Représentation graphique de fonctions discontinues

- N°36 - septembre 2013 -

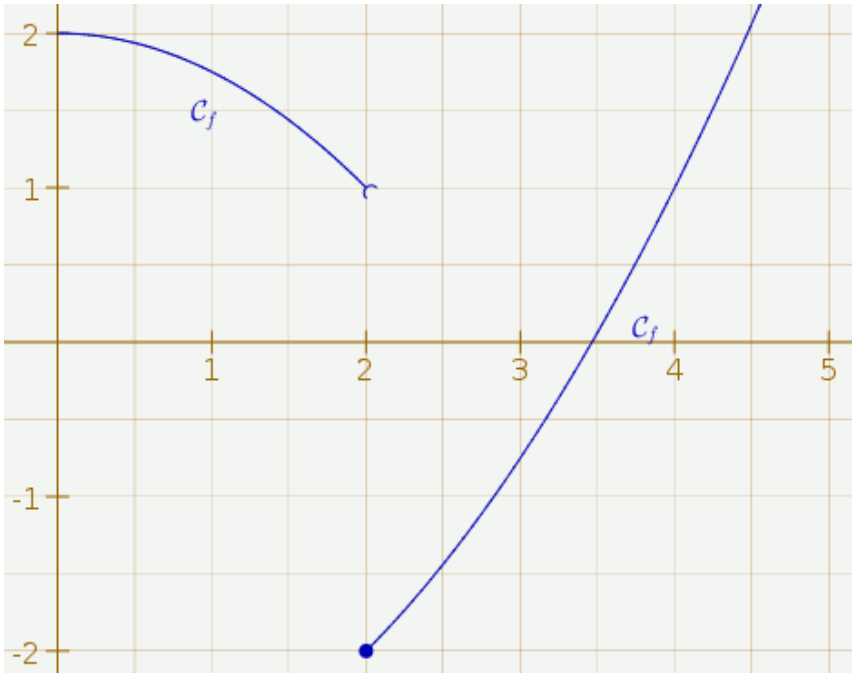
Date de mise en ligne : vendredi 14 juin 2013

Description :

On décrit comment on peut représenter graphiquement des fonctions discontinues avec CaRMetal 3.8, et sa gestion des discontinuités

Les nouvelles technologies pour l'enseignement des mathématiques

Depuis la version 3.8, CaRMetal permet de fabriquer des représentations graphiques de fonctions avec codage graphique des discontinuités :



Et l'export vectoriel (svg, eps ou pdf) fonctionne aussi avec ces nouveautés, on peut désormais insérer des représentations graphiques d'aspect professionnel dans des documents.

La genèse de cette fonctionnalité

À l'occasion de la rentrée de l'IREM, je covoiturais avec mon collègue Ludovic Fonquergne, sur le trajet Le Tampon-SinDni, et tout en cherchant des yeux d'éventuelles mégaptères tardives, nous discutions ... de maths ! Et dans le fil de la conversation, Ludovic m'a fait remarquer que les logiciels de géométrie dynamique que nous connaissons ne représentent pas les fonctions discontinues "comme dans le cours", avec des demi-cercles ou des points selon que l'intervalle est ouvert ou fermé. J'ai alors créé des macros CaRMetal permettant ce genre d'affichage, et envoyé ces macros à Pierre-Marc Mazat, qui les a tant appréciées que le jour même, il les a implémentées dans CaRMetal.

Il y a là un bel exemple de partage des tâches, entre utilisateurs et développeurs. Ce qui a été rendu possible par la communication entre tous les acteurs de cette histoire, il est parfois bon d'insister là-dessus...

Exemple

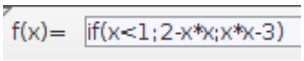
On considère la fonction f telle que

- f(x)=2-x2 si x<1 ;
- f(x)=x2-3 sinon.

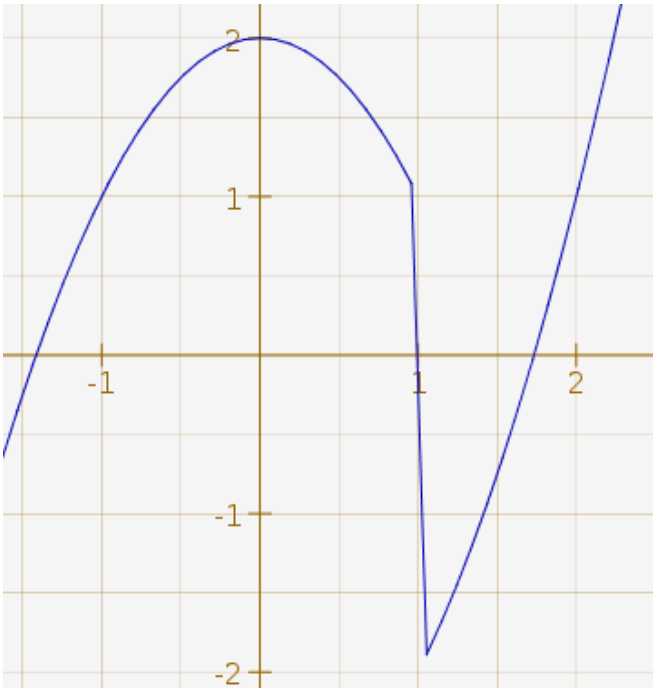
Le problème est de la représenter graphiquement...

