Les paradoxes de Zenon - Applications à Python

Denis Charrier B2 IA

September 2023

1 Achille et la tortue (zenon_achille.py)

Les paradoxes de Zenon sont des expériences de pensée dans lesquelles un raisonnement logique est appliqué à une situation réelle, aboutissant à une absurdité. Imaginons qu'une tortue soit à une distance d devant le héros rapide Achille, et que ce dernier veuille la rattraper. Selon Zenon c'est impossible car à chaque avancée d'Achille, la tortue progresse également et il ne pourra donc jamais la rattraper.
Le programme zenon_achille.py contient deux fonctions qui simulent cette course. La fonction anim_race() est une animation d'un carré rouge (Achille) et d'un carré vert (la tortue avec une distance d'avance) qui avancent dans la même direction. La fonction digit_race() résume la course avec un tableau dans lequel sont affichées les positions successives, puis affiche un graphique avec deux courbes affines qui représentent les distances parcourues.

2 Paradoxe de la dichotomie (zenon_dichotomie.py)

Zenon qui se trouve en A veut atteindre avec une pierre un point situé en B. Or pour aller en A il faut déjà parcourir la moitié de A. Arrivé à la moitié de A il faudra parcourir la moitié du chemin restant, et ainsi de suite à l'infini. Contre toute attente il serait impossible de faire parcourir une quelconque distance à un projectile. Ce paradoxe met en avant la notion abstraite d'infini dans un contexte réel. Pour illustrer ce paradoxe nous allons formaliser les données du problèmes:

On normalise la distance entre A et B par 1. En termes mathématiques, la moitié de la moitié de la... donne la série $\frac{1}{2}, \frac{1}{4}, \frac{1}{8}$...qu'on peut réécrire:

$$\sum_{n=1}^{\infty} \frac{1}{2^n} \tag{1}$$

On veut donc illustrer cette série dans un programme. Dans la vie courante, la trajectoire d'un projectile dessine une parabole dont les caractéristiques dépendent de l'accélération de la pesanteur, de la force du lancer, des contraintes météorologiques,etc. Pour cet exercice nous allons nous en tenir à des conditions simples et construire une suite de paraboles qui illustrera le problème graphiquement. On cherche donc une suite de paraboles dont la distance entre les points où elles s'annulent décroît de moitié pour chaque parabole de la suite, c'est à dire que la première s'annulera en 0 et en $\frac{1}{2}$, la seconde en $\frac{1}{2}$ et en $\frac{1}{2}+\frac{1}{4}=\frac{3}{4}$, la troisième en $\frac{3}{4}$ et en $\frac{3}{4}+\frac{1}{8}$,etc. Sans rentrer dans des généralisations inutiles, on part du principe qu'une parabole d'équation $-(x-\frac{1}{4})(x+\frac{1}{4})$ s'annule en $\frac{-1}{4}$ et $\frac{1}{4}$, qu'elle est renversée i.e qu'elle va vers l'infini dans les valeurs négatives, que la distance entre ses deux points d'annulation est égale à $\frac{1}{2}$ et enfin qu'elle est centrée en 0. De la sorte on construit facilement un ensemble de paraboles centrées en 0 et dont la distance diminue de moitié chaque fois. Leur équation générale est $-(x-\frac{1}{2^n})(x+\frac{1}{2^n})=-x^2+\frac{1}{2^{2n}}$. Finalement, il faut trouver un moyen de décaler ces paraboles pour qu'elles se suivent afin d'avoir notre représentation du lancer, on passe les détails techniques ayant permis d'y aboutir et on trouve la suite de fonction indexée par $n=2,3,4$:

$$f_n(x)=-(x-\frac{2^{n+2}-3}{2^n})^2-\frac{1}{2^{2n}} \tag{2}$$

On commence à $n=2$ pour avoir un premier lancer qui fait la moitié de 1 mais on peut toujours changer les indices pour adapter la formule. // Cette suite de parabole décroît également en amplitude donc les courbes deviennent vite petites et on ne les voit presque plus à partir de $n=6$. L'amplitude décroît en 2^{2n} à chaque terme donc j'ai multiplié par ce facteur afin d'avoir des lancers de même amplitude, tous normalisés à 1. Il y a quelques imprécisions graphiques, d'abord parce que plus on approche la fin du vecteur plus il y a de courbes à tracer et moins il y a de nombres. Ensuite les paraboles partagent un point $y=0$ deux à deux mais je n'ai pas pris le temps d'écrire une condition pour garder un seul 0 et non deux qui se suivent (ce qui a pour effet de décaler les courbes), j'ai simplement enlevé les 0 donc c'est normal si la courbe ne touche pas l'abcisse.

3 Flèche en vol (zenon_fleche.py)

Zenon affirme qu'une flèche ne peut jamais atteindre sa cible car sa trajectoire n'est qu'une suite d'instants pendant lesquels elle est immobile. Dans ce programme nous animons le parcours linéaire d'une flèche vers sa cible en décomposant la trajectoire par des points espacés régulièrement. On a décidé de ne pas offrir la possibilité à l'utilisat-eur-rice de changer les paramètres étant donné que ça n'avait pas grand intérêt pour la démonstration.