Snippets de TEXmaker pour PYMATH

ou comment débuter avec LATEX en s'aidant de morceaux de code tous prêts

Voici donc des morceaux de code (snippets) à utiliser lorsqu'on débute et que l'on souhaite obtenir un résultat rapidement.

Une fois TEXmaker ouvert, utiliser le menu `Options, Fichier de configuration, Remplacer le fichier de configuration par un autre` et installer le fichier « reglages_texmaker » donné par PYMATH.

Tout ceci est expliqué dans la quatrième vidéo d'installation et à la fin de l'article du bulletin 24 de Pymath.

Une fois que vous maîtrisez l'usage des snippets, vous pouvez utiliser les raccourcis clavier. Par exemple, dans l'exemple ci-dessous, en haut à droite du cadre, vous trouvez le raccourci à taper dans le document pour voir apparaître le snippet.

Exemple Complet Minimal `:ecm→`

Tous les snippets seront présentés de cette manière :

- dans un bandeau comme ci-dessus le nom du snippets à gauche. C'est ce nom que l'on retrouvera dans le panneau latéral à gauche de TEXmaker.
- dans le même bandeau à droite, le nom du raccourci à taper tel quel dans l'éditeur.
  par exemple, ici, on tape `:ecm→` (deux points e c m et la flèche vers la droite).
- En dessous du bandeau, des explications comme ici.
- Puis dans un cadre rouge, le snippet (code LATEX) proprement dit ainsi qu'un aperçu du résultat en dessous d'une ligne en pointillés

```
\documentclass{article}
\begin{document}
Bonjour le monde !
\end{document}
```

Bonjour le monde !

1 Colonnes

colonnes avec trait de séparation :colt→

nécessite un \usepackage{multicol} dans le préambule (déjà inclus dans la classe pymath)
ligne 1 { et ligne 11 } : Les commandes de réglages sont dans un groupe pour qu'ils soient locaux. Ainsi, une fois sorti du groupe, les réglages sont à nouveaux ceux de base.
lignes 3 et 6 : l'unité de longueurs peut s'exprimer en millimètres mm à la place du point pt.

```
1  { % les changements sont locaux au groupe
2  % espace entre les colonnes
3  \setlength{\columnsep}{25pt}
4
5  % largeur de la ligne séparatrice
6  \setlength{\columnseprule}{0.5pt}
7
8  \begin{multicols}{2} % 2 colonnes
9          \lipsum[8]
10 \end{multicols}
11 } % fin des colonnes
```

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Donec odio elit, dictum in, hendrerit sit amet, egestas sed, leo. Praesent feugiat sapien aliquet odio. Integer vitae justo. Aliquam vestibulum fringilla lorem. Sed neque lectus, consectetuer at, consectetuer sed, eleifend ac, lectus. Nulla facilisi. Pellentesque eget lectus. Proin eu metus. Sed porttitor. In hac habitasse platea dictumst. Suspendisse eu lectus. Ut mi mi, lacinia sit amet, placerat et, mollis vitae, dui. Sed ante tellus, tristique ut, iaculis eu, malesuada ac, dui. Mauris nibh leo, facilisis non, adipiscing quis, ultrices a, dui.

colonnes sans trait de séparation :cols→

nécessite un \usepackage{multicol} dans le préambule (déjà inclus dans la classe pymath)

```
1  { % les changements sont locaux au groupe
2  \setlength{\columnsep}{25pt} % espace entre les colonnes
3
4  \begin{multicols}{2} % 2 colonnes
5          \lipsum[8]
6  \end{multicols}
7  } % fin des colonnes
```

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Donec odio elit, dictum in, hendrerit sit amet, egestas sed, leo. Praesent feugiat sapien aliquet odio. Integer vitae justo. Aliquam vestibulum fringilla lorem. Sed neque lectus, consectetuer at, consectetuer sed, eleifend ac, lectus. Nulla facilisi. Pellentesque eget lectus. Proin eu metus. Sed porttitor. In hac habitasse platea dictumst. Suspendisse eu lectus. Ut mi mi, lacinia sit amet, placerat et, mollis vitae, dui. Sed ante tellus, tristique ut, iaculis eu, malesuada ac, dui. Mauris nibh leo, facilisis non, adipiscing quis, ultrices a, dui.