L'intérêt de ce genre de fonction, en Seconde, est algorithmique, ou ici, algébrique : La notion de test y est ici mise en oeuvre, et en plus c'est un test simple : Comparaison entre x et 1. Avec CaRMetal cela s'écrit

```
if(x<1;2-x*x;x*x-3)
```



La représentation graphique de cette fonction pose problème :

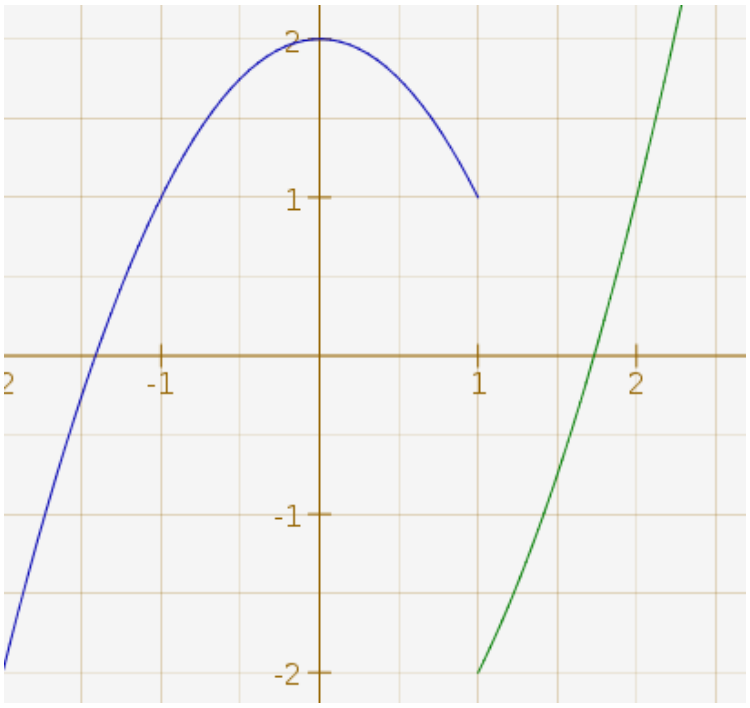


Pourquoi ce trait presque vertical ?

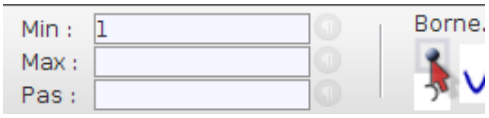
Dans CaRMetal (comme dans tout logiciel de géométrie dynamique), une fonction est représentée graphiquement par une approximation polygonale. Ici, des points voisins de (1 ;1) et de (1 ;-2) sont des sommets du polygone, et sont donc joints par un segment.

Ceci dit, on verra dans d'autres exemples que CaRMetal ne représente pas toujours graphiquement les segments trop verticaux ; la gestion des asymptotes des hyperboles par les logiciels de géométrie dynamique relève de la même logique.

Une solution à ce problème est de représenter graphiquement non une fonction, mais deux : Deux morceaux de parabole. Ici *f1* (en bleu) est la fonction *2-x*x* avec Xmax=1, et *f2* (en vert) est la fonction *x*x-3* avec Xmin=1 :



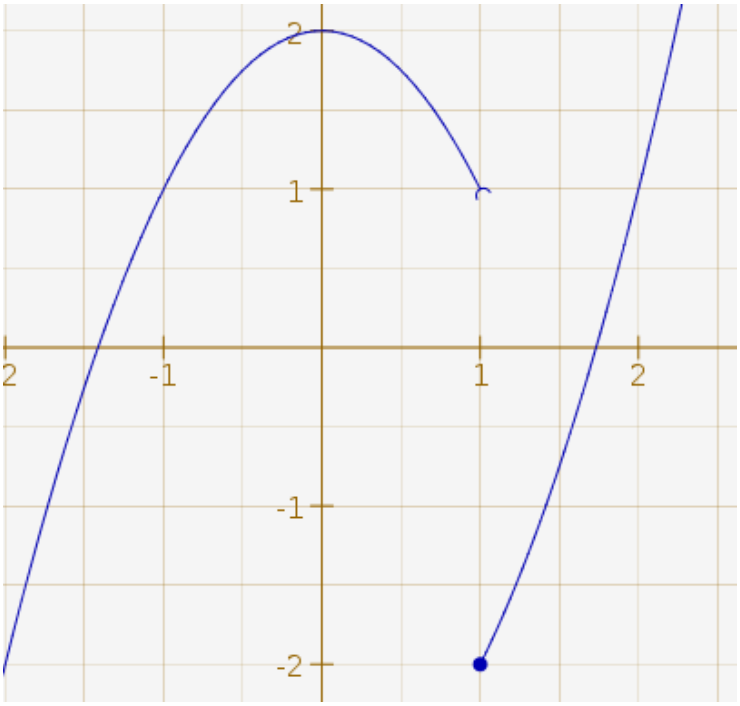
Il subsiste encore un problème : On ne voit pas sur la figure l'image de 1 lui-même. Or **depuis la version 3.7.8, CaRMetal permet de coder le caractère ouvert ou fermé de la courbe**. Pour fermer *f2* à gauche, on clique sur l'icône représentant un point :



