L'extension pour TEX et LATEX

systeme

v0.1

27 Février 2011

Manuel de l'utilisateur

Christian TELLECHEA
unbonpetit@gmail.com

Résumé

Cette petite extension met en forme des systèmes d'équations ou d'inéquations où les termes et les signes sont alignés verticalement, tout en permettant une saisie quasi naturelle.

Table des matières

0.1 Avant propos

Tout a recommencé, comme chaque année lorsque j'enseigne les systèmes d'équations, par un (petit) énervement concernant la difficulté de la saisie pour avoir une mise en forme acceptable. C'est à chaque fois un casse tête et une perte de temps conséquente de se battre avec les tableaux LaTeX pour obtenir *in fine* des systèmes avec un alignement correct, d'où le petit énervement, surtout lorsque, insouciant, on commence à taper ses sujets vers 23h pour le lendemain.

Fort de ce constat, je me suis dit qu'il allait falloir écrire des macros pour être débarrassé de la difficulté de la saisie. Et tant qu'à faire, autant écrire des macros en plain εTeX, que tout le monde puisse en profiter [1]. Ces macros sont réunies dans cette petite extension maintenant à peu près fonctionnelle.

1 Fonctionnalités de l'extension

1.1 La commande `\systeme`

Pour l'utiliser l'extension « systeme » , il faut écrire :

- `input systeme.tex` lorsqu'on est utilise TeX ou pdfTeX ;
- `\usepackage{systeme}` dans le préambule lorsqu'on utilise LaTeX.

L'extension **xstring** est requise et est chargée si cela n'a pas été le cas.

La commande principale est `\systeme` dont l'argument obligatoire contient les équations séparées par une virgule :

La commande `\systeme`

```
Résoudre \systeme{2a-3b+4c=2,
a+8b+5c=8,
-a+2b+c=-5}
```

Résoudre $\begin{cases} 2a-3b+4c=2 \\ a+8b+5c=8 \\ -a+2b+c=-5 \end{cases}$

La commande `\systeme` fonctionnera en mode math ou non et donnera un résultat correct si toutes les inconnues se trouvent dans le membre de gauche, le membre de droite étant celui des constantes. Les équations doivent être *développées*, c'est-à-dire que chaque terme est séparé de son voisin par un "+" ou un "-". De plus, les inconnues doivent être des lettres minuscules non accentuées, c'est-à-dire tout caractère de « a » à « z ». Comme l'extension est écrite en TeX et donc utilise un tableau fait à l'aide de `\halign`, il faudra se méfier du grabuge que peut provoquer cette primitive lorsqu'elle se trouve nue dans les environnements `tabular` de LaTeX et si l'on veut mettre la commande `\systeme` dans un tableau LaTeX, il faudra prendre la précaution de l'envelopper dans une `\hbox`.

L'alignement construit sera un tableau précédé d'une accolade et aura les spécificités suivantes :

- les signes d'égalité ou d'inégalité séparant les deux membres sont alignés ;
- les signes "+" ou "-" séparant chaque terme du membre gauche sont alignés ;
- chaque terme du membre de gauche se trouve dans une colonne au fer à droite ;
- le membre de droite se trouve dans une colonne au fer à gauche ;
- les espacements mathématiques entre colonnes seront corrects.

Si une inconnue est manquante dans une équation, la colonne du tableau reste vide :

Inconnues manquantes

```
\systeme{a-2b=3,
b-3c=4,
-a+4c=-1}
```

$\begin{cases} a-2b = 3 \\ b-3c = 4 \\ -a+4c = -1 \end{cases}$

On peut également avoir une, plusieurs, ou toutes les équations sans second membre :

1. Enfin, tout le monde, c'est vite dit ! Cette extension n'est pas compatible avec ConTeXt car, pour une raison que je ne m'explique pas, **xstring** n'est pas utilisable avec ConTeXt. Si quelqu'un a une explication (et éventuellement un remède), je lui serais très reconnaissant qu'il me contactepar **email** !

