

Tableaux de variations : ‘tabvar’

Daniel FLIPO
Daniel.Flipo@free.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 L^AT_EX.

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 pdfT_EX (`pdftex.map`). Ne pas oublier de mettre à jour les bases de données ls-R pour terminer (commande `mktexlsr` sous teT_EX ou T_EXLive).

1.2 Utilisation

L’environnement `tabvar` est une variante de l’environnement `array`, adaptée à la saisie de tableaux de variations.

1. La version présentée ici porte le numéro v1.7, dernière modification le 2013/01/21.

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`).

Un quatrième type de colonne, noté **U**² sert pour les pages 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 : `\niveau`, `\croit`, `\decroit`, `\constante`, `\dbarre` et `\discont`.

- `\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.
- Les commandes `\croit`, `\decroit` et `\constante` ne prennent pas d'argument, elles tracent les flèches montantes, descendantes ou horizontales.
- `\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'insérer 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)}`. Elle a besoin de connaître le nombre total de niveaux, elle doit donc être précédée d'une commande `\niveau{départ}{total}`.

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`.

La commande `\TVstretch` est à utiliser lorsqu'un élément du tableau colle à la ligne horizontale du dessus ou du dessous³. `\TVstretch{valeur}` ajoute un peu d'espace paramétrable (2pt par défaut)⁴ au dessus et en dessous de *valeur*.

La commande `\TVstretch` accepte un argument optionnel de type *dimension* qui s'utilise lorsqu'on souhaite n'ajouter d'espace vertical que d'un côté (au-dessus ou au-dessous). Si l'argument optionnel est une dimension positive, sa valeur sera ajoutée uniquement au-dessus et si c'est une dimension négative sa valeur absolue sera ajoutée uniquement en dessous.

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`.

1.3 Incompatibilités

`tabvar` est incompatible avec les extensions qui définissent les mêmes types de colonnes (C, L, R, U) pour d'autres usages, en particulier `tabulary`.

L'utilisation simultanée de `tabvar`, `cellspace`, et `siunitx` est pour l'instant impossible : `cellspace` et `siunitx` utilisent tous les deux le paramètre `S`, pour éviter le conflit, `siunitx` redéfinit le `S` de `cellspace` en `C` ce qui provoque un autre conflit avec `tabvar` ! Le problème a été signalé à l'auteur de `siunitx`.

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.

```
1 \newif\ifflechesMP \FlechesMPfalse
2 \DeclareOption{FlechesMP}{\FlechesMPtrue}
3 \DeclareOption{FlechesPS}{\FlechesMPfalse}
4 \ProcessOptions
```

3. Noter que les extensions `cellspace` et `tabls` qui règlent ce problème n'ont pas d'effet sur l'environnement `tabvar` et que `tabls` est incompatible avec `array` chargé par `tabvar`.

4. La valeur de l'espace ajouté au-dessus est définie par `\setlength{\TVextraheight}{2pt}`, celle de l'espace ajouté au-dessous est définie par `\setlength{\TVextradepth}{2pt}`.

2.2 Identification, extensions requises

Chargement des extensions utiles :

```
5 \RequirePackage{array}
6 \RequirePackage{colortbl}
7 \RequirePackage{varwidth}
8 \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.

```
9 \RequirePackage{graphicx}
10 \RequirePackage{ifpdf}
11 \ifpdf
12   \DeclareGraphicsRule{*}{mps}{*}{}
13 \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 `\TVarrowsscale` est alors 1.0 (échelle 1), l'utilisateur peut toujours redéfinir lui-même `\TVarrowsscale` selon ses besoins.

```
14 \providecommand{\f@size}{10}
15 \newcommand{\TVarrowsscale}{\strip@pt\dimexpr\f@size pt/10\relax}
```

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

`\TVarrowsscolstretch`

```
16 \newcommand*{\TVarrowsscolstretch}{1}
17 \newcommand*{\TV@arrowcol@stretch}[1]{%
18   \makebox[\TVarrowsscolstretch\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`

