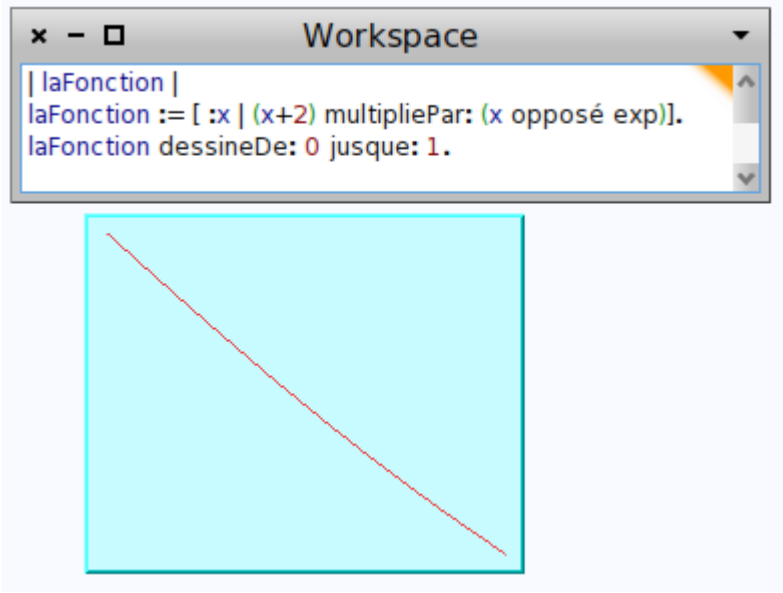
Traitement MathsOntologique du sujet Polynésie (bac S 2013)

MathsOntologie est à la fois un langage de programmation, et le logiciel permettant de programmer dans ce langage. Le langage se veut proche du Français, et est une traduction française de Smalltalk[1]. Le logiciel est téléchargeable ici :
https://dl.dropbox.com/u/10996692/MathsOntologie.zip
et le manuel d'utilisation du logiciel est ici :
http://www.reunion.iufm.fr/recherche/irem/IMG/pdf/mathsontologie.pdf
Plusieurs exemples d'utilisation sont décrits sur le site de l'IREM de La Réunion, notamment les sujets d'algorithmique du bac 2013 : http://www.reunion.iufm.fr/recherche/irem/spip.php?article644

Une approche un peu plus élémentaire du présent sujet y est d'ailleurs proposée. Cet article va donc se focaliser sur la puissance de l'approche fonctionnelle de MathsOntologie.

Dans MathsOntologie, les fonctions sont rédigées entre crochets, avec la liste des antécédents précédés d'un double-point, et la valeur retournée par la fonction. Par exemple la fonction de l'énoncé $x \to (x+2)e^{-x}$ va s'écrire [:x | (x+2)*(-x exp)]. Ce qui est puissant, c'est que cette fonction[2] peut être affectée à une variable laFonction[3] pour être réutilisée dans le programme.

MathsOntologie permet de décrire l'expression algébrique de f(x) par des mots :



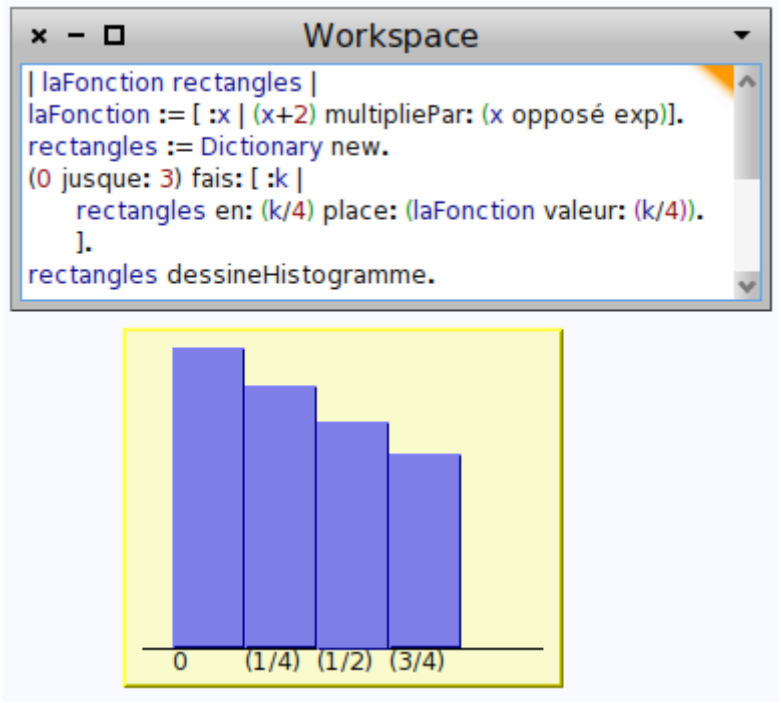
1 Le premier langage objet, conçu par Alan Kay dans les années 1970, pour gérer les souris, fenêtres etc. et pour mettre la programmation à la portée des enfants. C'est en Smalltalk qu'est programmé Scratch.
2 Sous cette forme, la fonction est dite anonyme, ou « lambda-fonction ». Le terme de « fonction jetable » serait plus approprié...
3 Pour calculer l'image de 2 par laFonction, on écrit laFonction valeur : 2

MathsOntologie permet d'esquisser la représentation graphique de la fonction pour vérifier qu'il n'y a pas d'erreur d'algèbre. On constate au passage

1. que le programme s'écrit dans une fenêtre spéciale appelée Workspace
2. que les variables (ici il n'y en a qu'une) doivent être déclarées au début, entre traits verticaux.

