,0) -- ++(-1,1) -- cycle;
  \draw[dotted] (3,3) parabola (2,0);
  \draw[gray!50] (2,3) node{$\bullet$};
\end{tikzpicture}
```

Tracé de graphes de fonctions. La commande `\draw [domain=-3:1.5] plot(\x,{exp(\x)});` produit le tracé du graphe de la fonction exponentielle entre les valeurs -3 et 1,5 de la variable (ici `\x`) :



Les axes en gris clair ont été ajoutés à la main : `\draw[lightgray,->] (-4,0) -- (2.5,0);` pour l'axe des abscisses.

Si on n'indique pas de valeurs pour le domaine, les valeurs par défaut sont -5 et 5. L'*opérateur de chemin* `plot` peut servir à tracer le graphe de nombreuses fonctions (voir TikZ pour l'impatient, §3.1.2), notons que par défaut l'argument des fonctions trigonométriques est considéré être une mesure d'angle en degrés, il faut donc indiquer que ce sont des radians le cas échéant. La figure :



est obtenue par les instructions :

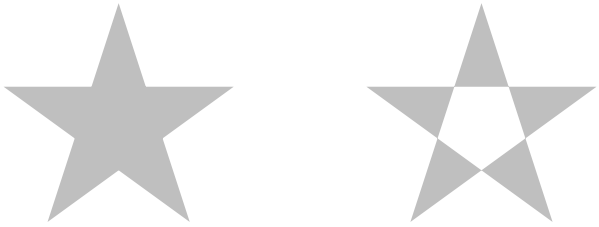
```
\draw[lightgray,thin] (-4,-1) grid (4,1);
\fill[lightgray] (0,0) node[below right]{$0$} circle (2pt);
\fill[lightgray] (pi,0) node[below]{$\pi$} circle (2pt);
\fill[lightgray] (-pi,0) node[below left]{$-\pi$} circle (2pt);
\draw [domain=-pi:pi] plot (\x,{sin(\x r)});
```

On a rajouté ici une grille et trois points pour aider au repérage. On verra dans le chapitre 5 comment automatiser le tracé de grilles et d'axes.

Enfin `plot` peut être utilisé à l'intérieur d'un chemin : est produit par

```
\fill[color=gray!20] (1,0) -- (1,1) -- plot [domain=1:2] (\x,1/\x) --
 (2,0) -- cycle;
```

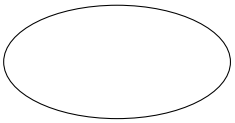
Remplissage de figures complexes. La commande `\fill` colore l'intérieur d'une figure, qui n'est pas toujours évident à déterminer, par exemple dans le cas d'un polygone étoilé ☆. Voici le remplissage par défaut (sans option) et celui avec l'option `even odd rule` :



4.2 Transformations, découpage, groupements

Signalons ici quelques unes des très nombreuses fonctionnalités supplémentaires offertes par TikZ, parmi les plus utiles.

Changements d'échelle. On a déjà vu comment « changer d'échelle » avec l'option `scale` , appliquée à toute la figure `\begin{tikzpicture}[scale=2]`. Elle s'applique également à de nombreux autres objets créés par TikZ : chemin (ou partie de chemin) `\draw[scale=2]`, nœud... On peut appliquer des changements d'échelles différents sur les deux axes avec `xscale` et `yscale`, ce qui donne un autre moyen de tracer une ellipse : `\draw[xscale=2] (0,0) circle (.75);`

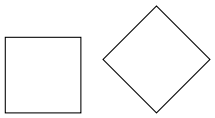


L'option `scale` peut être utilisée dans un nœud pour agrandir ou rapetisser le texte qu'il contient :

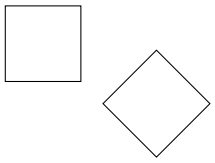
```
\draw (0,0) -- (4,0) node[near start,above,scale=2]{début} node[very
near end,below,scale=.5]{fin};
```

Translations et rotations. Il existe d'autres transformations, dont les principales sont `xshift` et `yshift` pour un déplacement horizontal ou vertical et `rotate` pour une rotation :

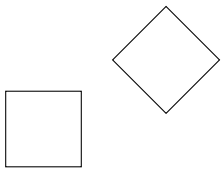
```
\draw (0,0) rectangle (1,1)
      [xshift=2cm,rotate=45] (0,0) rectangle (1,1);
```



Une translation oblique peut être indiquée sous la forme `shift{(2,-1)}` :

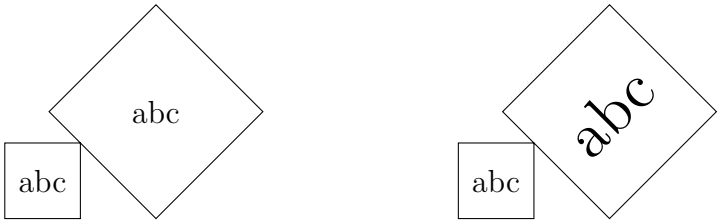


Attention à l'ordre des transformations ! Voici ce que donne l'option `[rotate=45,shift={(2,-1)}]` :



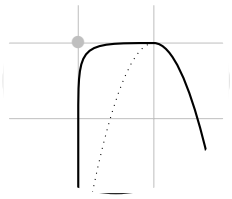
Texte et transformations. Rajoutons du texte dans les exemples ci-dessus pour voir que, si les translations appliquées à la figure ou à un chemin s'appliquent bien à celui-ci (le texte « suit » bien le nœud qui le contient), il n'en va pas de même des rotations : le texte ne « tourne » pas, ni du changement d'échelle. Le code suivant produit la figure de gauche ci-dessous :

```
\draw (0,0) rectangle (1,1) node[midway]{abc}
      [xshift=2cm,rotate=45,scale=2]
      (0,0) rectangle (1,1) node[midway]{abc};
```



On a ajouté l'option supplémentaire `transform shape` pour la figure de droite (dans les options de la deuxième partie du chemin).

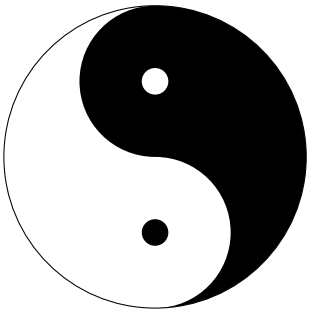
Découper une figure. La commande `\clip` permet de n'afficher qu'une partie d'une figure, délimitée par un chemin qu'on spécifie comme lorsqu'on souhaite le tracer : ajouter l'instruction `\clip (2.5,2.5) circle (1.5);` au début de l'encodage de la figure « combinée » ci-dessus donne :



La commande `\clip` n'agit que sur ce qui la suit, ce qui permet par exemple de créer un disque à demi-rempli ◑ :

```
\begin{tikzpicture}[scale=.25]
  \draw (0,0) circle (1);
  \clip (0,-1) rectangle (1,1);
  \fill (0,0) circle (1);
\end{tikzpicture}
```

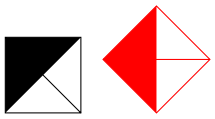
puis de le compléter en ajoutant 4 disques (bien placés dans le code !) pour obtenir le symbole du yin et du yang :



On peut aussi « clipper » en suivant le contour de la partie noire : `\clip (0,0) arc (270:90:1) arc (90:-90:2) arc (-90:90:1);`, il ne reste alors que les deux points à ajouter. Noter que `arc (270:90:1)` et `arc (-90:90:1);` créent des arcs différents !

Regrouper des instructions. Il peut être utile de créer des regroupements d'instructions à l'intérieur d'une figure, que ce soit pour y appliquer un certain style (qu'on ne veut pas appliquer à toute la figure) ou certaines transformations. C'est le sous-environnement `\begin{scope} ... \end{scope}` qui permet cela. De plus, les commandes `\clip` contenues dans un sous-environnement `scope` voient leur portée limitée à celui-ci.

