L'extension tablvar*

Antoine Missier
antoine.missier@ac-toulouse.fr

9 août 2024

Table des matières

1 Introduction

Cette extension permet de construire des tableaux de variations et de signes de manière simple et intuitive. Bien que plusieurs autres extensions soient déjà dédiées à cette tâche [1], nous pensons que tablvar possède quelques avantages : une syntaxe simple, une haute qualité graphique, une variété de positionnements et

*Ce document correspond à tablvar v2.1, dernière modification le 09/08/2024.

1. Mentionnons tabvar [2] de Daniel Flipo et variations [3] de Christian Obrecht ou, plus élaborés, tablor [4] de Guillaume Connan et tkz-tab [6] de Alain Matthes. Nous présentons un comparatif à la section 2.8

de réglages, une documentation soignée. Les dessins, pour les flèches ou les zones interdites, sont obtenus en exploitant les fonctionnalités des extensions graphiques PSTricks (plus exactement pst-node) ou Ti*k*Z. Les flèches sont réalisées à partir d'un *graphe* de *nœuds* venant se « superposer » au tableau lui-même.

Le parti pris est d'utiliser la même syntaxe que les environnements `array` (ou `tabular`) en laissant à LaTeX le soin de faire la composition du tableau et à l'extension graphique (PSTricks ou Ti*k*Z) celui de réaliser le dessin des flèches, automatisé et sans intervention de l'utilisateur. On a simplement besoin de préciser, dans une commande `\variations`, ce qui est en haut et ce qui est en bas.

L'extension possède deux options, utilisées pour le tracé automatique des flèches : `pstricks` ou `tikz`, que l'on invoque dans le préambule à l'appel de l'extension : `\usepackage[tikz]{tablvar}`. Lorsque l'option n'est pas précisée, l'extension choisira `tikz` pour un mode de sortie PDF direct (avec pdfLaTeX ou LuaLaTeX), `pstricks` sinon (avec LaTeX > dvips > ps2pdf ou XeLaTeX). Pour l'option `tikz`, il faut compiler *deux fois*, la première fois les flèches ne sont pas correctement dessinées (un message de compilation le rappelle).

Cette documentation donne une galerie d'exemples, décrit les commandes fournies ainsi que différentes possibilités de réglages et d'ajustements personnels.

2 Utilisation

2.1 Tableaux de variations simples

`tablvar` (*env.*) Un tableau de variations (ou de signes) se définit par un environnement `tablvar` qui *doit être en mode mathématique* (comme `array`).

`\haut` `\bas` `\mil` Voici un premier tableau tout simple avec les commandes de positionnement naïves `\haut`, `\bas` et `\mil` (milieu).

x	-5		-1		2		3		5
$f'(x)$		$+$	0	$-$	0	$+$	0	$-$	
$f(x)$	0	↗	2	↘	1	↗	4	↘	-3

```
\[\begin{tablvar}{4}
    \hline
    x & -5 && -1 && 2 && 3 && 5 \\
    \hline
    f'(x) & & + & 0 & - & 0 & + & 0 & - & \\
    \hline
    \variations{ \mil{f(x)} & \bas{0} && \haut{2} && \bas{1} &&
        \haut{4} && \bas{-3} }
    \hline
\end{tablvar}\]
```

L'argument obligatoire de l'environnement `tablvar` correspond au nombre d'intervalles du tableau c'est-à-dire aussi au nombre de flèches (ici 4). Les délimiteurs `&` correspondent aux changements de colonnes comme pour l'environnement `array`. Outre la première colonne de *légendes*, il y a deux types de colonnes : des colonnes de *valeurs*, de largeur fixe (`2em` par défaut) et centrées, et des colonnes *intervalles* de largeur fixe (`3em` par défaut) [2]. La longueur des flèches s'ajuste en fonction de la taille des valeurs qu'elles relient et ne sont pas contraintes à rester à l'intérieur des colonnes intervalles. Lorsqu'une valeur est trop large, elle déborde automatiquement dans les colonnes intervalles sans altérer les largeurs de colonne.

`\pos` Les commandes `\haut` et `\bas` reposent sur une macro plus générale qui est `\pos[`⟨*opt*⟩`]{`⟨*ligne*⟩`}{`⟨*valeur*⟩`}`. Celle-ci place la ⟨*valeur*⟩ sur la ⟨*ligne*⟩ indiquée en créant un nœud pour les flèches. La ligne des x et les lignes contenant dérivée et tableau de signes ont pour indice 0. Les lignes de la partie variations sont numérotées 1, 2, 3, *du haut vers le bas*, dans le sens de l'écriture et de la construction du tableau. Le paramètre optionnel ⟨*opt*⟩ [3] permet d'ajuster le positionnement des flèches : `c` (centered, par défaut), `b` (bottom) ou `t` (top).

`\pos*` La commande `\mil`, utilisée pour le $f(x)$ dans la première colonne, repose, elle, sur la commande plus générale `\pos*{`⟨*ligne*⟩`}{`⟨*valeur*⟩`}` qui se comporte comme `\pos` mais ne crée pas de nœud pour les flèches.

Le second tableau de l'exemple ci-dessous est obtenu avec le code suivant qui utilise les commandes `\pos` à la place de `\haut` et `\bas` (avec l'option `t` pour le max) et `\pos*` à la place de `\mil`. Comparer le positionnement des flèches.

x	0		$\frac{1}{2}$		$+\infty$
$f'(x)$		$+$	0	$-$	
$f(x)$	2	↗	$f(\frac{1}{2})$	↘	0

x	0		$\frac{1}{2}$		$+\infty$
$f'(x)$		$+$	0	$-$	
$f(x)$	2	↗	$f(\frac{1}{2})$	↘	0

```
\[\begin{tablvar}{2}
    \hline
    x & 0 & & \frac{1}{2} & & +\infty \\
    \hline
    f'(x) & & + & 0 & - & \\
    \hline
    \variations{ \pos*{2}{f(x)} & \pos{3}{2} &&
        \pos[t]{1}{f(\frac{1}{2})} && \pos{3}{0} }
    \hline
\end{tablvar}\]
```

`tablvar[`⟨*largeur*⟩`]` (*env.*) `intervalwidth` L'environnement `tablvar` accepte un argument optionnel qui règle la largeur des colonnes intervalles. L'on peut aussi passer diverses options avec la syntaxe

2. Une dimension peut être exprimée cm, pt, etc. Le `em` correspond à la largeur du « m » dans la fonte utilisée. Cette unité à l'avantage de s'ajuster lorsqu'on change de fonte sans avoir besoin de modifier le code. L'unité de même type utilisée pour les hauteurs est le `ex`, hauteur du « x ».

3. Ce paramètre optionnel n'est pas implémenté et est sans effet pour l'option `tikz`.

clé=valeur, par exemple la largeur des colonnes intervalles peut également se régler avec `intervalwidth=`⟨*largeur*⟩ (voir section 2.6 pour la liste détaillée des options).

`\variations[`⟨*nblignes*⟩`]` C'est la commande `\variations` qui se charge de la construction des flèches en fonction du positionnement des valeurs. Un paramètre optionnel permet de définir un nombre arbitraire de lignes pour les variations (3 par défaut).

Voici un exemple avec un seul intervalle, utilisant l'argument optionnel `4em` de `tablvar`, afin d'allonger un peu la largeur de l'intervalle, et une partie variations construite uniquement sur 2 lignes. La commande `\mil` positionne automatiquement son contenu avec un décalage vertical adéquat.

```
\[\begin{tablvar}[4em]{1}
    \hline
    x & -5 && +\infty \\
    \hline
    \variations[2]{ \mil{f(x)} & \haut{2} &&
        \bas{-1} }
    \hline
\end{tablvar} \]
```

x	-5		$+\infty$
$f(x)$	2	↘	-1

`colvalwidth` La largeur des colonnes de valeurs est fixée par défaut à `2em`, ce qui est légèrement plus large qu'un infini avec un signe. Lorsque toutes les valeurs sont étroites, on peut éventuellement diminuer la largeur des colonnes intervalles pour éviter des espaces larges sur les bords du tableau. Il est cependant rarement utile de modifier cette valeur.

x	0		1		2		3		5
$f(x)$	0	↗	1	→	1	↗	4	↘	0

Pour placer des nœuds de flèches au milieu (sur la ligne 2), on doit utiliser `\pos{2}` et non `\mil`. Lorsqu'il y a une autre option dans l'environnement `tablvar`, il faut utiliser la syntaxe complète `intervalwidth=...` pour modifier la largeur des colonnes intervalles.

```
\[\begin{tablvar}[colvalwidth=1em,intervalwidth=3.5em]{4}
    \hline
    x & -5 && 0 && 2 && 3 && 5 \\
    \hline
    \variations{ \pos*{2}{f(x)} & \pos{3}{-4} && \pos{2}{1} && \pos{2}{1}
        && \pos{1}{4} && \pos{3}{0} }
    \hline
\end{tablvar}\]
```

2.2 Tableaux de signes avec barres de séparation

`\barre` La macro `\barre`, présentée dans l'exemple ci-dessous, sert de séparateur de colonne dans les tableaux de signes. Elle est bien sûr facultative et possède un paramètre optionnel pour placer un 0 sur la barre. Voici un exemple avec

$$f(x) = -4x^3 + 3x^2 + 18x - 3 \quad \text{et} \quad f'(x) = 6(x+1)(-2x+3).$$

x	$-\infty$		-1		$\frac{3}{2}$		$+\infty$
$x+1$		$-$	0	$+$		$+$	
$-2x+3$		$+$		$+$	0	$-$	
Signe de $f'(x)$		$-$	0	$+$	0	$-$	
Variations de f	$+\infty$	$\searrow$	-14	$\nearrow$	$\frac{69}{4}$	$\searrow$	$-\infty$

La légende « Variations de f » est composée avec une coupure de ligne grâce à la commande `\makecell` de l'extension makecell [8] d'Olga Lapko.

```
\[\begin{tablvar}{3}
    \hline
    x & -\infty && -1 && \frac{3}{2} && +\infty \\
    \hline
    x+1 & & - & \barre[0] & + & \barre & + & \\
    \hline
    -2x+3 & & + & \barre & + & \barre[0] & - & \\
    \hline
    \mbox{Signe de }f'(x) & & - & \barre[0] & + & \barre[0] & - & \\
    \hline
    \variations{ \mil{\makecell{\mbox{Variations} \\ \mbox{de }f}} &
        \haut{+\infty} && \bas{-14} && \haut{\frac{69}{4}} &&
        \bas{-\infty} }
    \hline
\end{tablvar}\]
```

2.3 Doubles barres, discontinuités et limites

`\bb` Une double barre s'obtient avec la commande `\bb`.

`\discont` La commande `\discont` empêche le dessin d'une flèche entre deux nœuds, en particulier pour éviter de traverser une double barre. On peut en placer jusqu'à 4.

Une première solution est d'interrompre la double barre pour placer les limites qui resteront alors centrées au milieu de la colonne (mais il faut aimer).

x	$-\infty$		2		$+\infty$
$\frac{1}{x-2}$	0	$\searrow$	$+\infty$ $\Vert$ $-\infty$	$\searrow$	0

```
\[\begin{tablvar}{2}
    \hline
    x & -\infty && 2 &&  +\infty \\
    \hline
```

```
    \variations{\mil{\dfrac{1}{x-2}} & \pos{2}{0} && \bas{-\infty}
        \pos*{2}{\bb} \discont \haut{+\infty} && \pos{2}{0}  }
    \hline
\end{tablvar}\]
```

`\limg` `\limd` Les macros `\limg[`⟨*pos*⟩`]{`⟨*ligne*⟩`}{`⟨*valeur*⟩`}` et `\limd[`⟨*pos*⟩`]{`⟨*lgn*⟩`}{`⟨*val*⟩`}` offrent une solution plus élégante pour placer des limites à gauche ou à droite. L'option de positionnement ⟨*pos*⟩ correspond à celle de la macro `\pos` (`c`, `t` ou `b`).

`\limg*` `\limd*` Les variantes étoilées `\limg*` et `\limd*` sont des alias : `\limg*{+}` signifie `\limg{1}{+\infty}`, `\limg*{-}` correspond à `\limg{`n`}{-\infty}` (pour une partie variations sur n lignes), de même pour `\limd*`. Tout autre argument que `+` ou `-` génère un message d'erreur.

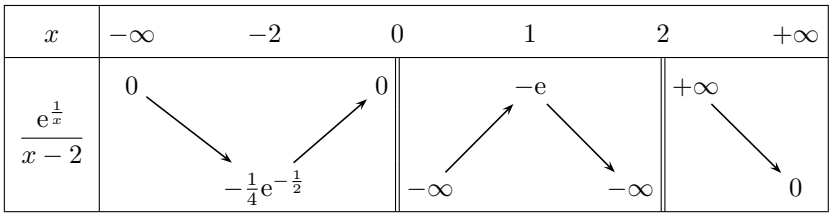
`firstcolsep` La colonne de légende peut être élargie en ajoutant de l'espace à ses extrémités avec l'option `firstcolsep` (dont la valeur par défaut est `5pt`).

x	$-\infty$	0	$+\infty$
$-\frac{1}{x}$	$+$	‖	$-$
$\mathrm{e}^{\frac{1}{x}}$	$1 \searrow 0$	‖	$+\infty \searrow 1$

```
\[\begin{tablvar}[firstcolsep=10pt,intervalwidth=4em]{2}
    \hline
    x & -\infty && 0 && +\infty \\
    \hline
    -\frac{1}{x} & & + & \bb & - & \\
    \hline
    \variations{ \mil{\e^{\frac{1}{x}}} & \haut{1} && \limg{3}{0} \bb
        \discont \limd*{+} && \bas{1} }
    \hline
\end{tablvar}\]
```

La macro `\e` compose le nombre d'Euler e en romain, comme c'est la règle pour les constantes mathématiques (cf. mismath [14] ou frenchmath [15]).