```
19 \newcommand*{\enearrow}{\enearrowi}
20 \newcommand*{\esearrow}{\esearrowi}
21 \newcommand*{\eatarrow}{\eatarrowi}
22 \newcommand*{\FlechesPS}[1]{%
23   \renewcommand*{\enearrow}{%
24     \csname enearrow\romannumeral#1\endcsname}%
```

```

25 \renewcommand*{\esearrow}{%
26   \csname esearrow\romannumeral#1\endcsname}%
27 \renewcommand*{\easarrow}{%
28   \csname easarrow\romannumeral#1\endcsname}%
29 }

```

\FlecheC Le tracé des trois types de flèches est fait par les commandes **\FlecheC**,
\FlecheD **\FlecheD** et **\FlecheH**. Le choix de la variante (MetaPost ou type1) est fait
\FlecheH au `\begin{document}`, ce qui économise une fonte mathématique parmi les 16
disponibles lorsqu'on choisit la variante MetaPost.

```

30 \AtBeginDocument{%
31   \ifFlechesMP
32     \newsavebox{\arup}%
33     \newsavebox{\ardown}%
34     \newsavebox{\arhor}%
35     \sbox{\arup}{\includegraphics[scale=\TVarrowscale]{tabvar.1}}%
36     \sbox{\ardown}{\includegraphics[scale=\TVarrowscale]{tabvar.2}}%
37     \sbox{\arhor}{\includegraphics[scale=\TVarrowscale]{tabvar.3}}%
38     \newcommand*{\FlecheC}{%
39       \TV@arrowcol@stretch{\raisebox{.5ex}{\usebox{\arup}}}%
40     \newcommand*{\FlecheD}{%
41       \TV@arrowcol@stretch{\raisebox{.5ex}{\usebox{\ardown}}}%
42     \newcommand*{\FlecheH}{%
43       \TV@arrowcol@stretch{\raisebox{.5ex}{\usebox{\arhor}}}%
44   \else
45     \DeclareFontFamily{U}{tv}{}%
46     \DeclareFontShape{U}{tv}{m}{n}{<->tabvar}{}%
47     \DeclareSymbolFont{tvsymbols}{U}{tv}{m}{n}%
48     \DeclareMathSymbol{\easarrowi}{\mathrel}{tvsymbols}{"21}%
49     \DeclareMathSymbol{\enearrowi}{\mathrel}{tvsymbols}{"25}%
50     \DeclareMathSymbol{\esearrowi}{\mathrel}{tvsymbols}{"26}%
51     \DeclareMathSymbol{\easarrowii}{\mathrel}{tvsymbols}{"31}%
52     \DeclareMathSymbol{\enearrowii}{\mathrel}{tvsymbols}{"35}%
53     \DeclareMathSymbol{\esearrowii}{\mathrel}{tvsymbols}{"36}%
54     \DeclareMathSymbol{\easarrowiii}{\mathrel}{tvsymbols}{"3B}%
55     \DeclareMathSymbol{\enearrowiii}{\mathrel}{tvsymbols}{"3F}%
56     \DeclareMathSymbol{\esearrowiii}{\mathrel}{tvsymbols}{"40}%
57     \DeclareMathSymbol{\easarrowiv}{\mathrel}{tvsymbols}{"46}%
58     \DeclareMathSymbol{\enearrowiv}{\mathrel}{tvsymbols}{"4A}%
59     \DeclareMathSymbol{\esearrowiv}{\mathrel}{tvsymbols}{"4B}%
60     \newcommand*{\FlecheC}{%
61       \TV@arrowcol@stretch{\ensuremath{\enearrow}}}%
62     \newcommand*{\FlecheD}{%
63       \TV@arrowcol@stretch{\ensuremath{\esearrow}}}%
64     \newcommand*{\FlecheH}{%
65       \TV@arrowcol@stretch{\ensuremath{\easarrow}}}%
66   \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 horizon-

tal supérieur.

```
67 \newdimen\TVextraheight
68 \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`).