L'aspect du bouton change légèrement pour signaler que maintenant, la fonction est "fermée" à gauche :

De même, l'autre fonction doit être ouverte en son maximum 1, ce qui se fait en cliquant sur l'outil d'ouverture du côté droit (puisque c'est le maximum de *x*) :

On a alors la représentation graphique souhaitée :

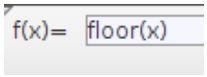


Par défaut, une fonction n'est ni ouverte, ni fermée ; et en cliquant à nouveau sur un bouton "enfoncé", on peut remettre la fonction dans cet état initial. De même, en fermant une fonction qui était ouverte, le bouton d'ouverture est désactivé.

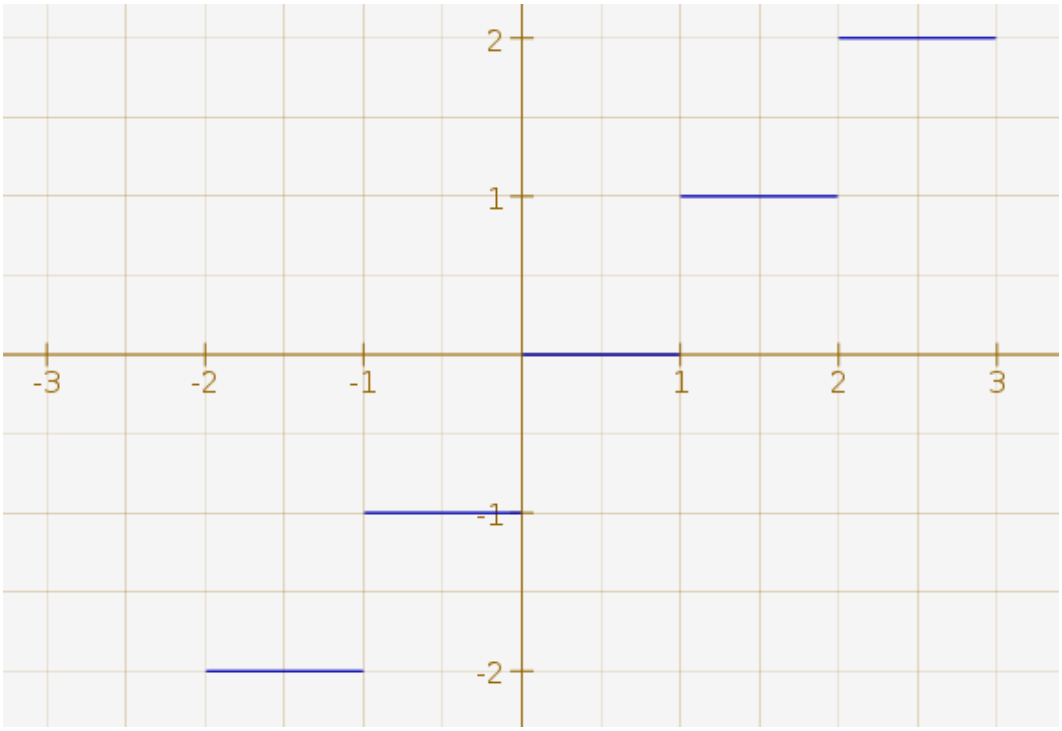
Les composantes connexes du graphe doivent être (re)définies une par une, et le codage doit être fait (ou non, on n'est pas obligé...) pour chacune d'entre elles. Mais le résultat, exporté à un format graphique vectoriel puis inséré dans un document (traitement de texte ou LaTeX), sort sur une imprimante laser avec une haute qualité, et sans trop d'effort il est plus facile de cliquer sur des icônes que de programmer en LaTeX...).

++++Partie entière

La fonction qui, à *x* réel, associe sa partie entière [1], se note *floor* dans CaRMetal :



Sa représentation graphique montre pourquoi on appelle ce genre de fonction, une *fonction en escalier* :

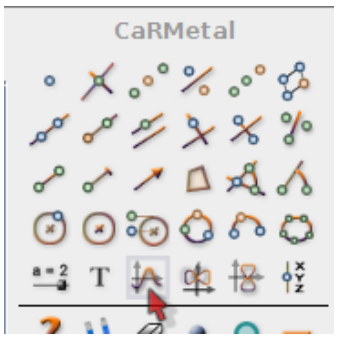


Pour coder sur le graphique le fait que la partie entière de 3 est 3 et non 2, il faut représenter plusieurs fonctions sur des intervalles de la forme *[n ;n+1]*, et pour cela, on utilise une boucle en JavaScript. Le script commence ainsi :

```
for(n=-8;n<=8;n++){

}
```

Pour se rappeler la syntaxe d'une représentation graphique en CaRScript, on clique sur l'icône correspondante :



Ce qui rappelle que l'on doit entrer les bornes de l'intervalle (ici *n* et *n+1*) et la valeur de la fonction (*n* à nouveau) :

```
for(n=-8;n<=8;n++){
  f=CartesianFunction(n,n+1,n);
}
```

