Simulations, en seconde, avec une calculatrice

Nous allons d'abord présenter quelques fonctions ou commandes implantées sur les calculatrices actuelles et qui vont nous permettre de réaliser quelques unes des simulations qui sont proposées dans le programme de mathématiques de Seconde Générale et Technologique applicable depuis la rentrée 2000. Il convient de noter que les exemples de programmes indiqués sont à utiliser sur des calculatrices de marque Casio. Des exemples de programmes qui automatisent certaines manipulations sont fournis, ils correspondent au langage de programmation des calculatrices récentes de marque Casio. L'enseignant pourra éventuellement, avec les élèves, construire l'algorithme d'un programme simple.

Afin d'éviter une saisie fastidieuse, les programmes Casio donnés dans cet article seront disponibles dans la conférence Maths/Info. Ils seront donnés sous la forme d'un fichier informatique nommé Seconde.fxi. Pour utiliser ces programmes et les transférer sur la calculatrice via l'ordinateur, il faut disposer de l'interface Casio (FX interface for Windows) dont le prix est d'environ 300 F.

LES OUTILS UTILES

Fonctions :

* Fonction RAND ou Ran# ou Random

Cette fonction, qui n'a pas besoin d'argument, renvoie un nombre décimal aléatoire appartenant à l'intervalle [0 , 1[. Le nombre de chiffres décimaux varie en fonction du modèle de calculatrice (10 ou plus). En activant plusieurs fois successivement cette fonction (il suffit d'appuyer sur la touche ENTER ou EXE plusieurs fois consécutives après avoir activé une première fois la fonction), on peut fabriquer une suite de nombres (pseudo) aléatoires.

* Fonction RANDINT *sur certaines calculatrices uniquement*

RANDINT(1,6) renvoie un nombre entier aléatoire compris, au sens large, entre 1 et 6.
Cette fonction accepte pour arguments deux listes de même longueur, elle renvoie alors une liste dont les éléments sont des nombres entiers aléatoires compris entre les entiers de même rang des listes arguments.
Remarque : les arguments doivent être nécessairement des entiers ou des listes d'entiers.

* Fonction Int

Int a renvoie la partie entière du nombre a ou bien la liste des parties entières des éléments de la liste si on lui donne comme argument une liste.
Exemple : Int 3.218 renvoie la valeur 3 ; Int{2.718 , 3.14} renvoie la liste {2 , 3}.

* Fonction Seq

Cette fonction, qui nécessite 5 arguments, renvoie une liste. Sa syntaxe est :

Seq(exp, var, deb, fin, pas)

argument 1 : exp	est une expression qui renvoie une valeur numérique, par exemple 3X+2.
argument 2 : var	est une variable, par exemple X.
argument 3 : déb	est la valeur de départ pour la variable, par exemple 1.
argument 4 : fin	est la valeur de fin pour la variable, par exemple 5.
argument 5 : pas	est le pas d'incrément pour la variable, par exemple 1.

Exemple 1 : Seq(3X+2,X,1,5,2) renvoie {5, 11, 17}, c'est-à-dire $(3 \times 1 + 2, 3 \times 3 + 2, 3 \times 5 + 2)$

Exemple 2 : Seq(RANDINT(1,6),X,1,90,1) renvoie une liste de 90 entiers aléatoires compris (au sens large) entre 1 et 6.

Opérateurs de comparaison :

Selon les modèles de calculatrice, ces opérateurs : $=$, $\neq$, $\geq$, $\leq$, $<$, $>$, peuvent se trouver dans un menu TEST ou bien dans le sous menu REL du menu PRGM ou encore dans un sous menu INEQ du menu MATH.
Ces opérateurs renvoient le nombre 1 ou 0 selon que la condition qu'ils expriment est réalisée ou non.
Exemple 1 : 3 = 5 renvoie la valeur 0,
Exemple 2 : 3 = INT(3.8) renvoie la valeur 1.
Ces opérateurs travaillent aussi avec des listes de même longueur, ils renvoient alors une liste constituée de 1 et de 0 qui correspondent aux résultats des comparaisons effectuées sur chacun des couples de nombres des éléments de même rang des deux listes.
Exemple 3 : {1, 5, 9} > {2, 3, 9} renvoie {0, 1, 0}.

