L. DIDIER & R. CABANE

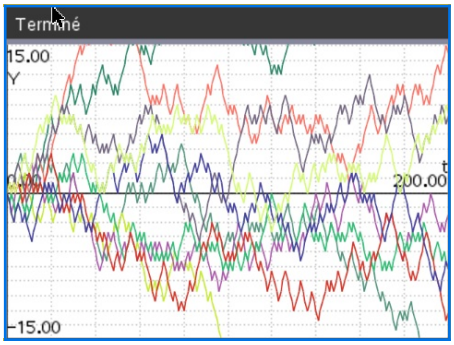
Marches aléatoires

Présentation et objectifs

Dans les programmes

Première : approfondissements possibles • Exemples de succession de plusieurs épreuves indépendantes. • Exemples de marches aléatoires.	**Terminale :** • Variables aléatoires indépendantes • Sommes de variables aléatoires. **Approfondissements :** marche aléatoire.

Situation déclenchante

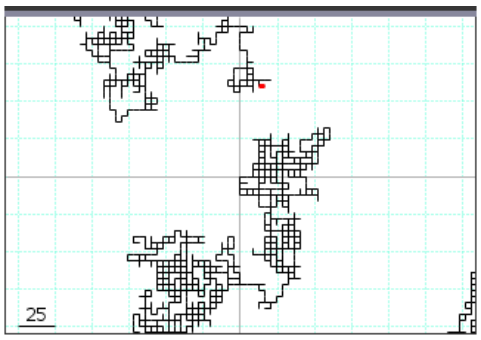


Les marches aléatoires sont des modélisations de phénomènes de nature chaotique comme le déplacement des molécules d'un gaz dans une enceinte fermée, celui de petites particules en suspension dans un liquide ou encore les cours des marchés financiers.
La marche aléatoire unidimensionnelle peut s'expliquer comme un jeu. On place un pion à l'origine d'un axe gradué, et on le déplace avec cette règle : à chaque unité de temps, le pion avance d'un pas (1 unité de longueur), soit à gauche soit à droite et de manière équiprobable. Il s'agit d'étudier le mouvement du pion sur une durée longue, et d'abord de le simuler.

Objectifs 1D (spécialité en Première)

1. On note Y l'abscisse du pion à chaque instant. Simuler la variable Y à l'aide du module `random`.
2. En utilisant le module `ti_plotlib`, représenter la marche aléatoire sur un graphique avec le temps en abscisse et Y en ordonnée (comme le graphe[1] ci-dessus, mais avec une seule courbe).

Objectifs 2D (spécialité en Terminale)



L'étude simultanée de plusieurs variables aléatoires permet d'envisager une marche aléatoire en 2 dimensions. Suivant la même idée, on déplace un pion dans le plan par « pas » suivant quatre directions possibles (équiprobables) : Nord, Sud, Ouest, Est, et on s'interroge sur le trajet suivi par le pion.

3. Simuler la marche aléatoire dans l'environnement Python.
   On pourra noter (X;Y) les coordonnées du pion.
4. Autre approche : justifier que (X;Y) peut aussi être obtenu à partir de deux marches aléatoires (U;V) en dimension 1, indépendantes, de la manière suivante : $X=(U+V)/2$, $Y=(U-V)/2$.
5. Représenter graphiquement la marche aléatoire en 2D (on pourra utiliser le module `turtle`).

1 Réalisé avec le logiciel Nspire CX II.

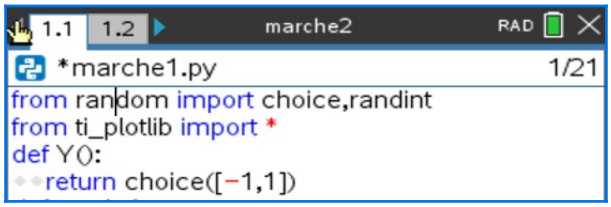
L. DIDIER & R. CABANE

Fiche méthode

Marche aléatoire 1D

▶ Objectif 1 :

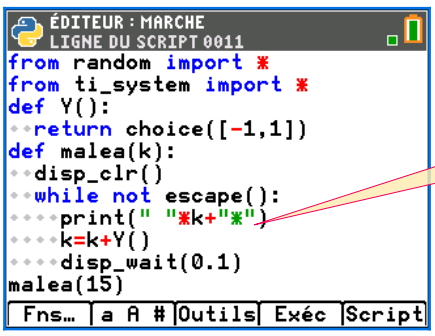
Pour simuler la variable aléatoire Y, on peut se servir de la fonction `randint` fournie par la bibliothèque (ou module) `random`, nécessitant deux paramètres pour préciser l'intervalle d'entiers entre lesquels les nombres au hasard seront fournis. Voir l'appendice 1 pour plus de détails.