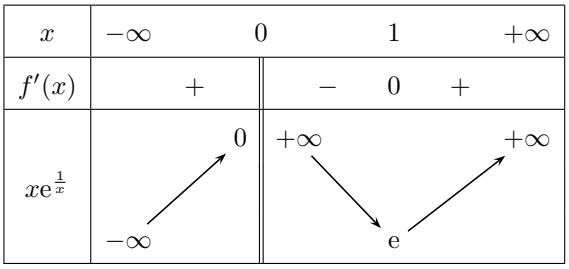
`\bblim` `\bblim*` Sauf sur les bords du tableau ou lorsqu'il y a une zone interdite, les doubles barres sont en général accompagnées à la fois d'une limite à gauche et d'une limite à droite. La commande `\bblim` permet alors d'alléger la syntaxe. Elle prend 4 arguments qui sont dans l'ordre ⟨*ligne*⟩ et ⟨*limite*⟩ à gauche puis à droite de la double barre. Cette macro trace la double barre, place les limites et crée une discontinuité. C'est un alias de `\limg{`⟨*ligne*⟩`}{`⟨*valeur*⟩`}\bb\discont\limd{`⟨*ligne*⟩`}{`⟨*valeur*⟩`}`. La seule différence est que `\bblim` ne possède pas d'option de positionnement des nœuds comme `\limg` et `\limd`. La forme étoilée offre une syntaxe encore plus légère lorsque les deux limites sont infinies : `\bblim*{-}{+}` correspond à `\bblim{3}{-\infty}{1}{+\infty}` (lorsqu'il y a 3 lignes de variations).



```
\[\begin{tablvar}{5}
    \hline
    x & -\infty && -1 && 0 && 1 && +\infty \\
    \hline
    \variations{ \mil{\dfrac{\e^{\frac{1}{x}}}{x-2}} & \haut{0} &&
        \bas{-\frac{1}{4}\e^{-\frac{1}{2}}} && \bblim{1}{0}{3}{-\infty} &&
        \haut{-\e} && \bblim*{-}{+} && \bas{0} }
    \hline
\end{tablvar}\]
```

`bordercolsep` `limsep`

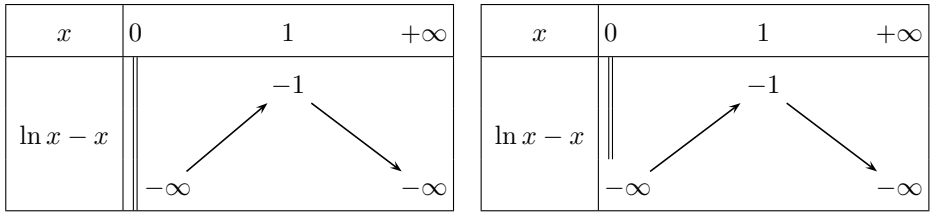
Le paramètre `bordercolsep` gère l'espace à gauche de la première colonne de valeurs et à droite de la dernière (`2pt` par défaut), `limsep` permet d'ajouter de l'espace entre une double barre et une limite (`1pt` par défaut). Il est rarement utile de les modifier, sauf si on préfère plus de blanc. On peut y compris leur donner des valeurs négatives si l'on veut s'approcher davantage des filets verticaux, par exemple `limsep=-1.5pt` amènerait au contact de la double barre).



```
\[\begin{tablvar}[bordercolsep=4pt,limsep=3pt]{3}
    \hline
    x & -\infty && 0 && 1 && +\infty \\
    \hline
    f'(x) & & + & \bb & - & 0 & + & \\
    \hline
    \variations{ \mil{x\e^{\frac{1}{x}}} & \bas{-\infty} &&
        \bblim{1}{0}{1}{+\infty} && \bas{\e} & & \haut{+\infty} }
    \hline
\end{tablvar}\]
```

`tablvar*` (*env.*)

L'environnement `tablvar*` sert à gérer correctement le positionnement des doubles barres lorsqu'elles se trouvent aux extrémités. La différence avec `tablvar` est que les colonnes de valeurs des extrémités ne sont plus centrées mais alignées à gauche pour la première et à droite pour la dernière.



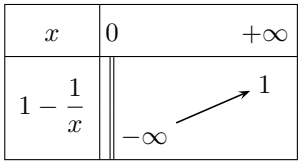
On observera l'utilisation de `\pos*` dans le second tableau, pour tracer une double barre, sauf sur la ligne 3 où on place $-\infty$.

```
\[\begin{tablvar*}{2}
    \hline
    x & 0 && 1 && +\infty \\
    \hline
    \variations{ \mil{\ln x -x} & \bb\limd*{-} && \haut{-1} &&
        \bas{-\infty} }
    \hline
\end{tablvar*}
\quad
\begin{tablvar*}{2}
    \hline
    x & 0 && 1 && +\infty \\
    \hline
    \variations{ \mil{\ln x -x} & \pos*{1}{\bb} \pos*{2}{\bb}
        \bas{-\infty} && \haut{-1} && \bas{-\infty} }
    \hline
\end{tablvar*}\]
```

`extleft` `extright`

Dans `tablvar*`, si, à une extrémité sans double barre, les valeurs ont des largeurs vraiment différentes leur alignement vers le bord peut être inélégant. L'environnement `tablvar` possède des options qui permettent de choisir un alignement à gauche ou à droite de manière séparée pour chaque extrémité. Il s'agit des options `extleft` pour aligner la première colonne à gauche et `extright` pour aligner la dernière colonne à droite (alors que `tablvar*` impose les deux). Ci-dessous, on n'a utilisé que l'option `extleft`[4] pour éviter que le 1 ne soit collé au bord droit et qu'il garde un bon centrage par rapport à $+\infty$.

```
\[\begin{tablvar}[extleft]{1}
    \hline
    x & 0 && +\infty \\
    \hline
    \variations[2]{ \mil{1-\dfrac{1}{x}}
        & \bb \limd*{-} && \haut{1} }
    \hline
\end{tablvar}\]
```



2.4 Valeurs remarquables

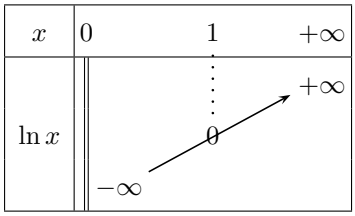
Nous appellerons *valeur remarquable*, un valeur particulière que l'on place dans un tableau de variations et qui sera reliée par des pointillés à son antécédent.

4. En fait `extleft` et `extright` sont des clés booléennes, la valeur `=true` étant facultative.

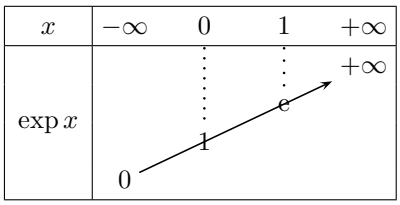
`\vr` Une première approche est placer ces valeurs remarquables par dessus les flèches de variations. Pour relier ces valeurs remarquables à leurs antécédents par des pointillés (tracés réalisés dans la commande `\variations`), on place une commande `\vr` sur la ligne des x et une commande `\vr` dans la partie variations.

Voici un exemple où la valeur remarquable est placée dans la colonne intervalle.

```
\[\begin{tablvar*}[6em]{1}
    \hline
    x & 0 & \vr{1} & +\infty \\
    \hline
    \variations{ \mil{\ln x} &
        \bb \limd*{-} & \vr{0} &
        \pos[t]{1}{+\infty} }
    \hline
\end{tablvar*}\]
```



Un autre exemple avec deux valeurs remarquables, placées cette fois dans des colonnes valeurs. Il y a alors 3 colonnes intervalles et la partie variations est composée sur 4 lignes.




```
\[\begin{tablvar}[stretch=1.2,intervalwidth=1em]{3}
    \hline
    x & -\infty && \vr{0} && \vr{1} && +\infty \\
    \hline
    \variations[4]{ \mil{\exp x} & \bas{0} &&
        \vr[3]{1} && \vr{\e} && \haut{+\infty} }
    \hline
\end{tablvar}\]
```

Comme la partie `\variations` est composée sur 4 lignes, les lignes sont numérotées de 1 (haut) à 4 (bas). La commande `\mil` opère automatiquement un décalage vertical approprié.

`\vr[`⟨*ligne*⟩`]` La commande `\vr` possède elle aussi un argument optionnel qui est la ligne sur laquelle placer la valeur remarquable, lorsqu'il s'agit de la partie variations (ligne 2 par défaut).

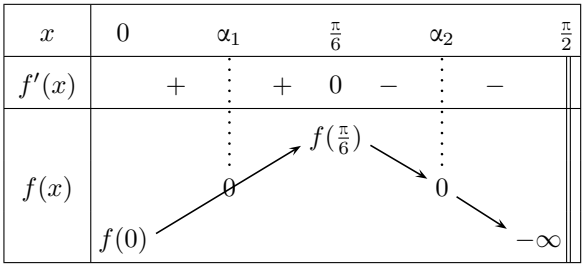
`stretch` À cause des 4 lignes de variations, nous avons utilisé l'option `stretch` qui est un facteur d'élasticité verticale, afin de diminuer la hauteur des lignes du tableau [5]. Sa valeur par défaut est de `1.6`, nous l'avons ramenée dans cet exemple à `1.2`, la valeur standard des environnements `array` étant de `1`.

On peut préférer que les flèches de variations ne passent pas à travers les valeurs remarquables (qui doivent alors être aussi des nœuds pour les flèches). L'exemple qui suit présente un tableau utilisant simultanément les deux manières de traiter

5. On peut aussi utiliser ce paramètre pour agrandir la hauteur des lignes mais l'effet sera global pour tout le tableau. Malheureusement un ajustement automatique des hauteurs, comme proposé dans l'extension cellspace [9] de Josselin Noirel, ne fonctionne pas pour les barres et doubles barres, mais, si l'on souhaite agrandir spécifiquement une ligne, on peut utiliser les commandes `\cstrut` ou `\vstrut` de l'extension spacingtricks (voir exemples page 13).

les valeurs remarquables. La taille des flèches s'ajuste automatiquement. Le second zéro correspond à la fois à un nœud pour les flèches et à une valeur remarquable (nœud pour les pointillés).

`\posvr` La commande `\vr` est inopérante lorsqu'elle est imbriquée avec une commande `\pos` (qui produit les nœuds). Nous fournissons pour cet usage la commande `\posvr[`⟨*opt*⟩`]{`⟨*ligne*⟩`}{`⟨*valeur*⟩`}`, où les 3 paramètres sont exactement les mêmes que ceux de `\pos` (le premier étant le paramètre optionnel `c`, `t` ou `b` pour le positionnement des flèches).



L'option `[t]` pour le max en ligne 1 est ici combinée avec l'option `[b]` en ligne 3 et permet de conserver un positionnement correct du 0 sur la flèche.

Pour placer la dernière valeur, on aurait pu utiliser `\limg[b]{3}{-\infty}`, mais ici `\pos[b]{3}{-\infty}` convient aussi à cause de l'option `extright` qui produit un alignement au fer à droite de la dernière colonne.

```
\[\begin{tablvar}[extright,intervalwidth=2em]{4}
    \hline
    x & 0 && \vr{\alpha_1} && \frac{\pi}{6} && \alpha_2 &&
        \frac{\pi}{2} \\
    \hline
    f'(x) && + && + & 0 & - && - & \bb \\
    \hline
    \variations{\mil{f(x)} & \pos[b]{3}{f(0)} && \vr{0} &&
        \pos[t]{1}{f(\frac{\pi}{6})} && \posvr{2}{0} &&
        \pos[b]{3}{-\infty}\bb }
    \hline
\end{tablvar}\]
```

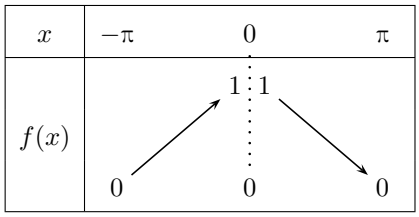
`\vrconnect` C'est la macro graphique `\vrconnect` qui gère le tracé de ces traits pointillés. Elle peut être redéfinie pour changer leur aspect (voir le code section 3.3).

On peut aussi placer les zéros sur les flèches, sans pointillés, auquel cas, on n'a pas besoin de `\vr` ou `\posvr`, il suffit d'utiliser `\pos*{2}{0}` (ou `\mil{0}`) pour le premier zéro de l'exemple précédent et `\pos{2}{0}` pour le second.

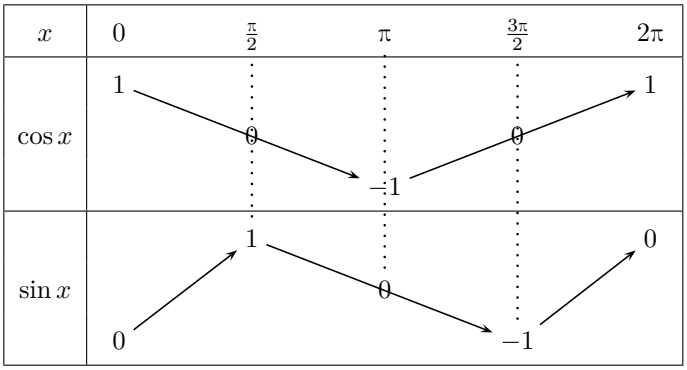
La commande `\vr` peut également servir à représenter des discontinuités particulières, comme dans l'exemple suivant avec la fonction définie par

$$f(x) = \begin{cases} \frac{\sin x}{x} & \text{si } x \neq 0 \\ 0 & \text{si } x = 0 \end{cases}.$$

```
\[\begin{tablvar}{2}
    \hline
    x & -\pi && \vr{0} && \pi \\
    \hline
    \variations{\mil{f(x)} & \bas{0} &&
        \limg{1}{1}\hspace{2pt}\discont
        \vr[3]{0}\hspace{2pt}\limd{1}{1}
        && \bas{0} }
    \hline
\end{tablvar}\]
```



Voici un dernier exemple, avec deux tableaux conjoints.

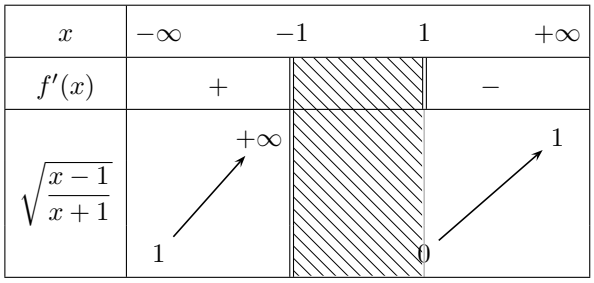



```
\[\begin{tablvar}{4}
    \hline
    x & 0 && \vr{\frac{\pi}{2}} && \vr{\pi} &&
        \vr{\frac{3\pi}{2}} && 2\pi \\
    \hline
    \variations{ \mil{\cos x} & \haut{1} && \mil{0} && \bas{-1} &&
        \mil{0} && \haut{1} }
    \hline
    \variations{ \mil{\sin x} & \bas{0} && \posvr{1}{1} &&
        \vr{0} && \posvr{3}{-1} && \haut{0} }
    \hline
\end{tablvar}\]
```

2.5 Zones interdites

Nous abordons pour finir le tracé de zones interdites, c'est-à-dire d'intervalles où la fonction n'est pas définie. On peut dessiner ces zones interdites en hachures ou en couleur.