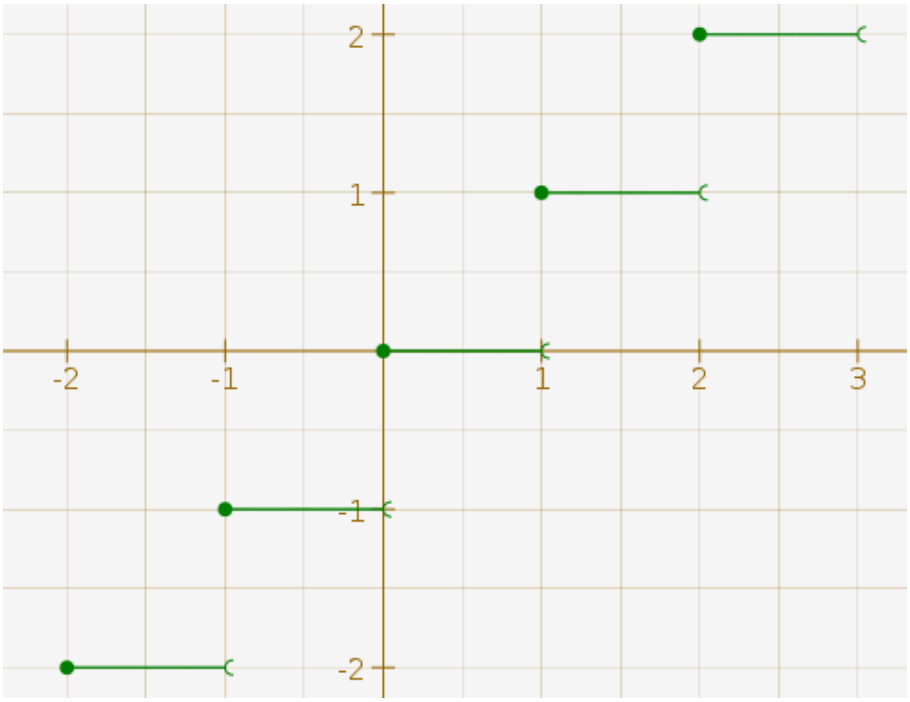
Ce script produit la même figure que *floor(x)* ci-dessus, mais on peut coder aussi en JavaScript les ouvertures et fermetures de segments sur le graphique :

- *SetMinOpen* pour un demi-cercle à gauche ;
- *SetMinClosed* pour un point à gauche ;
- *SetMaxOpen* pour un demi-cercle à droite ;
- *SetMaxClosed* pour un point à droite.

Dans chaque cas, le booléen correspondant doit être positionné à *true* :

```
for(n=-8;n<=8;n++){
  f=CartesianFunction(n,n+1,n);
  SetMinClosed(f,true);
  SetMaxOpen(f,true);
}
```

Ce script construit la représentation graphique "correcte" de la fonction *partie entière* :



++++partie fractionnaire

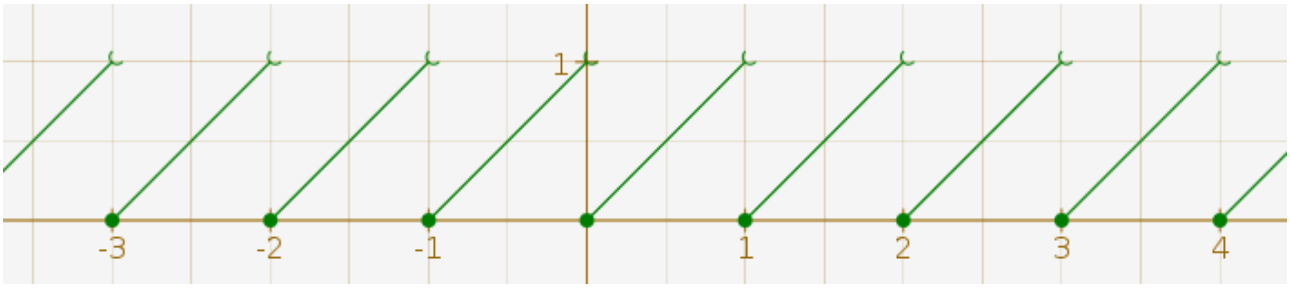
En soustrayant à un nombre *x* sa partie entière *floor(x)*, on obtient sa partie fractionnaire [2]. Cette fonction présente l'intérêt d'être périodique, et en plus, d'avoir une période plus simple que les fonctions trigonométriques : 1...

