Compte Rendu TP2
SIMULATION

Question 1 :

Après téléchargement, compilation et exécution du code, on remarque que les nombres générés aléatoirement sembles être les mêmes que ceux fournis avec le code. La génération pseudo aléatoire est donc bien déterministe comme attendu.

Question 2 :

Pour obtenir un nombre dans l'intervalle [a ; b] à partir d'un nombre aléatoire appartenant à [0 ; 1], il suffit d'appliquer la formule vu en cours. On note x le n :ombre de départ et y le nombre appartenant à l'intervalle voulu : alors y = x * (b – a) + a.

Question 3 :

On utilise la génération de nombre entre 0 et 1 et les propriétés des tableaux de probabilités cumulées pour générer un tirage ayant 3 possibilités. On pourrait réutiliser cette méthode pour créer un tirage ayant un nombre quelconque de possibilités. On remarque cependant qu'avec seulement 1 000 tirages, la répartition n'est pas suffisamment précise : il y a une erreur de l'ordre du centième de pourcent. Une fois atteint les 1 000 000 de tirages, la précision atteint le millionième de pourcent.

Question 4 :

On remarque bien dans les deux cas, que les petits nombres sont tirés plus de fois que les grands nombres. On peut donc en conclure que la distribution semble suivre une loi exponentielle négative. Attention toute fois, énormément de nombres sont générés loin après les bornes fixées. Il faut donc le prendre en compte lors de l'utilisation de fonctions de ce type.

Question 5 :

La génération fonctionne très bien sur l'intervalle [-2 ; 2], mais en dehors de cette intervalle la répartition est erronée : il y a plus d'éléments sur l'intervalle négatif que sur l'intervalle respectif positif. Pour résoudre ce problème, il suffirait de restreindre la génération à [-1 ; 1] (pour être sûr de la qualité de la répartition), puis de l'étendre en suivant le même principe que pour la question 2.

Question 6 :

Des librairies existent en C, C++ et JAVA pour permettre la génération de nombres aléatoires. En C, on peut tout d'abord citer celle vu dans ce TP (question 1). En C++, on peut réutiliser toutes les librairies du langage C. Cependant, il existe une implémentation de plusieurs algorithmes dans la librairie standard (Mersenne-Twister, congruence linéaire, etc.). Enfin en JAVA, on peut ici aussi réutilisé du code C, grâce à l'interface native de JAVA. Cependant, une implémentation JAVA existe

aussi dans la librairie standard : random. Random n’est pas très sécurisé (période de 2^{48}), il existe également SecureRandom qui possède une période de 2^{160}. Oracle fournit également une génération suivant la méthode de Mersenne-Twister dans son JDK.