Chronique 10

Multido

Dans la chronique 6 j'ai un peu parlé de l'instruction `\multido` qui nécessite l'extension `multido` ou l'extension `pstricks-add`.
Je vais un peu préciser les choses dans cette chronique-ci.

La documentation de l'extension se trouve dans le fichier `multido-doc.pdf` disponible à l'adresse :

`http://www.ctan.org/tex-archive/macros/generic/multido`

10.1 Rappel

La syntaxe de l'instruction `\multido` est : `\multido{var=début+pas}{nombre}{à répéter}` comme dans : `\multido{\i=0+5}{7}{\i\ }` (voir plus bas).

On peut également répéter quelque chose sans utiliser de variable.
Par exemple si on veut laisser de la place pour écrire une réponse comme dans _ _ _ _ _ _ _ _ _ _
il suffira d'entrer `\multido{}{10}{\_}` qui trace dix caractères `_` à la suite.

Mais il y a une chose importante à savoir quand on utilise l'instruction `\multido` avec des variables, c'est que ces variables sont typées : la première lettre de la variable détermine son type.

10.2 Variable de type `integer`

Si la variable a un nom qui commence par `i` ou par `I`, elle sera de type entier (`integer`).
Pour que le `\multido` soit valide, il faut que le nombre de départ et le pas de la variable utilisée soient des nombres entiers. Voyons quelques exemples :

	La séquence	donne
1	`\multido{\i=0+5}{7}{\i\ }`	0 5 10 15 20 25 30
2	`\multido{\i=0+5}{5}{(\i,\i)\ }`	(0,0) (5,5) (10,10) (15,15) (20,20)
3	`\multido{\i=0+5,\I=1+3}{5}{(\i,\I)\ }`	(0,1) (5,4) (10,7) (15,10) (20,13)
4	`\multido{\i=0+-5}{7}{\i\ }`	0 -5 -10 -15 -20 -25 -30
5	`$\multido{\i=0+-5}{7}{\i\ }$`	$0\ -5\ -10\ -15\ -20\ -25\ -30$
6	`\multido{\i=-5+1}{7}{$\i\ $}`	$-5\ -4\ -3\ -2\ -1\ 0\ 1$
7	`\multido{\i=0+0.5}{7}{\red \i\ }`	.50 0 0 0 0 0 0
8	`\multido{\i=0+5}{5}{\red 2*\i\ }`	2*0 2*5 2*10 2*15 2*20

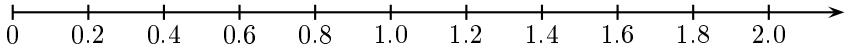
J'ai appelé \i la variable utilisée dans chaque exemple ; on aurait pu l'appeler \I ou \ia ou \Ia ou... (je n'ai pas essayé toutes les combinaisons de lettres possibles). Mais on ne peut pas l'appeler \i1 car cela donne un résultat surprenant (mais compréhensible).
Cette variable \i est une variable locale qui n'a d'existence que dans la boucle.

On voit dans les exemples que :

- ce que j'ai appelé `nombre` dans la syntaxe de l'instruction `multido` donne le nombre de résultats obtenus après l'exécution de \multido (1) ;
- on peut faire afficher des couples en utilisant une seule variable (2) ou deux (3). On peut de même utiliser des couples pour placer des points ou tracer des segments (on le verra dans une future chronique) ;
- un pas ne se soustrait pas mais peut être négatif ; la séquence \i=0-5 produit une erreur et doit être remplacée par \i=0+-5 (4) ;
- le nombre de départ peut être négatif (6) ;
- l'instruction \multido peut être utilisée en mode mathématique (5), mais il vaut mieux ne mettre en mode mathématique que ce que l'on affiche (6) ;
- si le pas est non entier, il n'y a pas de message d'erreur mais le résultat est faux (7). Il y a même quelque chose de bizarre au niveau de la couleur ;
- enfin on ne peut pas faire des calculs avec la variable utilisée (8).

