# Tableaux de variations : 'tabvar'

Daniel Flipo
Daniel.Flipo@univ-lille1.fr

## 1 Documentation

L'extension tabvar.dtx [1], a pour but de faciliter la saisie des tableaux de variations. Elle s'appuie sur les extensions array, colortbl, et varwidth. Les flèches sont dessinées en MetaPost, ce qui permet de travailler soit en LaTeX standard, soit en pdfLaTeX. Une autre possibilité est de faire appel, pour les flèches, à une fonte (type 1) spécifique créée par Michel Bovani. Un grand merci à Michel pour cette contribution et pour ses remarques qui m'ont été très utiles pour améliorer les versions préliminaires.

### 1.1 Installation

Assurez-vous que l'extension varwidth.sty est présente sur votre système, sinon récupérez-les sur CTAN : cherchez par exemple la chaîne `varwidth` sur `http://www.tex.ac.uk/CTANfind.html`
Les fichiers tabvar.1, ... , tabvar.3 ainsi que tabvar.sty et tabvar.cfg doivent être placés dans un répertoire vu par LaTeX.
Les flèches peuvent être prises dans la fonte type 1 tabvar.pfb à condition que celle-ci soit installée : il est nécessaire de placer le fichier tabvar.pfb dans un répertoire où elle sera prise en compte, par exemple, pour respecter l'architecture TDS : `texmf/fonts/type1/public/tabvar`. De même, son fichier de métriques tabvar.tfm devra être mis par exemple dans `texmf/fonts/tfm/public/tabvar`.
Un ligne donnant accès à cette fonte doit être ajoutée (voir tabvar.map) dans les fichiers `.map` utilisés par le pilote PostScript (psfonts.map) et par pdfTeX (pdftex.map). Ne pas oublier de mettre à jour les bases de données ls-R pour terminer (commande `mktexlsr` sous teTeX ou TeXLive).

### 1.2 Utilisation

L'environnement `tabvar` est une variante de l'environnement `array`, adaptée à la saisie de tableaux de variations.
Trois nouveaux types de colonnes, `C`, `L` et `R` sont utilisés à la place des types classiques (`c`, `l` et `r`) ; ils permettent de disposer du matériel sur plusieurs niveaux dans un même ligne du tableau (ce sont des colonnes de type `\parbox`).
Un quatrième type de colonne, noté `N` sert pour les plages où la fonction est non définie. La colonne est entièrement grisée par défaut, mais il est possible de choisir une autre couleur (voir le fichier de configuration tabvar.cfg).
La saisie des lignes contenant les valeurs de la variable et les signes des dérivées se fait exactement comme celles d'un tableau `array`. Seules les lignes contenant

1. La version présentée ici porte le numéro 0.8, dernière modification le 06/04/2003.

les variations de la ou des fonctions font appel à quatre commandes particulières : `\croit`, `\decroit`, `\constante`, `\niveau`, `\dbarre` et `\discont`.

– Les commandes `\croit`, `\decroit` et `\constante` ne prennent pas d'argument, elles tracent les flèches montantes, descendantes ou horizontales.
– `\niveau`{*départ*}{*total*} prend deux arguments : le niveau (hauteur) où doit être positionnée la première valeur de la fonction et le nombre total de niveaux qui seront utilisés dans la ligne.
– `\dbarre` trace un double trait vertical dont la hauteur est celle de la ligne du tableau ; elle indique les discontinuités de la fonction.
– `\discont`[*num*]{*valeur_gauche*}{< ou >}{*valeur_droite*} peut s'utiliser lorsque la fonction présente une discontinuité à la place de la double barre traditionnelle ; elle prend trois arguments obligatoires : les valeurs à gauche $f_-$ et à droite $f_+$ de la fonction, séparées par un signe `<` ou `>` selon que $f_- < f_+$ ou $f_- > f_+$. Enfin, l'argument optionnel, qui vaut 0 par défaut, permet d'intercaler *num* niveaux supplémentaires entre les valeurs de $f_-$ et $f_+$ si nécessaire.

Le fichier demo.pdf (joint) propose plusieurs exemples, accompagnés de leur code source, illustrant les utilisations possibles de l'environnement `tabvar`.