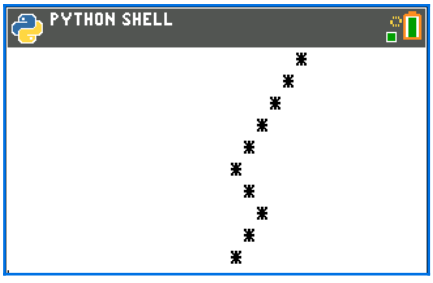


Ici, on attend des nombres 0 ou 1 (probabilité ½) ; pour obtenir des nombres -1 ou 1, une transformation arithmétique suffit alors (`2*randint(0,1)-1`). L'espérance est évidemment nulle.

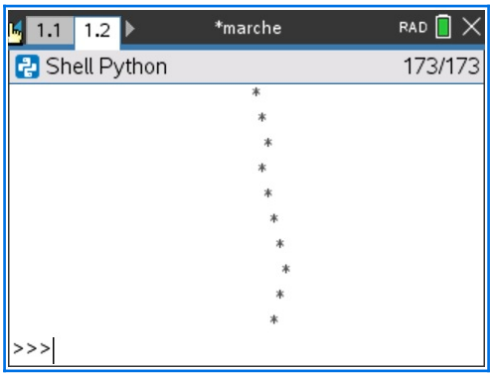
Une autre fonction fournie par ce module est `choice`, qui prend en paramètre la liste des valeurs entre lesquelles un tirage au hasard (équiprobable) est recherché. C'est le mécanisme du « pile ou face ».

Une représentation très simple de la marche aléatoire peut être obtenue en mode texte, en faisant défiler des lignes où une étoile est placée comme le pion. Pour positionner le caractère `*` dans la ligne, on le fait précéder d'un nombre variable d'espaces (c'est le rôle de la variable `k`).





Dans l'expression `" "*k+"*"`, le premier astérisque indique une répétition du caractère précédent et le signe + indique une concaténation.