`\ZI` Lorsque l'on veut définir une zone interdite, on place la commande `\ZI` dans les cellules correspondantes. La discontinuité des flèches est automatique.

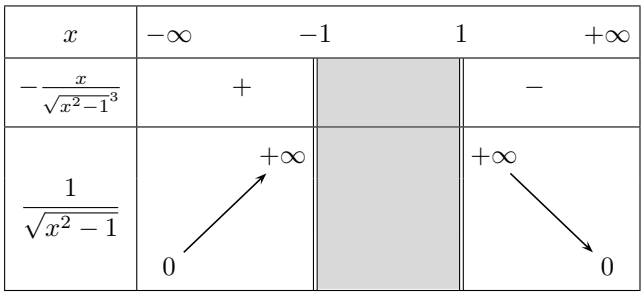



```
\[\begin{tablvar}{3}
    \hline
    x & -\infty & & -1 & & 1 & & +\infty \\
    \hline
    f'(x) && \hspace{-0.5em} + & \bb & \ZI & \bb & - & \\
    \hline
    \variations{ \pos*{2}{\sqrt{\dfrac{x-1}{x+1}}}
        & \bas{1} && \limg{1}{+\infty} \bb & \ZI
        & \bas{0} \barre && \haut{1} }
    \hline
\end{tablvar}\]
```

Le `\hspace{-0.5em}` devant `+` a pour effet de décaler le + affiché vers la gauche afin de le recentrer. En effet, les macros `\limg` et `\limd` placent les valeurs dans des boîtes de largeur nulle (grâce à la commande `\zbox`, voir section 3.7). De ce fait, le contenu d'une colonne valeur (qui a une largeur fixe) peut déborder en largeur et empiéter sur la colonne intervalle contiguë. C'est le cas ici avec le $+\infty$.

Pour la légende de la fonction, nous aurions pu utiliser `\mil` à la place de `\pos*{2}`. La différence est que `\mil` affiche son contenu dans une boîte de hauteur nulle et n'agrandit donc pas la ligne correspondante du tableau. Avec `\pos*{2}`, la seconde ligne des variations est légèrement plus haute.

`ZItype` Le type de rendu est défini par l'option `ZItype` qui peut prendre deux valeurs : `c`, pour une zone interdite colorée, ou `h` (par défaut), pour une zone interdite hachurée.



Dans la légende de la deuxième ligne, le dénominateur a été légèrement agrandi en hauteur, grâce à la commande LaTeX `\strut`, afin d'éviter que l'exposant 3 de la racine carrée n'arrive au contact de la barre de fraction (mais ceci n'a rien à voir en fait avec les tableaux de variations).

```
\[\begin{tablvar}[ZItype=c,intervalwidth=3.5em]{3}
```

```
    \hline
    x & -\infty && -1 && 1 && +\infty \\
    \hline
    -\frac{x}{\strut\sqrt{x^2-1}^3} &&  + & \bb & \ZI & \bb & - & \\
    \hline
    \variations{\pos*{2}{\dfrac{1}{\sqrt{x^2-1}}} & \bas{0} &&
        \limg*{+} \bb & \ZI &
        \bb \limd*{+} && \bas{0} }
    \hline
\end{tablvar}\]
```

`ZIcolor` Pour une zone interdite colorée, la couleur peut être modifiée avec l'option `ZIcolor`. On peut utiliser les noms de couleur prédéfinis ou créer son propre nom de couleur, par exemple `\definecolor{redblue}{rgb}{0.1,0.3,0.7}` [6]

x	$-\infty$	$-\frac{\sqrt{2}}{2}$	$\frac{\sqrt{2}}{2}$	$+\infty$
$\frac{2x}{\sqrt{2x^2-1}}$	$-$	‖	$+$	
$\sqrt{2x^2-1}$	$+\infty$ ↘	0	0	↗ $+\infty$

Ici nous avons agrandi spécifiquement la seconde ligne, contenant la fraction avec radical, grâce à la commande `\cstrut` de l'extension spacingtricks [11]. Cette commande permet de garantir une *profondeur* et une *hauteur* données, symétriques de part et d'autre de la ligne de centrage mathématique.

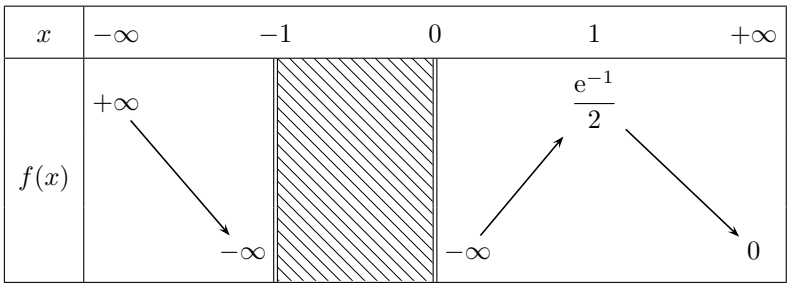
`ZIaddwidth` L'option `ZIaddwidth` permet d'ajuster finement la largeur des zones interdites. On a ajouté `1pt` ce qui donne un rendu un peu meilleur lorsque la zone interdite est encadrée de barres simples et non de double barres.

```
\[\begin{tablvar}[ZItype=c,ZIcolor=green,ZIaddwidth=1pt]{3}
    \hline
    x & -\infty && -\frac{\sqrt{2}}{2} && \frac{\sqrt{2}}{2} && +\infty \\
    \hline
    \frac{2x}{\sqrt{2x^2-1}} \cstrut{2.5ex} && - & \bb & \ZI & \bb
        & + & \\
    \hline
    \variations{ \mil{\sqrt{2x^2-1}} & \haut{+\infty} && \bas{0} \barre
        & \ZI & \bas{0} \barre && \haut{+\infty} }
    \hline
\end{tablvar}\]
```

lorsque des extrema de grande taille sont placés dans la partie *variations*, il est nécessaire de modifier la hauteur des lignes, pour éviter de toucher le filet ho-

6. Le rectangle coloré est affiché en transparence afin de laisser voir les traits sous-jacents. Il arrive que la compilation LaTeX + dvips + ps2pdf n'affiche pas cette transparence, ce qui semble être dû à un bug dans certaines versions de Ghostscript. Dans ce cas une solution peut être de compiler avec XeLaTeX ou LuaLaTeX (si luapstricks est installé sur votre distribution). La compilation pdfLaTeX avec l'option `tikz` ne pose pas ce problème.

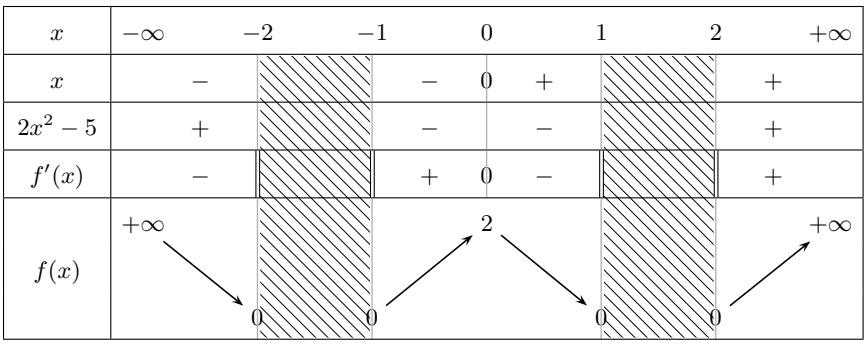
rizontal (au-dessus ou en-dessous) [7]. Cette modification peut se faire avec l'option `stretch` ou avec `\vstrut[`⟨*depth*⟩`]{`⟨*height*⟩`}` qu'il faut placer *dans la colonne de légendes* pour que `\ZI` prenne en compte sa hauteur. Dans l'exemple suivant, `stretch=2.2` aurait donné le même résultat mais une ligne des x plus haute.




```
\[\begin{tablvar}[4em]{4}
    \hline
    x & -\infty && -1 && 0 && 1 && +\infty \\
    \hline
    \variations{ \mil{f(x)} \vstrut[2ex]{4.5ex} & \haut{+\infty} &&
        \limg{3}{-\infty} \bb & \ZI & \bb \limd{3}{-\infty} &&
        \haut{\dfrac{\e^{-1}}{2}} && \bas{0} }
    \hline
\end{tablvar}\]
```

On peut créer jusqu'à 4 zones interdites sur des colonnes différentes, comme pour les discontinuités. Voici un tableau contenant deux zones interdites, avec

$$f(x) = \sqrt{(x^2-1)(x^2-4)} \quad \text{et} \quad f'(x) = \frac{x(2x^2-5)}{f(x)}.$$

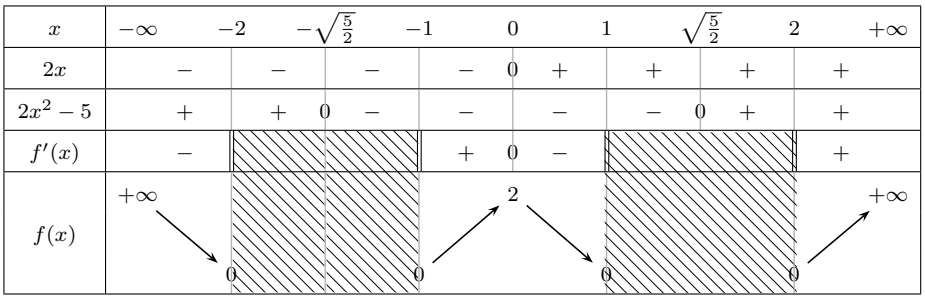



```
{ \small \[\begin{tablvar}[2em]{6}
    \hline
    x & -\infty && -2 && -1 && 0 && 1 && 2 && +\infty \\
    \hline
    x & &-& \barre & \ZI & \barre &-& \barre[0] &+& \barre & \ZI &
        \barre &+& \\
    \hline
```

7. L'extension tabularray [10] de Jianrui Lyu permettrait éventuellement une gestion automatisée des hauteurs de ligne dans une possible version future.

```
    2x^2-5 & &+& \barre & \ZI & \barre &-& \barre &-& \barre & \ZI &
        \barre &+& \\
    \hline
    f'(x) & &-& \bb & \ZI & \bb &+& \barre[0] &-& \bb & \ZI & \bb &+& \\
    \hline
    \variations{ \mil{f(x)} & \haut{+\infty} && \bas{0} \barre
        & \ZI & \bas{0} \barre && \haut{2} && \bas{0}
        \barre & \ZI & \bas{0} \barre && \haut{+\infty} }
    \hline
\end{tablvar}\] }
```

\hachure Lorsque l'on dessine des zones interdites hachurées sur plusieurs colonnes contiguës, les hachures ne seront en général pas jointives. Pour pallier cela, on peut utiliser la primitive graphique `\hachure` qui permet de définir manuellement le rectangle à hachurer. L'exemple suivant montre les deux solutions.



La commande `\hachure` peut être placée où l'on veut dans le tableau, il suffit de donner les coordonnées du rectangle visé à partir de la position courante (réglage manuel). Dans la partie variations, il faut mettre la commande `\hachure` dans un `\pos*`, sinon elle sera répétée pour chaque ligne des variations. Ce problème de discontinuité des lignes ne se pose pas avec des zones interdites colorées.

```
{ \footnotesize
\[ \begin{tablvar}[intervalwidth=1.8em,ZIaddwidth=1pt]{8}
    \hline
    x & -\infty && -2 && -\sqrt{\frac{5}{2}} && -1
        && 0 && 1 && \sqrt{\frac{5}{2}} && 2 && +\infty \\
    \hline
    2x & &-& \barre &-& \barre &-& \barre &-& \barre[0] &+&
        \barre &+& \barre &+& \barre &+& \\
    \hline
    2x^2-5 & &+& \barre &+& \barre[0] &-& \barre &-& \barre &-&
        \barre &-& \barre[0] &+& \barre &+& \\
    \hline
    f'(x) & &-& \bb & \ZI & \barre & \ZI & \bb &+&
        \barre[0] &-& \bb & & & & & \bb & + & \\
    \hline
    \variations{\mil{f(x)} & \haut{+\infty} && \bas{0}\barre & \ZI &
        \barre & \ZI & \bas{0}\barre && \haut{2} && \bas{0}\barre \discont
        && \pos*{3}{\hachure{-4.3em,16.4ex}{4.3em,-1.4ex}} &&
        \bas{0}\barre && \haut{+\infty} }
    \hline
\end{tablvar} \] }
```

\ZI* Dans un tableau de signes sans partie variations, il faut utiliser la commande \ZI* pour déclencher le tracé des hachures (ou de la zone colorée)[8]. Celle-ci doit être placée sur la dernière ligne de chaque bloc de hachures[9].

x	$-\infty$		-1		1		2		$+\infty$
$x-2$		$-$		$-$		$-$	0	$+$	
$\sqrt{x^2-1}$		$+$	0		0	$+$		$+$	
$(x-2)\sqrt{x^2-1}$		$-$	0		0	$-$	0	$+$	

```
\[\begin{tablvar}{4}
    \hline
    x & -\infty && -1 && 1 && 2 && +\infty \\
    \hline
    x-2 && - & \barre & - & \barre & - & \barre[0] & + & \\
    \hline
    \sqrt{x^2-1} && + & \barre[0] & \ZI & \barre[0] & + & \barre & + & \\
    \hline
    (x-2)\sqrt{x^2-1} && - & \barre[0] & \ZI* & \barre[0] & - &
        \barre[0] & + & \\
    \hline
\end{tablvar}\]
```

Pour tracer des rectangles de hachures ou des zones colorées sur des lignes non contiguës, dans un tableau de signes, il faut un appel à \ZI* pour déclencher le tracé de chaque rectangle.

x	$-\infty$		-1		1		2		$+\infty$
$\sqrt{x^2-1}$		$+$	0		0	$+$		$+$	
$x-2$		$-$		$-$		$-$	0	$+$	
$(x-2)\sqrt{x^2-1}$		$-$	0		0	$-$	0	$+$	

```
\[\begin{tablvar}{4}
    \hline
    x & -\infty && -1 && 1 && 2 && +\infty \\
    \hline
    \sqrt{x^2-1} && + & \barre[0] & \ZI* & \barre[0] & + & \barre & + & \\
    \hline
    x-2 && - & \barre & - & \barre & - & \barre[0] & + & \\
    \hline
    (x-2)\sqrt{x^2-1} && - & \barre[0] & \ZI* & \barre[0] & - &
```

8. La commande \ZI ne déclenche le tracé que sur la dernière ligne des variations et pas dans les lignes de signes ; les occurrences de \ZI qui précèdent ne font que cumuler la hauteur.