Équation sans second membre

```
\systeme{2a+3b-c=4,
b-2c,-a+2b+3c}
```

$$\begin{cases} 2a+3b-c=4 \\ b-2c \\ -a+2b+3c \end{cases}$$

Dans l'argument de la commande `\systeme`, lorsque deux virgules se suivent, une équation vide, c'est-à-dire une ligne vide est insérée. Malgré cette facilité, pour augmenter l'espacement vertical entre les équations, il vaut mieux utiliser la commande `\syslineskipcoeff`, voir page 4.

Ligne vide

```
\systeme{a-2b=3,,2a+5b=7}
```

$$\begin{cases} a-2b=3 \\ \\ 2a+5b=7 \end{cases}$$

1.2 Tri alphabétique

Quel que soit l'ordre dans lequel sont entrées les inconnues lors de la saisie, elles seront triées par ordre alphabétique à l'affichage :

Tri alphabétique

```
\systeme{2y+x-3z=4,
z-y+2x=-1,
-2x+3z-4y=0}
```

$$\begin{cases} x+2y-3z=4 \\ 2x-y+z=-1 \\ -2x-4y+3z=0 \end{cases}$$

Les signes "+" ne sont pas affichés lorsqu'ils précèdent le premier terme d'une équation, ce que l'on peut observer dans la première équation.

Le tri alphabétique est une facilité mais il peut s'avérer gênant surtout dans les systèmes 4×4 où, bien souvent, la 4e inconnue est « t » :

Tri alphabétique indésirable

```
\systeme{x+2y-3z+t=0,
2x-y-z+3t=4,
2y+3z+4t=-1,
3x-2z-2t=3}
```

$$\begin{cases} t+x+2y-3z=0 \\ 3t+2x-y-z=4 \\ 4t+2y+3z=-1 \\ -2t+3x-2z=3 \end{cases}$$

On aimerait bien que l'inconnue t soit en 4e position dans toutes les équations. Pour cela, il faut forcer un tri différent du tri alphabétique avec l'argument optionnel de la commande `\systeme`. Cet argument optionnel doit contenir la liste des inconnues, sans aucun espace entre elles, et dans l'ordre où l'on souhaite les voir affichées dans chaque équation. Si une inconnue est omise, elle ne sera pas affichée, et si une inconnue figure dans l'argument optionnel alors qu'elle n'existe pas dans le système, elle sera ignorée.

Ici, on affiche deux fois le même système avec deux ordres différents :

Tri forcé

```
\systeme[xyzt]{x+2y-3z+t=0,
2x-y-z+3t=4,
2y+3z+4t=-1,
3x-2z-2t=3}

\systeme[ztyx]{x+2y-3z+t=0,
2x-y-z+3t=4,
2y+3z+4t=-1,
3x-2z-2t=3}
```

$$\begin{cases} x+2y-3z+t=0 \\ 2x-y-z+3t=4 \\ 2y+3z+4t=-1 \\ 3x-2z-2t=3 \end{cases}$$

$$\begin{cases} -3z+t+2y+x=0 \\ -z+3t-y+2x=4 \\ 3z+4t+2y=-1 \\ -2z-2t+3x=3 \end{cases}$$

1.3 Inéquations

Dans chaque ligne, le signe susceptible de séparer les deux membres d'une équation sont l'un de cette liste : `=`, `<`, `>`, `<=`, `>=`, `\leq`, `\geq`, `\leqslant` et `\geqslant`. Les deux derniers ne sont utilisables que si l'extension

amssymb a été chargée.

Les signes <= et >= sont remplacés à l'affichage par \leq, \geq, qui donnent $\leq$ ou $\geq$.