```
\newcommand{\logo}{\draw (0,0) rectangle (1,1) (1,0) -- (0,1); \fill
(0,0) -- (0,1) -- (1,1) -- cycle;}
\begin{tikzpicture}
  \logo
  \begin{scope}[xshift=2cm,rotate=45,red]
    \logo
  \end{scope}
\end{tikzpicture}
```



Noter ici l'utilisation d'une *macro* LaTeX, définie par la commande **\newcommand** suivie du nom `\logo` et de la définition (le texte qui remplacera `\logo` à chacune de ses occurrences), elle peut être placée avant la figure ou même à l'intérieur de celle-ci dans l'environnement `tikzpicture` (auquel cas elle ne sera pas définie en dehors). La macro `\logo` n'est pas suivie d'un point-virgule lors de son utilisation dans `tikzpicture` car c'est une commande LaTeX (il est présent par contre à la fin du texte de remplacement).

4.3 Les boucles

On dispose dans TikZ d'une commande `\foreach` qui permet d'automatiser certains tracés. Elle est utilisable dès lors que le package TikZ est chargé, partout dans le document LaTeX, y compris en dehors d'un environnement TikZ. Ainsi

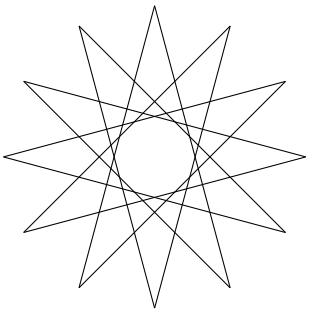
```
\foreach \x in {1,2,3} {$x=\x$, }
```

écrit : $x = 1$, $x = 2$, $x = 3$,

Noter l'absence de point-virgule à la fin de l'instruction (puisqu'elle n'est pas spécifique à l'environnement TikZ). La boucle `foreach` permet d'introduire une variable, ici `\x`, l'ensemble des valeurs qu'elle doit prendre, ici `{1,2,3}`, et l'instruction ou la suite d'instructions, entre accolades, à effectuer pour chaque valeur de `\x`, ici l'écriture du texte `$x=\x$, `.

Pour tracer le polygone étoilé à 12 côtés qui va de 5 en 5 (lorsqu'on considère les sommets disposés sur un cercle, ici de rayon 1), on peut faire :

```
\newcommand{\co}{12}\newcommand{\sa}{5}
\foreach \n in {1,2,...,\co}{\draw ({\n/\co*360*\sa}:1) -- ({(\n+1)/
\co*360*\sa}:1);}
```

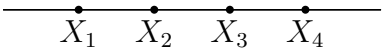


On a utilisé deux macros LaTeX pour stocker le nombre de côtés (`\co`) et la valeur du « saut » (`\sa`) en début de codage, de façon à pouvoir les modifier facilement. La boucle `foreach` introduit la variable `\n`, son ensemble de valeurs `{1,2,...,\co}`, c'est-à-dire toutes les valeurs entières entre 1 et 12 (puisque `\co` vaut 12), et l'instruction à effectuer pour chaque valeur de

\n, le tracé d'un trait reliant les sommets numéros \n et \n+1 (numéros dans le polygone étoilé cette fois).

La boucle permet aussi de définir des points (voir [MS07, Fig. 26], où la syntaxe est légèrement différente mais équivalente à celle que nous utilisons), ce qui permet de tracer le polygone étoilé d'une autre façon :

```
\foreach \i in {1,...,4}
         {
           \coordinate (X\i) at (\i,0);
           \draw (0,0) -- (5,0);
           \fill (X\i) node[below]{$X_\i$} circle (1.5pt);
         }
```

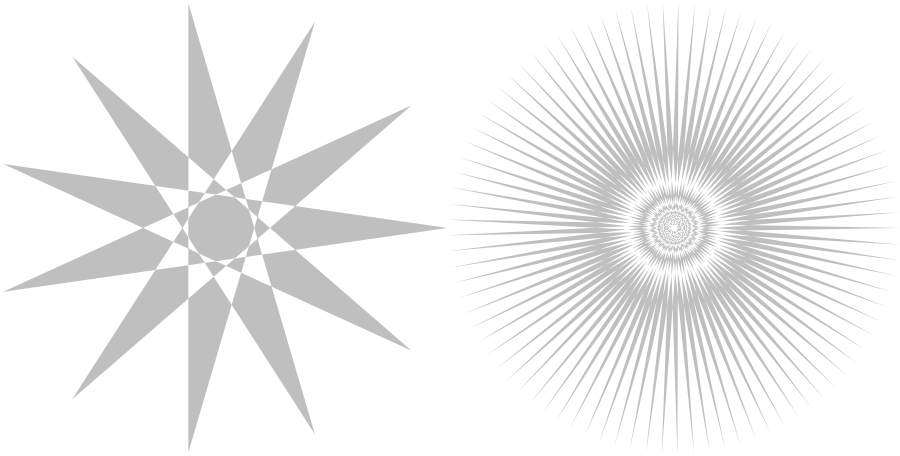


Boucle de chemin. Cette boucle est également disponible à l'intérieur d'un chemin, ce qui permet de simplifier le codage précédent en :

```
\draw (0:1) foreach \n in {1,...,\co} {-- ({\n/\co*360*\sa}:1)};
```

Cette syntaxe simplifie le codage dans ce cas, noter que foreach apparaît cette fois sans anti-slash (c'est maintenant une *opération de chemin*). On peut en profiter pour colorier l'intérieur des polygones étoilés imbriqués :

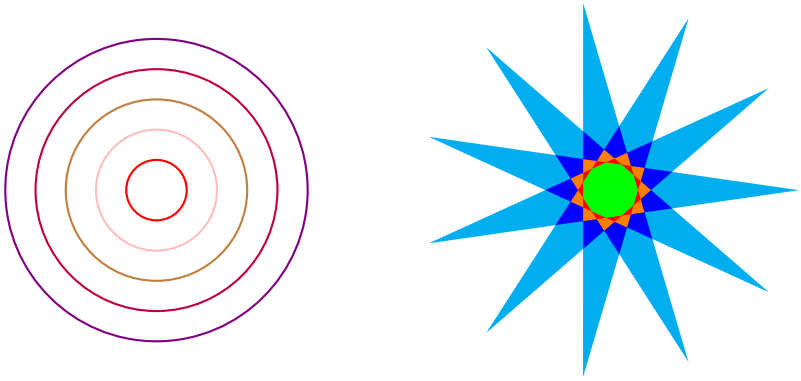
```
\newcommand{\co}{11}\newcommand{\sa}{5}
\fill[lightgray,even odd rule] (0:1) foreach \n in {1,2,...,\co} {--
({\n/\co*360*\sa}:1)};
\renewcommand{\co}{89}\renewcommand{\sa}{44}
\fill[lightgray,even odd rule,xshift=2cm] (0:1) foreach \n in
{1,2,...,\co} {-- ({\n/\co*360*\sa}:1)};
```



La commande LaTeX **\renewcommand** permet d'affecter de nouvelles valeurs aux macros \co et \sa pour la 2e figure (celle de droite, dont on pourra noter l'extrême précision en zoomant dans le visionneur de pdf).

Boucle à plusieurs variables (liées). La boucle `foreach` peut gérer plusieurs variables liées, sous la forme `\n/\m in {1/a,2/b,3/c}`, ce qui signifie que `\m` prend la valeur `a` quand `\n` vaut `1`, `\m` prend la valeur `b` quand `\n` vaut `2` et `\m` prend la valeur `c` quand `\n` vaut `3`. Par exemple (figure de gauche) :