Affectation :

Pour stocker en mémoire un résultat numérique ou une liste de résultats numériques, on utilise la commande STO ou encore → (selon le modèle de calculatrice) suivie d'un nom de mémoire (A, B, ……, Z) ou d'un nom de liste List1 (ou L1), List2 (ou L2), …, List6 (ou L6).
Pour utiliser ces valeurs stockées ou les afficher à l'écran, il suffit de taper le nom de la mémoire ou le nom de la liste et de valider.
Exemple 1 : 3,14159→A EXE permet de stocker la valeur approchée par défaut à 10^{-5} près de π dans la mémoire A. Pour retrouver ce nombre, il suffit de saisir A et d'appuyer sur EXE.

Exemple 2 : {1,2,3,4,5}→List1 EXE permet de stocker 1, 2, 3, 4, 5 dans la liste 1.

QUELQUES EXEMPLES DE SIMULATIONS

Des pièces :

* Pièce non truquée

- sans programmer

Pour un lancer de pièce, nous conviendrons que "Pile" sera représenté par 0 et "Face" par 1.

Ran#<0.5 cette instruction renvoie 0 ou 1.
Int(2Ran#) produit le même résultat.
RANDINT(0,1) produit le même résultat.

Si l'on appuie consécutivement 50 fois sur la touche EXE ou ENTER, on simule 50 lancers d'un pièce équilibrée.
Si l'on veut mémoriser les 50 résultats dans une liste, il suffit alors de taper l'instruction :

Seq(Ran#<0.5,X,1,50,1)→List1

La fonction Sum(List1) permet de calculer la somme des éléments de la liste 1. Saisir les instructions ci-dessous permet de :

Sum(List1=0)/50 faire afficher la fréquence de la modalité 0 de la liste 1.
Sum(List1=1)/50 faire afficher la fréquence de la modalité 1 de la liste 1.

- en programmant

Le premier programme (syntaxe Casio) est utilisable pour un nombre de lancers N ≤ 255 (au-delà, on ne peut utiliser les listes de nombres qui comportent au maximum 255 lignes). Le programme est le suivant :

Programme	Exemple d'affichage calculatrice
"NOMBRE DE LANCERS"?→N Seq(Ran#<0.5,X,1,N,1)→List 1 "FREQUENCE DE PILE" Sum (List 1=0)÷N◢ "FREQUENCE DE FACE" Sum (List 1=1)÷N	NOMBRE DE LANCERS? 250 FREQUENCE DE PILE 0.456 FREQUENCE DE FACE 0.544

Le programme ci-dessous permet d'effectuer un nombre de lancers supérieur à 255, sans utiliser les listes.

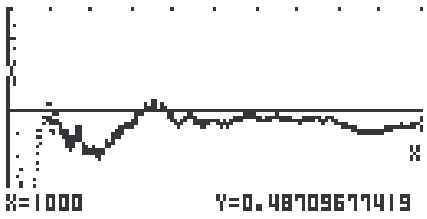
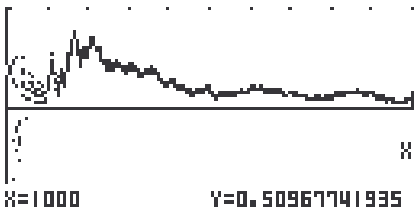
0→S	Remise à zéro de la mémoire S
"NOMBRE DE LANCERS"?→N	N sera le nombre de lancers souhaités
For 1→I To N	I sera la mémoire dans laquelle on rangera le nombre N de lancers effectués (I varie de 1 à N)
Ran#<0.5→V	V sera la mémoire dans laquelle on rangera les différentes valeurs des différents lancers
If V=0	Si V = 0 (Pile est obtenu)
Then 1+S→S	alors la valeur de la mémoire S augmente de 1
IfEnd	Instruction de "fin de Si"
Next	On répète l'opération précédente N fois
"FREQUENCE DE PILE"	Affichage du texte FRÉQUENCE DE PILE
S÷N◢	Calcul et affichage de la fréquence de Pile
"FREQUENCE DE FACE"	Affichage du texte FRÉQUENCE DE FACE
(N−S)÷N	Calcul et affichage de la fréquence de Face

En utilisant le programme ci-dessous, on peut illustrer graphiquement le fait que la fréquence de Pile (ou Face) se rapproche de 0,5 lorsque le nombre N de lancers est important.