2 Listes

liste de puces :puce→

Par défaut, itemize met des tirets.
C'est pour cela qu'on le force à mettre des petits ronds •(`\textbullet`) à la place.
On peut aussi mettre ▷ (`$\triangleright$`) ou ▶ (`$\blacktriangleright$`).
Il est possible de mettre tous les symboles qui nous plaisent, voir dans le panneau latéral à gauche de TEXmaker.

```
\begin{itemize}[label=\textbullet]
\item premier item
\item second item
\end{itemize}
```

- premier item
- second item

liste numérotée :num→

nécessite un `\usepackage{enumerate}` dans le préambule (déjà inclus dans la classe pymath)
le package enumerate permet d'ajouter des options sympathiques comme start pour démarrer de plus loin.

```
\begin{enumerate}[start=3] % changer le 3 si nécessaire
\item premier item
\item second item
\end{enumerate}
```

3. premier item
4. second item

liste numérotée suite :nums→

nécessite un `\usepackage{enumerate}` dans le préambule (déjà inclus dans la classe pymath)
le package enumerate permet d'ajouter des options sympathiques comme resume pour continuer une liste que l'on aurait fermé plus tôt pour insérer un texte ou une figure par exemple.

```
\begin{enumerate}[resume] % continue la liste précédente
\item premier item
\item second item
\end{enumerate}
```

1. premier item
2. second item

Remarque : l'exemple n'est pas représentatif car il est inclus dans un groupe. Pour les programmeurs, un groupe crée des variables locales donc les numérotations recommencent à zéro. En général, une série de questions reste dans un seul groupe.

3 Tableaux

tableau de valeurs :tabval→

nécessite un \usepackage{tabularx} dans le préambule (déjà inclus dans la classe pymath)
Ce package permet de définir la largeur du tableau.
Lorsque l'on veut faire des tableaux avec des cases vides à compléter par les élèves, LaTeX n'est pas des plus simples à utiliser.
En effet, LaTeX permet décrire un tableau sans se soucier de sa taille. Il s'arrange pour que les cases aient leur taille adaptée à leur contenu.
C'est pourquoi, ici, l'usage de tabularx permet d'élargir les colonnes (largeur uniforme adaptée à la taille du tableau) et l'usage d'un texte fantôme permet d'adapter la hauteur des lignes.
Lorsque LaTeX rencontre un texte fantôme, il réserve la place nécessaire au texte mais ne l'imprime pas.
Changer simplement :

- la largeur du tableau et le nombre de colonnes à la ligne 2
- les données séparées par des esperluètes (&) (attention au nombre de &) aux lignes 7 et 10

```
1  \begin{center}
2  \begin{tabularx}{10cm}{|Sc|*{7}{>{\centering \arraybackslash}SX|}}
3  % Changer la largeur du tableau (10cm) ci-dessus
4  % changer le nombre de valeurs ci-dessus (7)
5  \hline
6      $x$\vphantom{$\dfrac12$}&
7      1&2&3&4&5&& % mettre ici les valeurs séparées par &
8  \\\hline
9      $f(x)$\vphantom{$\dfrac12$} &
10     $\dfrac12$&&&&& 6&7 % mettre ici les valeurs séparées par &
11 \\\hline
12 \end{tabularx}
13 \end{center}
```

x	1	2	3	4	5		
$f(x)$	$\frac{1}{2}$					6	7

Sans aucun réglage, on aurait :

```
1  \begin{center}
2  \begin{tabular}{|c|*{7}{c|}}
3  \hline
4      $x$&1&2&3&4&5&&\\\hline
5      $f(x)$&$\dfrac12$&&&&&6&7\\\hline
6  \end{tabular}
7  \end{center}
```

x	1	2	3	4	5		
$f(x)$	$\frac{1}{2}$					6	7

tableau de signes :tabsigne→

nécessite un \usepackage{tkz-tab} dans le préambule (déjà inclus dans la classe pymath)
Ne pas hésiter à faire un texdoc tkz-tab dans une console ou un terminal pour avoir la documentation complète ou bien sur internet tkz-tab-screen.pdf ici : http://www.ctan.org/tex-archive/macros/latex/contrib/tkz/tkz-tab/doc
Tout se fait aux lignes 8 à 12.