# 2 Le code

## 2.1 Identification, extensions requises

```
1 ⟨*sty⟩
2 \NeedsTeXFormat{LaTeX2e}[1997/06/01]
3 \ProvidesClass{tabvar}[\filedate, v.\fileversion]
```

Chargement des extensions utiles :

```
4 \RequirePackage{array}
5 \RequirePackage{colortbl}
6 \RequirePackage{varwidth}
7 \RequirePackage{ifthen}
```

## 2.2 Dessin des flèches en MetaPost

Le fichier tabvar.mp (joint) contient le dessin des trois flèches.
La commande `mpost -tex=latex tabvar` produit trois fichiers tabvar.1... tabvar.3 qui contiennent les dessins des flèches ; en PDF, il faut indiquer qu'il s'agit de fichiers MetaPost.

```
8 \RequirePackage{graphicx}
9 \RequirePackage{ifpdf}
10 \ifpdf
11   \input{supp-pdf}
12   \DeclareGraphicsRule{*}{mps}{*}{}
13 \fi
```

`\FlecheC` `\FlecheD` `\FlecheH`

Le tracé des trois types de flèches est fait par les commandes `\FlecheC`, `\FlecheD` et `\FlecheH`.

```
14 \newsavebox{\arup}
```

```
\newsavebox{\ardown}
\newsavebox{\arhor}
\sbox{\arup}  {\includegraphics[scale=1.\@ptsize]{tabvar.1}}
\sbox{\ardown}{\includegraphics[scale=1.\@ptsize]{tabvar.2}}
\sbox{\arhor} {\includegraphics[scale=1.\@ptsize]{tabvar.3}}
\newcommand{\FlecheC}{\raisebox{.5ex}{\usebox{\arup}}}
\newcommand{\FlecheD}{\raisebox{.5ex}{\usebox{\ardown}}}
\newcommand{\FlecheH}{\raisebox{.5ex}{\usebox{\arhor}}}
```

## 2.3 Flèches comme caractères d'une fonte type 1

Si la fonte spécifique tabvar.pfb a été correctement installée, on pourra déclarer `\FlechesMPfalse` dans le fichier tabvar.cfg ou dans le préambule pour que les flèches soient prises dans cette fonte. Dans ce cas, la fonte sera déclarée et les commandes `\FlecheC`, `\FlecheD` et `\FlecheH` seront redéfinies au `\begin{document}`.

```
\newif\ifFlechesMP \FlechesMPtrue
\AtBeginDocument{%
  \ifFlechesMP
  \else
    \DeclareFontFamily{U}{tv}{}%
    \DeclareFontShape{U}{tv}{m}{n}{<->tabvar}{}%
    \DeclareSymbolFont{tvsymbols}{U}{tv}{m}{n}%
    \DeclareMathSymbol{\enearrow}{\mathrel}{tvsymbols}{"25}%
    \DeclareMathSymbol{\esearrow}{\mathrel}{tvsymbols}{"26}%
    \DeclareMathSymbol{\eastarrow}{\mathrel}{tvsymbols}{"21}%
    \renewcommand{\FlecheC}{\ensuremath{\enearrow}}%
    \renewcommand{\FlecheD}{\ensuremath{\esearrow}}%
    \renewcommand{\FlecheH}{\ensuremath{\eastarrow}}%
  \fi}
```

## 2.4 Positionnement vertical de éléments

La variable `\TVextraheight`, dont la valeur par défaut vaut `.7\baselineskip` permet d'écarter légèrement les valeurs maximales de la fonction, du filet horizontal supérieur.

```
\newdimen\TVextraheight
\setlength{\TVextraheight}{.7\baselineskip}
```

`\niveau` La commande `\niveau`, utilisée uniquement dans les lignes relatives aux valeurs des fonctions, permet d'initialiser les valeurs des compteurs `\@niveaux` (nombre total de niveaux utilisés dans la ligne) et `\@pos` (indicateur du niveau courant). Elle active également le drapeau `\if@socle` utilisé par la commande `\@socle`. Celle-ci place un filet invisible de hauteur `\TVextraheight` et ajoute `\@pos - 1` sauts de lignes (les colonnes sont alignées par le bas), ce qui assure le positionnement vertical de l'élément (valeur de la fonction ou flèche). Le drapeau `\if@socle` devra être mis localement à 'faux' dans certaines colonnes (cf. `\dbarre` et `discont`).

```
\newcount\@niveaux
```

```
\newcount\@pos
\newif\if@socle
\newcommand{\niveau}[2]{\global\@pos=#1 \global\@niveaux=#2
                        \global\@socletrue}
\newcommand{\@socle}{%
  \ifnum\@pos=1 \@soclefalse \fi
  \if@socle
    \rule{\z@}{\TVextraheight}%
    \@tempcnta=\@pos
    \advance\@tempcnta by -1
    \whiledo{\@tempcnta>0}{\TVnl \null \advance\@tempcnta by -1}%
  \fi}
```

