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 prises dans une fonte (type 1) spécialement créée par Michel Bovani. Depuis la version 1.5, quatre variantes de flèches sont proposées. Un grand merci à Michel pour cette contribution et pour ses remarques qui m'ont été très utiles pour améliorer les versions préliminaires.
Une autre possibilité (conservée uniquement pour préserver la compatibilité avec les premières versions de tabvar) consiste à faire appel, pour le dessin des flèches, à MetaPost : le fichier tabvar.mp permet de créer les trois flèches tabvar.1, tabvar.2, tabvar.3. Ceci est une solution de repli pour ceux qui auraient du mal à installer la fonte tabvar.pfb, à n'employer qu'en dernier recours.
Lorsqu'on travaille avec XeLaTeX ou LuaLaTeX il convient de placer l'appel à `\usepackage{tabvar}` *avant* `\usepackage{fontspec}`.

1.1 Installation

L'extension tabvar fait partie des distributions TeXLive, MacTeX, ProTeXt, MikTeX, etc. Si elle n'est pas installée chez vous, assurez-vous d'abord que l'extension varwidth.sty est présente sur votre système, sinon récupérez-la 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 sont prises dans la fonte type 1 tabvar.pfb à condition que celle-ci soit correctement installée : il est nécessaire de placer le fichier tabvar.pfb dans un répertoire où il sera pris 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` remplacent les 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`).

1. La version présentée ici porte le numéro v1.61, dernière modification le 2012/03/14.

Un quatrième type de colonne, noté `U`[2] sert pour les plages où la fonction n'est pas définie (`U` pour *Undefined*). La colonne est entièrement grisée par défaut, mais il est possible de choisir une autre couleur (voir le fichier de configuration tabvar.cfg). Désormais tabvar teste au `\begin{document}` si le type `N` a été défini par une autre extension, si c'est le cas un avertissement est affiché dans le fichier .log et tabvar n'écrase plus la définition du type `N`. Sinon, le type `N` est défini comme avant.
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 les variations de la ou des fonctions font appel à six 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. Le niveau le plus bas est numéroté 1.
- `\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.

Il est possible d'ajouter des filets d'alignement vertical en utilisant la commande `\barre{}` qui requiert un argument obligatoire, éventuellement vide : `\barre{}` trace un filet vertical dont la hauteur est celle de la ligne du tableau. Lorsqu'une valeur doit figurer sous le filet, on la passe en argument de la commande (`\barre{0}` par exemple), ainsi cette valeur sera centrée sur le filet. Ceci restreint évidemment l'usage de la commande `\barre` aux colonnes de type `C`. La couleur du filet (gris par défaut) est paramétrable, voir le fichier de configuration tabvar.cfg. Cette solution a été préférée à des pointillés qui posent des problèmes de raccordement d'une ligne à l'autre du tableau.
Depuis la version 1.5, quatre variantes sont proposées pour le dessin des flèches PostScript type 1, elles sont accessibles par les commandes `\FlechesPS1` (flèches « à moustaches » obtenues par défaut), `\FlechesPS2` (assorties à la police Fourier), `\FlechesPS3` et `\FlechesPS4`.
La commande `\TVcenter` prend un argument, elle sert à centrer verticalement le nom de la fonction, par exemple `\TVcenter{f(x)}`.
Le fichier demo.pdf (joint) propose plusieurs exemples, accompagnés de leur code source, illustrant les utilisations possibles de l'environnement `tabvar`.
Plusieurs commandes ou paramètres permettent de personnaliser l'aspect du tableau (augmentation de la largeur ou de la hauteur des cellules, etc.), voir le fichier tabvar.cfg.

2. Il était noté `N` jusqu'à la version 1.5, le type `N` est conservé pour assurer la compatibilité ascendante mais ne devrait plus être utilisé pour éviter un conflit avec l'extension numprint.sty.