Inéquations

```
\systeme{x+y-2z>4,
2x-y+z\geq-1,
3x-2y+z<=3}
```

$$\begin{cases} x+ \ y-2z>4 \\ 2x- \ y+ \ z\geq -1 \\ 3x-2y+ \ z\leq 3 \end{cases}$$

Pour choisir une autre substitution à « >= » ou « <= » ou pour en créer une pour tout autre signe d'égalité, on doit utiliser la commande :

```
\sysequivsign{signe}{substitution}
```

Voici le même système où l'on définit la substitution de « <= » avec « \leqslant » comme on l'observe à la troisième équation :

Redéfinir une substitution

```
\sysequivsign{<=}{\leqslant}
\systeme{x+y-2z>4,
2x-y+z\geq-1,
3x-2y+z<=3}
```

$$\begin{cases} x+ \ y-2z>4 \\ 2x- \ y+ \ z\geq -1 \\ 3x-2y+ \ z\leqslant 3 \end{cases}$$

1.4 Nouveaux signes d'égalité

Avec la commande \sysaddeqsign, on peut créer un nouveau signe susceptible de séparer les deux membres des équations. Il faut écrire :

```
\sysaddeqsign{<nouveau signe>}
```

Mettons ici que l'on créé le nouveau signe [2] d'égalité « ~ » en écrivant :

```
\sysaddeqsign{~}
```

Puis, mettons que l'on veuille ensuite remplacer ce nouveau signe par « \approx » dans l'affichage final. On devra écrire :

```
\sysequivsign{~}{\approx}
```

En voici l'illustration dans cet exemple :

Nouveau signe

```
\sysaddeqsign{~}
\sysequivsign{~}{\approx}
\systeme{2a+b-c~6,a-4b~4}
```

$$\begin{cases} 2a+ \ b-c\approx 6 \\ a-4b \quad \approx 4 \end{cases}$$

Par la suite, on peut supprimer ce signe ou n'importe quel autre déjà existant il faut utiliser la commande \sysremoveeqsign et écrire :

```
\sysremoveeqsign{~}
```

1.5 Coefficients décimaux

À première vue, la virgule étant utilisée pour séparer les équations, il n'est pas possible d'écrire des coefficients décimaux. On peut spécifier un autre caractère pour séparer les différentes équations avec le second argument optionnel de la commande \système. Ici, on prend « : » [3] ce qui permet d'écrire des coefficients décimaux. Le comportement de la virgule est redéfinit à l'intérieur de la commande \systeme de telle sorte qu'elle ne soit pas suivie d'une espace, comme c'est le cas en mode mathématique.

2. La création d'un nouveau signe est possible même si son code de catégorie est actif.
3. Ici encore, il est possible de choisir un caractère de code de catégorie actif, comme c'est le cas de « : » lorsque l'option « frenchb » est spécifiée au package babel.

Coefficients décimaux

```
\systeme[][:]{1,5x-0,45y=0,7:x-0,8y=1,4}
```

$$\begin{cases} 1{,}5x - 0{,}45y = 0{,}7 \\ x - 0{,}8y = 1{,}4 \end{cases}$$

1.6 Espacement des lignes

On peut faire varier l'espacement entre les lignes avec la commande `\syslineskipcoeff` dont l'argument est un nombre qui viendra multiplier la valeur de `\baselinskip`. Par défaut, l'argument vaut 1.

Espacement variable

```
\systeme{x+2y-z=0,2x-y+z=1,x-3y+2z=1}
\syslineskipcoeff{1.5}\quad
\systeme{x+2y-z=0,2x-y+z=1,x-3y+2z=1}
```

$$\begin{cases} x + 2y - z = 0 \\ 2x - y + z = 1 \\ x - 3y + 2z = 1 \end{cases} \quad \begin{cases} x + 2y - z = 0 \\ 2x - y + z = 1 \\ x - 3y + 2z = 1 \end{cases}$$

1.7 Inconnues entre accolades

La détection des lettres représentant les inconnues se fait même lorsque ces inconnues sont dans des accolades. On peut donc écrire des équations où se trouve par exemple `\frac{x+1}{2}` :