Pour exécuter le programme une fois qu'on l'a rédigé, on doit sélectionner le script avec la souris[4], puis faire un clic droit[5] et choisir « Do It ! »

Remarque : On peut aussi dessiner les fameux rectangles dont la somme des aires approche l'intégrale de la fonction sur [0;1]. Pour cela, on crée un pseudo tableau d'effectifs en plaçant des valeurs de x dans la liste des entrées[6] d'un « dictionnaire » et leurs images dans la liste des valeurs du dictionnaire. Puis, bien que ces valeurs ne soient ni des effectifs ni des fréquences, MathsOntologie peut dessiner l'histogramme :



L'algorithme du bac sera lui aussi une fonction de MathsOntologie ; mais elle aura deux antécédents :

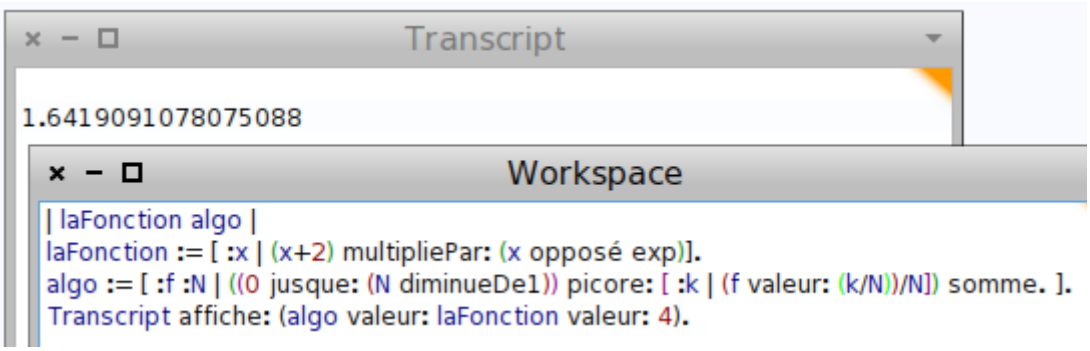
- une fonction (il s'agit donc d'une fonctionnelle, une fonction qui porte sur des fonctions et non sur des nombres)
- un entier N : le nombre de rectangles

Cet algorithme se traduit en MathsOntologie par les étapes suivantes :

1. on construit une liste d'entiers, allant de 0 à N-1 ;
2. on la fait picorer par la fonction f ; on obtient alors une liste de valeurs de f (ou plus précisément, de f/N)
3. on se procure la somme des éléments de cette nouvelle liste

4 Ou par le raccourci clavier Alt+A, qui sélectionne tout
5 Ou le raccourci clavier Alt+D, qui exécute le script sélectionné
6 Ou clés, en Smalltalk « keys »

Pour avoir la somme des aires des 4 rectangles, on appelle alors l'algorithme avec pour valeurs laFonction et 4 :