9. Un appel à \ZI* dans chaque cellule ne permettrait pas de garantir la continuité des hachures.

```
        \barre[0] & + & \\
    \hline
\end{tablvar}\]
```

2.6 Options et réglages

Comme indiqué dans l'introduction, l'extension tablvar possède deux options, `pstricks` ou `tikz`, pour choisir la méthode graphique utilisée. Ce sont les seules options reconnues à l'appel `\usepackage`.

`\tablvarset{`⟨*keyval*⟩`}`

Mais les environnements `tablvar` et `tablvar*` possèdent également un jeu de paramètres qui peuvent être définis de manière globale, ou locale dans un environnement (ou un groupe `{...}`), soit en utilisant des variables spécifiques, soit (depuis la version 2.0) avec la commande `\tablvarset{`⟨*keyval*⟩`}`, basée sur le mécanisme *clé=valeur*.

`tablvar[`⟨*keyval*⟩`]` (*env.*)
`tablvar*[`⟨*keyval*⟩`]` (*env.*)

Ces paramètres peuvent aussi être réglés localement, indépendamment des valeurs globales, en tant qu'options d'un environnement `tablvar[`⟨*keyval*⟩`]{`⟨*num*⟩`}` (ou `tablvar*`), avec la même syntaxe *clé=valeur*.

Le tableau suivant résume les options disponibles, précise leur valeur par défaut et les variables de réglage spécifiques (longueurs, booléens, etc.) auxquelles elles sont associées et qui peuvent aussi être modifiées directement. Les liens hypertextes (en bleu) renvoient vers un exemple dans la documentation.

Option	Valeur par défaut	Variable associée
intervalwidth	3em	\intervalwidth
colvalwidth	2em	\colvalwidth
firstcolsep	5pt	\firstcolsep
bordercolsep	2pt	\bordercolsep
limsep	1pt	\limsep
stretch	1.6	\tablvarstretch
extleft	false	extleft
extright	false	extright
ZItype	h	\ZItype
ZIcolor	gray	ZIcolor
ZIaddwidth	0pt	\ZIaddwidth

Le premier groupe de paramètres ci-dessus gère les largeurs de colonnes et les espacements horizontaux :

`intervalwidth` règle la largeur des colonnes « intervalles ».

`colvalwidth` définit la largeur des colonnes de valeurs ; il est rarement utile de modifier ce paramètre car lorsqu'une valeur est trop large, elle peut déborder dans les colonnes intervalles adjacentes ; la valeur par défaut est légèrement plus grande qu'un infini avec un signe.

firstcolsep définit l'espace de séparation par rapport aux filets verticaux autour de la première colonne (légendes).

bordercolsep définit l'espace de séparation à gauche de la première et à droite de la dernière colonne de valeurs, par rapport aux filets verticaux.

limsep règle l'espace de séparation à gauche ou à droite d'une limite.

La hauteur des lignes se règle avec :

stretch qui est un facteur d'élasticité verticale agissant sur toutes les lignes du tableau [10] ; une valeur de 1 correspond à la valeur standard d'un environnement `array`.

Les deux paramètres suivants règlent l'alignement des valeurs dans les colonnes aux extrémités du tableau :

extleft aligne les valeurs à gauche dans la première colonne ; l'effet est adéquat lorsqu'il y a une double barre à l'extrémité gauche.

extright aligne les valeurs à droite dans la dernière colonne, adéquat pour une double barre à l'extrémité droite ; l'environnement `tablvar*` fixe `extleft` et `extright` à `true`.

Enfin le dernier groupe de paramètres gère l'aspect des zones interdites :

ZYtype peut prendre deux valeurs : `h` pour une zone interdite hachurée ou `c` pour une zone interdite colorée.

ZIcolor définit la couleur des zones interdites colorées qui doit être un nom de couleur déjà définie ; ce paramètre est sans effet si `ZItype=h`.

ZIaddwidth permet de jouer sur un réglage fin de la largeur des zones interdites ; la manière dont les zones interdites arrivent au contact des barres ou double barres peut s'ajuster en faisant varier cette valeur (de $\pm$`1pt`).

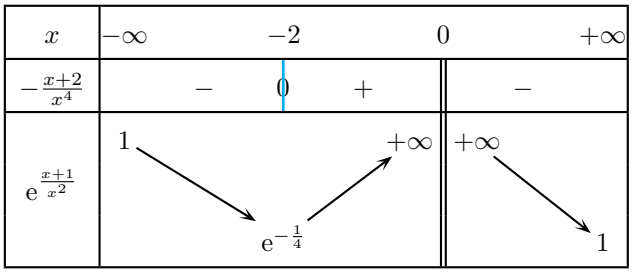
`tablvar[`⟨*largeur*⟩`]` (*env.*) L'argument optionnel des environnements `tablvar` et `tablvar*` peut également être une simple dimension (à condition qu'il n'y ait aucune autre option), qui sera alors interprétée comme la largeur donnée à `intervalwidth`.

Le rendu des commandes graphiques peut également être modifié, en redéfinissant ces commandes, en particulier `\fleche`, `\vrconnect` et `\hachure` grâce aux nombreuses options offertes par PSTricks ou Ti*k*Z. C'est le cas aussi de l'aspect des filets et barres du tableau qui sont définis par les commandes `\tvrulewidth` (`0.4pt` par défaut), `\tvbarrewidth` (`0.5pt` par défaut), `tvbarrecolor` (`{gray}{0.7}` par défaut) et `\bbrulewidth` (`0.4pt` par défaut). Cette redéfinition sera locale si on la place dans l'environnement math du tableau.

Ci-dessous des flèches plus grasses, plus proches des nœuds et dont la pointe est plus effilée, codé ici pour PSTricks (voir le code section 3.3 pour les commandes Ti*k*Z). L'épaisseur des filets du tableau et des barres a été doublée, la couleur de

10. Pour augmenter la hauteur d'une ligne particulière, on pourra utiliser la commande `\vstrut[`⟨*depth*⟩`]{`⟨*height*⟩`}` (du package spacingtricks [11]), où ⟨*depth*⟩ désigne la *profondeur* et ⟨*height*⟩ la *hauteur* minimales à atteindre par rapport à la ligne de *base*, ou encore `\cstrut{`⟨*height*⟩`}` qui garantit une hauteur minimale en-dessous et au-dessus de la ligne de centrage mathématique.

la barre traversant 0 a été modifiée et les infinis ont été « collés » aux bords avec `bordercolsep=-1pt`.




```
\[ \renewcommand{\fleche}{\ncline[linewidth=0.8pt,arrowsize=2pt 3,
       arrowinset=0.5,nodesep=1.5pt]{->}}
\setlength{\tvrulewidth}{0.8pt}
\setlength{\bbrulewidth}{0.8pt}
\setlength{\tvbarrewidth}{1pt}
\definecolor{tvbarrecolor}{named}{cyan}
\begin{tablvar}[intervalwidth=4em,bordercolsep=-1pt]{3}
    \hline
    x & -\infty && -2 && 0 && +\infty \\
    \hline
    -\frac{x+2}{x^4} && - & \barre[0] & +  & \bb & - & \\
    \hline
    \variations{\mil{\e^{\frac{x+1}{\vstrut{1.2ex} x^2}}} & \haut{1} &&
        \bas{\e^{-\frac{1}{4}}} && \bblim*{+}{+} && \bas{1} }
    \hline
\end{tablvar} \]
```

Enfin deux autres paramètres de type longueur `\rowtopsep` et `\rowbottomsep` garantissent un espace minimum (de `2pt` par défaut) entre les filets horizontaux et le contenu des cellules, pour la colonne des légendes.

2.7 Nouveautés depuis la version 2.0

La version 2.0 présente de nombreuses nouveautés :

— *Il n'y a plus d'option par défaut* à l'appel de l'extension. C'était `pstricks` dans les versions précédentes. Dorénavant, si aucune option n'est précisée, l'extension choisira elle-même en fonction du compilateur utilisé, donc `tikz` pour pdfLaTeX qui ne supporte pas le code PSTricks.

— *Les colonnes de valeurs ont des largeurs fixes* (`2em` *par défaut)* et non plus variables en fonction du contenu, comme c'était le cas jusqu'à la version 1.2. L'avantage est d'améliorer le centrage pour les signes, lorsque les colonnes de valeurs ont des largeurs très différentes, et surtout d'éviter le recours à des `\zbox` pour les colonnes de valeurs bordant les zones interdites.

— Le paramètre `\innercolsep` qui gère l'espacement entre les colonnes valeurs et les colonnes intervalles, valait `4pt` et a été fixé à `0pt` car, en l'absence de filets verticaux, il ne sert plus à rien avec les colonnes à largeur

fixe. Mais, en contrepartie, *la valeur par défaut de* `\intervalwidth` *a été augmentée de* `2.5em` *à* `3em`.

— Les réglages peuvent désormais être effectués, avec la syntaxe *clé = valeur* et la commande `\tablvarset`, ou comme options des environnements `tablvar` et `tablvar*`.

— Un nouveau paramètre `\firstcolsep` (ou l'option `firstcolsep`) permet d'élargir la colonne des légendes.

— Deux options permettent d'aligner de manière différenciée la première colonne de valeurs à gauche (`extleft`) *ou* la dernière à droite (`extright`) contrairement à l'environnement `tablvar*` qui fait obligatoirement les deux.

— Le paramètre `\tablvarstetch` (géré par l'option `stretch`) a été augmenté de `1.4` à `1.6`.

— Une nouvelle macro `\posvr` permet de définir conjointement un nœud pour les flèches *et* pour les valeurs remarquables.

— Deux nouvelles macros `\limg` et `\limd` gèrent désormais le bon positionnement des limites à gauche et à droite, en particulier au bord des doubles barres (en plus de `\bblim`) et surtout au bord des zones interdites. Le recours explicite à `\zbox` n'est plus nécessaire.

— Un nouveau paramètre `\limsep` permet d'ajuster la proximité des limites avec les doubles barres.

— Les zones interdites sont, à présent, définies par une unique commande `\ZI`, avec une option `ZItype`. Les anciennes commandes `\ZIh`, pour les zones interdites hachurées, ou `\ZIc`, pour les zones interdites colorées, ne sont plus utiles, mais elles ont été conservées et permettraient, par exemple, de dessiner dans un même tableau une zone interdite hachurée et une autre colorée, même si cela ne viendrait à l'esprit de personne.

— Il n'y a, en principe, plus besoin d'ajustement de hauteur des zones interdites avec un paramètre optionnel `\ZI[`⟨*height*⟩`]`, lorsque des contenus de grande hauteur sont placés dans la colonne des légendes. Néanmoins ce paramètre a (pour l'instant) été conservé, pour faire éventuellement de micro-ajustements.

— On peut désormais régler l'épaisseur des filets verticaux et double barres avec `\tvrulewidth` et `\bbrulewidth`.

— Le type de colonne `i`, utilisé pour les colonnes intervalles, est maintenant interne à l'environnement `tablvar` et il n'y a donc plus de risque d'incompatibilité avec une autre extension qui utiliserait ce même identifiant comme type de colonne. De nouveaux types de colonnes ont été définis pour les valeurs et légendes, mais sont également internes à l'environnement `tablvar`.

— Les macros `\bas` et `\mil` peuvent à présent être utilisées même lorsque le nombre de lignes de variations est différent de 3. Dans ce cas, commande `\mil` calcule automatiquement le décalage vertical ; l'utilisateur n'a plus besoin d'un appel explicite à `\vdecal`.

Dans la version 2.1 :

— Ont été ajoutés deux nouveaux paramètres `\rowtopsep` et `\rowbottomsep` pour garantir un espace minimum (de `2pt` par défaut) entre les filets horizontaux et le contenu pour la colonne des légendes.

— Il est désormais possible d'utiliser l'option `pstricks` avec LuaLaTeX, si luapstricks est installé sur votre distribution.

— Le calcul du décalage vertical de la commande `\mil` a été amélioré, dans le cas d'un nombre pair de lignes de variations.

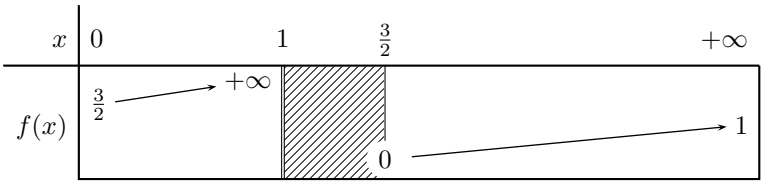
— Les nouvelles commandes étoilées `\limg*`, `\limd*` et `\bblim*` permettent une écriture simplifiée de `\limg`, `\limd` et `\bblim`, pour des limites ∞.

Des problèmes de composition peuvent apparaître avec des tableaux réalisés antérieurement à la version 2.0, en particulier pour les limites au bord des doubles barres, lorsque l'on n'utilise ni `\bblim`, ni `tablvar*`. S'il apparaît une limite positionnée par dessus une double barre, il suffit de remplacer les `\zbox` utilisées antérieurement par `\limg` ou `\limd`. D'autre part, les ajouts de hauteur qui peuvent avoir été utilisés comme argument optionnel de `\ZIh` ou `\ZIh*` sont à éliminer.