Équations avec fractions

```
\systeme{\frac{x+1}{3}+\frac{y}{3}=\frac{1}{2},
\frac{x}{2}-\frac{3y-1}{2}=-2}
```

$$\begin{cases} \frac{x+1}{3} + \frac{y}{3} = \frac{1}{2} \\ \frac{x}{2} - \frac{3y-1}{2} = -2 \end{cases}$$

Certes, le rendu est discutable mais ce n'est pas le but de cette extension de traiter ce genre d'équations…

2 Algorithme

Voici les notations utilisées dans l'algorithme :

- les principales variables utilisées sont en bleu ;
- les ⟨constantes⟩ sont entre chevrons ;
- $\texttt{car}_{\texttt{n}}$(variable) est le caractère n de la variable ;
- x ⟵ y est une assignation qui signifie que x reçoit y ;
- x $=\!\oplus$ y est une concaténation qui signifie que la chaine y est ajoutée à la fin de x. De la même façon, x $\oplus\!=$ y ajoute la chaine y au début de x ;
- ∃ x signifie que la variable x existe ;
- gauche(var1,var2) est dans var1 ce qui se trouve à gauche de la première occurrence de var2. Même chose pour droite(var1,var2) sauf que c'est ce qui est à droite.

Dans les grandes lignes, voici l'algorithme qui est utilisé pour parcourir, analyser, découper, trier et reconstruire un système avec la commande

```
\systeme[arg_opt#1][arg_opt#2]{argument obligatoire}
```

1. insérer un ⟨\begingroup⟩
2. ⟨\mathcode`\,⟩ ⟵ ⟨"013B⟩
3. si ⟨arg_opt#1⟩ = ⟨vide⟩
   tri_auto ⟵ ⟨vrai⟩
   list_inconnues ⟵ ⟨vide⟩
   sinon
   tri_auto ⟵ ⟨faux⟩
   list_inconnues ⟵ ⟨arg_opt#1⟩
4. si ⟨arg_opt#2⟩ = ⟨vide⟩
   séparateur ⟵ ⟨,⟩

sinon
séparateur ⟵ ⟨arg_opt#2⟩

5. numligne ⟵ ⟨1⟩
arg_restant ⟵ ⟨argument obligatoire⟩
6. si séparateur ∈ arg_restant
éq_actuelle ⟵ gauche(arg_restant,séparateur)
arg_restant ⟵ droite(arg_restant,séparateur)
sinon
éq_actuelle ⟵ arg_restant
arg_restant ⟵ ⟨vide⟩
7. si l'éq_actuelle contient un signe contenu dans la liste des signes d'égalité
signe[numligne] ⟵ signe
membre_G ⟵ gauche(éq_actuelle,signe)
membre_D[numligne] ⟵ droite(éq_actuelle,signe)
sinon
membre_G ⟵ éq_actuelle
8. si $\mathtt{car}_1$(membre_G) ∉ {⟨+⟩, ⟨-⟩}
membre_G ⊕= ⟨+⟩
9. signe_actuel ⟵ $\mathtt{car}_1$(membre_G)
membre_G ⟵ droite(membre_G,signe_actuel)
   (a) si membre_G contient ⟨+⟩ ou ⟨-⟩
signe ⟵ première occurrence de ⟨+⟩ ou ⟨-⟩ dans membre_G
terme_actuel ⟵ gauche(membre_G,signe)
membre_G ⟵ droite(membre_G,signe)
sinon
terme_actuel ⟵ membre_G
membre_G ⟵ ⟨vide⟩
   (b) chercher alpha, la lettre représentant l'inconnue dans le terme_actuel
   (c) signe[numligne,alpha] ⟵ signe_actuel
   (d) terme[numligne,alpha] ⟵ terme_actuel
   (e) si tri_auto = ⟨vrai⟩ et alpha ∉ list_inconnues
insérer alpha à sa place alphabétique dans list_inconnues
   (f) si membre_G ≠ ⟨vide⟩
signe_actuel ⟵ signe
retourner en 9a
10. si arg_restant ≠ ⟨vide⟩
numligne ⟵ numligne + 1
aller en 6
11. nb_inconnues ⟵ nombre de caractères de list_inconnues
12. nb_lignes ⟵ numligne
13. fabriquer le préambule du \halign (1 colonne au fer à droite pour la première inconnue, 2 de ces colonnes pour les autres inconnues, 1 colonne pour le signe d'égalité et 1 colonne au fer à gauche pour le membre de droite) :
   (a) code_systeme ⟵ ⟨vide⟩
   (b) code_systeme =⊕ ⟨\hfil#&\hfil#&⟩ × (nb_inconnues − 1)
   (c) code_systeme =⊕ ⟨\hfil#&#&#\hfil\null\cr⟩
14. numligne ⟵ ⟨1⟩
15. numlettre ⟵ ⟨1⟩ premier_terme ⟵ ⟨vrai⟩
16. alpha ⟵ $\mathtt{car}_{\mathtt{numlettre}}$(list_inconnues)
   (a) si ∃ signe[numligne,alpha]
si signe[numligne,alpha] = ⟨+⟩