```
\foreach \r/\c in {1/red,2/pink,3/brown,4/purple,5/violet} \draw[
thick,\c] (0,0) circle (\r mm);
```



Utilisons cette possibilité pour colorier l'hendécagone emboîté avec des couleurs différentes pour chaque sous-polygone (figure de droite ci-dessus). On fait une boucle sur les différentes valeurs du saut, descendant de la valeur maximale (ici 5) jusqu'à 1, en liant une couleur à chacune comme ci-dessus, avec la commande

```
\foreach \sb/\couleur [evaluate=\sb] in {\sa/cyan,\sa-1/blue,\sa-2/
orange,\sa-3/red,\sa-4/green}
```

Le plus grand polygone (valeur du saut 5) sera donc de couleur `cyan`, le suivant (valeur du saut 4) de couleur `blue`... jusqu'au plus petit (valeur du saut 1) de couleur `green`. Noter l'option `[evaluate=\sb]` qui assure que LaTeX calcule effectivement les différences `\sa-1`..., `\sa-4`, de sorte que la variable `\sb` prend des valeurs numériques.

À chaque étape i, il faut calculer le rayon R_{s-i} du nouveau polygone (valeur du saut $s-i$), avec la formule

$$R_{s-i} = \cos\left(\frac{180s}{N}\right) \Big/ \cos\left(\frac{180(s-i)}{N}\right)$$

(où s est la valeur du saut initiale `\sa`, N est le nombre de somments `\co` et le rayon du polygone initial est 1) et ajuster son orientation pour que les polygones dessinés à chaque étape s'imbriquent correctement les uns dans les autres (l'ajustement proposé ne convient que pour des valeurs impaires de `\co` et `\sa`). Enfin, chaque polygone est tracé à l'aide d'une boucle comme auparavant, on a donc deux boucles imbriquées dans le codage :

```
\newcommand{\co}{11}\newcommand{\sa}{5}
\foreach \sb/\couleur [evaluate=\sb] in
{\sa/cyan,\sa-1/blue,\sa-2/orange,\sa-3/red,\sa-4/green}
{
```

```
  \fill[rotate={180*(\sb+1)/\co},\couleur]
  (0:{cos(180*\sa/\co)/cos(180*\sb/\co)})
  foreach \n in {1,2,...,\co}
  {-- ({\n/\co*360*\sb}:{cos(180*\sa/\co)/cos(180*\sb/\co)})};
}
```

Automatisation de la coloration. L'inconvénient du codage précédent est qu'on doit spécifier les couleurs une par une, ce qui limite les paramètres qu'on peut utiliser. On peut éviter ce problème en utilisant un dégradé de couleurs créé automatiquement en fonction de la valeur de la variable **\sb** utilisée dans la boucle principale. Pour cela, on ajoute une option

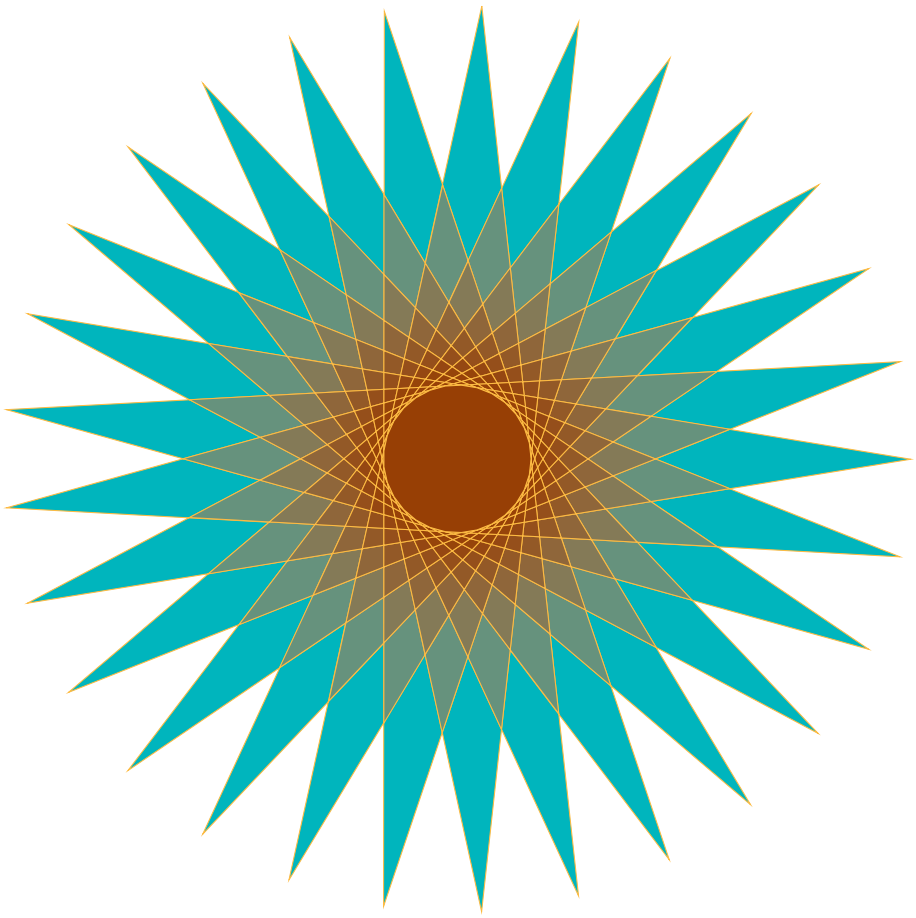
```
evaluate=\sb as \sbval using 100*((\sb-1)/(\sa-1))^4
```

dans la première boucle, qui indique de quelle manière la nouvelle variable `\sbval` dépend de **\sb**, puis on utilise cette nouvelle variable pour créer une couleur spécifique à chaque étape : l'option

```
fill=\coulext!\sbval!\coulint
```

mélange la couleur « extérieure » et la couleur « intérieure », définies par des macros LaTeX en début de codage (ici on utilise des couleurs de la famille `dvipsnames`), en utilisant `\sbval`% de couleur extérieure. On ajoute une couleur pour les côtés, également définie par une macro LaTeX, qu'on fait apparaître avec `\filldraw` :