- lignes 7 à 10 : on écrit d'abord ce que l'on veut sur la première colonne sous la forme « texte / taille (hauteur en cm) » (le symbole / est nécessaire) séparés par des virgules et entourés d'accolades.
- ligne 11 : on décrit la première ligne, n « valeurs » séparées par des virgules et entourées d'accolades.
- ligne 12 : on décrit la ligne des signes avec $2n-1$ « valeurs » séparées par des virgules et entourées d'accolades. Ces « valeurs » prennent en compte aussi les intervalles de séparation. Les valeurs possibles sont décrites dans la documentation. Quelques valeurs sont précisées aux lignes 13 à 16.

```
1  \begin{center}
2  \begin{tikzpicture}   % toutes les dimensions sont en cm
3  \tkzTabInit[lgt=3,    % largeur de la première colonne
4                      espcl=1.5 % espace entre les valeurs de la
5                                % première ligne
6                     ]
7  % ce qui suit décrit la première colonne
8  {$x$ / 1 , % titre de la première ligne et sa hauteur
9  signe de $f(x)$ /1  % titre de la seconde ligne et sa hauteur
10 }%
11 {$v_1$ , $v_2$ , $v_3$ , $v_4$, $v_5$}% n valeurs de la première ligne
12 \tkzTabLine{ t, + , z ,- , d , - , 0, +, } % 2n-1 "valeurs"
13                                           % t pour un trait vertical
14                                           % z pour un zéro sur un trait
15                                           % d pour une double barre
16                                           % et + et - bien sûr !
17                                           % 0 pour un zéro sans barre
18 % rajouter des lignes avec \tkzTabLine{ , , , , }
19 \end{tikzpicture}
20 \end{center}
```

x	v_1	v_2	v_3	v_4	v_5
signe de $f(x)$	$+$	0	$-$	$-$ 0 $+$	

tableau de variation :tabvar→

nécessite un \usepackage{tkz-tab} dans le préambule (déjà inclus dans la classe pymath)
Ne pas hésiter à faire un texdoc tkz-tab dans une console ou un terminal pour avoir la documentation complète ou bien sur internet tkz-tab-screen.pdf ici : http://www.ctan.org/tex-archive/macros/latex/contrib/tkz/tkz-tab/doc
Tout se fait aux lignes 8 à 12.

- lignes 7 à 10 : on écrit d'abord ce que l'on veut sur la première colonne sous la forme « texte / taille (hauteur en cm) » (le symbole / est nécessaire) séparés par des virgules et entourés d'accolades.
- ligne 11 : on décrit la première ligne, n « valeurs » séparées par des virgules et entourées d'accolades.
- ligne 12 : on décrit la ligne des signes avec $2n-1$ « valeurs » séparées par des virgules et entourées d'accolades. Ces « valeurs » prennent en compte aussi les intervalles de séparation. Les valeurs possibles sont décrites dans la documentation. Quelques valeurs sont précisées aux lignes 13 à 16.

Si vous avez l'impression d'avoir déjà lu ce texte c'est parce que c'est la même méthode que pour les tableaux de signe ;-)
Merci à l'auteur de ce package et à sa documentation très bien faite.

```
1  \begin{center}
2  \begin{tikzpicture} % toutes les dimensions sont en cm
3  \tkzTabInit[lgt=2,  % largeur de la première colonne
4                      espcl=3 % espace entre les valeurs de la
5                              % première ligne
6                     ]
7  % ce qui suit décrit la première colonne
8  {$x$ / 1 , % titre de la première ligne et sa hauteur
9  {variation\\ de $f$} /2  % titre de la seconde ligne et sa hauteur
10 }%
11 {$v_1$ , $v_2$ , $v_3$ }% n valeurs de la première ligne
12 \tkzTabVar{+/$M_1$ , -/m, +/$M_2$}% + pour écrire en haut
13                                     % - pour écrire en bas
14                                     % +D ou -D pour une
15                                     % double barre
16 \end{tikzpicture}
17 \end{center}
```