si numlettre $\neq 1$ et premier_terme $= \langle$faux$\rangle$
code_systeme $=\oplus$ signe[numligne,alpha]
sinon
si numlettre $= 1$
terme[numligne,alpha] $\oplus=$ signe[numligne,alpha]
sinon
code_systeme $=\oplus$ signe[numligne,alpha]
premier_terme $\longleftarrow \langle$faux$\rangle$

(b) si numlettre $\neq 1$
code_systeme $=\oplus \langle$&$\rangle$

(c) si $\exists$ terme[numligne,alpha]
code_systeme $=\oplus$ terme[numligne,alpha]

(d) code_systeme $=\oplus \langle$&$\rangle$

(e) si numlettre < nb_inconnues
numlettre $\longleftarrow$ numlettre+$\langle$1$\rangle$
aller en 16

(f) si $\exists$ signe[numligne]
code_systeme $=\oplus$ signe[numligne]

(g) si $\exists$ membre_D[numligne]
code_systeme $=\oplus \langle$&$\rangle$membre_D[numligne]

(h) si numligne $<$ nb_lignes
code_systeme $=\oplus \langle$\cr$\rangle$
numligne $\longleftarrow$ numligne $+ \langle$1$\rangle$
aller en 15

17. code_systeme $=\oplus \langle$\cr$\rangle$
18. se mettre en mode mathématique s'il y a besoin et insérer $\langle$\left{\vcenter{\halign{code_systeme}}\right.$\rangle$
19. insérer $\langle$\endgroup$\rangle$.

Pour finir et rendre les choses compréhensibles, voici un système où les frontières des 7 colonnes (3 inconnues et donc $2 \times 3 + 1 = 7$ colonnes) sont visibles :

$$\left\{\begin{array}{|rcrcrcr|}-10x&+&4y&-&5z&=&-1\\ x&&&-&7z&=&3\\ 2x&-&y&&&=&0\end{array}\right.$$

À faire : dans les systèmes saisis, il faut permettre que les inconnues soit des lettres indicées comme "x_1", "x_2", etc. Pour cela, il faudra améliorer l'algorithme utilisé et regarder si les lettres détectées sont suivies de "_". Ça sera pour la prochaine version !

$$\begin{array}{c}\star\\ \star\quad\star\end{array}$$

J'espère que cette extension vous sera utile et surtout que le code ne comporte pas trop de bugs car il a été écrit assez rapidement sans vraiment faire les tests qui s'imposent... Un **email** pour me signaler tout dysfonctionnement, toute proposition d'amélioration ou même tout commentaire sur cette extension sera le bienvenu.

Christian Tellechea