```
\newcommand{\co}{29}\newcommand{\sa}{11}
\newcommand{\coulext}{Aquamarine}\newcommand{\coulint}{RawSienna}
\newcommand{\coulbord}{Dandelion}
\foreach \sb [evaluate=\sb,evaluate=\sb as \sbval using 100*((\sb-1)
/(\sa-1))^4] in {\sa,...,1}
     {
       \filldraw[rotate={180*(\sb+1)/\co},fill=\coulext!\sbval!
\coulint,draw=\coulbord]
       (0:{cos(180*\sa/\co)/cos(180*\sb/\co)})
       foreach \n in {1,2,...,\co}
       {-- ({\n/\co*360*\sb}:{cos(180*\sa/\co)/cos(180*\sb/\co)})};
     }
```



Chapitre 5

Visualisation de données avec `pgfplots`

On a vu ci-dessus comment utiliser les fonctions de base du package TikZ de LaTeX. À l'aide de celles-ci, le tracé de courbes représentant le graphe d'une fonction est aisé mais il faut ajouter les axes, légendes, grille... à la main ; celui de courbes représentant des données est possible mais il faut entrer les coordonnées de tous les points (encore qu'il soit possible de les extraire d'un fichier style tableur à *deux* colonnes, voir TikZ pour l'impatient, §5.2), construire leur représentation (point, barre, secteur...), ajouter là encore des axes, grille, repères numériques, légendes...

Ces diverses opérations peuvent être automatisées à l'aide d'une autre surcouche de `pfg` (l'ensemble de commandes de bas niveau sur lequel TikZ est construit), appelée `pgfplots`, qui fonctionne en complément de TikZ et qui le contient. Il s'agit d'un package LaTeX à charger dans le préambule du document (à la place de TikZ) :

```
\usepackage{pgfplots}
\pgfplotsset{compat=1.17}
```

La deuxième ligne est inhabituelle et optionnelle : elle indique la version de `pgfplots` utilisée dans le document, afin d'éviter que les figures soient modifiées lors de compilations ultérieures, utilisant d'autres versions du package. Le numéro de version peut être trouvé dans le fichier `.log` produit par la compilation par LaTeX :

> Package pgfplots notification 'compat/show suggested version=true' : you might benefit from `\pgfplotsset{compat=1.17}` (current compat level : 1.13).

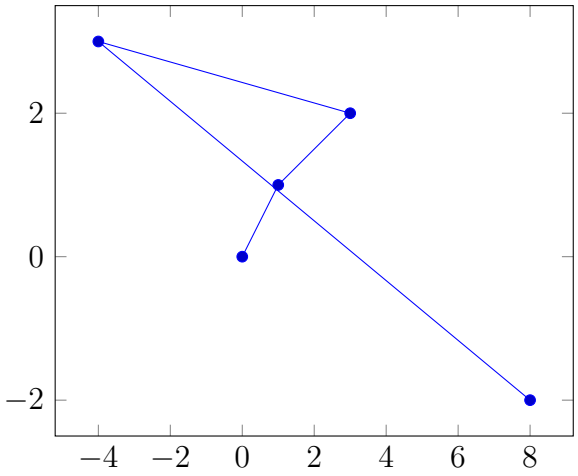
L'automatisation de la présentation visuelle des données (axes, grille, légendes,...) permise par `pgfplots` s'applique aussi bien aux graphes de fonctions qu'à des listes de données incluses dans le code ou à des données externes tirées d'un fichier au format `.csv` ('comma separated values', format d'enregistrement minimaliste offert par les tableurs). Dans ce dernier cas, `pgfplots` permet de choisir les colonnes à utiliser pour la représentation (elles doivent être pourvues d'une entête, c'est-à-dire un nom sur la première ligne de chaque colonne). Enfin `pgfplots` permet de réaliser à la fois des graphiques 2D et 3D.

5.1 Principes de base

L'utilisation est très simple : dans un environnement `tikzpicture`, on inclut un environnement `axis` avec (comme toujours) `\begin{axis} ... \end{axis}`, celui-ci crée automatiquement les axes convenables pour la représentation des données. On utilise la commande `\addplot` pour lancer un tracé, que ce soit à partir d'une liste de points, de l'expression d'une fonction ou d'un fichier externe.

```
\begin{tikzpicture}
\begin{axis}
\addplot coordinates {(0,0) (1,1) (3,2) (-4,3) (8,-2)};
\end{axis}
\end{tikzpicture}
```

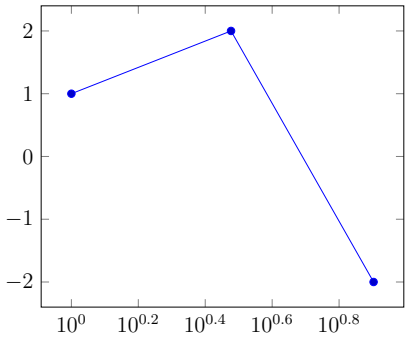
produit le graphe :



On observe ici le comportement par défaut, sans utiliser aucune option... et avec les points donnés dans un ordre aléatoire, ordre que `pgfplots` respecte pour le tracé des lignes qui les relient ! Dans la suite, on les mettra dans l'ordre croissant des abscisses.

L'intervalle de valeurs de chacun des axes est choisi pour contenir toutes les valeurs tirées des données qui lui appartiennent et les graduations sont adaptées à ces intervalles (indépendamment pour chacun des axes).

Diagrammes (semi-)logarithmiques. On peut tracer des diagrammes dans un repère logarithmique pour l'une ou l'autre des variables ou pour les deux en remplaçant `axis` par `semilogxaxis`, `semilogyaxis` ou `loglogaxis`. Nos données ne s'y prêtent pas vraiment mais essayons quand même (sur l'axe des x) :



Les deux valeurs impossibles $(0,0)$ et $(-4,3)$ ont été ignorées (la fonction log n'est définie que sur les réels strictement positifs) et `pgfplots` ajuste comme toujours la graduation des axes aux valeurs restantes.

Données externes. On peut récupérer les données dans un fichier externe au format `csv` avec la syntaxe