## 2.5 Nouveaux types de colonnes

Ces définitions nécessitent les extensions `array` et `varwidth`. L'environnement `varwidth`, comme `minipage`, redéfinit la commande `\\`. On la renomme à l'intérieur des environnements `varwidth`, de façon à éviter la confusion entre passage à la ligne à l'intérieur d'une colonne et passage à la ligne suivante du tableau : `\TVnl` (commande interne) provoque un changement de ligne à l'intérieur d'une colonne, l'utilisateur peut continuer à utiliser `\\` pour terminer une ligne du tableau. La commande `\TVtabularnewline`, définie dans l'environnement `tabvar`, provoque un changement de ligne dans le tableau (`\tabularnewline`) et affecte la valeur 'vrai' au drapeau `\ifreset@niveaux`, ce qui commande la réinitialisation des compteurs `\@pos` et `\@niveaux` à la valeur 1. Cette réinitialisation aura lieu *après* que la commande `\@socle` ait placé les valeurs de la fonction et les flèches à la bonne hauteur.

```
\newif\ifreset@niveaux
\newcommand{\reset@niveaux}{%
  \ifreset@niveaux
    \global\@niveaux=1 \global\@pos=1 \global\@soclefalse
  \fi}
```

On définit des variantes `C`, `L` et `R`, des colonnes `c`, `l` et `r` : ce sont des *minipage* alignées par le bas, dont la largeur est celle de la ligne la plus longue, avec un maximum de `5em`, (voir la documentation de l'extension varwidth.sty).

```
\newcolumntype{C}{%
   >{\begin{varwidth}[b]{5em}\let\TVnl=\\ \let\\=\TVtabularnewline $}%
   c%
   <{\@socle \reset@niveaux
     $\@finalstrut\@arstrutbox\end{varwidth}}}
\newcolumntype{L}{%
   >{\begin{varwidth}[b]{5em}\let\TVnl=\\ \let\\=\TVtabularnewline $}%
   l%
   <{\@socle \reset@niveaux
     $\@finalstrut\@arstrutbox\end{varwidth}}}
\newcolumntype{R}{%
   >{\begin{varwidth}[b]{5em}\let\TVnl=\\ \let\\=\TVtabularnewline $}%
```

```
   r%
   <{\@socle \reset@niveaux
     $\@finalstrut\@arstrutbox\end{varwidth}}}
```

On définit également un type `N` pour les domaines où la fonction n'est pas définie : la colonne est coloriée en faisant appel à l'extension colortbl. La couleur peut être choisie par l'utilisateur, par exemple :
`\definecolor{TVcolor}{rgb}{0.66, 0.8, 0}`
donne un vert, voir color.sty pour la façon de définir des couleurs.

```
\definecolor{TVcolor}{gray}{0.7}
\newdimen\TVarraycolsep
\newdimen\TVcolorLeftSep
\newdimen\TVcolorRightSep
\setlength{\TVcolorLeftSep}{\TVarraycolsep}
\setlength{\TVcolorRightSep}{\TVarraycolsep}
\newcolumntype{N}{%
   >{\columncolor{TVcolor}[\TVcolorLeftSep][\TVcolorRightSep]}
   c}
```

## 2.6 Commandes de saisie

Les valeurs à afficher dans chaque ligne peuvent être saisies directement (`1.4`, `+`, `-`, etc.) comme dans un tableau normal. Les lignes correspondant aux valeurs des fonctions comportent plusieurs étages, nous disposons deux compteurs, `\@niveaux` qui contient le nombre total de niveaux (ou étages) utilisés dans la ligne, `\@pos` qui indique le niveau courant.