2.8 Comparatif avec les autres extensions

Le dépôt d'archives CTAN (Comprehensive TeX Archive Network) contient quelques autres extensions dédiées à la réalisation de tableaux de variations, dans la rubrique « Maths tabvar ». Nous les décrivons ici brièvement.

tableaux de Nicolas Kisselhoff [1]. Propose un pseudo-code PSTricks, certainement plus commode que d'écrire toutes les commandes PSTricks. Attention l'extension doit être appelée avec `\usepackage{tableau}`, sans `x`. Voici un exemple repris de la doc de tableaux avec le code en suivant.




```
\newpsstyle{hachured}{fillstyle=hlines,hatchwidth=0.2pt,hatchsep=2pt}
\setlength{\TabTitreL}{1cm}

\begin{MonTableau}{1}{9}{1.5}
\psframe[style=hachured,linestyle=none](.3,1)(.45,0)

\TabNewCol{0}
\rTabPut{Br}{-.15}{.25}{$x$}
\rTabPut{Br}{-.15}{.40}{$f(x)$}

\TabNewCol{0}
\rTabPut{Bl}{.15}{.25}{$0$}
\rTabPut{Bl}{.15}{.60}{$\frac{3}{2}$}
```

```
\TabNewCol{.30}
\rTabPut{B}{0}{.25}{$1$}
\rTabPut{Br}{-.15}{.80}{$+\infty$}
\psline[style=TabDblBarre](.3,0)(.3,1)

\TabNewCol{.45}
\rTabPut{B}{0}{.25}{$\frac{3}{2}$}
\rTabPut[2]{B}{0}{.10}{\TabZ}

\TabNewCol{1}
\rTabPut{Br}{-.15}{.25}{$+\infty$}
\rTabPut{Br}{-.15}{.40}{$1$}

\TabFleche{B1}{C1}
\TabFleche{D1}{E1}
\end{MonTableau}
```

tabvar de Daniel Flipo [2], l'auteur, entre-autres de babel-french, extension bien connue de tous les utilisateurs francophones de LaTeX. Permet de réaliser des tableaux plutôt symboliques. Voici un tableau simple et son code tabvar.

```
\[\begin{tabvar}{|C|CCCCR|}
    \hline
    x & 0 & & \frac{1}{2} & & 1 \\
    \hline
    f'(x) && + & \barre{0} & - & \\
    \hline
    \niveau{1}{2}\TVcenter{f(x)} & 0 & \croit
        & \frac{1}{4} & \decroit & 0 \\
    \hline
\end{tabvar}\]
```

x	0		$\frac{1}{2}$		1
$f'(x)$		$+$	0	$-$	
$f(x)$	0	↗	$\frac{1}{4}$	↘	0

variations de Christian Obrecht [3]. Extension vraiment simple d'usage, qui a l'avantage d'avoir une documentation très courte. Voici un exemple repris de la doc de variations (\mI et \pI désignent $-\infty$ et $+\infty$).

x	$-\infty$		0		$+\infty$
$-\frac{1}{x^2}$		$-$		$-$	
$\frac{1}{x}$	0	↘	$-\infty$ $\|$ $+\infty$	↘	0

```
\[\begin{variations}
    x & \mI &&& 0 &&& \pI \\
    \filet
    -\frac{1}{x^2} && - && \bb && - & \\
    \m{\dfrac{1}{x}} & \h{0} & \d & \mI & \bb & \h{\pI} & \d & 0 \\
\end{variations}\]
```

tablor de Guillaume Connan [4]. Permet d'automatiser de manière poussée la composition *et la conception* du tableau grâce à la librairie de calcul formel Giac/Xcas (qui doit être installée). Il suffit de renseigner l'expression de la fonction, l'intervalle souhaité, les éventuelles valeurs interdites. Voici par exemple le code pour le tableau de la fonction $g : t \mapsto \frac{t^2}{t^2-1}$ sur $[-10, +\infty[$, avec variations et signe de la dérivée, très synthétique !

```
\begin{TV}
TV([-10,+infinity],[-1,1],"g","t",x^2/(x^2-1),1,n,\tv)
\end{TV}
```

On peut aussi définir un tableau avec simplement deux listes, les antécédents et les images, et tablor prend en charge les zones interdites (mais pas la continuité des hachures de cellules contiguës). On pourra consulter la documentation pour de jolis exemples.

Pour fonctionner, tablor utilise tableauVariations.mp de Frédéric Mazoit [5] qui compose le tableau en METAPOST. Xcas oblige à compiler en mode `shell-escape` pour autoriser les sorties de la compilation TEX.

tkz-tab d'Alain Matthes [6]. Une très belle extension qui fait a peu près tout ce que fait tablvar et même un peu plus : par exemple tangentes horizontales sur les extrema, cellules en couleur. Un de ses principaux bénéfices par rapport à tablvar est de ne pas nécessiter de double compilation [11]. Signalons quand même quelques avantages de tablvar :

— avec tkz-tab, un tableau n'est pas globalement en mode mathématique, il faut donc mettre les $ du mode math autour de chaque valeur,

— et les hauteurs de ligne doivent être précisées de manière explicite et uniquement cm (unité fixe qui ne s'ajuste pas à la fonte),

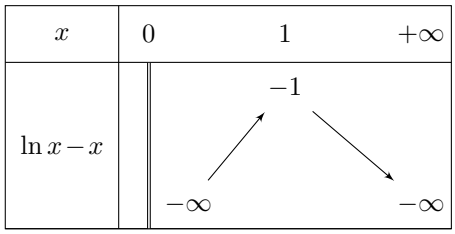
— tablvar offre la possibilité de mettre des valeurs par dessus les flèches,

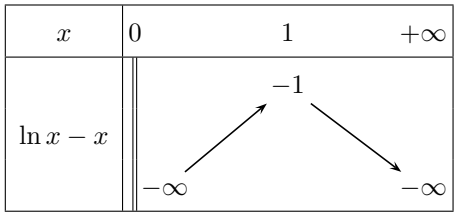
— dans tkz-tab, les double barres aux extrémités du tableau sont assez éloignées des filets, ce qui laisse un blanc important sur les bords.

Voici un exemple illustrant ce dernier point, avec son code. Pour comparer, le tableau obtenu avec tablvar a été placé dessous avec son code en suivant.

```
\begin{center}\begin{tikzpicture}
    \tkzTabInit[lgt=1.5,espcl=1.8,deltacl=0.4]%
        {$x$/0.75, $\ln x - x$/2.2}{0, 1, $+\infty$}
    \tkzTabVar{D-/$-\infty$ , +/$-1$  , -/$-\infty$}
\end{tikzpicture}\end{center}
```

11. La double compilation avec tablvar n'est cependant nécessaire qu'avec l'option `tikz`.






```
\[\begin{tablvar*}{2}
    \hline
    x & 0 & & 1 & & +\infty \\
    \hline
    \variations{ \mil{\ln x -x} & \bb\bas{-\infty} && \haut{-1} &&
        \bas{-\infty} }
    \hline
\end{tablvar*}\]
```

On remarquera que les réglages tablvar sont automatiques et ne nécessitent aucune option (si ce n'est l'appel à la version étoilée `tablvar*`).

3 Le code

3.1 Extensions requises et options du package

```
\RequirePackage{array}
\RequirePackage{ifthen}
\RequirePackage{multido}
\RequirePackage{xkeyval}
\RequirePackage{iftex}

\newif\iftv@tikz % false par défaut
\DeclareOption{tikz}{\tv@tikztrue}
\newif\iftv@pstricks
\DeclareOption{pstricks}{\tv@pstrickstrue}
\ProcessOptions \relax

\iftv@pstricks
    \ifpdf
        \ifluatex
            \PackageInfo{tablvar}{Option 'pstricks' works
                with luapstricks}
        \else
            \PackageError{tablvar}{Don't use the 'pstricks' option with
```

```
                PDF output mode}{Use 'tikz' option or change the engine}
        \fi
    \else
        \PackageInfo{tablvar}{Option 'pstricks' is used}
    \fi
\else\iftv@tikz
    \AtEndDocument{\PackageWarningNoLine{tablvar}{Option 'tikz' is used.
        \MessageBreak
        Maybe rerun to draw the arrows correctly}
    }
\else\ifpdf
    \AtEndDocument{\PackageWarningNoLine{tablvar}{Output is in PDF mode,
        I'm using the 'tikz' option.
        \MessageBreak
        Maybe rerun to draw the arrows correctly}
    }
    \tv@tikztrue
\else
    \PackageWarningNoLine{tablvar}{Output is in DVI mode,
        I'm using the 'pstricks' option}
    \tv@pstrickstrue
\fi\fi\fi

\iftv@tikz
    \RequirePackage{tikz}
    \usetikzlibrary{patterns}
    \usetikzlibrary{patterns.meta}% pour le paramétrage des hachures
\else % pstricks=true
    \RequirePackage{pst-node}
\fi
```

3.2 Les paramètres généraux

Dans cette section sont présentés tous les paramètres que l'on peut redéfinir pour modifier l'aspect des tableaux.

`\intervalwidth` Définit la largeur par défaut des colonnes « intervalles », valeur prédéfinie à `3em`.

```
\newlength{\intervalwidth}
\setlength{\intervalwidth}{3em} % largeur des "intervalles"
```

`\colvalwidth` Définit la largeur des colonnes de valeurs, valeur prédéfinie à `2em`.

```
\newlength{\colvalwidth}
\setlength{\colvalwidth}{2em} % largeur des colonnes de valeurs
```

`\firstcolsep` Définit l'espace de séparation autour de la première colonne (légendes), valeur prédéfinie à `5pt`, qui est la valeur par défaut dans les tableaux LaTeX.

```
\newlength{\firstcolsep}
\setlength{\firstcolsep}{5pt} % valeur LaTeX par défaut
```

`\bordercolsep` Définit l'espace de séparation aux extrémités du tableau, pour les première et dernière colonnes de valeurs, prédéfinie à `2pt`. Sa valeur par défaut a été réduite par rapport à la longueur LaTeX standard `\arraycolsep`, afin que les valeurs des extrémités (souvent des ∞) soient plus proches des filets verticaux bordant le tableau.

```
\newlength{\bordercolsep}
\setlength{\bordercolsep}{2pt}
```

`\limsep` Règle l'espace avant une limite à droite et après une limite à gauche, utilisé dans les macros `\limg`, `\limd` et `\bblim`. La macro `\bb` composant la double barre comprend déjà `1.5pt` d'espace autour des traits verticaux. La macro `\limsep` ajoute `1pt` par défaut. Si on règle ce paramètre à `-1.5pt`, on élimine l'espace créé par `\bb` et on arrive au contact du filet de la double barre.

```
\newlength{\limsep}
\setlength{\limsep}{1pt}
```

`\tablvarstretch` Permet de régler la valeur relative de l'espacement vertical des lignes du tableau. La valeur par défaut est `1.6` (1 correspondant à la valeur standard d'un environnement `array`).

```
\newcommand{\tablvarstretch}{1.6}
```

`\ZItype` Définit le type de zone interdite, `h` pour « hachurée » ou `c` pour « colorée » (`h` par défaut).

```
\newcommand{\ZItype}{h}
```

`ZIcolor` Définit la couleur des zones interdites, « colorées » (gris par défaut).

```
\definecolor{ZIcolor}{named}{gray}
```

`\ZIaddwidth` Définit une largeur supplémentaire (positive ou négative) à ajouter à la largeur des zones interdites, par exemple `1pt` pour arriver au contact du trait de `\barre`.

```
\newlength{\ZIaddwidth}
\setlength{\ZIaddwidth}{0pt}
```

`\tablvarset` Les paramètres précédents peuvent être modifiés avec `\setlength`, `\setboolean`, `\renewcommand` ou `\definecolor`, mais ils peuvent également être gérées avec le mécanisme *clé=valeur*, grâce à l'extension xkeyval [12] de Hendri Adriaens. Nous avons fait le choix de ne pas en faire des options de l'extension tablvar elle-même, mais de fournir une commande spéciale, `\tablvarset`, pour régler ces options. La couleur définie par l'option `ZIcolor` doit être une couleur « nommée ».

```
\define@key{tablvar}{intervalwidth}{\setlength{\intervalwidth}{#1}}
\define@key{tablvar}{colvalwidth}{\setlength{\colvalwidth}{#1}}
\define@key{tablvar}{firstcolsep}{\setlength{\firstcolsep}{#1}}
\define@key{tablvar}{bordercolsep}{\setlength{\bordercolsep}{#1}}
\define@key{tablvar}{limsep}{\setlength{\limsep}{#1}}
\define@key{tablvar}{stretch}{\renewcommand{\tablvarstretch}{#1}}
\define@boolkey{tablvar}[]{extleft}[true]{} % false si non appelé
\define@boolkey{tablvar}[]{extright}[true]{}
\define@choicekey{tablvar}{ZItype}{h,c}{\renewcommand{\ZItype}{#1}}
\define@key{tablvar}{ZIcolor}{\definecolor{ZIcolor}{named}{#1}}
\define@key{tablvar}{ZIaddwidth}{\setlength{\ZIaddwidth}{#1}}

\newcommand\tablvarset[1]{\setkeys{tablvar}{#1}}
```

D'autres paramètres, présentés ci-dessous, qui n'ont, en principe, pas vocation à être modifiés, ne sont pas gérés par `\tablvarset`.

`\tvrulewidth` Définit l'épaisseur des filets du tableau (`0.4pt` par défaut).

```
\newlength{\tvrulewidth}
\setlength{\tvrulewidth}{0.4pt}
```

`\tvbarrewidth` Définit l'épaisseur des barres de séparation verticales coupant les 0 d'un tableau de signe : `0.5pt` par défaut.

```
\newlength{\tvbarrewidth}
\setlength{\tvbarrewidth}{0.5pt}
```

`tvbarrecolor` Définit la couleur des mêmes barres de séparation verticales. La valeur par défaut `{gray}{0.7}` correspond à un niveau de gris. On peut redéfinir la couleur par une couleur nommée `\definecolor{named}{`⟨*couleur*⟩`}` ou selon un modèle comme rgb, cmyk ou gray : `\definecolor{tvbarrecolor}{rgb}{`⟨*x, x, x*⟩`}`.

```
\definecolor{tvbarrecolor}{gray}{0.7}
```

`\bbrulewidth` Définit l'épaisseur des traits verticaux d'une double barre.

```
\newlength{\bbrulewidth}
\setlength{\bbrulewidth}{0.4pt}
```

`\innercolsep` Définit la largeur entre les colonnes valeurs et les colonnes intervalles du tableau. Cette largeur a été fixée à `0pt`, car l'augmenter est équivalent à augmenter `\intervalwidth` de la même longueur, vu qu'il n'y a pas de filet de séparation entre les colonnes de valeurs et les colonnes intervalles.

```
\newlength{\innercolsep}
\setlength{\innercolsep}{0pt}
```

`maxdiscount` La gestion de discontinuités permettant de ne pas relier certains nœuds consécutifs se fait grâce à la commande `\discont`. Le compteur `maxdiscont` est fixé à 4 par défaut ; on peut l'augmenter (dans le préambule) si l'on veut produire un tableau de variations avec plus de 4 discontinuités.

```
\newcounter{maxdiscont}
\setcounter{maxdiscont}{4} % nb max de discontinuités
```

`\rowtopsep` Cette longueur définit l'espace minimal entre le filet vertical supérieur et le contenu, pour les cellules de la colonne des légendes. Elle ne s'applique pas pour les lignes de variations puisque, dans la colonne des légendes, seule la ligne « milieu » est censée posséder un contenu.

```
\newlength{\rowtopsep}
\setlength{\rowtopsep}{2pt}
```

`\rowbottomsep` Cette longueur définit l'espace minimal entre le filet vertical inférieur et le contenu, pour les cellules de la colonne des légendes. Elle ne s'applique pas pour les lignes de variations.

```
\newlength{\rowbottomsep}
\setlength{\rowbottomsep}{2pt}
```

3.3 Les commandes graphiques PSTricks/TikZ

Nous présentons ici les commandes graphiques permettant le dessin des flèches, les pointillés des valeurs remarquables, les hachures des zones interdites. Celles-ci sont définies différemment s'il s'agit de l'option `tikz` ou `pstricks` (plus précisément liées à l'extension pst-node).