```
\begin{tikzpicture}
  \begin{axis}
    \addplot table{donnees.csv};
  \end{axis}
\end{tikzpicture}
```

Il y a d'autres extensions possibles pour le nom de fichier de données externes, par exemple `.dat`. Par défaut, un « point » correspond à une ligne du fichier, dans laquelle les différentes « coordonnées » sont séparées par des virgules, et `\addplot` traite les deux premières colonnes. On peut utiliser d'autres séparateurs que le saut de ligne et la virgule (notamment l'espace) et on peut, si les colonnes ont des entêtes en première ligne, modifier les colonnes à utiliser pour le tracé.

Par exemple, supposons que le fichier `donnees.csv` se présente ainsi :

abs	ord	hau
0	0	3
1	2	6
2	3	8
3	2	1
4	4	0
5	3	7
6	5	1

La première colonne a pour entête `abs`, la deuxième `ord` et la troisième `hau` ; le séparateur est l'espace plutôt que la virgule ici, ce qui ne pose pas de problème. On trace le diagramme des données de la troisième colonne en fonction de celles de la première en indiquant `table[x=abs,y=hau]{donnees.csv}` :

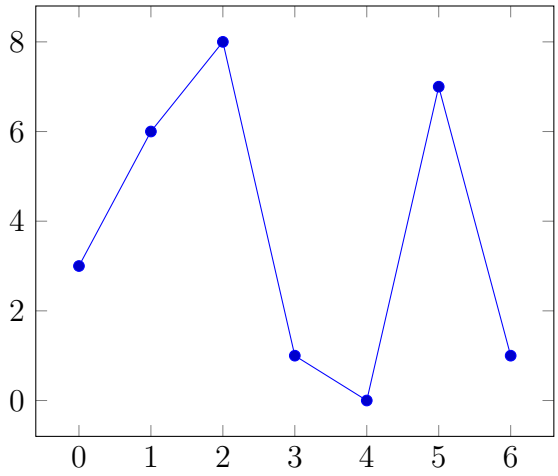
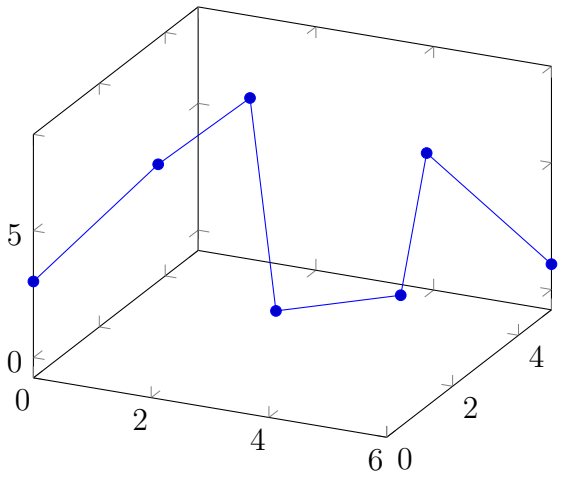


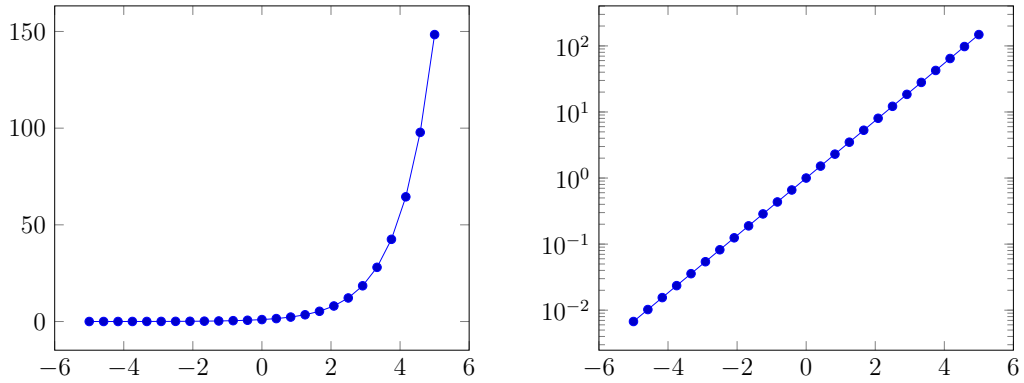
Diagramme 3D. Alternativement, avec un tableau à 3 colonnes comme ci-dessus, on peut tracer tout aussi facilement un graphe 3D avec la commande `\addplot3` :