2 Le code

2.1 Options

Depuis la version 0.9, les flèches utilisées par défaut sont prises dans la fonte type 1 de Michel Bovani.
Si cette fonte spécifique n'a pas pu être correctement installée, on pourra déclarer `\FlechesMPtrue` dans le fichier tabvar.cfg ou dans le préambule, ou encore utiliser l'option `FlechesMP` pour que les flèches MetaPost utilisées à la place. Cette solution est à proscrire lorsqu'on travaille avec XeLaTeX ou LuaLaTeX.

```
\newif\ifFlechesMP \FlechesMPfalse
\DeclareOption{FlechesMP}{\FlechesMPtrue}
\DeclareOption{FlechesPS}{\FlechesMPfalse}
\ProcessOptions
```

2.2 Identification, extensions requises

Chargement des extensions utiles :

```
\RequirePackage{array}
\RequirePackage{colortbl}
\RequirePackage{varwidth}
\RequirePackage{ifthen}
```

2.3 Dessin des flèches

Le fichier tabvar.mp (joint) contient le dessin des trois flèches en MetaPost.
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.

```
\RequirePackage{graphicx}
\RequirePackage{ifpdf}
\ifpdf
  \DeclareGraphicsRule{*}{mps}{*}{}
\fi
```

La mise à l'échelle des flèches MetaPost se fait à partir de la valeur de `\f@size` qui contient normalement la taille en points de la police de base (10 en 10pt). Si la classe utilisée ne définit pas `\f@size`, on donne la valeur 10 à `\f@size`, la valeur par défaut de `\TVarrowscale` est alors 1.0 (échelle 1), l'utilisateur peut toujours redéfinir lui-même `\TVarrowscale` selon ses besoins.

```
\providecommand{\f@size}{10}
\newcommand{\TVarrowscale}{\strip@pt\dimexpr\f@size pt/10\relax}
```

La commande `\TVarrowscolstretch`, dont la valeur par défaut est 1, permet d'augmenter la largeur des colonnes contenant les flèches.

`\TVarrowscolstretch`

```
\newcommand*{\TVarrowscolstretch}{1}
\newcommand*{\TV@arrowcol@stretch}[1]{%
     \makebox[\TVarrowscolstretch\width][c]{#1}}
```

La commande `\FlechesPS` suivie d'un chiffre compris entre 1 et 4 permet d'accéder à quatre variantes pour les flèches PostScript.

\FlechesPS

```
\newcommand*{\enearrow}{\enearrowi}
\newcommand*{\esearrow}{\esearrowi}
\newcommand*{\eastarrow}{\eastarrowi}
\newcommand*{\FlechesPS}[1]{%
  \renewcommand*{\enearrow}{%
    \csname enearrow\romannumeral#1\endcsname}%
  \renewcommand*{\esearrow}{%
    \csname esearrow\romannumeral#1\endcsname}%
  \renewcommand*{\eastarrow}{%
    \csname eastarrow\romannumeral#1\endcsname}%
}
```