Pour la représenter avec le codage des discontinuités, on peut faire comme dans l'onglet précédent, avec ce

CaRScript :

```
for(n=-8;n<=8;n++){
   f=CartesianFunction(n,n+1,"x-floor(x)");
   SetMinClosed(f,true);
   SetMaxOpen(f,true);
}
```

On obtient alors cette frise (ce qui montre bien qu'elle est 1-périodique) :



Si elle est 1-périodique, elle a une série de Fourier. Or il se trouve que celle-ci est relativement simple : Outre le terme constant qui est 1/2, les autres termes sont de la forme sin(2Ï€nx)/(Ï€n) ; ce qui permet de dire, en notant E(x) la partie entière de x :

$$\forall x \in \mathbb{R}, x = E(x) + \frac{1}{2} - \sum_{n=1}^{\infty} \frac{sin(2\pi x)}{n\pi}$$

La vidéo des sommes partielles de Fourier, de l'ordre 1 à l'ordre 16, montre que la vitesse de convergence est plus rapide au centre des segments qu'à leurs bords :

Le fichier au format CaRMetal, avec son curseur, est téléchargeable en bas d'article. Le fait que les approximations par sommes de Fourier se laissent emporter par leur élan à chaque saut, et dépassent la valeur à atteindre, est le

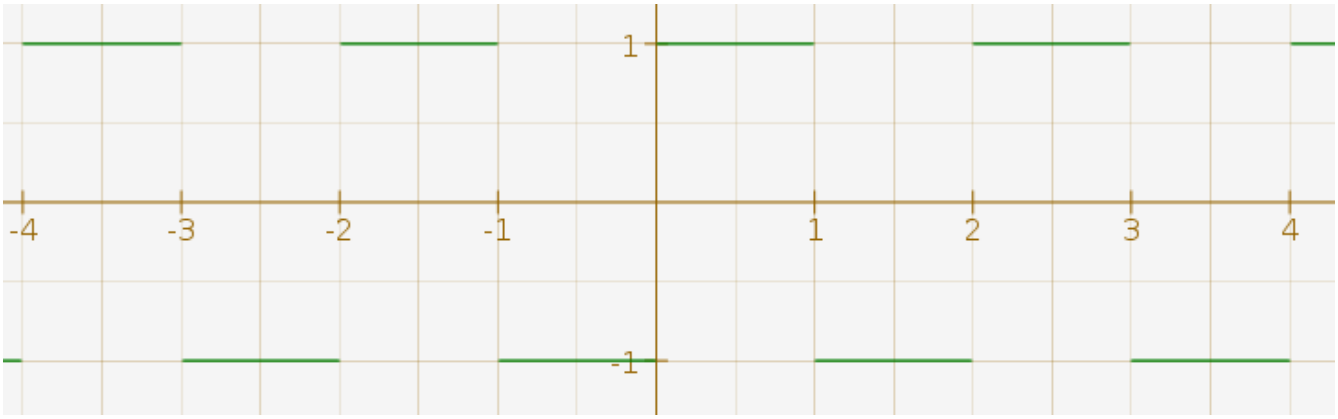
phénomène de Gibbs, qu'on va illustrer sur un autre exemple dans les onglets suivants.

++++vers Gibbs

La fonction que voici a une série de Fourier plus simple (que des sinus) parce qu'elle est impaire :

f(x)= sign(rsin(pi*x))

La fonction *sign* de CaRMetal associe à un nombre réel, le nombre 1 s'il est positif et le nombre -1 s'il est négatif ; la fonction *sign(rsin(pi*x))* est donc périodique de période 2 (*rsin* désigne le sinus en radians sous CaRMetal). Mais elle se dessine sous forme de segments :

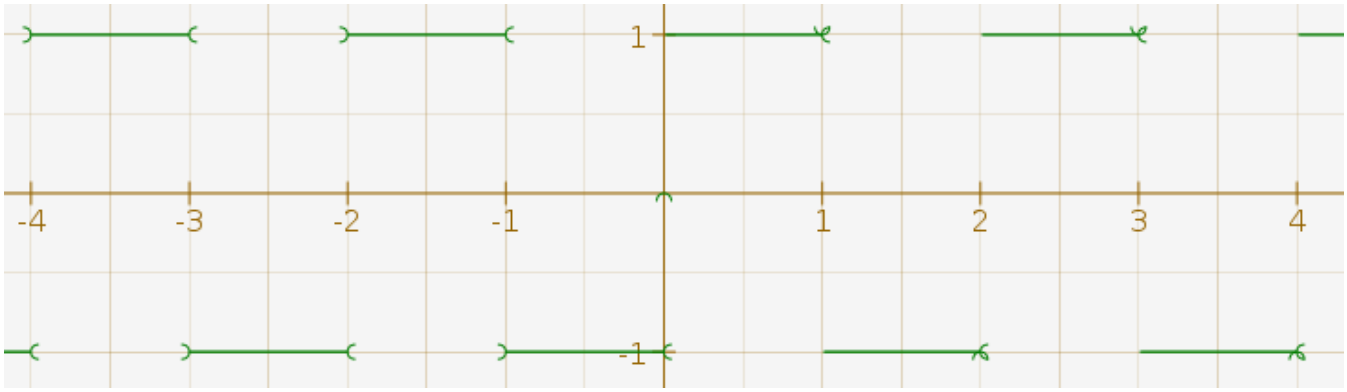


D'après un théorème de Dirichlet (séries de Fourier), la série de Fourier d'une fonction réglée tend vers la moyenne entre les limites à gauche et à droite de f(x). Si f est continue en x, cette moyenne vaut f(x) et la série converge. Donc si f est discontinue en 0, on a intérêt à poser que f(0)=(fg(0)+fd(0))/2 (fg(0) et fd(0) désignant les limites à gauche et à droite de f en 0).