```
\begin{axis}
  \addplot3 table{donnees.csv};
\end{axis}
```



Tracé de graphes de fonctions. La syntaxe pour le tracé du graphe d'une fonction est encore plus simple ! Pour un diagramme 2D (figure de gauche) :

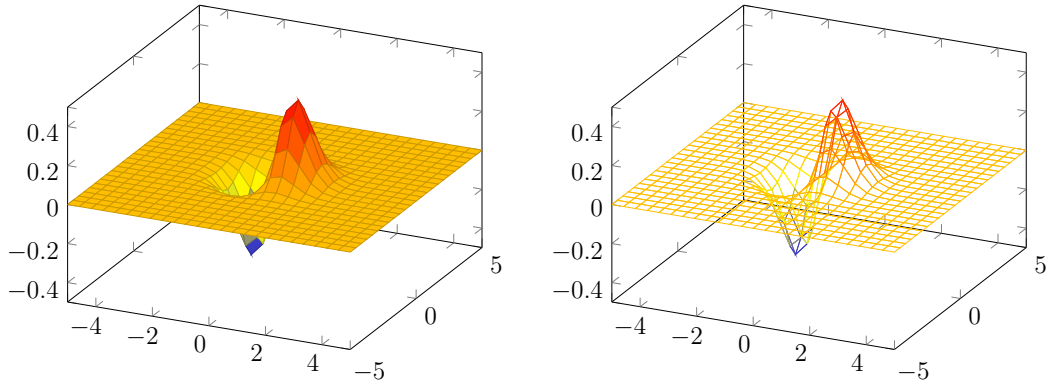
```
\begin{tikzpicture}
  \begin{axis}
    \addplot {exp(x)};
  \end{axis}
\end{tikzpicture}
```



La figure de droite est le tracé de la même fonction exp dans l'environnement `semilogyaxis`. Comme on le constate, le domaine des valeurs de la variable `x` est l'intervalle $[-5, 5]$ par défaut.

Pour tracer le graphe 3D d'une fonction de deux variables, c'est aussi facile :

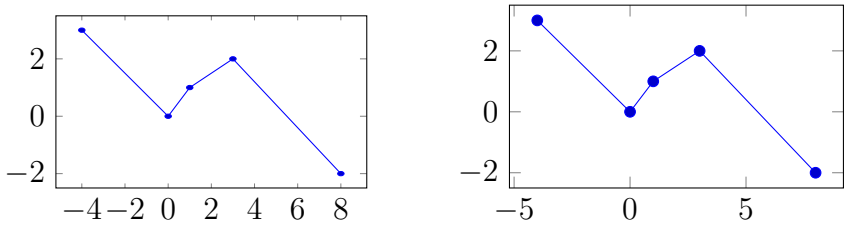
```
\addplot3[surf] {exp(-x^2-y^2)*x};  % ou \addplot3[mesh]
```



On a utilisé l'option `surf` pour `\addplot3` à gauche, `mesh` à droite.

5.2 Options pour l'environnement `axis`

Dimensions. L'option `scale` peut être appliquée à `axis`, ainsi que ses variantes dans chaque direction, voir l'effet de `xscale=.6,yscale=.4` ci-dessous (figure de gauche). On peut indiquer également des dimensions, comme `height=4cm,width=6cm`, si on veut imposer la taille du graphique (figure de droite) :



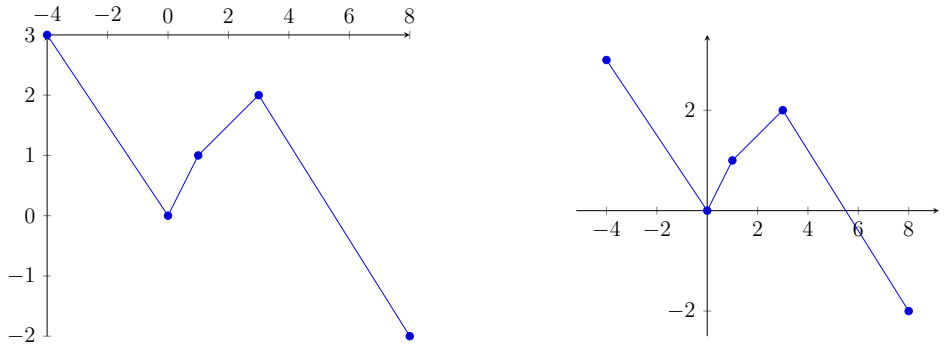
Noter que la taille des repères numériques indiqués le long des axes n'est pas affectée par cette option, par contre le nombre de graduation est automatiquement adapté par `pgfplots` à la place disponible (avec apparemment des critères différents dans les deux figures). Si besoin, on peut imposer les graduations des axes en indiquant par exemple `xtick={-4,-2,0,2,4,6,8}` dans les options de `axis`.

On aurait bien sûr pu utiliser l'option `scale` au niveau de l'environnement `tikzpicture`, c'est ce qu'on fera dans la suite (`\begin{tikzpicture}[scale=.7]`) pour que l'affichage du graphe prenne moins de place. On notera que, dans ce cas, le changement d'échelle s'applique aussi aux repères numériques indiqués le long des axes.

Placement des axes. Pour chacun des axes, on peut choisir les valeurs de `axis x line` et `axis y line` (ou `axis lines` si elles sont identiques) parmi les suivantes :

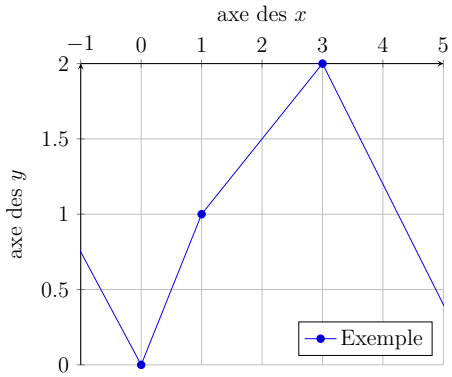
- `box` : par défaut (voir ci-dessus) ;
- `none` : pas d'axe ;
- `top` ou `bottom` : placement tout en haut ou tout en bas (axe des x) ;
- `left` ou `right` : placement tout à gauche ou tout à droite (axe des y) ;
- `middle` ou `center` : axe croisant l'autre à la valeur 0 (ou à sa plus petite valeur si 0 n'est pas dans son intervalle de valeurs).

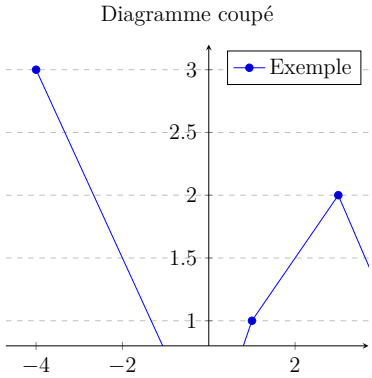
Voici ce que donnent les choix `axis x line=top,axis y line=left` (à gauche) et `axis lines=middle` (à droite) :



Pour la deuxième figure, on a ajouté `enlargelimits=true` aux options de `\begin{axis}`, ce qui agrandit légèrement les axes par rapport aux valeurs minimales et maximales tirées des données.

Valeurs minimales et maximales, grilles, titre, légendes. On peut spécifier des valeurs minimales et maximales pour chaque axe avec les options `xmin=-1`, `xmax=5` (figure de gauche), `ymin=1` (figure de droite) :





On en a profité pour illustrer les options `grid=major`, `xlabel={axe des $x$}` et `ylabel={axe des $y$}` (à gauche), `ymajorgrids`, `grid style=dashed` et `title={Diagramme coupé}` (à droite).

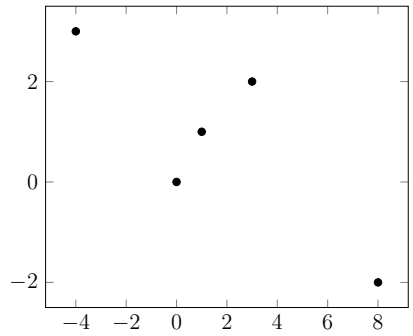
Enfin, c'est une commande spécifique qui permet d'écrire la légende du diagramme :

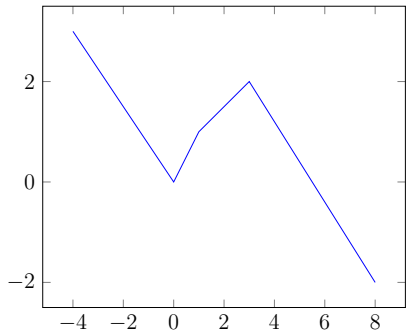
```
\begin{axis}[axis x line=top,axis y line=left,xmin=-1,xmax=5,grid=
major,xlabel={axe des $x$},ylabel={axe des $y$},legend pos=south east
]
  \addplot coordinates {(-4,3) (0,0) (1,1) (3,2) (8,-2)};
  \legend{Exemple}
\end{axis}
```

Par défaut elle est placée dans le coin en haut à droite (deuxième figure), on peut modifier son emplacement dans les options de `axis` (première figure). On aurait pu utiliser également `outer north east` qui sort la légende du diagramme, ou `north west` (`south west` et `north east` sont moins adaptés ici).

5.3 Options de la commande addplot

Là aussi il y a de nombreuses options disponibles. On peut n'afficher que les points, en indiquant `\addplot[only marks]`, ou que les traits : `\addplot+[mark=none]` (noter qu'il n'y a pas de 's' à `mark` dans le deuxième cas) :

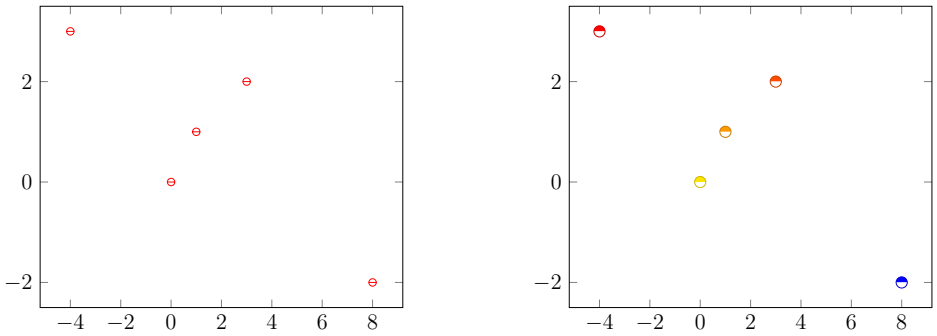




La couleur bleue par défaut du graphe a disparu dans le premier cas (le fait d'indiquer une option à `addplot` enlève les options par défaut) mais pas dans le deuxième grâce au `+` qui

indique que l'option s'ajoute aux précédentes. On peut évidemment modifier la couleur en l'ajoutant dans les options : `\addplot[only marks,red]` colore les points en rouge (alors que `\begin{axis}[red]` colore toute la figure en rouge, y compris les axes, repères...).

Points. L'apparence des points offre de nombreuses possibilités. Dans la figure de gauche, on a indiqué `mark=halfcircle` dans les options (en plus de `marks only` et `red`), cela produit des cercles coupés en deux par un diamètre. Dans celle de droite, cette option est modifiée par une étoile : `mark=halfcircle*`, ce qui colore la moitié supérieure du cercle ; la taille des points est augmentée en stipulant `mark size=3pt`, leur couleur est réglée par l'option `scatter` qui choisit celle-ci en fonction des valeurs d'un paramètre (ici, par défaut, la deuxième coordonnée).

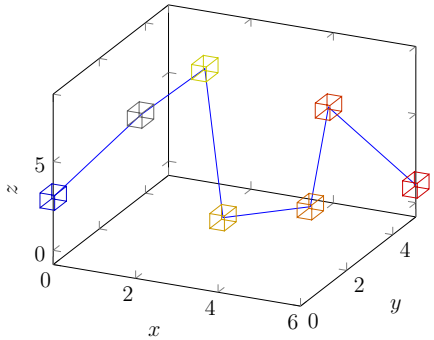


La liste des apparences possibles des points comprend notamment (manuel de pgfplots §4.7) :