`\fleche` La commande `\fleche{`⟨*nœud1*⟩`}{`⟨*nœud2*⟩`}` possède deux paramètres qui sont les noms des nœuds à relier. La création des nœuds est obtenue avec la commande `\noeud` appelée par `\pos` et le tracé des flèches est réalisé automatiquement par la commande `\variations`. Pour modifier l'aspect des flèches on peut redéfinir la commande `\fleche`.

```
\newcommand*{\fleche}[2]{
    \iftv@tikz
        \tikz[remember picture,overlay]{\draw[->,>=stealth,
            line width=0.6pt] (#1) -- (#2);}
    \else
        \ncline[arrowsize=2pt 2,arrowinset=0.4,nodesep=3pt,
            linewidth=0.6pt]{->}{#1}{#2}
    \fi
}
```

`\vrconnect` La commande `\vrconnect{`⟨*nœud1*⟩`}{`⟨*nœud2*⟩`}` relie les nœuds définis par `\vr` (valeurs remarquables) et le tracé est réalisé automatiquement par la commande `\variations`. Par défaut, les lignes sont en pointillés d'épaisseur `1pt`.

```
\newcommand*{\vrconnect}[2]{
    \iftv@tikz
        \tikz[remember picture,overlay]{\draw[dotted,line width=1pt]
            (#1) -- (#2);}
    \else
        \ncline[nodesep=5pt,linestyle=dotted,linewidth=1pt]{-}{#1}{#2}
    \fi
}
```

`\noeud` `\noeud[`⟨*pos*⟩`]{`⟨*nœud*⟩`}{`⟨*valeur*⟩`}` définit les nœuds des flèches et valeurs remarquables ; le 1er paramètre, optionnel, correspond à l'option `t` (top), `b` (bottom) ou `c` (centered, par défaut) permettant d'ajuster la manière dont la flèche arrive sur le nœud (pas implémenté avec l'option `tikz`) ; le 2e paramètre est le nom du nœud (qui est donné automatiquement par les commandes de positionnement) ; le 3e paramètre est la valeur affichée dans le tableau.

```
\newcommand*{\noeud}[3][c]{
    \iftv@tikz
        % fonctionne mal avec autre chose que 'anchor=base'
        \tikz[remember picture,baseline]{% surtout pas de overlay ici
            \node[anchor=base,inner sep=0pt,outer sep=4pt] at (0,0) (#2)
                {$#3$};}
    \else
        \rnode[#1]{#2}{#3}
    \fi
}
```

`\hachure` Définition des hachures pour les zones interdites. La macro prend deux arguments qui sont des paires de longueurs, par exemple `\hachure{-3em,12ex}{3em,-1ex}`, représentant les extrémités du rectangle à hachurer par rapport à la position courante où la macro est appelée. Pour l'option `tikz`, on aurait pu simplifier la macro en utilisant `\fill[pattern=north east lines]` mais, en utilisant la bibliothèque `patterns.meta`, on peut obtenir des hachures plus élégantes, correspondant à celles de l'option `pstricks`.

```
\newcommand*{\hachure}[2]{
    \iftv@tikz
        \tikz[remember picture,overlay]{%
            \fill[pattern={Lines[distance=3pt,angle=135,line width=0.2pt]}]
            (#1) rectangle (#2);}
    \else
        \psframe[linestyle=none,fillstyle=vlines,hatchwidth=0.2pt,
            hatchsep=3pt](#1)(#2)
    \fi
}
```

`\ZIcouleur` Trace une zone interdite en couleur, mêmes arguments que `\hachure`.

```
\newcommand*{\ZIcouleur}[2]{
    \iftv@tikz
        \tikz[remember picture,overlay]{\fill[color=ZIcolor,opacity=0.3]
            (#1) rectangle (#2);}
    \else
        \psframe[linestyle=none,fillstyle=solid,opacity=0.3,
            fillcolor=ZIcolor](#1)(#2)
    \fi
}
```

3.4 Longueurs et compteurs internes

```
\newcounter{var@ligne} % numéro de ligne des variations
\newcounter{var@noeud} % numéro du nœud des variations
\newcounter{numvr} % numéro de la valeur remarquable
\newcounter{nb@intervals} % nombre de colonnes "intervalles"
\newcounter{numdiscont} % numéro de la discontinuité
```

Un compteur est créé pour chaque discontinuité : `discont1`, `discont2`, etc. Le compteur `discont`$\langle i \rangle$ contient le numéro du nœud précédant la i-ème discontinuité. La flèche partant de ce nœud ne sera pas tracée. Il faut un compteur de plus que le nombre de discontinuités.

```
\AtBeginDocument{% car maxdiscont a pu être modifié dans le préambule
    \stepcounter{maxdiscont}
    % il faut un compteur de plus que le nb de discontinuités
    \multido{\I=1+1}{\themaxdiscont}{\newcounter{discont\I}}
}
```

La commande `\mil` positionne son contenu sur la colonne `mil@row`, et réalise un décalage vertical de `\mil@shift` si le nombre de lignes de variations est pair. Pour cela nous avons besoin d'une variable de dimension et d'un compteur.

```
\newlength\mil@shift
\newcounter{mil@row}
```

Pour gérer correctement les zones interdites, avec la commande `\ZI`, de nombreuses variables, longueurs et compteurs, sont nécessaires. On peut créé des zones interdites sur 4 colonnes différentes maximum (on ne peut pas augmenter cette valeur sans revoir le code). Pour chacune, la hauteur des lignes est cumulée dans les variables `\ZIheighti`, ..., `\Ziheightiv`, et le numéro de colonne « intervalle » correspondant est enregistré dans un compteur `ZI1`, ..., `ZI4`. La variable `\ZIheight` permet de renvoyer la hauteur de la ZI courante pour le calcul final avant tracé des hachures. Les paramètres qui suivent sont également dédiés à la gestion des

dimensions et au tracé des zones interdites, à l'exception de `\tv@cellbox` qui sert aussi à la gestion des colonnes de valeurs.

```
\newcounter{maxZI} % nb max de ZI
\setcounter{maxZI}{4}
\newlength{\ZIheight}
\newlength{\ZIheighti}
\newlength{\ZIheightii}
\newlength{\ZIheightiii}
\newlength{\ZIheightiv}
\newlength{\ZIdepth}
\newlength{\ZIwidth}
\newcounter{nbZI} % nombre de ZI utilisées
\newcounter{numZI} % numéro de ZI courant
\newcounter{nbvarlignes} % nombre de lignes des variations
\multido{\I=1+1}{\themaxZI}{\newcounter{ZI\I}}
\newcounter{tv@icol} % numéro de colonne "intervalle"
\newcounter{tv@row} % numéro de ligne du tableau
\newsavebox{\tv@cellbox}
\newlength{\tv@cellheight}
\newlength{\tv@celldepth}
```

3.5 Les environnements `tablvar` et `tablvar*`

`\tablvarinit` Cette commande d'initialisation des compteurs est appelée au début de chaque environnement `tablvar`. On retire 1 à `nb@intervals` à cause de la manière dont on appelle les colonnes dans `tablvar` (voir plus bas). Les redéfinitions de `\extrarowheight` et `\arraystretch` seront locales à l'environnement `tablvar`. L'instruction `\setcounter{var@ligne}{0}` est nécessaire ici et pas seulement dans la partie variations car le compteur est utilisé par `\ZI` dans les lignes de signes.

```
\newcommand*{\tablvarinit}[1]{
    \setlength{\extrarowheight}{0pt} % paramètre de l'extension array
    \renewcommand{\arraystretch}{\tablvarstretch}
    \setlength{\arrayrulewidth}{\tvrulewidth}
    \setcounter{var@ligne}{0}
    \setcounter{numvr}{0}
    \setcounter{tv@row}{0}
    \setcounter{nb@intervals}{#1}
    \addtocounter{nb@intervals}{-1}
}
```

`\ZIinit` Initialisation des longueurs et compteurs qui peuvent être utilisés par `\ZI`. Le compteur `nbvarlignes` doit être non nul pour un tableau de signe seul.

```
\newcommand{\ZIinit}{
    \global\ZIheighti=0pt
    \global\ZIheightii=0pt
    \global\ZIheightiii=0pt
    \global\ZIheightiv=0pt
    \setcounter{nbZI}{0}
    \multido{\I=1+1}{\themaxZI}{\setcounter{ZI\I}{0}}
    \setcounter{nbvarlignes}{3}
}
```

`\tv@setheight` Cette macro sert à calculer la hauteur et la profondeur d'une cellule, qui seront enregistrées dans des variables globales afin que la commande `\ZI` puisse les récupérer. Elle est appelée à chaque « sortie » de cellule de la colonne de légendes grâce à la définition d'un nouveau type de colonne `A` (voir macro `\tvcoltypes`). La hauteur de la boîte de cellule `\@arstrutbox` ne donne pas la hauteur réelle de la ligne, car celle-ci peut être calibrée par d'autres cellules de la même ligne [12].

L'idée ici est de comparer la hauteur de `\@arstrutbox` avec la hauteur dans la 1re colonne de légendes, sur la même ligne, car c'est en général elle qui contient des éléments de grande hauteur. Évidemment si l'on place un contenu de grande hauteur ailleurs, dans une colonne de valeurs et non dans la colonne de légendes, le calcul ne fonctionnera pas. Mais dans ce cas, l'extension array gère de toute façon mal les hauteurs de ligne et le contenu de la cellule agrandie va toucher le filet horizontal. La solution est alors de placer un `\vstrut` ou `\cstrut` dans la colonne de légendes pour augmenter sa hauteur qui, de ce fait, sera bien enregistrée par `\tv@setheight`.

```
\newcommand\tv@setheight{%
    \global\tv@cellheight=\ht\tv@cellbox
    \ifthenelse{\value{var@ligne}=0}{
        \global\advance\tv@cellheight by \rowtopsep}{}
    \ifdim \tv@cellheight < \ht\@arstrutbox
        \global\tv@cellheight = \ht\@arstrutbox
    \fi
    \global\tv@celldepth=\dp\tv@cellbox
    \ifthenelse{\value{var@ligne}=0}{
        \global\advance \tv@celldepth by \rowbottomsep}{}
    \ifdim \tv@celldepth < \dp\@arstrutbox
        \global\tv@celldepth = \dp\@arstrutbox
    \fi
    \vrule height \tv@cellheight depth \tv@celldepth width 0pt
}
```

`\tvcoltypes` Grâce à l'extension array, nous pouvons définir 3 nouveaux types de colonnes : `A` pour la colonne des légendes (1re colonne), `i` pour les colonnes intervalles et `v{`⟨*pos*⟩`}` pour les colonnes de valeurs. L'argument ⟨*pos*⟩ peut prendre les valeurs `c` (centré), `l` (aligné au fer à gauche) ou `r` (aligné au fer à droite). En faire un paramètre optionnel ne fonctionne pas ici. Les colonnes d'intervalles sont en fait du type `p`, paragraphe centré, avec une largeur `\intervalwidth` et incrémentent le compteur interne `tv@icol`. La première colonne de légende, remet à zéro le compteur `tv@col` et incrémente le compteur de ligne `tv@row`.

La commande `\tvcoltypes` est appelée au début de chaque environnement `tablvar` et, de ce fait, ces types de colonnes ne sont reconnus qu'à l'intérieur d'un `tablvar` et ne créent pas d'incompatibilités avec d'autres extensions qui utiliseraient les mêmes lettres comme type de colonne.

```
\newcommand{\tvcoltypes}{
    % type de colonne A pour les légendes à gauche du tableau
    \newcolumntype{A}{%
        >{\setcounter{tv@icol}{0}\stepcounter{tv@row}\begin{lrbox}%
```

12. L'utilisation d'un type de colonne permettant un ajustement automatique de la hauteur tel que fourni par l'extension cellspace [9] de Josselin Noirel ne fonctionne pas ici : les barres et double barres ne sont pas correctement dessinées ; mais l'on pourra peut-être envisager une évolution grâce à l'extension tabularray [10] de Jianrui Lyu.

```
            \tv@cellbox $}%
        c%
        <{$\end{lrbox}\usebox{\tv@cellbox}\tv@setheight}}
    % type de colonne i pour les intervalles
    \newcolumntype{i}{>{\stepcounter{tv@icol}
        \centering\arraybackslash$}p{\intervalwidth}<{$}}
    % type de colonne v pour les valeurs
    \newcolumntype{v}[1]{% un argument optionnel ne fonctionne pas ici
        >{\ifthenelse{\value{tv@row}=1}{\begin{lrbox}\tv@cellbox $}{}}%
        ##1%
        <{\ifthenelse{\value{tv@row}=1}{$\end{lrbox}%
            \makebox[\colvalwidth][##1]{\usebox\tv@cellbox}}{}}}
}
```

`tablvar` (*env.*) La syntaxe de `tablvar` est : `\begin{tablvar}[`⟨*options*⟩`]{`⟨*nbintervals*⟩`}`. Le 1er paramètre optionnel permet de régler les options sous la forme *clé=valeur* ou bien on peut y placer un unique argument dimensionnel qui représente alors la largeur des colonnes intervalles. Le 2e paramètre (obligatoire) est le nombre d'intervalles. Fondamentalement, cet environnement n'est rien d'autre qu'un `array` dans lequel, après avoir effectué les initialisations, on a choisi les bons types de colonnes.

On commence par prendre en compte les options grâce à `\setkeys*` fourni par l'extension xkeyval. La version étoilée de `\setkeys` a l'avantage de ne pas produire d'erreur lorsqu'une clé n'est pas reconnue (celle-ci est passée à `\XKV@rm`). On l'interprète [13] alors comme une dimension pour le paramètre `\intervalwidth`.