La fonction *f* ci-dessus est discontinue en les nombres entiers. La moyenne de 1 et -1 valant 0, on pose f(n)=0 si n est entier. Pour le coder sur la représentation graphique, il faut donc rajouter des points de coordonnées (n ;0) pour tout *n*, et ouvrir tous les segments. Pour ce qui est de l'ouverture des segments, l'algorithme ci-dessous devrait convenir :

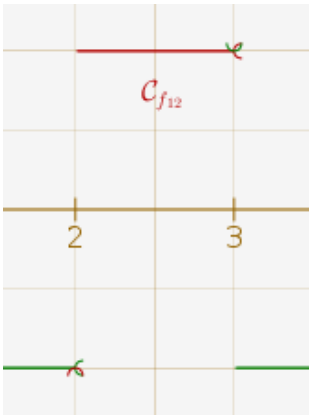
```
1 for(n=-8;n<=8;n++){
2
  f=CartesianFunction(n,n+1,"sign(rsin(pi*x))");
3    SetMinOpen(f,true);
4    SetMaxOpen(f,true);
5 }
```

Le rendu est effectivement correct dans les nombres négatifs, mais pas dans les nombres positifs :



(aux coordonnées (2 ;-1) par exemple, on voit qu'il y a un problème).

Pour voir ce qui ne va pas, on peut changer de couleur la représentation graphique d'un des morceaux (ici, la fonction *f12* qui vaut 1 pour *x* entre 2 et 3) :

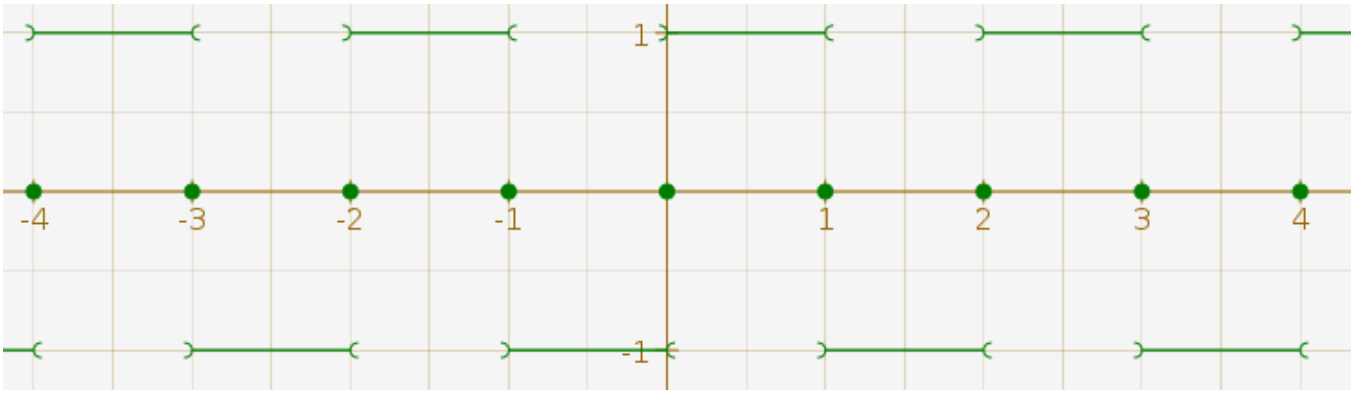


Sur ce coup-là, CaRMetal a considéré que le signe de 0 était -1 et non 0 (une petite erreur d'approximation, amplifiée par la discontinuité). Ce qui suggère la remédiation suivante : Remplacer 2 par 2,001. Avec les points, créés et mis en gras à la volée, le script complet devient

```
for(n=-8;n<=8;n++){

f=CartesianFunction(n+0.001,n+1,"sign(rsin(pi*x))");
   SetMinOpen(f,true);
   SetMaxOpen(f,true);
   SetThickness(Point(n,0),"thick");
}
```

Il produit l'effet suivant :

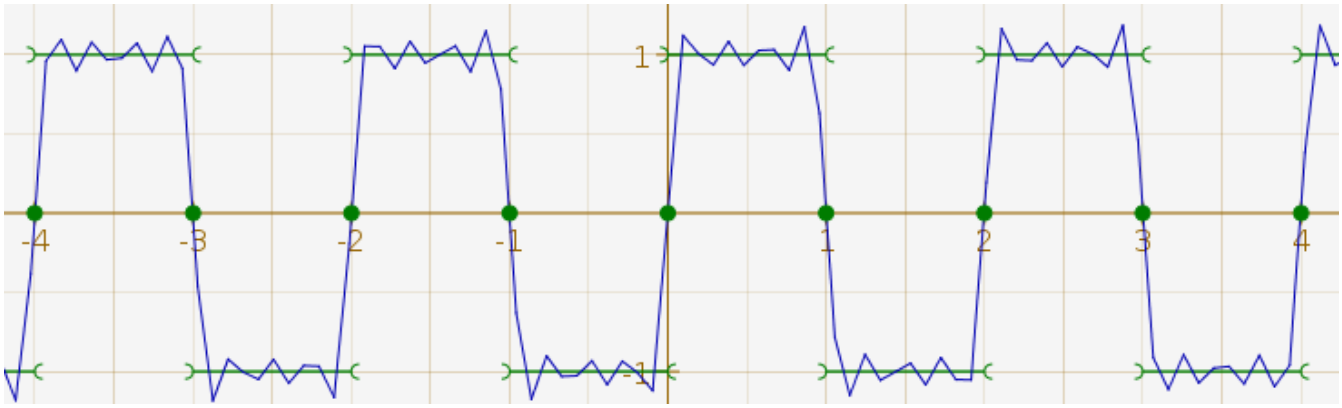


++++Gibbs

Pour illustrer le phénomène de Gibbs, on ajoute à cette représentation graphique, une somme partielle de la série de Fourier de f. Par exemple à l'ordre 7 :