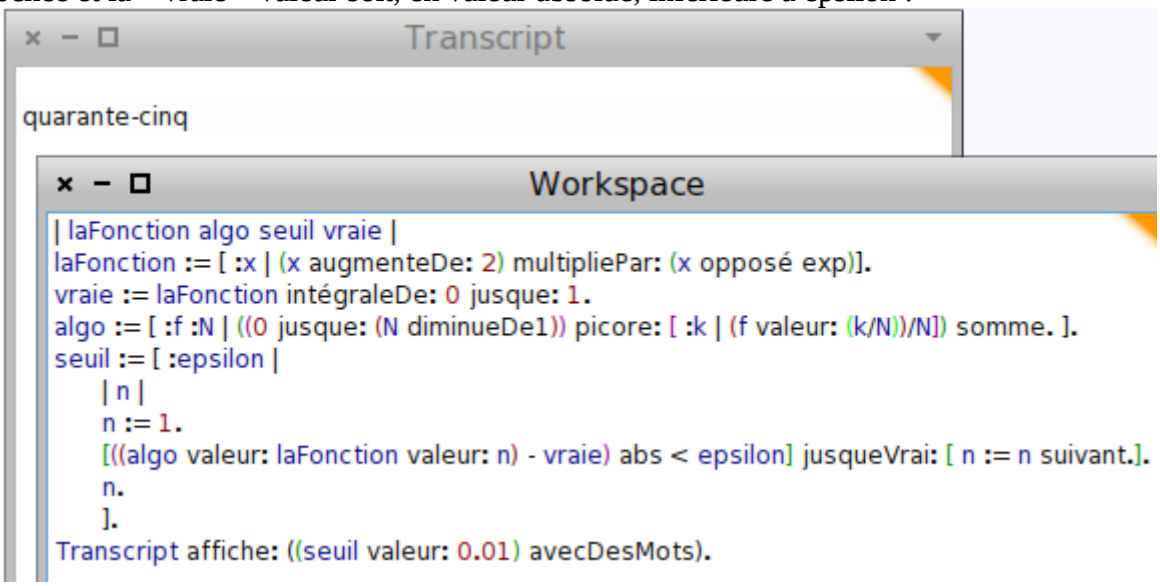


On apprend au passage que pour afficher un nombre, on demande au transcript (une fenêtre autre que le workspace) de l'afficher.

Maintenant que le sujet du bac est traité, on peut profiter de ce que l'algorithme est stocké dans une fonction, pour déterminer à partir de combien de rectangles une précision donnée est atteinte pour le calcul de l'intégrale. Bien qu'on puisse faire cela dans une simple boucle (incrémenter n jusqu'à ce que la précision voulue soit atteinte), on gagnera plus tard, à le faire encore une fois dans une fonction appelée seuil, et ayant pour variable epsilon, la précision voulue. Pour calculer une valeur précise de l'intégrale, on peut demander à laFonction quelle est son intégrale entre 0 et 1, par

laFonction intégraleDe : 0 jusque : 1

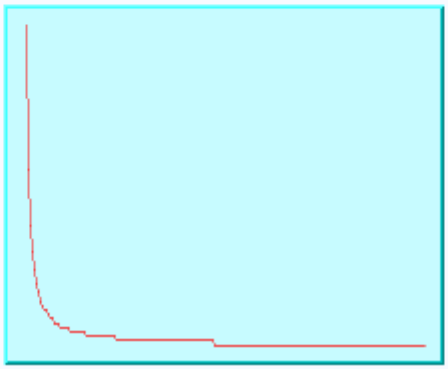
La fonction seuil possède, outre sa variable d'entrée (epsilon), une variable locale appelée n. Celle-ci est initialisée à 1, puis incrémentée (par n:= n suivant) jusqu'à ce que la différence entre la valeur approchée et la « vraie » valeur soit, en valeur absolue, inférieure à epsilon :



Ensuite il suffit de calculer l'image de 0,01 par cette fonction pour apprendre qu'il faut 45 rectangles minimum pour que la somme de leurs aires approche l'intégrale à 0,01 près. Mais également, puisqu'on a une fonction définie sur $]\ 0;+\infty\ [$, on peut la représenter graphiquement :

Workspace

```
| laFonction algo seuil vraie |
laFonction := [ :x | (x+2) multipliePar: (x opposé exp)].
vraie := laFonction intégraleDe: 0 jusque: 1.
algo := [ :f :N | ((0 jusque: (N diminueDe1)) picore: [ :k | (f valeur: (k/N))/N]) somme. ].
seuil := [ :epsilon |
    | n |
    n := 1.
    [((algo valeur: laFonction valeur: n) - vraie) abs < epsilon] jusqueVrai: [n := n
suivant.].
    n.
    ].
seuil dessineDe: 0.001 jusque: 1.
```

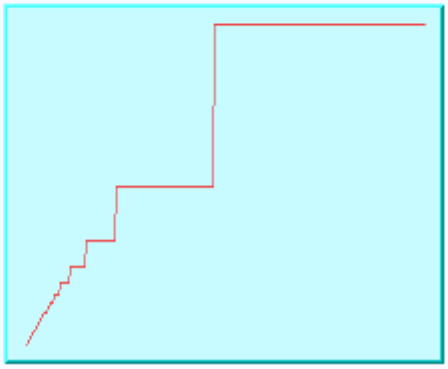


On s'en doutait un peu, la fonction est en escalier, et tend vers l'infini en 0. Mais est-ce que cet escalier ressemble à une hyperbole[7] ? Pour le savoir, il suffit de retourner, au lieu de n, son inverse :

7 Autrement dit, est-ce que le nombre de rectangles est, ou non, inversement proportionnel à la précision souhaitée ?

Workspace

```
| laFonction algo seuil vraie |
laFonction := [ :x | (x+2) multipliePar: (x opposé exp)].
vraie := laFonction intégraleDe: 0 jusque: 1.
algo := [ :f :N | ((0 jusque: (N diminueDe1)) picore: [ :k | (f valeur: (k/N))/N]) somme. ].
seuil := [ :epsilon |
    | n |
    n := 1.
    [((algo valeur: laFonction valeur: n) - vraie) abs < epsilon] jusqueVrai: [n := n
suivant.].
    n inverse.
    ].
seuil dessineDe: 0.001 jusque: 1.
```



Les points en bas à gauche n'ont pas vraiment l'air alignés. Mais une étude plus poussée (par exemple avec une régression) permettrait d'y répondre.