```
69 \newcount\@niveaux
70 \newcount\@pos
71 \newif\if@socle
72 \newcommand{\niveau}[2]{\global\@pos=#1 \global\@niveaux=#2
73 \global\@socletrue}
74 \newcommand{\@socle}{%
75 \ifnum\@pos=1 \@soclefalse \fi
76 \if@socle
77 \rule{\z@}{\TVextraheight}%
78 \@tempcnta=\@pos
79 \advance\@tempcnta by -1
80 \whiledo{\@tempcnta>0}{\TVnl \null \advance\@tempcnta by -1}%
81 \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.

```
82 \newif\ifreset@niveaux
83 \newcommand{\reset@niveaux}{%
84 \ifreset@niveaux
85 \global\@niveaux=1 \global\@pos=1 \global\@soclefalse
86 \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`).

```

87 \newdimen\TVmaxcolwidth
88 \setlength{\TVmaxcolwidth}{\linewidth}
89 \newcolumnntype{C}{%
90   >{\begin{varwidth}[b]{\TVmaxcolwidth}\let\TVnl=\
91     \let\=\TVtabularnewline $}%
92   c%
93   <{\@socle \reset@niveaux
94     $\@finalstrut\@arstrutbox\end{varwidth}}}
95 \newcolumnntype{L}{%
96   >{\begin{varwidth}[b]{\TVmaxcolwidth}\let\TVnl=\
97     \let\=\TVtabularnewline $}%
98   l%
99   <{\@socle \reset@niveaux
100     $\@finalstrut\@arstrutbox\end{varwidth}}}
101 \newcolumnntype{R}{%
102   >{\begin{varwidth}[b]{\TVmaxcolwidth}\let\TVnl=\
103     \let\=\TVtabularnewline $}%
104   r%
105   <{\@socle \reset@niveaux
106     $\@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.

```

107 \definecolor{TVcolor}{gray}{0.7}
108 \newdimen\TVarraycolsep
109 \newdimen\TVcolorLeftSep
110 \newdimen\TVcolorRightSep
111 \setlength{\TVcolorLeftSep}{\TVarraycolsep}
112 \setlength{\TVcolorRightSep}{\TVarraycolsep}
113 \newcolumnntype{U}{%
114   >{\columncolor{TVcolor}[\TVcolorLeftSep][\TVcolorRightSep]}
115   c}
116 \AtBeginDocument{%
117   \@ifundefined{NC@find@N}%
118     {\newcolumnntype{N}{U}}%
119     {\PackageWarning{tabvar}{Le type de colonne N est d'efini par
120       ailleurs. \MessageBreak Remplacer N par
121       U dans \protect\begin{tabvar}{...N...}
122       \MessageBreak}}%
123 }

```

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` 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
`\decroit`
`\constante` lorsque l'une de ces commandes fait sortir de la plage de niveaux déclarés par la commande `\niveau`.

```

124 \newcommand{\decroit}{\FlecheD
125             \global\advance\@pos by -1
126             \ifnum\@pos<1
127             \PackageError{tabvar.sty}%
128             {Les arguments la commande
129             \protect\niveau\space sont incorrects}%
130             \fi}
131 \newcommand{\croit}  {\raisebox[-\baselineskip]{\FlecheC}%
132             \global\advance\@pos by 1
133             \ifnum\@pos>\@niveaux
134             \PackageError{tabvar.sty}%
135             {Les arguments la commande
136             \protect\niveau\space sont incorrects}%
137             \fi}
138 \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*).