f(x)= 4/pi*(rsin(pi*x)+rsin(3*pi*x)/3+rsin(5*pi*x)/5+rsin(7*pi*x)/7)

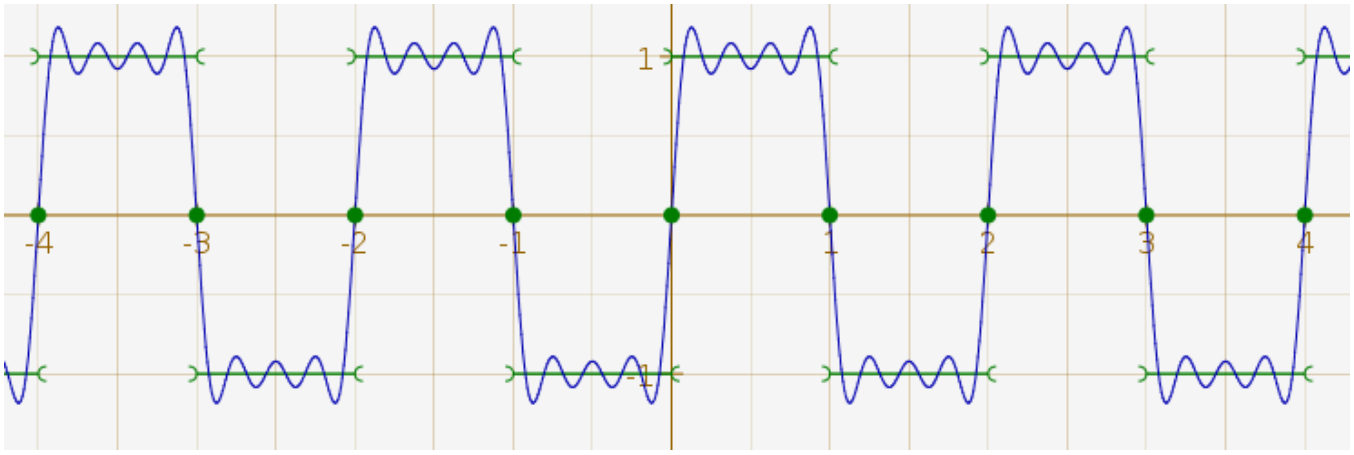
La voici, en bleu, superposée à la représentation graphique de l'onglet précédent :



Pas terrible : L'algorithme de tracé de CaRMetal est rapide mais au prix d'une approximation polygonale un peu grossière. Pour y remédier, il suffit d'imposer un pas assez petit, par exemple 0,001 (trouvé empiriquement) :

Min : -4
Max : 4
Pas : 0.001

L'affichage est nettement meilleur :

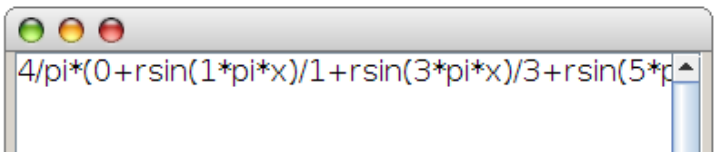


Pour avoir une somme de Fourier à un ordre plus grand, le mieux est de construire son expression algébrique avec un CaRScript ; l'algorithme peut se décrire ainsi :

- initialiser une chaîne de caractères avec le début de l'expression : `4/pi*(0`
- faire une boucle sur l'indice *n*, allant de 1 à 7 par pas de 2 (par exemple), dans laquelle, pour chaque valeur de n, on ajoute un terme de la somme de Fourier : `+rsin(_n*pi*x)`
- après la boucle, refermer la parenthèse en ajoutant à la chaîne de caractères, ladite parenthèse : `)`

L'algorithme, en JavaScript et avec affichage de la chaîne pour vérification, s'écrit ainsi :

```
1 alg="4/pi*(0";
2 for(n=1;n<=7;n+=2){
3   alg+="+rsin("+n+"*pi*x)/"+n;
4 }
5 alg+=")"
6 Println(alg);
7
8
```



