Chronique 11

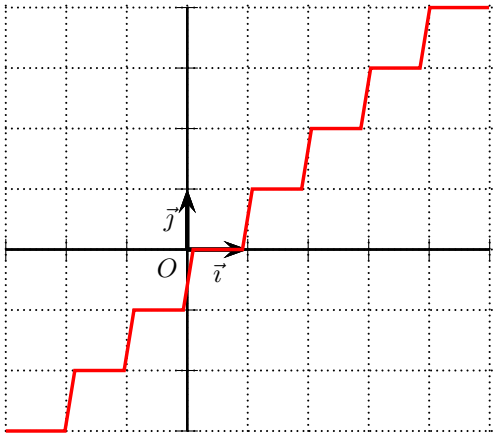
Fonction partie entière

Pourquoi consacrer une chronique à la fonction partie entière ?
Parce que cette fonction permet de voir des petites choses nouvelles et de travailler un peu l'instruction `\multido` détaillée dans la chronique précédente.

11.1 Première version

La fonction partie entière s'appelle `floor` comme dans GeoGebra.
On peut donc essayer de tracer sa représentation graphique au moyen d'un `\psplot` :

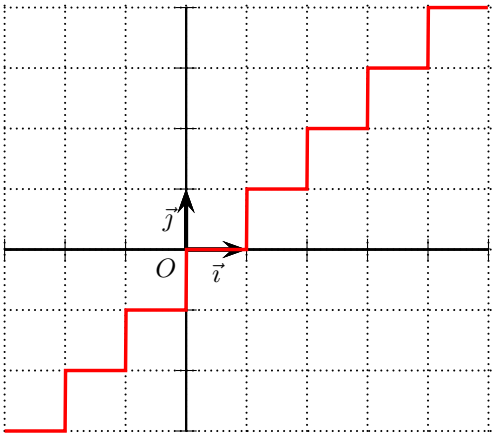
```
\psset{unit=0.8cm}
\def\xmin{-3}  \def\xmax{5}
\def\ymin{-3}  \def\ymax{4}
\begin{pspicture}(\xmin,\ymin)(\xmax,\ymax)
\psgrid[subgriddiv=1,griddots=10,gridlabels=0]
\psaxes[labels=none](0,0)(\xmin,\ymin)(\xmax,\ymax)
\uput[dl](0,0){$0$}
\psaxes[linewidth=1.5pt]{->}(0,0)(1,1)
\uput[d](0.5,0){$\vec{\imath}$}
\uput[l](0,0.5){$\vec{\jmath}$}
\psset{linecolor=red,linewidth=1.3pt}%  ligne qui sera modifiée
\psplot{\xmin}{4.99}{x floor}%          ligne qui sera modifiée
\end{pspicture}
```



Ce n'est vraiment pas fameux !

Un début de solution consiste à augmenter le nombre de points par un `plotpoints=1000` comme option dans `\psplot`.

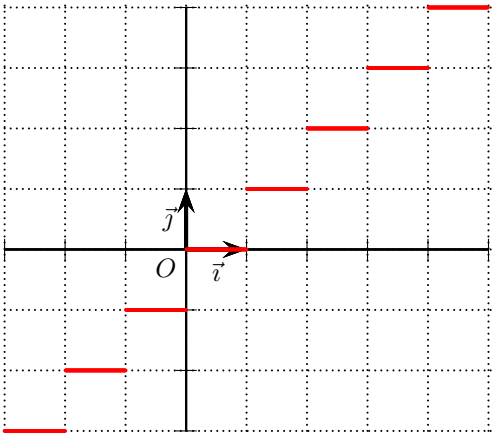
C'est mieux, mais ce n'est toujours pas acceptable : les points sont reliés entre eux, même quand il y a discontinuité, c'est-à-dire pour chaque valeur entière de la variable x.
Un peu comme sur les calculatrices.



11.2 Deuxième version

Comme on peut le faire sur les calculatrices, on va tracer la fonction point par point sans relier les points entre eux ; pour cela on va entrer comme option `plotstyle=dots` dans `\psplot`.