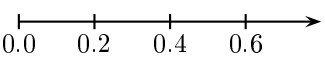
10.3 Variable de type `number`

Comment faire si l'on veut graduer un axe tous les centimètres de 0,2 en 0,2 ?

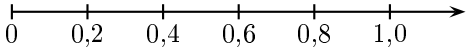


Il ne faut pas utiliser une variable de type `integer` puisqu'on a vu qu'on ne pouvait pas y ajouter un nombre non entier ; on va utiliser une variable de type `number` dont la valeur de début peut être un nombre non entier, tout comme le pas.
Pour tracer le graphique du dessus, il faut deux variables : \i (de type `integer`) qui va permettre de tracer les graduations tous les centimètres (avec \psline), et une variable de type `number` pour écrire 0, 0.2, ..., 2.0 ; cette variable partira de 0 et aura pour incrément 0.2.
Une variable de type `number` doit avoir un nom commençant par `n` ou `N` comme \n, \N, \nx, \Ny, etc. et ne pas contenir de chiffres dans son nom.
Les deux variables \i et \n doivent prendre le même nombre de valeurs.
Le code du graphique est :

```
\psset{unit=1cm,arrowsize=2pt 3}%  paramètres
\begin{pspicture}(-1,-0.5)(11,0.2)
\psline{->}(0,0)(11,0)%            tracé de l'axe
\multido{\i=0+1,\n=0+0.2}%         définition des variables
        {11}%                      nombre de valeurs
        {
        \psline(\i,-0.1)(\i,0.1)%  tracé de la graduation
        \uput[d](\i,0){\n}%        légendes
        }
\end{pspicture}
```

On obtient : 

en définissant \n ainsi : `\n=0.0+0.2`.

On verra plus tard comment obtenir : 

avec des nombres décimaux écrits avec des virgules et pas des points.

Quelques exemples de boucles traitées avec une variable de type `number` :

	La séquence	donne
1	`\multido{\n=10+5}{5}{\n\ }`	10 15 20 25 30
2	`\multido{\n=10+0.05}{5}{\n\ }`	10 10.05 10.10 10.15 10.20
3	`\multido{\n=10.1+1.5}{5}{\n\ }`	10.1 11.6 13.1 14.6 16.1
4	`\multido{\n=10.1+0.05}{5}{\red \n\ }`	10.1 10.06 10.11 10.16 10.21
5	`\multido{\n=10.01+0.5}{5}{\red \n\ }`	10.01 10.6 11.1 11.6 12.1
6	`\multido{\n=10.01+0.50}{5}{\n\ }`	10.01 10.51 11.01 11.51 12.01
7	`\multido{\n=10+5}{5}{\red 2+\n\ }`	2+10 2+15 2+20 2+25 2+30

Quelques remarques que l'on peut tirer de ces exemples :

- pas de problème si on emploie `\n` comme un entier (1) ;
- pas de problème non plus si la valeur de départ est entière, et l'incrément non entier (2) ;
- toujours pas de problème si la valeur de départ contient une décimale, tout comme l'incrément (3) ;
- si le nombre de départ a une décimale et l'incrément deux (4), ou le contraire (5), il n'y a pas de message d'erreur mais les résultats sont faux ;
- ça redevient juste si le nombre de départ et l'incrément ont chacun deux décimales (6) ;
- les calculs avec la variable ne sont toujours pas possibles (7).

Moralité ?
Si on veut obtenir un résultat conforme aux espérances avec une variable de type `number`, on s'arrangera pour que le nombre de départ et l'incrément aient le même nombre de chiffres après la virgule ; c'est toujours possible, il suffit de rajouter quelques 0 là où il faut.
Si le nombre de départ est entier, ça fonctionnera toujours.

On peut donc faire un peu ce que l'on veut avec le type `number`.