```

139 \newcommand{\barre@nth}{%
140     \ifnum\@niveaux=1
141     \@tempdimc=\TVarraystretch\baselineskip
142     \else
143     \@tempcnta=\@niveaux
144     \advance\@tempcnta by -1
145     \@tempdimc=\@tempcnta\baselineskip
146     \@tempdimb=\TVextraheight
147     \ifdim\@tempdimb<.7\baselineskip
148     \@tempdimb=.7\baselineskip
149     \fi
150     \advance\@tempdimc by \@tempdimb
151     \advance\@tempdimc by \dp\@arstrutbox
152     \fi
153     \@tempdimb=\dp\@arstrutbox}

```

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

```

154 \newcommand{\dbarre}{%
155   \barre@PTH
156   \rule[-\@tempdimb]{.5\p@}{\@tempdimc}%
157   \kern 2\p@
158   \rule[-\@tempdimb]{.5\p@}{\@tempdimc}%
159   \@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).

```

160 \newsavebox{\tab@box}
161 \definecolor{TVbarrecolor}{gray}{0.7}
162 \newcommand{\barre}[1]{%
163   \sbox{\tab@box}{\ensuremath{#1}}%
164   \barre@PTH
165   \@tempcnta=\@pos
166   \advance\@tempcnta by -1
167   \advance\@tempdimb by \@tempcnta\baselineskip
168   \raisebox[-\@tempdimb][Opt][Opt]{%
169     \makebox[\wd\tab@box][c]{\color{TVbarrecolor}%
170       \rule{.5\p@}{\@tempdimc}}}%
171   \kern-\wd\tab@box\usebox{\tab@box}%
172 }

```

`\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 ou à gauche de la valeur à déplacer.

```

173 \newcommand{\discont}[4][0]{%
174   \settowidth{\@tempdimc}{\ensuremath{#2}}%
175   \settowidth{\@tempdimb}{\ensuremath{#4}}%
176   \ifdim\@tempdimc<\@tempdimb \@tempdimc=\@tempdimb\fi
177   \rule{\z@}{\TVextraheight}%
178   \@soclefalse
179   \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.

```

180   {\makebox[\@tempdimc]{\ensuremath{#4}}}%
181   \@tempcnta=#1
182   \whiledo{\@tempcnta>0}{\TVnl \null \advance\@tempcnta by -1}%
183   \TVnl
184   \makebox[\@tempdimc]{\ensuremath{#2}}}%
185   \@tempcnta=\@pos
186   \advance\@tempcnta by -1
187   \whiledo{\@tempcnta>0}{\TVnl \null \advance\@tempcnta by -1}%
188   \global\advance\@pos by 1
189   \global\advance\@pos by #1
190   }%
191   {\ifthenelse{\equal{#3}{>}}}%

```

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

```

192   {\makebox[\@tempdimc]{\ensuremath{#2}}}%
193   \@tempcnta=#1
194   \whiledo{\@tempcnta>0}{\TVnl \null \advance\@tempcnta by -1}%
195   \TVnl
196   \makebox[\@tempdimc]{\ensuremath{#4}}}%
197   \@tempcnta=\@pos
198   \advance\@tempcnta by -2
199   \advance\@tempcnta by -#1
200   \whiledo{\@tempcnta>0}{\TVnl \null \advance\@tempcnta by -1}%
201   \global\advance\@pos by -1
202   \global\advance\@pos by -#1
203   }%

```

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

```

204   {\PackageError{tabvar.sty}%
205     {Le second argument de \protect\discont\space doit \^etre
206     \MessageBreak soit '<' soit '>'}}%
207   }%
208 }

```

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

```

209 \newcommand*\TVcenter}[1]{%
210   \@tempcnta=\@niveaux \advance\@tempcnta by -1 \divide\@tempcnta by 2
211   \@tempdimb=\@tempcnta\baselineskip
212   \ifodd\@niveaux\else\advance\@tempdimb by .5\baselineskip\fi
213   \@pos=1\raisebox{\@tempdimb}{\ensuremath{#1}}}%
214 }

```

\TVstretch La commande `\TVstretch{}` prend un argument obligatoire, le nom de la valeur à écarter des lignes horizontales et un argument optionnel de type *dimension*. Elle

place son argument dans une boîte et en mesure la hauteur et la profondeur. En l'absence d'argument optionnel, elle ajoute respectivement `\TVextraheight` à la hauteur `\TV@tempa` et `\TVextradepth` à la profondeur `\TV@tempb`.

Si l'argument optionnel est une dimension positive, sa valeur sera ajoutée uniquement à la hauteur et si c'est une dimension négative sa valeur absolue sera ajoutée uniquement à la profondeur. La valeur par défaut de l'argument optionnel est `0pt`.