Il a fallu quand même réduire la taille des points ; pour cela il a suffi de modifier la variable `dotscale` dans `\psset`.



Le résultat est assez satisfaisant.
Les deux lignes à modifier dans le code du paragraphe 11.1 sont à remplacer par :

```
\psset{linecolor=red,dotscale=0.4}
\psplot[plotpoints=1000,plotstyle=dots]{\xmin}{4.99}{x floor}
```

Au passage, on remarque que l'on s'arrête à 4.99 (comme dans le graphique du paragraphe 11.1) et pas à `\xmax` pour éviter un point (ou un segment) disgracieux en dehors du quadrillage.

Et si on veut marquer les points de coordonnées $(-3,-3)$, $(-2,-2)$, etc. ?
On va utiliser `\multido` qui va servir également à tracer les segments.

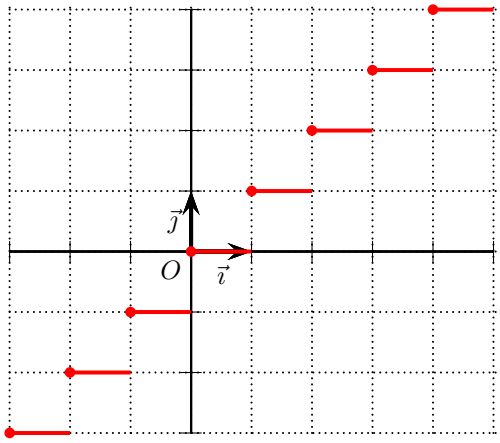
11.3 Troisième version

Que faut-il comme variables dans une boucle `\multido` pour placer les points et tracer les segments qui vont représenter la fonction partie entière ?

- Pour tracer les points, il faut une variable de type entier (appelée `\i`) qui prendra toutes les valeurs entières entre -3 et 4 soit 8 valeurs. On utilisera `\psdots` pour placer ces points.
- On tracera ensuite 8 segments de $(-3,-3)$ à $(-2,-3)$, de $(-2,-2)$ à $(-1,-2)$, etc., et de (4,4) à (5,4) ; il faut donc une deuxième variable de type entier (appelée `\I`) qui prendra 8 valeurs entières entre -2 et 5. Les segments seront tracés avec `\psline`.

Les deux lignes à modifier dans le code du paragraphe 11.1 sont à remplacer par :

```
\psset{linecolor=red,linewidth=1.5pt}
\multido{\i=\xmin+1,\I=-2+1}
        {8}
        {
        \psdots[linewidth=1pt](\i,\i)
        \psline(\i,\i)(\I,\i)
        }
```



On trace les points de coordonnées `(\i,\i)` et les segments qui vont de `(\i,\i)` à `(\I,\i)`.

Pas mal ! Mais on peut faire mieux en n'utilisant qu'une seule variable.

11.4 Quatrième version

En fait, dans le tracé de la fonction partie entière, les segments sont tous les mêmes ; par rapport au point, on avance d'une unité en abscisse et on garde la même ordonnée.

Ce serait bien si `\pstricks` autorisait les déplacements relatifs.
C'est évidemment possible ! Sinon je n'en parlerais pas...

On va utiliser l'instruction `\rlineto` qui va tracer un segment relativement au point courant, comme s'il s'agissait d'un vecteur de coordonnées (1,0) ; on place le point courant au bon endroit en utilisant l'instruction `\moveto`.

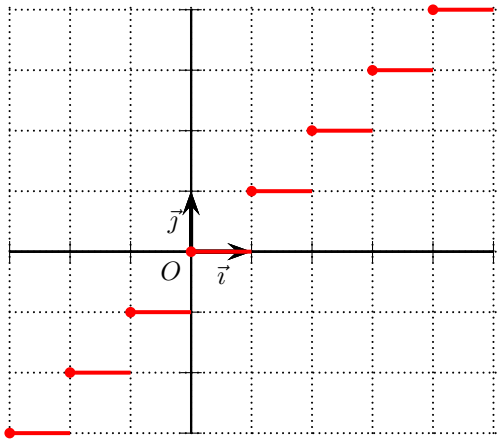
Et quand je vous aurai dit qu'il faut placer le tout dans un environnement personnalisé de type `\pscustom` (déjà vu en saison 1 lors du tracé d'aires sous une courbe), vous saurez tout !