```
*, x , +, |, o, asterisk, star, 10-pointed star, Mercedes star,
Mercedes star flipped, halfdiamond*, halfsquare*, halfsquare right*,
halfsquare left*, text,
oplus, otimes, square, triangle, diamond, halfcircle, pentagon, cube
```

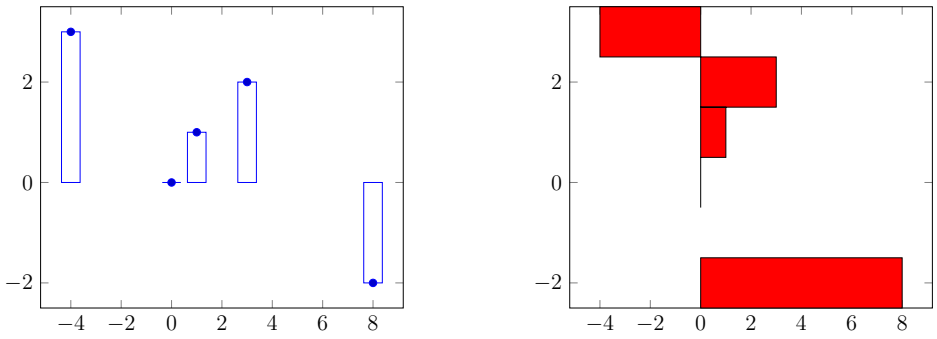
où :

- `*` trace le point "par défaut" ● ;
- `text` utilise par défaut la lettre 'p' comme marqueur (modifier avec `text mark=...` : tout texte ou image est accepté) ;
- les 8 dernières possibilités ont une variante avec `*` qui les colore ;
- `cube` et `cube*` sont réservés aux figures 3D :



On a utilisé la commande `\addplot3+[mark=cube,mark size=8pt,scatter,point meta={x}] table{donnees.csv};`. La couleur des points est individualisée ici aussi mais elle dépend maintenant de la valeur de la première colonne grâce à l'option supplémentaire `point meta={x}` (voir manuel de pgfplots, §4.8).

Barres. Rien de plus facile que de transformer notre figure en un diagramme en barres ! Pour des barres verticales, on utilise l'option `ybar`, et `xbar` si on préfère des barres horizontales. On peut ou non laisser les marques précédentes pour les points en plus des barres (mais penser à enlever `only marks`).

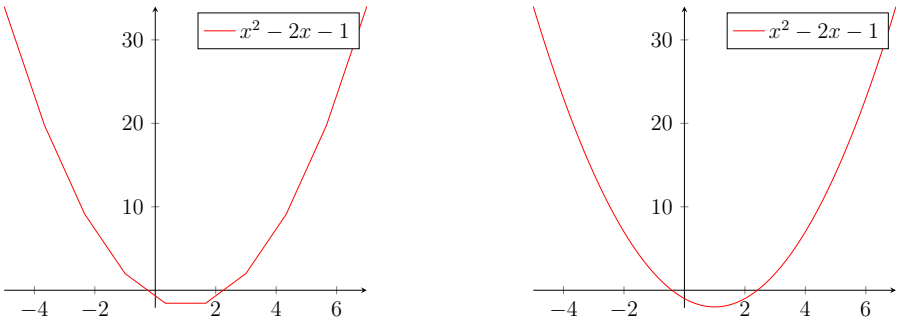


À gauche, on a utilisé `\addplot+[ybar]` qui montre l'affichage par défaut, avec le `+` qui permet de garder la couleur bleue *et* l'affichage des points (mais pas les traits qui les reliaient). À droite la commande est `\addplot[xbar,fill=red,bar width=1]` : remplissage des barres en rouge, réglage de leur largeur avec `bar width=1`, qui fait qu'elles se touchent (celles qui sont centrées sur les points d'ordonnées 0, 1, 2 et 3). Sans indication, l'encadrement des barres est en noir (on n'a pas mis de `+`), on peut le modifier en ajoutant `draw=red`.

Pour un diagramme en bâtons, on peut utiliser l'option `bar width` pour réduire la largeur des barres ou remplacer `xbar` et `ybar` par `xcomb` et `ycomb` (voir manuel pgfplots, §4.5.7).

5.4 Options pour les graphes de fonctions

Graphes 2D. Les deux principales options de `\addplot` concernent le choix du domaine, ici `domain=-5:7`, et le nombre de points à calculer pour dessiner la courbe : `samples=10` à gauche et `samples=100` à droite.

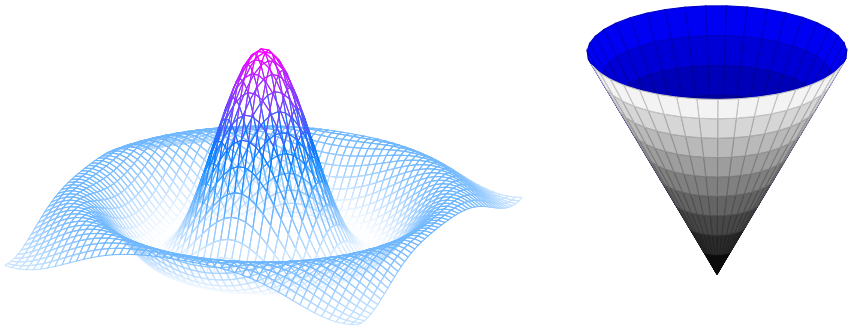


Rappelons que le choix des axes se coupant à l'origine $(0, 0)$ est obtenu en passant `axis lines =middle` en option de `\begin{axis}`.

Graphes 3D. Pour les graphes 3D, les deux options `domain` et `samples` s'appliquent également (à `\addplot3` ou même à `\begin{axis}`), elles portent sur la première variable `x`, ainsi que sur la deuxième `y` sauf si `y domain` ou `samples y` est spécifié.

L'expression de la fonction peut prendre deux formes : une formule $f(x,y)$ donnant la valeur de z en fonction de x et y ou un triplet $(f_1(x,y), f_2(x,y), f_3(x,y))$ donnant les trois coordonnées en fonction de deux paramètres x et y.