Les colonnes de valeurs sont de type `c` (centré) par défaut mais les booléens `extleft` et `extright` permettent de différencier l'alignement dans la première colonne de valeur (type `B` comme begin) ou la dernière (type `E` comme end).

```
\newenvironment{tablvar}[2][]{%
    \setkeys*{tablvar}{#1} % fourni par xkeyval
    \if\XKV@rm\empty \else \setlength{\intervalwidth}{#1} \fi
    \tvcoltypes
    \ifextleft\newcolumntype{B}{v{l}}\else\newcolumntype{B}{v{c}}\fi
    \ifextright\newcolumntype{E}{v{r}}\else\newcolumntype{E}{v{c}}\fi
    \tablvarinit{#2}
    \ZIinit
    \begin{array}{%
        |@{\hspace{\firstcolsep}}%
        A@{\hspace{\firstcolsep}}%
        |@{\hspace{\bordercolsep}}%
        B@{\hspace{\innercolsep}}%
        i@{\hspace{\innercolsep}}%
        *{\value{nb@intervals}}{
            v{c}@{\hspace{\innercolsep}}%
            i@{\hspace{\innercolsep}}%
        }%
        E@{\hspace{\bordercolsep}}|%
    }
}{\end{array}}
```

`tablvar*` (*env.*) L'environnement `tablvar*` est une variante de `tablvar` (même syntaxe) où les

13. Un message d'erreur de dimension erronée peut donc signifier que l'on s'est trompé dans l'orthographe d'une clé.

première et dernière colonnes de valeurs sont alignées respectivement au fer à gauche (`l`) et à droite (`r`). L'appel à `\setkeys*` enregistre les paramètres optionnels dans les variables correspondantes qui seront donc utilisés par `tablvar`, mais `extleft` et `extright` seront fixés.

```
\newenvironment{tablvar*}[2][]{%
    \setkeys*{tablvar}{#1} % fourni par xkeyval
    \if\XKV@rm\empty \else \setlength{\intervalwidth}{#1} \fi
    \begin{tablvar}[extleft,extright]{#2}
}{\end{tablvar}}
```

3.6 La commande \variations

`\varloop` La commande `\varloop{`⟨*iter*⟩`}{`⟨*code*⟩`}` répète ⟨*code*⟩ (⟨*iter*⟩−1) fois (car la dernière ligne des variations doit subir un traitement particulier). Nous avons créé notre propre commande de boucle car les usuelles `\multido`, `\Multido` ou `\whiledo` plantent sur `\\` ou `\@arraycr` et la commande `\variations` a besoin d'utiliser une boucle dans un tableau. `\varloop` n'est autre qu'un `\ifthenelse` récursif.

```
\newcounter{loop@counter}
\newcommand{\varloop}[2]{%
    \setcounter{loop@counter}{#1}
    \addtocounter{loop@counter}{-1}% on boucle 1 fois de moins que #1
    \ifthenelse{\value{loop@counter}=0}{}{%
        #2 \varloop{\value{loop@counter}}{#2}%
    }
}
```

`\variations` La syntaxe est `\variations[`⟨*nblignes*⟩`]{`⟨*code*⟩`}` où ⟨*nblignes*⟩ est le nombre de lignes pour les variations (3 par défaut) ; ⟨*code*⟩ contient les commandes de positionnement et les séparateurs de colonnes `&`.
Le principe est que l'on parcourt ⟨*nblignes*⟩ fois le contenu de `\variations` ; à chaque itération, le compteur `var@ligne` est incrémenté, le compteur `var@noeud` est remis à 0 puis incrémenté à chaque commande `\pos`, mais le contenu de `\pos` n'est affiché et le nœud n'est effectivement créé que si la valeur du compteur `var@ligne` correspond à l'argument de ligne de `\pos`.
Les flèches et pointillés sont dessinés à la fin, quand tous les nœuds sont créés, mais il faut les tracer avant le `\\` final, sinon la compilation plante ! Le compteur `var@ligne` doit être remis à 0 au cas où il y a plusieurs parties variations. Cette remise à zéro s'effectue à la fin au cas où on mettrait des lignes de signes après une ligne variations.

```
\newcommand*{\variations}[2][3]{% #1=nblignes (3 par défaut)
    % (ré)initialisation des compteurs
    \setcounter{nbvarlignes}{#1}
    \setcounter{numdiscont}{0}
    \multido{\I=1+1}{\themaxdiscont}{\setcounter{discont\I}{0}}
    % boucle : on exécute le code #2 un nb de fois égal à (#1)-1
    \varloop{#1}{%
        \setcounter{var@noeud}{0}\setcounter{numvr}{0}
        % à chaque tour de boucle on réinitialise les compteurs de nœuds
        \stepcounter{var@ligne} % le numéro de ligne est incrémenté
        #2 % les nœuds sont fabriqués par le code #2 (avec \pos et \vr)
```

```
        \\ % retour ligne
        }
    % dernière itération -> flèches tracées AVANT \\ sinon bug !?
    \setcounter{var@noeud}{0}\setcounter{numvr}{0}
    \stepcounter{var@ligne} #2
    % tracé des flèches
    \addtocounter{var@noeud}{-1} % 1 flèche de moins que le nb de nœuds
    \setcounter{numdiscont}{1}
    \multido{\Ix=1+1,\Iy=2+1}{\thevar@noeud}{
        \ifthenelse{\value{discont\thenumdiscont}=\Ix}{
            % on saute les discontinuités
            \stepcounter{numdiscont}}{
            % sinon on trace la flèche N1->N2 puis N2->N3, etc.
            \fleche{N\Ix}{N\Iy}
        }
    }
    % tracé des pointillés pour les valeurs remarquables
    \multido{\Ix=1+1}{\thenumvr}{\vrconnect{X\Ix}{Y\Ix}}
    \setcounter{var@ligne}{0}
    \\ % dernier retour ligne du tableau
}
```

3.7 Les commandes de positionnement

\valpos `\valpos[`⟨*opt*⟩`]{`⟨*ligne*⟩`}{`⟨*valeur*⟩`}` sert à positionner les valeurs dans la partie variations ; ⟨*ligne*⟩ désigne la ligne où il faut placer ⟨*valeur*⟩ et produire le nœud, numéroté avec le compteur `var@noeud` et défini en appelant la commande `\noeud`. Les lignes de variations sont numérotées *du haut vers le bas* (et les lignes de signes portent toutes le numéro 0). Le 1er argument optionnel, `c` (centered, par défaut), `t` (top) ou `b` (bottom), est utilisé pour le positionnement des flèches. La macro `\valpos` sert de macro sous-jacente à la macro `\pos`.

```
\newcommand*{\valpos}[3][c]{
    \stepcounter{var@noeud}
    \ifthenelse{\thevar@ligne=#2}{
        \noeud[#1]{N\thevar@noeud}{#3}
    }{} % si ligne != #2, on ne fait rien
}
```

\zbox Place son contenu dans une boite de largeur nulle : affiche le contenu mais considère que l'espace occupé est nul pour ne pas altérer le calcul de la largeur de colonne. Cette macro est similaire à `\mathclap` de l'extension `mathtools` [13]. Sa syntaxe est : `\zbox[`⟨*pos*⟩`]{`⟨*contenu*⟩`}` où ⟨*opt*⟩ = `c` (par défaut), `l` (left) ou `r` (right).

```
\newcommand*{\zbox}[2][c]{\makebox[0pt][#1]{$#2$}}
```

\pos Dans `\pos`, si l'option `extleft` est activée, la première colonne de valeurs, celle où le compteur des colonnes intervalles, `\tv@icol`, est encore à 0, est alignée au fer à gauche : l'option de positionnement `l` (left) est enregistrée dans `\val@@pos`. Si c'est `extright` qui est activée, la dernière colonne de valeurs est alignée au fer à droite (option de positionnement `\val@@pos=r`). On se trouve dans cette dernière colonne, lorsque le compteur des colonnes intervalles a dépassé `nb@intervals` qui contient le nombres d'intervalles −1.

\pos* Dans la version étoilée, `\pos*{`⟨*ligne*⟩`}{`⟨*valeur*⟩`}`, la différence est qu'aucun

nœud n'est créé. Ceci est utile en particulier pour la toute première colonne contenant la légende ou pour positionner une valeur par dessus une flèche. Il n'y a pas lieu de se préoccuper de l'alignement, sauf si on voulait positionner une valeur aux extrémités sans la relier par des flèches, auquel cas on pourrait utiliser `\zbox[`⟨*pos*⟩`]` dans `\pos*`. Les anciennes commandes internes `\@pos` et `@@pos` ont été remplacées par `\tv@pos` et `\tv@@pos`, pour éviter un conflit avec la macro `\@pos` déjà utilisée par Daniel Flipo dans tabvar.

```
\newcommand*{\tv@pos}[3][c]{
    \def\val@@pos{c}
    \ifextleft
        \ifthenelse{\thetv@icol=0}{\def\val@@pos{l}}{}
    \fi
    \ifextright
        \ifthenelse{\thetv@icol>\thenb@intervals}{\def\val@@pos{r}}{}
    \fi
    \zbox[\val@@pos]{\valpos[#1]{#2}{#3}}
}
\newcommand*{\tv@@pos}[2]{\ifthenelse{\thevar@ligne=#1}{#2}{}}
\newcommand*{\pos}{\@ifstar{\tv@@pos}{\tv@pos}}
```

Voici les commandes de positionnement plus abstraites qui peuvent être utilisés à la place des commandes `\pos` et `\pos*`.

`\haut` `\haut{`⟨*valeur*⟩`}` place ⟨*valeur*⟩ sur la première ligne des variations. L'option de `\pos` n'est pas prise en charge.

```
\newcommand*{\haut}{\pos{1}}
```

`\bas` `\bas{`⟨*valeur*⟩`}` place ⟨*valeur*⟩ sur la dernière ligne des variations. L'option de `\pos` n'est pas prise en charge.

```
\newcommand*{\bas}{\pos{\value{nbvarlignes}}}
```

`\vdecal` La macro `\vdecal{`⟨*decal*⟩`}{`⟨*contenu*⟩`}` permet de réaliser un décalage vertical : le 1er paramètre est le décalage (positif = vers le haut ou négatif = vers le bas), le second est le contenu à placer. Elle est utilisée en particulier par `\mil`.

```
\newcommand*{\vdecal}[2]{\smash{\raisebox{#1}{$#2$}}}
```

`\smash` a pour effet d'annuler la hauteur de la boite afin de ne pas agrandir la ligne courante ainsi quelque soit le décalage, le tableau ne bouge pas (on pourrait même faire sortir le contenu à afficher du tableau).

`\mil` Pour un nombre n impair de lignes de variations, la commande `\mil{`⟨*valeur*⟩`}` positionne ⟨*valeur*⟩ sur la ligne $\frac{n+1}{2}$ de la partie variations. Cela correspond bien à la ligne du milieu : ligne 2 lorsqu'il y a 3 lignes. Par contre si n est pair, on place ⟨*valeur*⟩ sur la ligne $\frac{n}{2}$ mais avec un décalage vertical négatif (vers le bas) grâce à `\vdecal`. Ce décalage correspond à une profondeur de cellule augmenté de `0.5ex` (écart entre la ligne de base du texte et la ligne de centrage mathématique). Le résultat de ce calcul est stocké dans la variable de dimension `\mil@shift`. Au cas où l'on ne serait pas satisfait du résultat de `\mil`, on pourrait alors régler le décalage manuellement avec `\pos*{`⟨*ligne*⟩`}{\vdecal{`⟨*decal*⟩`}{`⟨*contenu*⟩`}}`.

```
\newcommand*{\mil}[1]{%
    \setcounter{mil@row}{\value{nbvarlignes}}
    \ifthenelse{\isodd{\value{nbvarlignes}}}{
```

```
        \addtocounter{mil@row}{1}
        \divide\value{mil@row} by 2
        \pos*{\themil@row}{\smash{#1}}
    }{
        \divide\value{mil@row} by 2
        \mil@shift = \dp\@arstrutbox
        \advance\mil@shift by 0.5ex
        \pos*{\themil@row}{\vdecal{-\mil@shift}{#1}}
    }
}
```

3.8 Barres, discontinuités, limites et valeurs remarquables

\barre La macro \barre permet de tracer une barre verticale pour marquer les séparations de colonne dans un tableau de signe, en passant à travers les 0. Son aspect est contrôlé par les paramètres \tvbarrewidth et \tvbarrecolor. Sa syntaxe est : \barre[⟨*valeur*⟩] où, en principe, on met 0 comme argument optionnel ou rien. Le \hspace{-0.5\tvbarrewidth} sert à obtenir un centrage parfait du \vrule en particulier si on souhaite un trait épais.

```
\newcommand*{\barre}[1][]{\makebox[0pt]{$#1$}%
    \color{tvbarrecolor}%
    \hspace{-0.5\tvbarrewidth}\vrule width \tvbarrewidth}
```

\bb La macro \bb, qui produit une double barre, reprend celle de l'extension variations de Christian Obrecht : \def\bb{\vrule\kern1pt\vrule}. Nous avons ajouté 1.5pt d'espace avant et après, afin d'aligner au mieux la double barre avec un 0 sur la ligne des x, lorsque la double barre se trouve à une extrémité. L'épaisseur de la double barre a été paramétrée par \bbrulewidth.

```
\newcommand*{\bb}{%
    \kern1.5pt\vrule width \bbrulewidth\kern1pt
    \vrule width \bbrulewidth\kern1.5pt}
```

\limg \limd Les macros \limg et \limd servent à positionner des limites à gauche ou à droite, en particulier aux bords des double barres. Le premier paramètre, optionnel, précise l'option de positionnement (c, l ou r) et le second *obligatoire* et le numéro de ligne, enfin le troisième est la valeur à placer.