"NOMBRE DE LANCERS"?→N	N sera le nombre de lancers souhaités
ViewWindow 0,N,100,0.4,0.6,0.1	On défini les paramètres de la fenêtre du graphique
Graph Y=0.5	On fait tracer la droite d'équation y = 0,5
0→S	Remise à zéro de la mémoire S
For 1→I To N	I sera la mémoire dans laquelle on rangera le nombre N de lancers effectués (I varie de 1 à N)
Ran#<0.5→V	V sera la mémoire dans laquelle on rangera les différentes valeurs des différents lancers
If V=0	Si V = 0 (Pile est obtenu)
Then 1+S→S	alors la valeur de la mémoire S augmente de 1
IfEnd	Instruction de "fin de Si"
Plot I,S÷I	On place graphiquement le point de coordonnées (I ; S÷I)
Next	On répète l'opération précédente N fois

Voici deux exemples obtenus pour 1 000 lancers consécutifs (la fréquence affichée est celle de "Pile" et la ligne horizontale symbolise la droite d'équation y = 0,5)

.

Exemple n°1	*Exemple n°2*
	

* Pièce truquée

Pour simuler le lancer d'une pièce truquée telle que la probabilité d'obtenir "Pile" soit p, il suffit d'utiliser l'instruction :

Ran#≤p

* Distribution de fréquences **du "nombre de "Pile" lors de 4 lancers consécutifs**"

- sans programmer

On peut simuler 100 séries de 4 lancers (*ou plus selon les capacités de la calculatrice*). En réalité on effectue 4 séries de 100 lancers que l'on range dans les listes List1, List2, List3 et List4. Ensuite on additionne les 4 listes dans la liste List5 et il suffit de calculer la fréquence des 0, 1, 2, 3 et 4 dans la liste List5.

Seq(Ran#<0.5,X,1,100,1)→List1

Remarque importante : il est inutile de retaper l'instruction pour la liste List2, il suffit d'appuyer sur la flèche de droite du pavé fléché (ou d'utiliser la commande ENTRY qui réécrit la dernière instruction et les précédentes en activant encore ENTRY si besoin est) et remplacer List1 par List2 et ainsi de suite.

Ensuite, on tape :

List1 + List2 + List3 + List4 → List5

Pour obtenir les fréquences des modalités 0 (4 fois "Pile" pour 4 lancers consécutifs d'une pièce), 1 (une fois "Pile"), 2 (deux fois "Pile"), 3 (trois fois "Pile") et 4 (quatre fois "Pile"), on tape :

Sum(List5=0)/100 , Sum(List5=1)/100, … Sum(List5=4)/100

On obtient ainsi la distribution de fréquences correspondant à 100 lancers. On peut comparer les résultats obtenus par chacun des élèves et constater que ces résultats varient d'un tableau à un autre même pour les mêmes marques de calculatrice : *c'est la fluctuation d'échantillonnage.* En effet on peut considérer que chaque élève a tiré un échantillon de taille 100, de la population virtuelle de l'infinité des résultats de 4 lancers consécutifs d'une pièce. Si on regroupe les résultats des élèves par groupe de 5, on voit alors que la variabilité est plus faible, et par groupe de 10 encore plus faible.

Cette situation est à rapprocher avec celle d'avoir une fille (ou un garçon), 2 filles, 3 filles ou 4 filles dans une famille de 4 enfants.

- en programmant

Les opérations précédentes peuvent être programmées pour N séries de 4 lancers (avec N ≤ 255). Voici le listing du programme et l'affichage calculatrice.

Programme	Exemple d'affichage calculatrice
"NBR SERIES 4 LANCERS"?→N Seq(Ran#<0.5,X,1,N,1)→List 1 Seq(Ran#<0.5,X,1,N,1)→List 2 Seq(Ran#<0.5,X,1,N,1)→List 3 Seq(Ran#<0.5,X,1,N,1)→List 4 List 1+List 2+List 3+List 4→List 5 "FREQUENCE 4 PILE" Sum (List 5=0)÷N◢ "FREQUENCE 3 PILE" Sum (List 5=1)÷N◢ "FREQUENCE 2 PILE" Sum (List 5=2)÷N◢ "FREQUENCE 1 PILE" Sum (List 5=3)÷N◢ "FREQUENCE 0 PILE" Sum (List 5=4)÷N	NBR SERIES 4 LANCERS? 100 FREQUENCE 4 PILE 0.02 FREQUENCE 3 PILE 0.31 FREQUENCE 2 PILE 0.36 FREQUENCE 1 PILE 0.23 FREQUENCE 0 PILE 0.08

Des dés non pipés :

* 1 lancer

Si l'on dispose de la fonction RANDINT alors : **RANDINT(1,6)**
sinon **Int(6×Ran#) + 1**
en effet, $0 \leq$ **Ran#** < 1
$0 \leq$ **6×Ran#** < 6
$0 \leq$ **Int(6×Ran#)** ≤ 5
$1 \leq$ **Int(6×Ran#) + 1** ≤ 6

Pour relancer le dé il suffit d'appuyer à nouveau sur EXE ou ENTER.