\FlecheC
\FlecheD
\FlecheH

Le tracé des trois types de flèches est fait par les commandes \FlecheC, \FlecheD et \FlecheH. Le choix de la variante (MetaPost ou type 1) est fait au \begin{document}, ce qui économise une fonte mathématique parmi les 16 disponibles lorsqu'on choisit la variante MetaPost.

```
\AtBeginDocument{%
  \ifFlechesMP
    \newsavebox{\arup}%
    \newsavebox{\ardown}%
    \newsavebox{\arhor}%
    \sbox{\arup}{\includegraphics[scale=\TVarrowscale]{tabvar.1}}%
    \sbox{\ardown}{\includegraphics[scale=\TVarrowscale]{tabvar.2}}%
    \sbox{\arhor}{\includegraphics[scale=\TVarrowscale]{tabvar.3}}%
    \newcommand*{\FlecheC}{%
        \TV@arrowcol@stretch{\raisebox{.5ex}{\usebox{\arup}}}}%
    \newcommand*{\FlecheD}{%
        \TV@arrowcol@stretch{\raisebox{.5ex}{\usebox{\ardown}}}}%
    \newcommand*{\FlecheH}{%
        \TV@arrowcol@stretch{\raisebox{.5ex}{\usebox{\arhor}}}}%
  \else
    \DeclareFontFamily{U}{tv}{}%
    \DeclareFontShape{U}{tv}{m}{n}{<->tabvar}{}%
    \DeclareSymbolFont{tvsymbols}{U}{tv}{m}{n}%
    \DeclareMathSymbol{\eastarrowi}{\mathrel}{tvsymbols}{"21}%
    \DeclareMathSymbol{\enearrowi}{\mathrel}{tvsymbols}{"25}%
    \DeclareMathSymbol{\esearrowi}{\mathrel}{tvsymbols}{"26}%
    \DeclareMathSymbol{\eastarrowii}{\mathrel}{tvsymbols}{"31}%
    \DeclareMathSymbol{\enearrowii}{\mathrel}{tvsymbols}{"35}%
    \DeclareMathSymbol{\esearrowii}{\mathrel}{tvsymbols}{"36}%
    \DeclareMathSymbol{\eastarrowiii}{\mathrel}{tvsymbols}{"3B}%
    \DeclareMathSymbol{\enearrowiii}{\mathrel}{tvsymbols}{"3F}%
    \DeclareMathSymbol{\esearrowiii}{\mathrel}{tvsymbols}{"40}%
    \DeclareMathSymbol{\eastarrowiv}{\mathrel}{tvsymbols}{"46}%
    \DeclareMathSymbol{\enearrowiv}{\mathrel}{tvsymbols}{"4A}%
    \DeclareMathSymbol{\esearrowiv}{\mathrel}{tvsymbols}{"4B}%
    \newcommand*{\FlecheC}{%
            \TV@arrowcol@stretch{\ensuremath{\enearrow}}}%
    \newcommand*{\FlecheD}{%
            \TV@arrowcol@stretch{\ensuremath{\esearrow}}}%
    \newcommand*{\FlecheH}{%
            \TV@arrowcol@stretch{\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 `\TVmaxcolwidth` fixé à `\linewidth` par défaut, (voir la documentation de l'extension varwidth.sty).

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

On définit également un type `U` 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. L'ancien nom `N`, conservé pour la compatibilité ascendante, tant qu'il n'y a pas conflit, mais ne devrait plus être utilisé.

```
\definecolor{TVcolor}{gray}{0.7}
\newdimen\TVarraycolsep
\newdimen\TVcolorLeftSep
\newdimen\TVcolorRightSep
\setlength{\TVcolorLeftSep}{\TVarraycolsep}
\setlength{\TVcolorRightSep}{\TVarraycolsep}
\newcolumntype{U}{%
   >{\columncolor{TVcolor}[\TVcolorLeftSep][\TVcolorRightSep]}
   c}
\AtBeginDocument{%
  \@ifundefined{NC@find@N}%
     {\newcolumntype{N}{U}}%
     {\PackageWarning{tabvar}{Le type de colonne N est d\'efini par
                              ailleurs. \MessageBreak Remplacer N par
                              U dans \protect\begin{tabvar}{...N...}
                              \MessageBreak}}%
}
```

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`, `\@tempdimc` contient la hauteur totale (*totalheight*) et `\@tempdimb` la profondeur (*depth*).

```
\newcommand{\barre@dth}{%
   \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
   \@tempdimb=\dp\@arstrutbox}
```

On fait appel à `\rule` pour le tracé de `\dbarre`.

```
\newcommand{\dbarre}{%
   \barre@dth
```

```
    \rule[-\@tempdimb]{.5\p@}{\@tempdimc}%
    \kern 2\p@
    \rule[-\@tempdimb]{.5\p@}{\@tempdimc}%
    \@soclefalse}
```

`\barre` La commande `\barre` prend un argument obligatoire. `\barre{}` trace un filet vertical centré dans une colonne. Lorsque l'argument est non vide, celui-ci est superposé (centré) sur le filet. Le filet est tracé en gris par défaut (couleur paramétrable).