x	v_1	v_2	v_3
variation de f	M_1 ↘	m	↗ M_2

tableau complet étude de fonction :tabsigvar→

nécessite un \usepackage{tkz-tab} dans le préambule (déjà inclus dans la classe pymath)
Lire les explication des deux tableaux précédents car celui ci n'en est qu'une compilation.

```
\begin{center}
\begin{tikzpicture} % toutes les dimensions sont en cm
\tkzTabInit[lgt=3,  % largeur de la première colonne
                    espcl=3 % espace entre les valeurs de la
                            % première ligne
                   ]
% ce qui suit décrit la première colonne
{$x$ / 1 , % titre de la première ligne et sa hauteur
{signe de $f'(x)$} /1, % titre de la seconde ligne et
                                                % sa hauteur
{variation\\ de $f$} /2  % titre de la troisème ligne
                                                % et sa hauteur
}%
{$v_1$ , $v_2$ , $v_3$ }% n valeurs de la première ligne
\tkzTabLine{ t, + , z ,- ,d } % 2n-1 "valeurs"
                                        % t pour un trait vertical
                                        % z pour un zéro sur un trait
                                        % d pour une double barre
                                        % et + et - bien sûr !
% rajouter des lignes avec \tkzTabLine{ , , , , }
\tkzTabVar{+/$M_1$ , -/m, +D/$M_2$}% + pour écrire en haut
                                                % - pour écrire en bas
                                                % +D ou -D pour une
                                                % double barre
\end{tikzpicture}
\end{center}
```

x	v_1	v_2	v_3
signe de $f'(x)$	+	0	−
variation de f	M_1 ↘	m	↗ M_2

4 Arbres

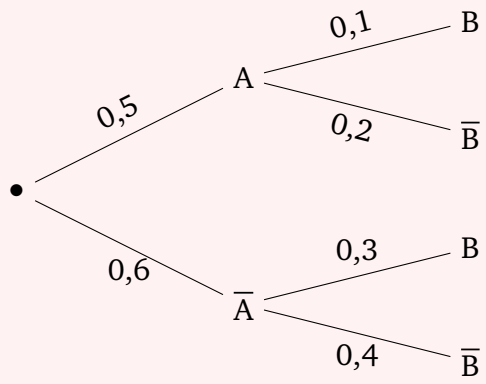
arbre 2×2 pondéré :arbre2x2→

nécessite un \usepackage{tikz} dans le préambule (déjà inclus dans la classe pymath)
Les probabilités sur les branches ont été choisies pour que vous puissiez les retrouver facilement dans le code.

- level distance = longueur des branches
- sibling distance = écart vertical entre les éventualités

Il existe des valeurs par défaut mais elles ne sont pas très belles.

```
\begin{tikzpicture}
% level distance = longueur des branches
% sibling distance = écart vertical entre les éventualités
\tikzstyle{level 1}=[level distance=3cm, sibling distance=3cm]
\tikzstyle{level 2}=[level distance=3cm, sibling distance=1.5cm]
\node{$\bullet$}[grow'=right] % l'arbre pousse à droite
% modifier le nom des "noeuds" (node) et les probabilités
        child{node{$A$}
                child{node{$B$}
                        edge from parent node[above,sloped]{0,1}}
                child{node{$\overline{B}$}
                        edge from parent node[below,sloped]{0,2}}
        edge from parent node[above,sloped]{0,5}}
        child{node{$\overline{A}$}
                child{node{$B$}
                        edge from parent node[above]{0,3}}
                child{node{$\overline{B}$}
                        edge from parent node[below]{0,4}}
        edge from parent node[below]{0,6}}
        ;
\end{tikzpicture}
```