* une série de 100 lancers

- sans programmer

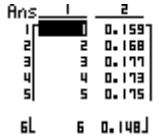
Comme pour les pièces, on aura : **Seq(Int(6×Ran#)+1,X,1,100,1)→List1**

- en programmant

Ces instructions sont programmables. Les fréquences des tirages des faces de la pièce ne sont pas rangées dans des mémoires comme pour le jeu de Pile ou Face (c'est pourtant réalisable), mais dans une liste (ici liste 1). Les tirages n'apparaissent dans aucune des listes, ce qui permet un nombre de lancers supérieur à 255.

"NOMBRE DE LANCERS"?→N Seq(X,X,1,6,1)→List 1 Seq(0,X,1,6,1)→List 2 For 1→I To N Int (6Ran#+1)→X 1+List 2[X]→List 2[X] Next List 2÷N→List 2 List→Mat(List 1,List 2)	N sera le nombre de lancers souhaités On fait afficher 1, 2, 3, 4, 5 et 6 dans la liste 1 On définit le nombre de lignes de la liste 2. On obtient 6 lignes comportant chacune un zéro I sera la mémoire dans laquelle on rangera le nombre N de lancers effectués (I varie de 1 à N) X sera la mémoire contenant le résultat du lancer On augmente de 1 le contenu de la $X^{ième}$ ligne de la liste 2 On répète l'opération précédente N fois Tous les tirages étant effectués, on remplace les 6 lignes de la liste 2 par la fréquence correspondante On fait afficher le contenu des listes 1 et 2

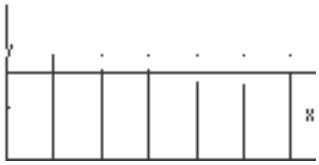
Un exemple de l'affichage de la calculatrice.



Il est possible de tracer un diagramme en bâtons permettant de visualiser les fréquences d'apparition des 6 faces.

"NOMBRE DE LANCERS"?→N	N sera le nombre de lancers souhaités
Seq(0,X,1,6,1)→List 1	On définit le nombre de lignes de la liste 1. On obtient 6 lignes comportant chacune un zéro
For 1→I To N	I sera la mémoire dans laquelle on rangera le nombre N de lancers effectués (I varie de 1 à N)
Int (6Ran#+1)→X	X sera la mémoire contenant le résultat du lancer
1+List 1[X]→List 1[X]	On augmente de 1 le contenu de la $X^{ième}$ ligne de la liste 1
Next	On répète l'opération précédente N fois
List 1÷N→List 1	Tous les tirages étant effectués, on remplace les 6 lignes de la liste 1 par la fréquence correspondante
List 1◢	On fait afficher le contenu de la liste 1
ViewWindow 0,6.5,1,0,Max(List 1)+0.1,0.1	On défini les paramètres de la fenêtre du graphique
For 1→A To 6	Pour des valeurs de la mémoire A allant de 1 à 6, on place les points de coordonnées (A ; 0), c'est le point bas du premier bâton et (A ; List 1[A]), c'est le point haut du premier bâton
Plot A,0	
Plot A,List 1[A]	On relie les deux points à chaque fois pour tracer le bâton
Line	
Next	
Graph Y=1÷6	On trace une ligne horizontale pour situer la fréquence de 1/6

La calculatrice affiche d'abord le contenu de la liste 1 (les fréquences d'apparition des faces de 1 à 6), puis un diagramme en bâton similaire à celui ci-dessous où la ligne horizontale symbolise la droite d'équation $y = \frac{1}{6}$.