```
\begin{tikzpicture}
  \begin{axis}[axis lines=none, colormap/cool]
    \addplot3[mesh, samples=50, domain=-8:8] {sin(deg(sqrt(x^2+y^2)))
/sqrt(x^2+y^2)};
  \end{axis}
  \begin{scope}[xshift=6cm]
    \begin{axis}[axis lines=none,
      domain=0:1, y domain=0:2*pi,
      xmin=-1.5, xmax=1.5, ymin=-1.5, ymax=1.5, zmin=0.0,
      mesh/interior colormap={blueblack}{color=(black) color=(blue)},
      colormap/blackwhite,
      samples=10,samples y=40,
      z buffer=sort]
      \addplot3[surf] ({x*cos(deg(y))},{x*sin(deg(y))},{x});
    \end{axis}
  \end{scope}
\end{tikzpicture}
```



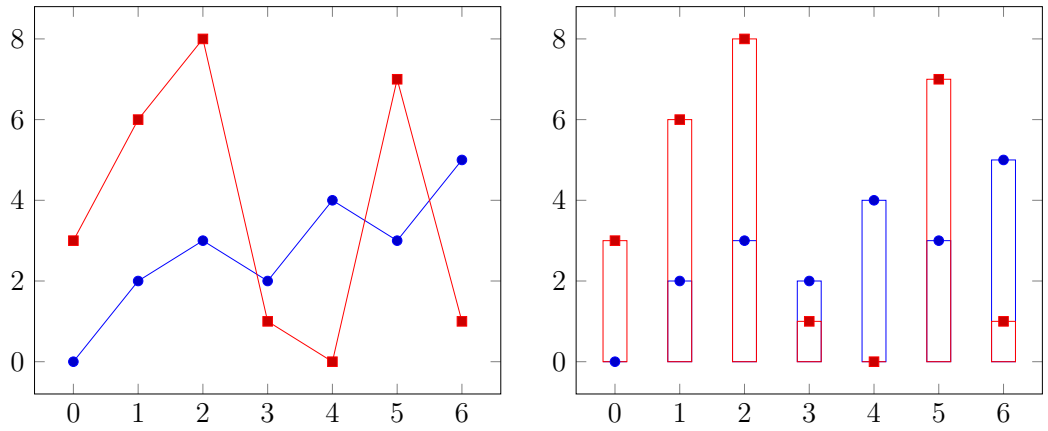
On retrouve les deux options `mesh` (à gauche) et `surf` (à droite) de `\addplot3`. Ici on a de plus fait varier les *modèles de couleur* (`colormap`) utilisés pour colorier les tracés : `cool` à gauche, `blueblack` à l'intérieur et `blackwhite` à l'extérieur de la surface à droite. Les autres modèles disponibles sont `hot`, `hot2`, `jet`, `bluered`, `greenyellow`, `redyellow` et `violet` (voir manuel pgfplots §4.7.6). Enfin l'option `z buffer=sort` applique un procédé plus précis que celui utilisé par défaut pour déterminer quelle partie de la surface doit apparaître au premier plan de la représentation (manuel pgfplots §4.6.5).

Il y a de nombreuses autres options, notamment pour modifier l'apparence des surfaces (obtenues avec `surf`), voir le manuel de pgfplots §4.6.6.

5.5 Avec plusieurs séries de nombres.

Si on veut représenter deux séries de nombres sur le même graphique, on utilise deux fois la commande `\addplot` :

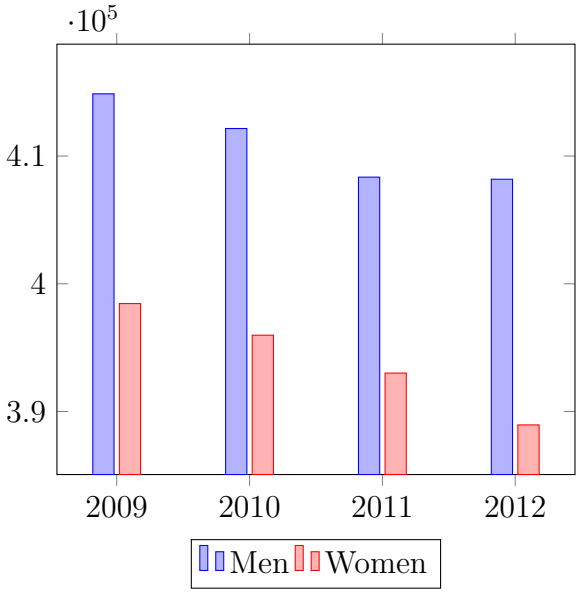
```
\begin{tikzpicture}
  \begin{axis}
    \addplot table[x=abs,y=ord]{donnees.csv};
    \addplot table[x=abs,y=hau]{donnees.csv};
  \end{axis}
\end{tikzpicture}
```



À droite c'est la version avec `\addplot+[ybar]` à chacune des deux lignes. Sur cet exemple simple, on voit l'intérêt de garder les marques des points (en plus des barres), ils permettent de voir les petites valeurs de la première série (en bleu), dont les barres sont recouvertes par celles de la deuxième lorsque celle-ci prend des valeurs supérieures.

Plus de style. Il est possible de décaler les deux séries de barres, un environnement spécifique est décrit dans [Feu15, p. 81], nous le reproduisons ici :

```
\begin{axis}
  [
    x tick label style={
      /pgf/number format/1000 sep=},
    enlargelimits=0.15,
    legend style={at={(0.5,-0.15)},
      anchor=north,legend columns=-1},
    ybar,
    bar width=7pt,
  ]
  \addplot coordinates {(2012,408184) (2011,408348) (2010,412156)
(2009,414870)};
  \addplot coordinates {(2012,388950) (2011,393007) (2010,395972)
(2009,398449)};
  \legend{Men,Women}
\end{axis}
```



Ici `ybar` est en option de `\begin{axis}` et affecte tous les `\addplot` qui suivent. On peut modifier les paramètres donnés en option, attention cependant au résultat : sur la page `Pgfplots_package` de [Ove], le même diagramme apparaît avec des données manquantes (celles de 2009).

On trouvera de très nombreuses autres possibilités d'affichages de données multiples dans [Feu15, §4.5], y compris avec plus de deux séries de données.

Bibliographie

[Feu15] Christian Feuersänger. *Manual for Package pgfplots. 2D/3D Plots in LaTeX*. Version 1.12.1. 2015. URL : http://sourceforge.net/projects/pgfplots.

[Gou10] Philippe Goutet. *Aide-mémoire LaTeX*. 2010. URL : http://pgoutet.free.fr/latex/aide-memoire.pdf.

[Ker21] Uwe Kern. *Extending LaTeX's color facilities : the xcolor package*. Version v2.13. 2021. URL : http://www.ukern.de/tex/xcolor.html.

[MS07] Andrew Mertz et William Slough. "Graphics with TikZ". In : *The PracTEX Journal* 1 (2007), p. 1-22.

[Ove] Overleaf. *Documentation*. URL : https://fr.overleaf.com/learn.

[Riv16] Gonzalo Rivero. "Presentations in LaTeX. Introduction to the beamer class". In : 2016.

[TD15] Gérard Tisseau et Jacques Duma. *TikZ pour l'impatient*. 2015. URL : http://math.et.info.free.fr/TikZ/.

[Wik] Wikibooks. *LaTeX*. URL : https://fr.wikibooks.org/wiki/LaTeX.