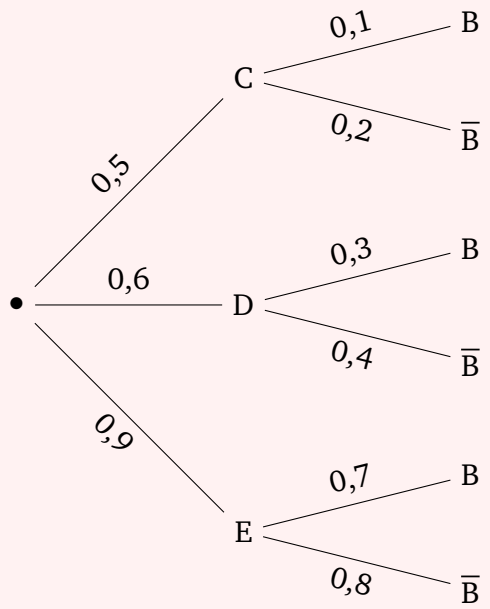


Remarque : Regardez l'effet du retrait de l'option slope sur les trois dernières branches.

arbre 3×2 pondéré :arbre3x2→

nécessite un \usepackage{tikz} dans le préambule (déjà inclus dans la classe pymath)
Voir cas précédent

```
\begin{tikzpicture}
% level distance = longueur des branches
% sibling distance = écart vertical entre les éventualités
\tikzstyle{level 1}=[level distance=3cm, sibling distance=3cm]
\tikzstyle{level 2}=[level distance=3cm, sibling distance=1.5cm]
\node{$\bullet$}[grow'=right] % l'arbre pousse à droite
% modifier le nom des "noeuds" (node) et les probabilités
        child{node{$C$}
                child{node{$B$}
                        edge from parent node[above,sloped]{0,1}}
                child{node{$\overline{B}$}
                        edge from parent node[below,sloped]{0,2}}
        edge from parent node[above,sloped]{0,5}}
        child{node{$D$}
                child{node{$B$}
                        edge from parent node[above,sloped]{0,3}}
                child{node{$\overline{B}$}
                        edge from parent node[below,sloped]{0,4}}
        edge from parent node[above,sloped]{0,6}}
        child{node{$E$}
                child{node{$B$}
                        edge from parent node[above,sloped]{0,7}}
                child{node{$\overline{B}$}
                        edge from parent node[below,sloped]{0,8}}
        edge from parent node[below,sloped]{0,9}}
        ;
\end{tikzpicture}
```



5 Graphiques

Une méthode simple est d'utiliser Geogebra qui permet d'exporter les figures aux deux formats LaTeX les plus connus : TikZ et PStricks. Voir la vidéo LaTeXGeogebra expliquant comment faire sur le site de Pymath.

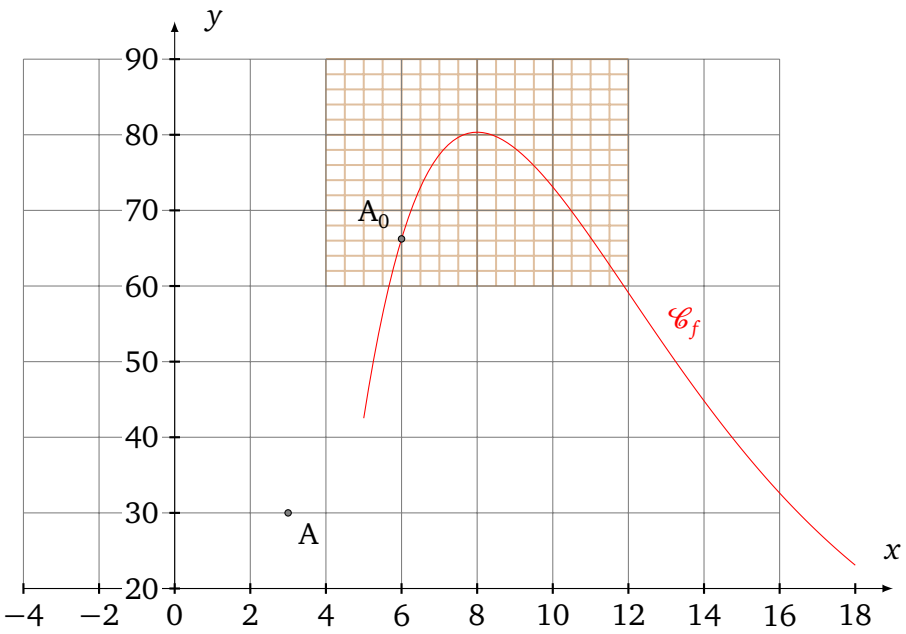
Les avantages d'exporter dans ces formats sont que :

- Le graphique est inclus dans le fichier source ;
- Le graphique peut être modifié sans retourner dans Geogebra (entre autres, on peut changer les noms des points ou rajouter des textes ou des formules) ;
- la police de caractère utilisée est celle du texte, d'où une unité graphique plus agréable.