Petit détail qui m'a agacé : l'instruction `\psdots` qui place les points doit être écrite en dehors de l'environnement `\pscustom` ; ne me demandez pas pourquoi !

Une seule variable de type `integer` suffit alors dans la boucle `\multido`.

Les deux lignes à modifier dans le code du paragraphe 11.1 sont à remplacer par :

```
\psset{linecolor=red,linewidth=1.5pt}
\multido{\i=\xmin+1}
        {8}
        {
        \psdots[linewidth=1pt](\i,\i)
        \pscustom{
                \moveto(\i,\i)
                \rlineto(1,0)
                }
        }
```



Le résultat est évidemment le même que dans la troisième version.

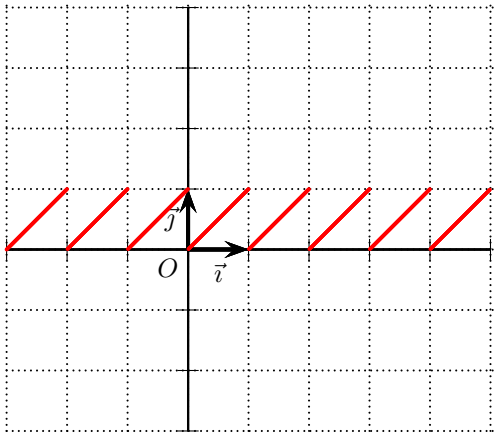
11.5 Application

Pour réemployer les méthodes que l'on vient de voir, on va tracer la représentation graphique de la fonction $x \longmapsto x - E(x)$ où E désigne la fonction partie entière ; on va le faire de deux façons : point par point puis en utilisant `\rlineto`.

Les deux lignes à modifier dans le code du paragraphe 11.1 sont à remplacer par :

```
\psset{linecolor=red,dotscale=0.4}
\psplot[plotpoints=1000,plotstyle=dots]
    {\xmin}{4.99}{x x floor sub}
```

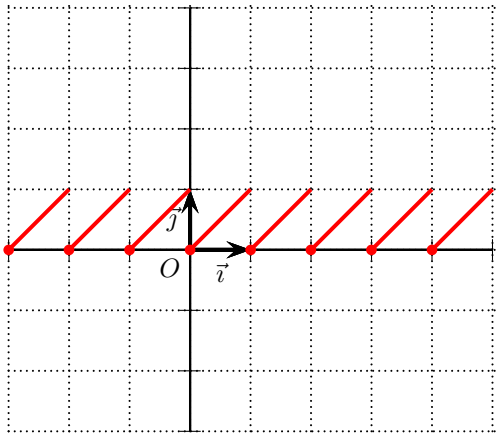
En entrant l'option `algebraic=true` dans `\psset`, on peut écrire `{x-floor(x)}` à la place de `{x x floor sub}`.



Pour aller du début d'un segment à son extrémité, on augmente l'abscisse de 1 et l'ordonnée de 1 : c'est donc facile d'utiliser l'instruction `\rlineto`.
On place des points en $(-3,0)$, $(-2,0)$, ...,$(4,0)$ et à partir de chacun de ces points on trace au moyen de `\rlineto` un vecteur (sans flèche) de coordonnées (1,1).

Les deux lignes à modifier dans le code du paragraphe 11.1 sont à remplacer par :

```
\psset{linecolor=red,linewidth=1.5pt}
\multido{\i=\xmin+1}
        {8}
        {
        \psdots[linewidth=1pt](\i,0)
        \pscustom{
                \moveto(\i,0)
                \rlineto(1,1)
                }
        }
```



On verra d'autres applications de `\multido` dans une future chronique consacrée aux tracés de polygones réguliers.