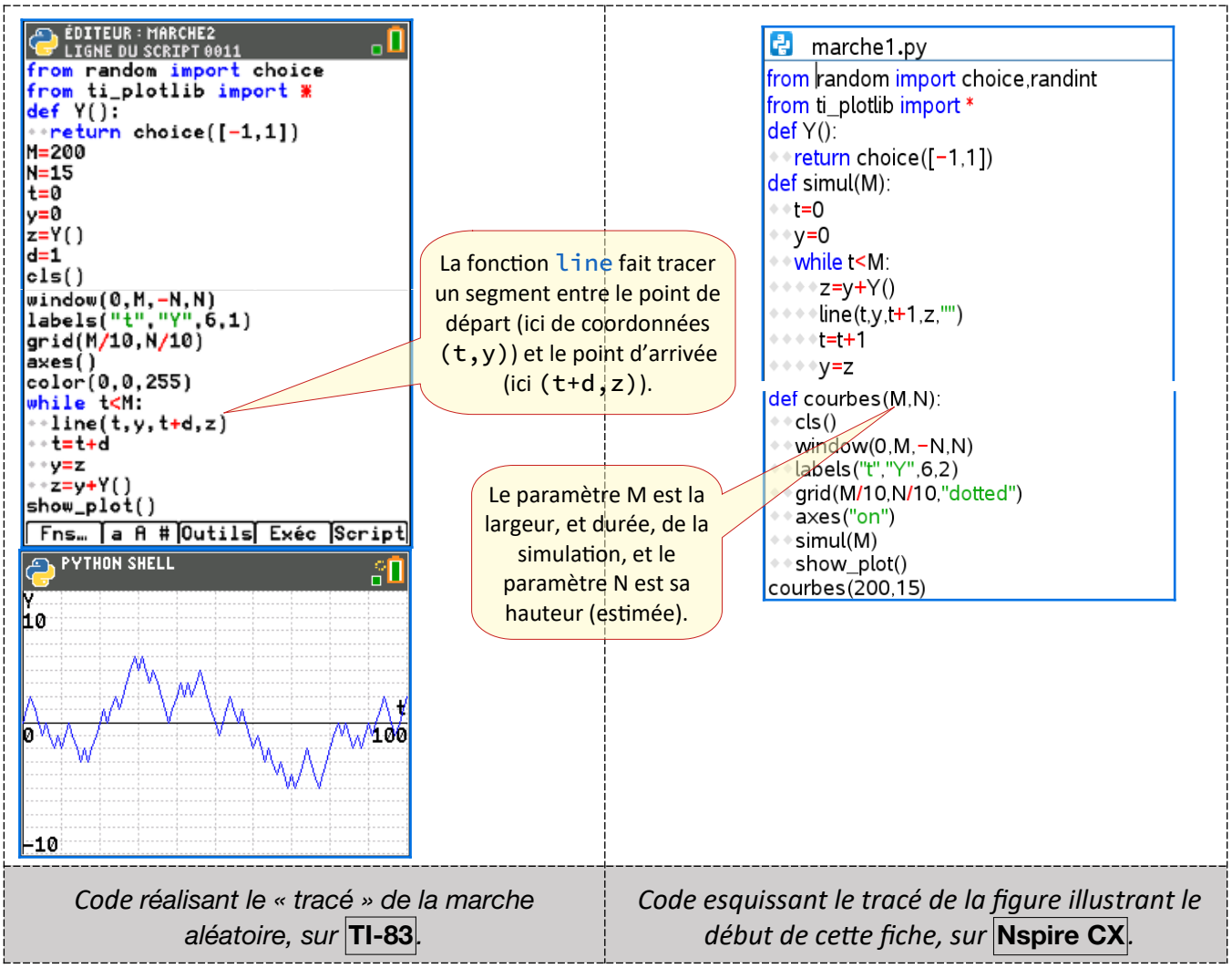


Codage pour TI-83 CE	*Codage pour Nspire CX-II*

On constate que la marche aléatoire s'écarte assez peu du point de départ (traduisant l'espérance nulle), et y revient régulièrement (c'est un théorème dont la démonstration n'est pas au niveau du programme de spécialité).

▶ Objectif 2 :

Pour l'affichage graphique, il convient de définir une échelle, ici conventionnellement prise égale à 100 en abscisse et à 10 en ordonnées. À chaque « pas » on doit recalculer la position du pion (variable `z`), tracer le trait correspondant (instruction `line`) et mettre à jour abscisse (`t`) et ordonnée (`y`).



Code réalisant le « tracé » de la marche aléatoire, sur **TI-83**.

Code esquissant le tracé de la figure illustrant le début de cette fiche, sur **Nspire CX**.

Marche aléatoire en 2D

▶ Objectif 3 : une simulation de (X;Y)

Supposons que les coordonnées soient entières et que les « pas » du pion soient d'une unité de longueur. Il nous faut un premier tirage « pile ou face » (-1 ou 1) pour décider si le pion va se déplacer verticalement ou horizontalement, et un second pour décider de la direction qu'il prendra. Le code Python est alors très simple, ci-contre avec un exemple d'exécution associé.

À noter : les trois appels de `b` ne désignent pas la même valeur, puisque chaque appel à la fonction `randint` ou `choice` « relance la pièce ».

▶ Objectif 4 : autre approche

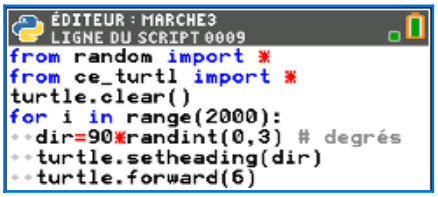
En posant $X=\frac{U+V}{2}, Y=\frac{U-V}{2}$, on a l'initialisation $(X;Y)=(0\,;0)$, et les évolutions possibles détaillées ci-contre, ce qui se trouve en accord avec les règles de déplacement du pion. Grâce au principe d'additivité de l'espérance, on en déduit que les espérances de X et Y sont nulles.

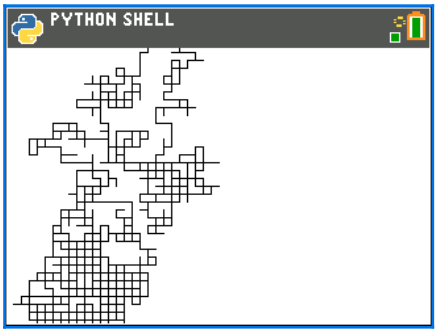
U	V	X	Y
+1	+1	+1	0
−1	−1	−1	0
+1	−1	0	+1
−1	+1	0	−1

▶ Objectif 5 : représentation graphique

Nous allons maintenant employer le module additionnel Turtle (présenté dans l'avant-propos et détaillé dans l'appendice 2).

Le programme ci-contre[2] est très simple : à chaque étape il tire un angle au hasard, multiple de 90°, oriente la tortue en conséquence puis provoque son avancement de 6 « pas » (ou pixels). La figure tracée ressemble à celle qu'on voit à droite. On constate que, cette fois, le « pion » ne repasse que très rarement, voire jamais, sur son point de départ ; d'autres conjectures peuvent être faites (éloignement, franchissement de droites, etc.).





D'autres expérimentations

Il est dès lors facile de simuler une marche aléatoire évoluant avec des angles de 60° au lieu de 90°, ou avec des angles un peu quelconques. Autant le « retour à l'origine » est possible avec un angle de 60°, autant il n'a pratiquement plus jamais lieu avec d'autres angles. À essayer !

2 Ce programme devra être adapté selon la calculatrice employée et les mises à jour du module Turtle.