```
\newsavebox{\tab@box}
\definecolor{TVbarrecolor}{gray}{0.7}
\newcommand{\barre}[1]{%
    \sbox{\tab@box}{\ensuremath{#1}}%
    \barre@dth
    \@tempcnta=\@pos
    \advance\@tempcnta by -1
    \advance\@tempdimb by \@tempcnta\baselineskip
    \raisebox{-\@tempdimb}[0pt][0pt]{%
       \makebox[\wd\tab@box][c]{\color{TVbarrecolor}%
                                  \rule{.5\p@}{\@tempdimc}}}%
    \kern-\wd\tab@box\usebox{\tab@box}%
}
```

`\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 '>'}}%
     }%
}
```

`\TVcenter` La commande `\TVcenter{}` prend un argument, le nom de la fonction à centrer verticalement dans sa colonne.

```
\newcommand*{\TVcenter}[1]{%
  \@tempcnta=\@niveaux \advance\@tempcnta by -1 \divide\@tempcnta by 2
  \@tempdimb=\@tempcnta\baselineskip
  \ifodd\@niveaux\else\advance\@tempdimb by .5\baselineskip\fi
  \@pos=1\raisebox{\@tempdimb}{\ensuremath{#1}}%
}
```

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}
   {\typeout{loading tabvar.cfg}}
   {\typeout{tabvar.cfg not found, using default values}}
```

3 Fichier de configuration

```
%% Fichier de configuration de l'extension `tabvar.sty'.
%%
%% D\'ecommenter la ligne suivante pour que les variantes MetaPost
%% des fl\`eches soient utilis\'ees \`a la place de la fonte tabvar.pfb
%% (d\'econseill\'e en g\'en\'eral et *jamais* sous LuaLaTeX ou XeLaTeX).
%%
%%\FlechesMPtrue
%%
%% Choix d'une des 4 variantes pour les fl\`eches PostScript
%%
%%\FlechesPS1   % (d\'efaut)
%%\FlechesPS2   % assorties \`a la police Fourier
%%\FlechesPS3
%%\FlechesPS4
%%
%% Ce param\`etre permet d'augmenter la largeur des colonnes contenant
%% des fl\`eches (essayer 1.3, 1.5, etc.), sa valeur par d\'efaut est 1 :
%%
%%\renewcommand*{\TVarrowscolstretch}{1}
%%
%% Ce param\`etre permet d'ajuster la hauteur des lignes
%% de `tabvar' correspondant aux variations d'une fonction ;
%% sa valeur par d\'efaut est :
%%
%%\setlength{\TVextraheight}{0.7\baselineskip}
%%
%% Valeur de \arraycolsep utilis\'ee dans `tabvar'.
%%
%%\setlength{\TVarraycolsep}{1pt}
%%
%% Valeur de \arraystretch utilis\'ee dans `tabvar'.
%%
%%\renewcommand{\TVarraystretch}{1.5}
%%
%% Largeur maximale des colonnes de type C, L ou R.
%%
%%\setlength{\TVmaxcolwidth}{\linewidth}
%%
%% Exemples de d\'efinitions de couleurs pour les colonnes `U'
%% o\`u la fonction est non d\'efinie.
%%
%%\definecolor{TVcolor}{gray}{0.5}
%%\definecolor{TVcolor}{rgb}{0.33, 0.12, 0}
%%\definecolor{TVcolor}{cmyk}{0.91,0,0.88,0.12}
%%
%% Les valeurs suivantes assurent que les colonnes `U' sont
```

```
%% colori\'ees sur toute leur largeur.
%%
%%\setlength{\TVcolorLeftSep}{\TVarraycolsep}
%%\setlength{\TVcolorRightSep}{\TVarraycolsep}
%%
%% On peut ajuster comme ci-dessus la couleur des filets
%% tra\c{c}\'es par la commande \barre{}.
%%
%%\definecolor{TVbarrecolor}{gray}{0.7}
```

Change History