`\croit` `\decroit` `\constante`

Les commandes `\croit`,`\decroit` et `\constante` tracent les flèches à la hauteur adéquate et mettent à jour le compteur `\@pos`. Un message d'erreur est affiché lorsque l'une de ces commandes fait sortir de la plage de niveaux déclarés par la commande `\niveau`.

```
\newcommand{\decroit}{\FlecheD
                      \global\advance\@pos by -1
                      \ifnum\@pos<1
                      \PackageError{tabvar.sty}%
                        {Les arguments la commande
                         \protect\niveau\space sont incorrects}%
                      \fi}
\newcommand{\croit}  {\raisebox{-\baselineskip}{\FlecheC}%
                      \global\advance\@pos by 1
                      \ifnum\@pos>\@niveaux
                      \PackageError{tabvar.sty}%
                        {Les arguments la commande
                         \protect\niveau\space sont incorrects}%
                      \fi}
\newcommand{\constante}{\FlecheH}
```

`\dbarre`

La commande `\dbarre` sert à tracer les doubles barres La commande `\vline` ne peut pas être utilisée à cette fin dans les environnements de type `\parbox`, car sa

portée est limitée à un interligne.
On calcule la hauteur exacte de la rangée, dans les deux cas `\@niveaux=1` et `\@niveaux>1`, et on fait appel à `\rule` pour le tracé.

```
\newcommand{\dbarre}{\ifnum\@niveaux=1
                       \@tempdimc=\TVarraystretch\baselineskip
                     \else
                       \@tempcnta=\@niveaux
                       \advance\@tempcnta by -1
                       \@tempdimc=\@tempcnta\baselineskip
                       \@tempdimb=\TVextraheight
                       \ifdim\@tempdimb<.7\baselineskip
                         \@tempdimb=.7\baselineskip
                       \fi
                       \advance\@tempdimc by \@tempdimb
                       \advance\@tempdimc by \dp\@arstrutbox
                     \fi
                     \rule[-\dp\@arstrutbox]{.5\p@}{\@tempdimc}%
                     \kern 2\p@
                     \rule[-\dp\@arstrutbox]{.5\p@}{\@tempdimc}%
                     \@soclefalse}
```

`\discont` La commande `\discont` s'utilise lorsque la fonction présente une discontinuité, elle réclame 3 arguments obligatoires : le premier est la limite à gauche $f_-$, le deuxième le signe '<' ou '>', le troisième est la limite à droite $f_+$. LaTeX ne peut pas toujours comparer facilement les valeurs de $f_-$ et $f_+$ (penser à $f_- = \sqrt{e}$, $f_+ = \pi/2$), le deuxième argument précise si $f_- < f_+$ ou si $f_- > f_+$.
En plus de ces 3 arguments obligatoires, un argument optionnel (entier positif) permet d'écarter verticalement les valeurs $f_-$ et $f_+$ ; la valeur de cet entier donne le nombre de niveaux supplémentaires à intercaler (0 par défaut).
On commence par mesurer la largeur des deux arguments `#2` et `#4` pour pouvoir les centrer ensuite dans une boîte de largeur égale au maximum des deux largeurs. Si cette disposition ne convient pas, on pourra toujours ajouter un `\hfill` à droite où à gauche de la valeur à déplacer.

```
\newcommand{\discont}[4][0]{%
    \settowidth{\@tempdimc}{\ensuremath{#2}}%
    \settowidth{\@tempdimb}{\ensuremath{#4}}%
    \ifdim\@tempdimc<\@tempdimb \@tempdimc=\@tempdimb\fi
    \rule{\z@}{\TVextraheight}%
    \@soclefalse
    \ifthenelse{\equal{#3}{<}}%
```

Cas où $f_- < f_+$ : on pose la valeur de $f_+$ (`#4`), puis on saute autant de lignes supplémentaires qu'indiqué dans l'argument optionnel, ensuite on passe à la ligne et on pose la valeur de $f_-$ (`#2`), enfin on ajoute en dessous `\@pos - 1` sauts de lignes pour positionner le tout en hauteur. Il reste à ajuster le compteur `\@pos` pour que la flèche suivante soit placée à la bonne hauteur.

```
      {\makebox[\@tempdimc]{\ensuremath{#4}}%
       \@tempcnta=#1
```

```
      \whiledo{\@tempcnta>0}{\TVnl \null \advance\@tempcnta by -1}%
      \TVnl
      \makebox[\@tempdimc]{\ensuremath{#2}}%
      \@tempcnta=\@pos
      \advance\@tempcnta by -1
      \whiledo{\@tempcnta>0}{\TVnl \null \advance\@tempcnta by -1}%
      \global\advance\@pos by 1
      \global\advance\@pos by #1
     }%
     {\ifthenelse{\equal{#3}{>}}%
```