* Distribution de fréquences de "**la somme des chiffres portés sur les faces supérieures lors du lancer de 2 dés**"

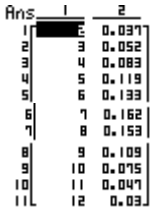
- sans programmer

On effectue, par exemple, cette simulation sur 100 lancers de 2 dés. On a donc, sans commentaire :

Seq(Int(6×Ran#)+1,X,1,100,1)→List1
Seq(Int(6×Ran#)+1,X,1,100,1)→List2
List1+List2→List3
Sum(List3=2)/100 ; Sum(List3=3)/100 ; …….. ; Sum(List3=12)/100

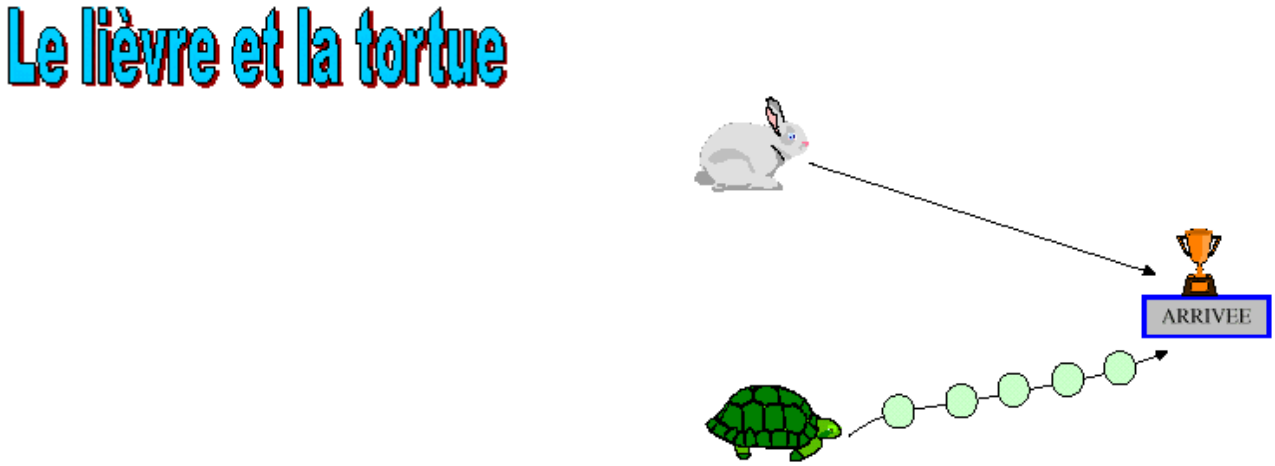
- en programmant

Le programme correspondant peut être le suivant :

Programme	Exemple d'affichage calculatrice
"NOMBRE DE LANCERS"?→N Seq(X,X,2,12,1)→List 1 Seq(0,X,0,10,1)→List 2 For 1→I To N Int (6Ran#+1)+Int (6Ran#+1)→X 1+List 2[X−1]→List 2[X−1] Next List 2÷N→List 2 "LES FREQUENCES DES SOMMES 2, 3, 4,...,11 ET 12 SONT :" ◢ List→Mat(List 1,List 2)	

Pour un nombre important de lancers, chaque élève pourra comparer ses résultats avec les autres et essayer de dégager une tendance.

Le jeu du lièvre et de la tortue : (extrait du document d'accompagnement des programmes de mathématiques de la classe de seconde).



Une partie du jeu du lièvre et de la tortue se déroule ainsi :

On lance un dé. Si le dé tombe sur 1, 2, 3, 4 ou 5 la tortue avance d'une case. Il lui reste alors 5 cases à franchir avant d'atteindre l'arrivée. Si lors des 5 lancers suivants le dé ne tombe pas sur 6, la partie est alors terminée et la tortue à gagné.
Si le dé tombe sur 6 lors de l'un des 6 lancers, le lièvre atteint directement l'arrivée. La partie est alors terminée et le lièvre à gagné.
Quelle est la situation la plus enviable : celle du lièvre ou de la tortue ?

Pour une partie, le programme peut être le suivant :

```
For 1→I To 6
Int (6Ran#+1)→X
If X=6
Then "×× LE LIEVRE A GAGNE"
Stop
Else "LA TORTUE AVANCE"
Next
"LA TORTUE A GAGNE ××"
```

Pour un nombre N de parties, le programme peut être le suivant :

```
"NOMBRE DE COURSES"?→N
0→L
0→T
For 1→J To N
0→A
For 1→I To 6
Int (6Ran#+1)→X
If X=6
Then L+1→L
1→A
Break
IfEnd
Next
A=1⇒Goto 1
T+1→T
Lbl 1
Next
"PART.GAGNEES PAR L"
L◢
"PART.GAGNEES PAR T"
T
```

Pour un même nombre de courses, chaque élève pourra annoncer ses résultats et il sera possible de dégager une tendance pour voir si le jeu est réellement équitable.