\limg* \limd* Les versions étoilées n'acceptent comme argument que + ou -, \limg*{+} correspond à \limg{1}{+\infty}, \limg*{-} correspond à \limg{n}{-\infty} (pour une partie variations sur n lignes et de même pour \limd*.

```
\newcommand*{\@limg}[3][c]{%
    \zbox[r]{\valpos[#1]{#2}{#3\hspace{\limsep}}}}
\newcommand*{\@limd}[3][c]{%
    \zbox[l]{\valpos[#1]{#2}{\hspace{\limsep}#3}}}
\newcommand*{\@@limg}[1]{%
    \ifthenelse{\equal{#1}{+}}{\@limg{1}{+\infty}}{
    \ifthenelse{\equal{#1}{-}}{\@limg{\value{nbvarlignes}}{-\infty}}{
        \PackageError{tablvar}{Invalid argument for \string\limg*}
            {Only + or - are valid arguments for \string\limg*}
    }}
}
\newcommand*{\@@limd}[1]{%
```

```
    \ifthenelse{\equal{#1}{+}}{\@limd{1}{+\infty}}{
    \ifthenelse{\equal{#1}{-}}{\@limd{\value{nbvarlignes}}{-\infty}}{
        \PackageError{tablvar}{Invalid argument for \string\limd*}
            {Only + or - are valid arguments for \string\limd*}
    }}
}
\newcommand{\limg}{\@ifstar{\@@limg}{\@limg}}
\newcommand{\limd}{\@ifstar{\@@limd}{\@limd}}
```

`\discont` Associe un numéro de nœud à un compteur de discontinuité (chaque discontinuité a son propre compteur). La flèche entre le nœud précédent `\discont` (enregistré dans le compteur) et le nœud suivant ne sera pas tracée.

```
\newcommand*{\discont}{
    \ifthenelse{\thevar@ligne=1}{
        % on ne compte les discontinuités qu'une seule fois, sur ligne 1
        \ifthenelse{\thenumdiscont=0}{
            \ifthenelse{\thevar@noeud > 0}{% pas avant le 1er nœud
                \stepcounter{numdiscont}
                \setcounter{discont\thenumdiscont}{\thevar@noeud}
            }{}
        }{% on ne compte pas 2 fois la même discontinuité
            \ifthenelse{\thevar@noeud > \value{discont\thenumdiscont}}{
                \stepcounter{numdiscont}
                \setcounter{discont\thenumdiscont}{\thevar@noeud}
            }{}
        }
    }{}
}
```

`\bblim` `\bblim*` Ces macros servent à positionner des limites à gauche *et* à droite d'une double barre. Elles tracent la double barre, placent les limites et appellent `\discont`.
La syntaxe de `\bblim` est :
`\bblim{`⟨*ligne gauche*⟩`}{`⟨*limite gauche*⟩`}{`⟨*ligne droite*⟩`}{`⟨*limite droite*⟩`}`.
La syntaxe de`\bblim*` reprend celle de `\limg*` et `\limd*` : `\bblim*{`⟨+|-⟩`}{`⟨+|-⟩`}`.

```
\newcommand*{\@bblim}[4]{\limg{#1}{#2}\bb\discont\limd{#3}{#4}}
\newcommand*{\@@bblim}[2]{\limg*{#1}\bb\discont\limd*{#2}}
\newcommand*{\bblim}{\@ifstar{\@@bblim}{\@bblim}}
```

`\vr` La commande `\vr` fabrique un nœud pour chaque valeur remarquable. Les nœuds sont désignés par X1, X2… sur la ligne des x et Y1, Y2… sur les lignes de variations. Sa syntaxe est : `\vr[`⟨*ligne*⟩`]{`⟨*valeur*⟩`}`. Le paramètre optionnel ⟨*ligne*⟩ vaut 2 par défaut pour Y, et n'est pas pris en compte pour X (ligne 0), le second paramètre est la valeur à afficher. Les nœuds seront ensuite reliés par la commande `\vrconnect` (appelée par `\variations`) en fonction de leur numéro.

```
\newcommand*{\vr}[2][2]{% ligne 2 par défaut sauf si tv@row=1
    \stepcounter{numvr}
    \ifthenelse{\thetv@row=1}{\noeud{X\thenumvr}{#2}}{
        \ifthenelse{\thevar@ligne=#1}{\noeud{Y\thenumvr}{#2}}{}
    }
}
```

`\posvr` Malheureusement, on ne peut pas imbriquer une commande `\vr` et une commande `\pos` l'une dans l'autre. Mais on peut en fait imbriquer une commande `\noeud`

dans une autre. Ici ce ne sont pas les `\ifthenelse` contenues dans les commandes `\vr` ou `\pos` qui posent problème, mais les `\stepcounter` qui ne sont pas supportés à l'intérieur d'une commande `\noeud`. Nous avons donc créé la commande `\posvr` pour placer une valeur dans la partie variations qui soit à la fois un nœud pour les flèches et pour les pointillés de valeur remarquable. Sa syntaxe est la même que celle de la commande `\pos`.

```
\newcommand*{\posvr}[3][c]{%
    \stepcounter{numvr}
    \tv@pos[#1]{#2}{\noeud{Y\thenumvr}{#3}}}
```

3.9 Zones interdites

La gestion des zones interdites nécessite quelques sophistications algorithmiques car il faut ne dessiner un rectangle de hachures qu'une seule fois pour un ensemble de cellules superposées, sinon les hachures ne seront pas jointives. Pour les zones interdites colorées, on utilisera les mêmes macros qui donnent un ajustement idéal.

Une zone interdite possède un numéro et chaque appel de `\ZI` va cumuler la hauteur de cellule jusqu'au tracé, déclenché par `\ZI*`. On doit vérifier si une zone interdite non tracée existe déjà dans la même colonne, sinon on en crée une nouvelle.

`\ZIfind` On cherche s'il existe déjà une zone interdite (non tracée) pour la même colonne, auquel cas on renvoie son numéro (dans `numZI`), sinon on renvoie 0. Les ZI sont indicées par un compteur `ZI1`, `ZI2`, `ZI3` ou `ZI4` qui contient le numéro de la colonne intervalle à laquelle cette ZI appartient. Le compteur `nbZI` contient le nombre de ZI actives déjà créées. On repère l'indice de ZI en comparant la valeur du compteur `ZI\I` avec le numéro de la colonne intervalle courante.

```
\newcommand\ZIfind{% calcule numZI
   \setcounter{numZI}{0}
    \multido{\I=1+1}{\thenbZI}{% il existe déjà des ZI à la même colonne
        \ifthenelse{\value{ZI\I}=\value{tv@icol}}{
            \setcounter{numZI}{\I}
        }{}
    }
}
```

`\ZInew` Créé une nouvelle zone interdite en lui affectant un numéro (indice) et mémorise le numéro de la colonne intervalle où cette ZI est créée. Si le numéro de colonne (intervalle) courante ne correspond à aucune ZI enregistrée dans les compteurs `ZI\I` alors on incrémente le compteur `nbZI` et on enregistre ce numéro de colonne dans le compteur de `ZI\thenbZI` correspondant. Si jamais on dépasse 4, le nombre max de ZI possibles, `\ZInew` va générer une erreur en appelant `ZI\thenumZI` qui n'existe pas. Par contre, si une ZI de la même colonne a déjà été tracée précédemment (cela ne peut arriver que dans les tableaux de signes), le numéro de ZI de cette colonne sera remis à 0 par `\ZIreset` (voir macro suivante) auquel cas on récupère son indice pour créer cette nouvelle ZI.

```
\newcommand\ZInew{
    \multido{\I=1+1}{\thenbZI}{% on cherche s'il y a des ZI\I=0
        \ifthenelse{\value{ZI\I}=0}{\setcounter{numZI}{\I}}{}
```

```
    }
    \ifthenelse{\value{numZI}>0}{}{
        % si pas de ZI disponible il faut augmenter nbZI
        \stepcounter{nbZI}
        \setcounter{numZI}{\value{nbZI}}
    }
    \setcounter{ZI\thenumZI}{\value{tv@icol}}
}
```

`\ZIreset` `\ZIreset` est appelée lors du tracé d'une zone interdite dans la partie signe, afin de libérer l'indice de ZI correspondant qui pourra être réaffecté plus bas dans la même colonne. Son argument obligatoire est l'indice de ZI à remettre à 0. Son rôle est aussi de remettre à 0, la hauteur de cette ZI nouvellement créée.

```
\newcommand*\ZIreset[1]{
    \ifnum #1 > 0 \setcounter{ZI#1}{0} \fi
    \ifnum #1 = 1
        \global\ZIheighti=0pt
    \else \ifnum #1 = 2
        \global\ZIheightii=0pt
    \else \ifnum #1 = 3
        \global\ZIheightiii=0pt
    \else \ifnum #1 = 4
        \global\ZIheightiv=0pt
    \fi\fi\fi\fi
}
```

`\ZIaddheight` La macro `\ZI@addheight` sert à cumuler la hauteur et la profondeur de la ligne courante dans une des 4 variables de dimension `\ZIheighti`, ..., `\ZIheightiv` correspondant à la zone interdite appelante. On récupère hauteur et profondeur qui ont été enregistrés dans les variables globales `\tv@cellheight` et `\tv@celldepth`, par la colonne de légendes de la même ligne. Le numéro de la ZI (son indice) est passé en argument.

```
\newcommand*\ZIaddheight[1]{
    \ZIheight=0pt
    \advance\ZIheight by \tv@cellheight
    \advance\ZIheight by \tv@celldepth
    \advance\ZIheight by 0.5\arrayrulewidth
    \ifnum #1 = 1
        \global\advance\ZIheighti by \ZIheight
    \else \ifnum #1 = 2
        \global\advance\ZIheightii by \ZIheight
    \else \ifnum #1 = 3
        \global\advance\ZIheightiii by \ZIheight
    \else \ifnum #1 = 4
        \global\advance\ZIheightiv by \ZIheight
    \fi\fi\fi\fi
}
```

`\ZIgetheight` On récupère, avant le tracé, la hauteur globale de la ZI (dont le numéro est donné en argument) dans la variable `\ZIheight`.

```
\newcommand*\ZIgetheight[1]{%
    \ifnum #1=1
```

```
        \global\ZIheight=\ZIheighti
    \else \ifnum #1 = 2
        \global\ZIheight=\ZIheightii
    \else \ifnum #1 = 3
        \global\ZIheight=\ZIheightiii
    \else \ifnum #1 = 4
        \global\ZIheight=\ZIheightiv
    \fi\fi\fi\fi
}
```

\ZI La macro \ZI est à placer dans les cellules où l'on souhaite produire une zone interdite. Elle ne déclenche le tracé des hachures que sur la dernière ligne des variations (par un appel à \ZI*).

On crée une discontinuité puis on récupère l'indice de ZI (numZI) grâce à \ZIfind. Si \ZIfind renvoie 0, aucune ZI active n'existe pour la colonne intervalle courante, auquel cas on crée un nouvelle ZI avec \ZInew. Et enfin on cumule la hauteur de la ligne de cette ZI grâce à \ZIaddheight.

La macro possède un paramètre optionnel \ZI[⟨*hauteur*⟩] qui est un supplément de hauteur global. Celui-ci n'est, en principe, plus nécessaire, mais nous l'avons conservé pour des raisons de compatibilité avec les versions antérieures à tablvar 2.0. Ce paramètre permettrait éventuellement de faire un ajustement fin de la hauteur.

\ZI* C'est en fait la macro ZI* qui déclenche le tracé de la zone interdite en appelant la commande \hachure ou la commande \ZIcouleur en fonction de l'option ZItype. Elle utilise la hauteur cumulée, précédemment enregistrée dans la variable de dimension correspondant à la ZI (par les commandes \ZI placés dans la même colonne). Dans la macro \variations, \ZI* est en fait appelée par ZI sur la dernière ligne des variations, sans que l'utilisateur n'ait à intervenir.

Par contre, dans un tableau de signes sans partie variations il faut placer explicitement des \ZI* à la place des \ZI, au moment où l'on souhaite déclencher le tracé des hachures, sur la dernière ligne d'un bloc de hachures. La commande \ZI* possède le même argument optionnel ZI*[⟨*hauteur*⟩].

```
\newcommand*{\@ZI}[1][0pt]{%
    \discont
    \ifthenelse{\thevar@ligne=\value{nbvarlignes}}{\@@ZI[#1]}{%
        \ZIfind
        \ifnum \thenumZI = 0 \ZInew \fi
        \ZIaddheight{\thenumZI}
    }
}

\newcommand*{\@@ZI}[1][0pt]{
    \discont
    \ZIfind
    \ifnum \thenumZI > 0 \ZIgetheight{\thenumZI} \else \ZIheight=0pt \fi
    \advance\ZIheight by \tv@cellheight
    \advance\ZIheight by 0.5\arrayrulewidth
    \advance\ZIheight by #1
    \ZIdepth = \tv@celldepth
    \ZIwidth = \intervalwidth
    \advance\ZIwidth by 2\innercolsep
```

```
    \advance\ZIwidth by \colvalwidth
    \advance\ZIwidth by -1pt % au bord des double barres
    \advance\ZIwidth by \ZIaddwidth
    \ifthenelse{\equal{\ZItype}{h}}{
        \hachure{-0.5\ZIwidth,-\ZIdepth}{0.5\ZIwidth,\ZIheight}
    }{
        \ZIcouleur{-0.5\ZIwidth,-\ZIdepth}{0.5\ZIwidth,\ZIheight}
    }
    \ZIreset{\thenumZI}
}

\newcommand*{\ZI}{\@ifstar{\@@ZI}{\@ZI}}
```

`\ZIh` `\ZIc` Nous avons conservé les noms des anciennes macros `\ZIh` et `ZIc` pour les zones interdites hachurées ou colorées pour des raisons de compatibilité, mais elles sont désormais basées sur `\ZI` ou `\ZI*` (en fait sur `\@ZI` ou `\@@ZI`). La macro `\ZIh` possède toujours une version étoilée `\ZIh*` pour le tracé des hachures dans les lignes de signe, `\ZIc` n'en possédait pas.

```
\newcommand*{\ZIc}[1][0pt]{\renewcommand{\ZItype}{c}\@@ZI[#1]}
\newcommand{\ZIh}{\renewcommand{\ZItype}{h}\ZI}
```

Références

[1] *Tableaux*, Nicolas Kisselhoff, CTAN.

[2] *Tableaux de variations : 'tabvar'*, Daniel Flipo, CTAN, v1.8 16/07/2022.

[3] *L'extension variations*, Christian Obrecht, CTAN, v0.3 13/09/2006.

[4] *tablor.sty La machine à créer des tableaux de signes et de variations*, Guillaume Connan, CTAN, v4.07 09/05/2010.

[5] *tableauVariations – Variation tables in METAPOST*, Frédéric Mazoit, CTAN, 2005.

[6] *Tkz-Tab*, Alain Matthes, CTAN, v2.12c 29/04/2020.

[7] *A new implementation of LATEX's tabular and array environment*, Frank Mittelbach, David Carlisle, CTAN, v2.5g revised 16/10/2023.

[8] *The makecell package*, Olga Lapko, CTAN, v0.1e 03/08/2009.

[9] *The cellspace package*, Josselin Noirel, CTAN, v1.9.0 04/01/2022.

[10] *Tabularray Typeset Tabulars and Arrays with LaTeX3*, Jianrui Lyu, CTAN, v2023A, 01/03/2023. v1.9.0 04/01/2022.

[11] *The spacingtricks package*, Antoine Missier, CTAN, v1.7 28/07/2023.

[12] *The xkeyval package*, Hendri Adriaens, CTAN, v2.9 16/06/2022.

[13] *The mathtools package*, Morten Høgholm, Lars Madsen, CTAN, v1.23 17/01/2020.

[14] Miscellaneous mathematical macros – The *mismath package*, Antoine Missier, CTAN, v3.1 16/06/2024.

[15] *L'extension frenchmath*, Antoine Missier, CTAN, v3.1 07/05/2024.