```

215 \newsavebox\TVbox
216 \newdimen\TVextraheight
217 \newdimen\TVextradepth
218 \setlength{\TVextraheight}{2pt}
219 \setlength{\TVextradepth}{2pt}
220 \newdimen\TV@tempa
221 \newdimen\TV@tempb
222 \newcommand{\TVstretch}[2][0pt]{%
223   \edef\tmp{#1}%
224   \sbox{\TVbox}{\ensuremath{#2}}%
225   \settoheight{\TV@tempa}{\usebox{\TVbox}}%
226   \settodepth{\TV@tempb}{\usebox{\TVbox}}%
227   \ifdim\tmp=0pt
228     \addtolength{\TV@tempa}{\TVextraheight}%
229     \addtolength{\TV@tempb}{\TVextradepth}%
230   \else
231     \ifdim\tmp>0pt
232       \addtolength{\TV@tempa}{\tmp}%
233     \else
234       \addtolength{\TV@tempb}{-\tmp}%
235   \fi
236 \fi

```

Il reste à afficher la boîte initiale et à lui adjoindre une `\rule` invisible (de largeur nulle) de profondeur `\TV@tempb` et de hauteur totale `\TV@tempa + \TV@tempb`.

```

237 \usebox{\TVbox}%
238 \addtolength{\TV@tempa}{\TV@tempb}%
239 \rule[-\TV@tempb]{0pt}{\TV@tempa}%
240 }

```

2.7 Environnement ‘tabvar’

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

`tabvar`

```

241 \newcommand{\TVarraystretch}{1.5}
242 \setlength{\TVarraycolsep}{1pt}
243 \newenvironment{tabvar}[1]
244   {\renewcommand{\arraystretch}{\TVarraystretch}}%
245   \setlength{\arraycolsep}{\TVarraycolsep}}%
246   \global@niveaux=1 \global@pos=1 \global@soclefalse
247   \def\TVtabularnewline{\reset@niveauxtrue\tabularnewline}%

```

```
248 \begin{array}{#1}}
249 {\end{array}}
```

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

```
250 \InputIfFileExists{tabvar.cfg}
251 {\typeout{loading tabvar.cfg}}
252 {\typeout{tabvar.cfg not found, using default values}}
```

3 Fichier de configuration

```
253 %% Fichier de configuration de l'extension 'tabvar.sty'.
254 %%
255 %% D\ecommenter la ligne suivante pour que les variantes MetaPost
256 %% des fl\eches soient utilis\ees \a la place de la fonte tabvar.pfb
257 %% (d\econseill\e en g\en\eral et *jamais* sous LuaLaTeX ou XeLaTeX).
258 %%
259 %%\FlechesMPtrue
260 %%
261 %% Choix d'une des 4 variantes pour les fl\eches PostScript
262 %%
263 %%\FlechesPS1 % (d\efault)
264 %%\FlechesPS2 % assorties \a la police Fourier
265 %%\FlechesPS3
266 %%\FlechesPS4
267 %%
268 %% Ce param\etre permet d'augmenter la largeur des colonnes contenant
269 %% des fl\eches (essayer 1.3, 1.5, etc.), sa valeur par d\efault est 1 :
270 %%
271 %%\renewcommand*\TVarrowscolstretch}{1}
272 %%
273 %% Ce param\etre permet d'ajuster la hauteur des lignes
274 %% de 'tabvar' correspondant aux variations d'une fonction ;
275 %% sa valeur par d\efault est :
276 %%
277 %%\setlength{\TVextraheight}{0.7\baselineskip}
278 %%
279 %% Valeur de \arraycolsep utilis\ee dans 'tabvar'.
280 %%
281 %%\setlength{\TVarraycolsep}{1pt}
282 %%
283 %% Valeur de \arraystretch utilis\ee dans 'tabvar'.
284 %%
285 %%\renewcommand{\TVarraystretch}{1.5}
286 %%
287 %% Largeur maximale des colonnes de type C, L ou R.
288 %%
289 %%\setlength{\TVmaxcolwidth}{\linewidth}
290 %%
291 %% Valeur des espaces verticaux ajout\es par la commande
292 %% |\TVstretch{ }|.
293 %%
294 %%\setlength{\TVextraheight}{2pt}
```