10.4 Variable de type `real`

Il existe un autre type de variable, le type `real` dont le nom doit commencer par `r` ou `R`. Le `\multido` donnera un résultat correct s'il y a moins de 4 chiffres de chaque côté du point décimal. Hélas, le mode d'emploi lui-même signale qu'il peut y avoir « occasional small errors » ; en effet, si on tape : `\multido{\r=0.10+0.05}{6}{\r\ }`
on obtient : 0.1 0.15001 0.20001 0.25002 0.30002 0.35002
C'est plutôt moins bien qu'avec le type `number` ; à oublier, donc.

10.5 Variable de type `dimension`

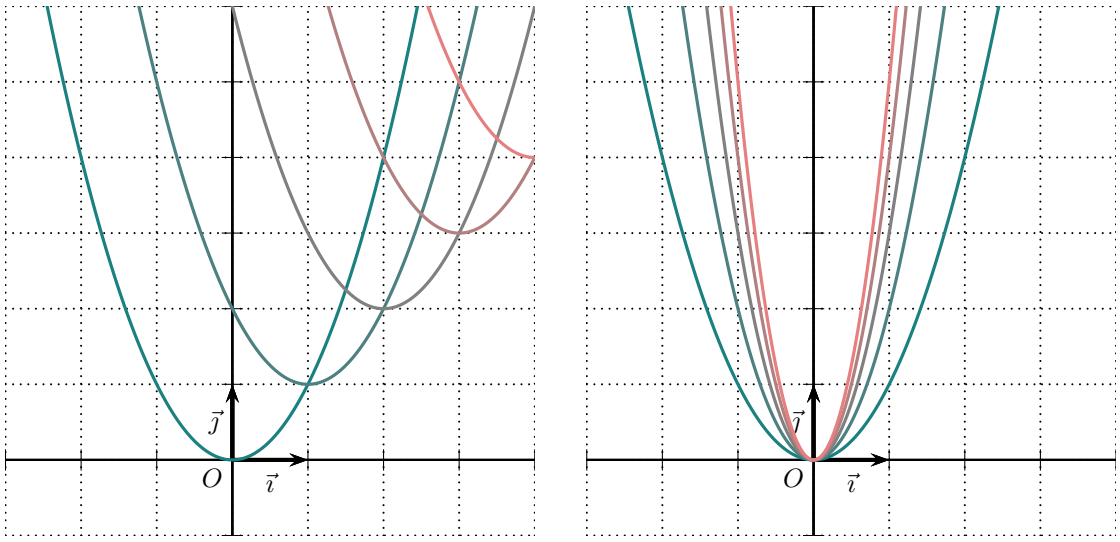
Je signale pour mémoire le type `dimension` dont je n'ai pas bien vu l'intérêt.
L'initiale du nom de la variable doit être `d` ou `D`, on s'en doutait !

On ajoute des centimètres et des points et ça donne des `sp` (`scaled points`) :
`\multido{\d=1cm+10pt}{4}{\d\ }` donne 1864679sp 2520039sp 3175399sp 3830759sp
Il faut 65536 `sp` pour faire 1 point.

Les curieux iront voir dans la documentation (dans laquelle il n'y a pas grand-chose !!!).

10.6 Applications graphiques

On peut utiliser `\multido` pour tracer des familles de fonctions.
Voyons deux variations sur la fonction carré :



Voici le code du graphique de gauche :

```
\psset{unit=1cm,algebraic}
\def\xmin{-3}  \def\xmax{4}
\def\ymin{-1}  \def\ymax{6}
\begin{pspicture*}(\xmin,\ymin)(\xmax,\ymax)
\psgrid[subgriddiv=1,griddots=10,gridlabels=0]
\psaxes[labels=none](0,0)(\xmin,\ymin)(\xmax,\ymax)
\uput[dl](0,0){$0$}
\psaxes[linewidth=1.5pt,arrowsize=2pt 2]{->}(0,0)(1,1)
\uput[d](0.5,0){$\vec{\imath}$}
\uput[l](0,0.5){$\vec{\jmath}$}
\psset{linewidth=1.2pt,plotpoints=1000}
\multido{\i=0+1,\n=0.1+0.2}
        {5}
        {
        \definecolor{couleur}{rgb}{\n,0.5,0.5}
        \psplot[linecolor=couleur]{\xmin}{\xmax}{(x-\i)^2+\i}
        }
\end{pspicture*}
```

On peut voir que l'on a utilisé la variable `\i` dans des calculs.