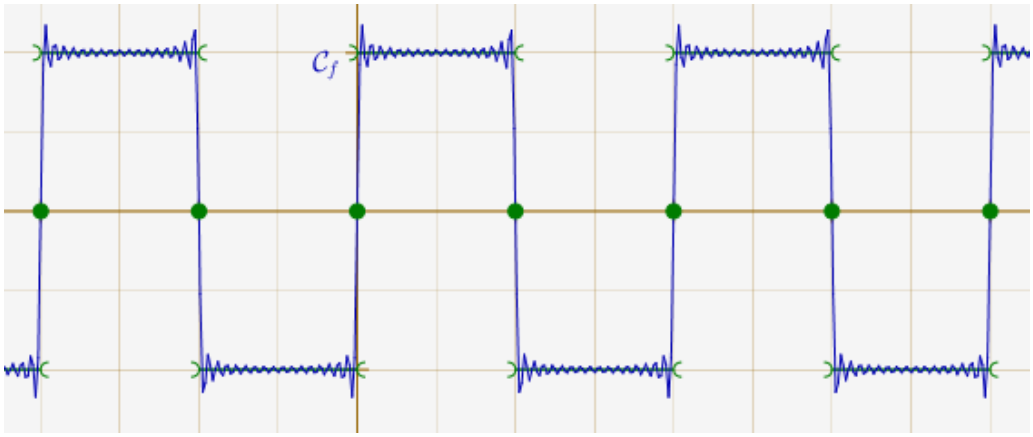
Puisque la chaîne de caractères obtenue a l'air correcte, on peut tranquillement remplacer le 7 par un 33 et créer une fonction ayant cette expression algébrique pour valeur :

```
1 alg="4/pi*(0";
2 for(n=1;n<=33;n+=2){
3   alg+="+rsin("+n+"*pi*x)/"+n;
4 }
5 alg+=")"
6 f=CartesianFunction(-8,8,alg);
7 SetColor(f,"blue");
```

pour avoir la somme d'ordre 33 :

f(x)= n(27*pi*x)/27+rsin(29*pi*x)/29+rsin(31*pi*x)/31+rsin(33*pi*x)/33)

sur laquelle le phénomène de Gibbs se constate mieux :



++++Gibbs cursorisé

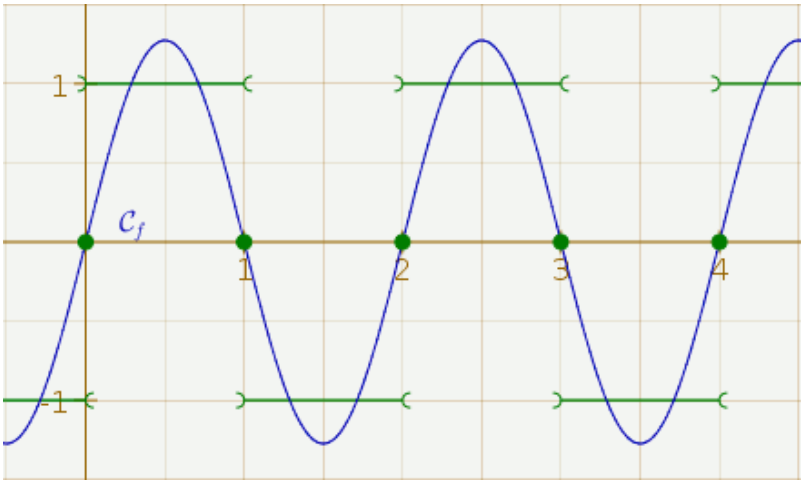
En production de documents, CaRMetal devient donc irremplaçable sur ce sujet. Mais c'est quand même avant tout un logiciel de géométrie dynamique, et dans l'exemple présent, il serait intéressant de pouvoir changer l'ordre de la série de Fourier par un curseur.

L'astuce [3] consiste à multiplier chaque terme par un booléen (en fait converti en entier), égal à 0 ou 1 selon que le terme doive être affiché ou non. Or c'est précisément le cas de *N>5* qui est égal à 1 si le curseur *N* est suffisamment grand, et à 0 sinon. Donc, comme les termes à afficher sont de toute manière impairs, on multiplie le terme d'odre *n* par *2*N+1>n* pour que sa visibilité soit dépendante du curseur comme souhaité :

```
alg="4/pi*(0";
for(n=1;n<=33;n+=2){
   alg+="+(2*N+1>"+n+")*rsin("+n+"*pi*x)/"+n;
}
alg+=")"
f=CartesianFunction(-8,8,alg);
SetColor(f,"blue");
```

L'allure de la courbe va alors de la simple sinusoïde à la figure précédente selon la valeur du curseur *N*. Le résultat est téléchargeable ci-dessous, mais si on l'ouvre avec une ancienne version de CaRMetal, l'affichage des fonctions en vert sera moins bien codé.

En actionnant le curseur, on voit l'évolution des oscillations, comme le montre cette vidéo :



Post-scriptum :Il faudrait vraiment que les concepteurs du sujet de BTS groupement A 2013 utilisent CaRMetal pour faire leurs figures, parce que là, avec leurs segments verticaux, les figures de l'énoncé sont contradictoires avec le cours de Seconde...

---

[1] définie comme le plus grand entier inférieur ou égal à *x*

[2] on peut aussi l'appeler *x modulo 1* et dans certains langages de programmation, la calculer ainsi

[3] dûe semble-t-il à Éric Hakenholz, l'idée était en gestation, sans les multiplications, dans le journal d'Alice (nostagie...), se retrouve plus algébriquement exprimée ici, toujours sans les multiplications, et est citée en 2009 ici, cette fois-ci avec multiplications