```

295 %%\setlength{\TVextradepth}{2pt}
296 %%
297 %% Exemples de d\efinitions de couleurs pour les colonnes 'U'
298 %% o\ 'u la fonction est non d\efinie.
299 %%
300 %%\definecolor{TVcolor}{gray}{0.5}
301 %%\definecolor{TVcolor}{rgb}{0.33, 0.12, 0}
302 %%\definecolor{TVcolor}{cmymk}{0.91,0,0.88,0.12}
303 %%
304 %% Les valeurs suivantes assurent que les colonnes 'U' sont
305 %% colori\ 'ees sur toute leur largeur.
306 %%
307 %%\setlength{\TVcolorLeftSep}{\TVarraycolsep}
308 %%\setlength{\TVcolorRightSep}{\TVarraycolsep}
309 %%
310 %% On peut ajuster comme ci-dessus la couleur des filets
311 %% tra\c{c}\ 'es par la commande \barre{ }.
312 %%
313 %%\definecolor{TVbarrecolor}{gray}{0.7}

```

Historique des versions

tabvar-0.9	General : Ajout de deux options pour le choix du type de flèches (suggestion de Frank Stengel). . . 3	tabvar-1.3	General : Valeur de la profondeur <code>\@tempdimb</code> corrigée dans <code>\barre</code> . Bug signalé par Frank Stengel. 9
	Par défaut, les flèches sont prises maintenant dans la fonte type 1 de Michel Bovani. 3	tabvar-1.4	General : Ajout d'une commande <code>\TVarrowscolstretch</code> permettant d'augmenter la largeur des colonnes contenant les flèches. . 4
tabvar-1.0	General : Ajout d'un paramètre <code>\TVmaxcolwidth</code> pour choisir la largeur maximale des colonnes de type C, L, R, au lieu d'une valeur fixe. 6	tabvar-1.5	General : La définition des flèches en MetaPost est déplacée (At-BeginDocument) pour éviter un message d'erreur avec XeLaTeX. Bug signalé par Thierry Wybrecht. 4
tabvar-1.1	General : Ajout de la commande <code>\barre</code> . Ajout de <code>\barre@dt</code> pour le calcul de la hauteur d'une rangée (utilisée par <code>\barre</code> et <code>\dbarre</code>). 8	tabvar-1.6	Trois nouvelles formes de flèches dessinées par Michel Bovani. Ajout de la commande <code>\FlechesPS</code> pour y accéder. . . . 4
tabvar-1.2	General : <code>\@ptsize</code> peut prendre des valeurs négatives (classes koma-script avec tailles inférieures à 10pt). Utiliser <code>\f@size</code> à la place (patch proposé par Ulrike Fisher, merci Ulrike). . . 4		
tabvar-1.2b	General : Augmenter la valeur de <code>\TVmaxcolwidth</code> à <code>\linewidth</code>). 6		

après tabvar et inversement (cf. array.dtx).	7	phics.sty appelle pdftex.def qui charge supp-pdf.mkii.	4
tabvar-1.61 General : Charger supp-pdf.tex est inutile : en mode PDF, gra-		tabvar-1.7 General : Ajout de la commande <code>\TVstretch</code>	10