Le logiciel de statistique R permet aussi d'exporter ces graphiques au format pdf que LaTeX sait importer.

Sinon, vous pouvez apprendre les commandes des packages PSTricks, Tikz, Asymptote, ... (ou d'un seul). C'est plus long mais aussi puissant pour ceux qui voudraient devenir sorcier ;-)

Dans l'exemple qui suit, il est nécessaire d'installer Gnuplot, voir la cinquième vidéo d'installation du bulletin 24 sur le site de Pymath.



Graphique Fonction :grfct→

nécessite un `\usepackage{tkz-fct}` dans le préambule (déjà inclus dans la classe pymath)
Ne pas hésiter à faire un `texdoc tkz-fct` dans une console ou un terminal pour avoir la documentation complète ou bien sur internet tkz-tab-screen.pdf ici : http://www.ctan.org/tex-archive/macros/latex/contrib/tkz/tkz-fct/doc
Ce graphique montre une bonne part de ce qui est utile en analyse pour un professeur.
Toutefois, il n'est pas exhaustif. La documentations du package est très bien faite.
Pour ce graphique, il est nécessaire d'installer Gnuplot, voir la cinquième vidéo d'installation.

```
\begin{tikzpicture}[scale=1] % on peut agrandir,
                        % de base l'unité est de 1 cm.
\tkzInit[ % les différents réglages de base
xmin=-4,
xmax=18,
xstep=2, % l'unité est de 1 cm (scale) pour xstep
ymin=20,
ymax=90,
ystep=10 % l'unité est de 1 cm (scale) pour ystep
]
% remarque : les commandes sont dans l'ordre d'affichage
% ainsi la courbe sera tracée au dessus des grilles
\tkzGrid[sub, % une "sous-grille" pour mieux lire les
subxstep=0.5, % coordonnées. Exemple, ici on a subdivisé
subystep=2,   % l'unité en 4 car subxstep = xstep/4
color=brown](4,60)(12,90) % on donne les coordonnées des
                                                % deux coins opposés.
\tkzGrid(-4,20)(16,90) % une grille qui n'a pas forcément
                                            % la même taille que les axes
                                            % on donne les coordonnées des
                                            % deux coins opposés.

\tkzDrawX[label={$x$},above=10pt] % dessine l'axe des x
                                        % avec un "titre" ou unité (label)
                                        % décalé légèrement vers le haut
\tkzLabelX % écrit les nombres sur l'axe des x.
\tkzDrawY[label={$y$},right=10pt] % dessine l'axe des x
                                        % avec un "titre" ou unité (label)
                                        % décalé légèrement vers la droite
\tkzLabelY % écrit les nombres sur l'axe des y.

\tkzFct[ % les différents réglages pour tracer la courbe
domain = 5:18,        % domaine de définition
samples=100,        % nombre de points
                            % penser à mettre 2 points pour une droite
color=red
]%
{(\x-4)*exp(-0.25*\x+5)} % formule avec \x au lieu de x

\tkzText(13.5,55){$\mathcal{C}_{f}$} % ajoute un texte

\tkzDefPoint(3,30){A} % définit un point
\tkzDrawPoint(A)         % trace le point
\tkzLabelPoints(A)         % écrit le nom du point

\tkzDefPointByFct(6) % définit un point sur la courbe
\tkzGetPoint{B}                 % attrape ce point et le nomme B
\tkzDrawPoint(B)         % trace le point B
\tkzLabelPoint[above left](B){$A_0$} % réécrit le nom
% du point B en l'appelant A0 et écrit au dessus à gauche
\end{tikzpicture}
```