La variable `\n` sert à modifier la couleur des tracés ; voir les explications de `\definecolor` dans la chronique 1 de la saison 2 (page 4). L'instruction `\definecolor` est située à l'intérieur de la boucle pour que la couleur change à chaque tracé de courbe.

La fonction doit être écrite en notation infixée : il faut activer l'option `algebraic` dans `\psset` en écrivant `algebraic=true` ou `algebraic` tout court.

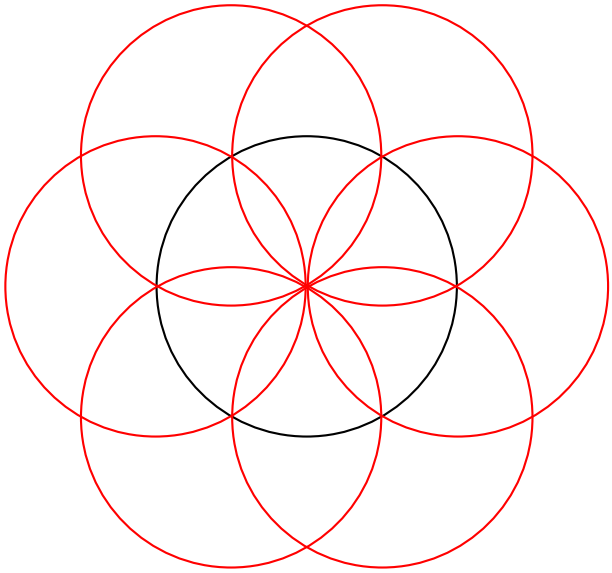
Le deuxième graphique s'obtient en remplaçant la ligne `\psplot...` par :

```
\psplot[linecolor=couleur]{\xmin}{\xmax}{\i*x^2}}
```

et en remplaçant `\i=0+1` par `\i=1+1`.

10.7 Rosace

À ce stade, on sait presque réaliser ce dessin :



Il y a un cercle en noir de rayon 2, tracé avec `\pscircle` ; pas de problème.
Il y a six cercles en rouge de même rayon que le cercle noir, dont les centres sont situés aux sommets d'un hexagone régulier inscrit dans le premier cercle.
Pour désigner les centres de ces cercles, on va utiliser le repérage polaire.
En `pstricks`, quand dans un couple (x,y) les nombres sont séparés par une virgule, il s'agit de coordonnées cartésiennes.
Quand les nombres sont séparés par un point virgule, on passe en repérage polaire (avec un angle exprimé en degrés).
Les centres des cercles sont donc repérés en polaire par les couples $(2\,;0)$, $(2\,;60)$, $(2\,;120)$,..., $(2\,;300)$; on définira une variable entière représentant l'angle, qui partira de 0 et qui ira de 60 en 60 en prenant 6 valeurs.

Voici le code :

```
\psset{unit=1cm}
\def\xmin{-4.5}  \def\xmax{4.5}
\def\ymin{-4.5}  \def\ymax{4.5}
\begin{pspicture}(\xmin,\ymin)(\xmax,\ymax)
\pscircle(0,0){2}%                cercle en noir
\psset{linecolor=red}%            on passe en rouge
\multido{\i=0+60}%                définition de la variable
        {6}%                      nombre de valeurs prises
        {\pscircle(2;\i){2}}%     cercles en rouge
\end{pspicture}
```