Cas où $f_- > f_+$ : *idem* en permutant $f_-$ et $f_+$.

```
     {\makebox[\@tempdimc]{\ensuremath{#2}}%
      \@tempcnta=#1
      \whiledo{\@tempcnta>0}{\TVnl \null \advance\@tempcnta by -1}%
      \TVnl
      \makebox[\@tempdimc]{\ensuremath{#4}}%
      \@tempcnta=\@pos
      \advance\@tempcnta by -2
      \advance\@tempcnta by -#1
      \whiledo{\@tempcnta>0}{\TVnl \null \advance\@tempcnta by -1}%
      \global\advance\@pos by -1
      \global\advance\@pos by -#1
     }%
```

Cas où le deuxième argument n'est ni < ni > : erreur

```
     {\PackageError{tabvar.sty}%
        {Le second argument de \protect\discont\space doit \^etre
         \MessageBreak soit '<' soit '>'}}%
     }%
}
```

## 2.7 Environnement 'tabvar'

L'environnement `tabvar` est un `array` où sont redéfinis `\TVarraystretch`, `\TVarraycolsep` et `\tabularnewline`.

`tabvar`

```
\newcommand{\TVarraystretch}{1.5}
\setlength{\TVarraycolsep}{1pt}
\newenvironment{tabvar}[1]
  {\renewcommand{\arraystretch}{\TVarraystretch}%
   \setlength{\arraycolsep}{\TVarraycolsep}%
   \global\@niveaux=1 \global\@pos=1 \global\@soclefalse
   \def\TVtabularnewline{\reset@niveauxtrue\tabularnewline}%
   \begin{array}{#1}}
  {\end{array}}
```

Chargement du fichier de préférences, si il en existe un.

```
\InputIfFileExists{tabvar.cfg}
```

```
159    {\typeout{loading tabvar.cfg}}
160    {\typeout{tabvar.cfg not found, using default values}}
```
</sty>

# 3 Fichier de configuration

```
161 ⟨*cfg⟩
162 %%% Fichier de configuration de l'extension `tabvar.sty'.
163 %%
164 %%% D\'ecommenter la ligne suivante pour que les fl\`eches
165 %%% soient prises dans la fonte tabvar.pfb, au lieu d'\^etre
166 %%% dessin\'ee en MetaPost.
167 %%\FlechesMPfalse
168 %%
169 %%% Ce param\`etre permet d'ajuster la hauteur des lignes
170 %%% de `tabvar' correspondant aux variations d'une fonction ;
171 %%% sa valeur par d\'efaut est :
172 %%\setlength{\TVextraheight}{0.7\baselineskip}
173 %%
174 %%% Valeur de \arraycolsep utilis\'ee dans `tabvar'.
175 %%\setlength{\TVarraycolsep}{1pt}
176 %%
177 %%% Valeur de \arraystretch utilis\'ee dans `tabvar'.
178 %%\renewcommand{\TVarraystretch}{1.5}
179 %%
180 %%% Exemples de d\'efinitions de couleurs pour les colonnes `N'
181 %%% o\`u la fonction est non d\'efinie.
182 %%\definecolor{TVcolor}{gray}{0.5}
183 %%\definecolor{TVcolor}{rgb}{0.33, 0.12, 0}
184 %%\definecolor{TVcolor}{cmyk}{0.91,0,0.88,0.12}
185 %%
186 %%% Les valeurs suivantes assurent que les colonnes `N' sont
187 %%% colori\'ees sur toute leur largeur.
188 %%\setlength{\TVcolorLeftSep}{\TVarraycolsep}
189 %%\setlength{\TVcolorRightSep}{\TVarraycolsep}
190 %%
191 ⟨/cfg⟩
```
<*sty>
</sty>