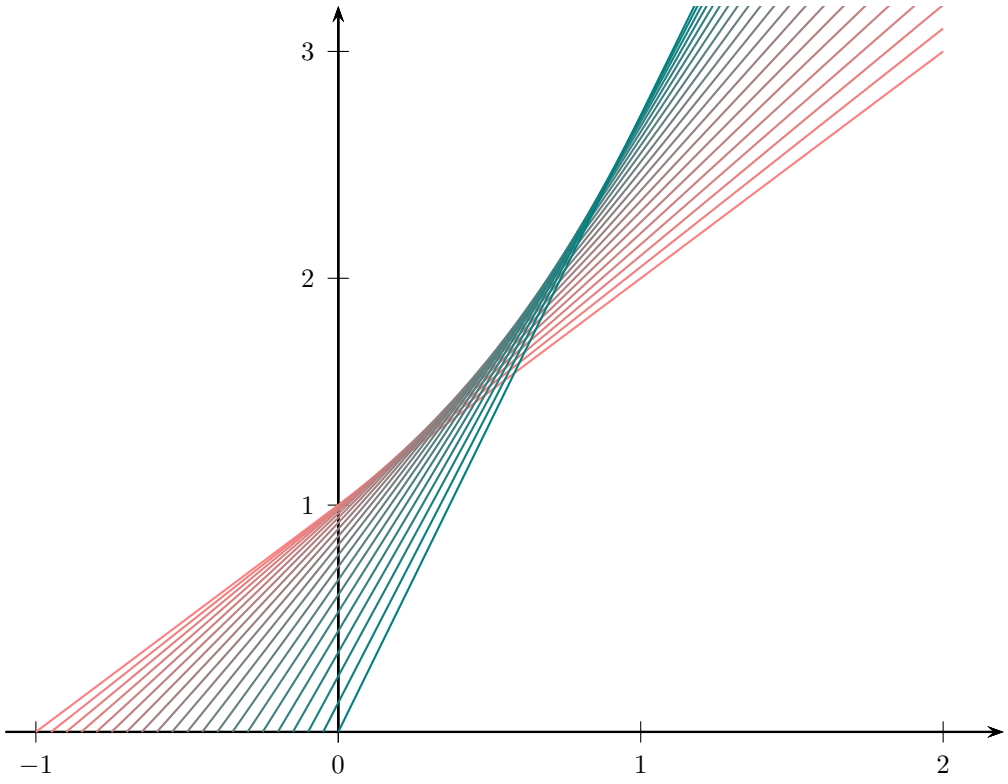
Petite remarque

Ce n'est pas que pour faire joli et pour montrer ce que l'on peut faire avec `\multido` que j'ai dessiné cette rosace ; on peut aussi l'utiliser en classe.
En sixième, on peut faire reproduire ce dessin pour travailler l'apprentissage du compas.
Plus tard dans la scolarité on peut faire décrire cette figure aux élèves, voire même leur faire écrire un texte mathématique qui permettrait à d'autres élèves de tracer cette rosace.

10.8 Bouquet final

Je ne peux pas terminer cette chronique sans vous présenter ce graphique créé par Jean-Éric Visca et qui m'a inspiré pour tracer les graphiques du paragraphe 10.6.
J'ai un peu modifié le code (notamment en simplifiant l'équation de la droite) ; on trouve l'original à l'adresse :

`http://calque.pagesperso-orange.fr/latex/latexmultido.html`



Il s'agit de tracer les tangentes à la courbe de la fonction exponentielle pour 21 points de cette courbe dont les abscisses sont comprises entre 0 et 1.
Chaque tangente est construite avec une couleur différente qui est définie dans la boucle.

Voici le code de ce beau graphique (qui nécessite l'extension `pstricks-add`) :

```
\psset{xunit=4,yunit=3,algebraic}
\begin{pspicture}(-1,0)(1,3)
\psaxes{->}(0,0)(-1,0)(1.1,3.2)
\multido
   {\n=-1+0.05}
   {21}
   {
   \definecolor{couleur}{rgb}{-\n,0.5,0.5}
   \psplot[linecolor=couleur]{\n}{1}{2.718^(\n +1)*(x-\n)}
   }
\end{pspicture}
```

Les matheux vérifieront les calculs !