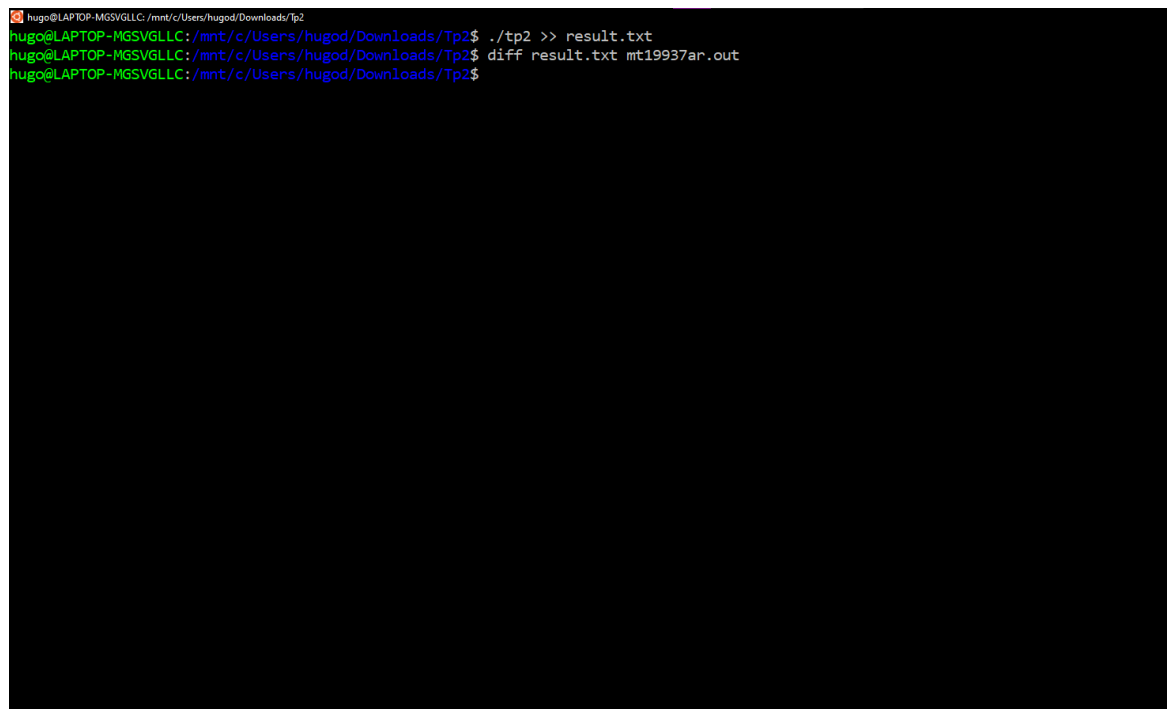


# Compte Rendu TP2 Simulation



Dans ce rapport je vais donc vous présenter les sorties des programmes que j'ai fait afin de répondre aux questions du TP. Le code est disponible en pièce jointe du mail. Pour tirer aléatoirement les réels j'ai utilisé la fonction `genrand_real2()` tout le long du TP. Toutes les fonctions sont en commentaires dans le main afin de pouvoir les retester si besoin.

- 1) Afin de savoir si la sortie de mon programme était la même que celle attendue dans le fichier `mt19937arc.out` j'ai mis le résultat de ma sortie dans un fichier `.txt`, j'ai ensuite utilisé la commande « diff » de Linux afin de voir les différences entre les 2 fichiers pour voir si tout les nombres pseudos aléatoires étaient les mêmes ce qui était le cas puisque la commande n'a rien retourné comme nous pouvons le voir sur ce screenshot.



```
hugo@LAPTOP-MGSVGLLC:/mnt/c/Users/hugod/Downloads/tp2$ ./tp2 >> result.txt
hugo@LAPTOP-MGSVGLLC:/mnt/c/Users/hugod/Downloads/tp2$ diff result.txt mt19937ar.out
hugo@LAPTOP-MGSVGLLC:/mnt/c/Users/hugod/Downloads/tp2$
```

- 2) Pour la deuxième question, je suis reparti du code de la fonction uniforme vue en cours en utilisant `genrand_real2()` à la place de `F_Rand()`.

```
hugo@LAPTOP-MGSVGLLC: /mnt/c/Users/hugod/Downloads/Tp2
hugo@LAPTOP-MGSVGLLC:/mnt/c/Users/hugod/Downloads/Tp2$ ./tp2
-52.93379586 -56.72652739 -72.98647670 50.32218194 54.45828949
24.40695019 24.78901002 -81.46745899 -61.67750219 -1.17389721
-2.22796873 20.95571795 -67.23001940 -38.94079005 55.01865154
-50.35964045 42.04884912 20.21808551 -28.95046497 -50.51846374
-40.89814703 -46.06094588 7.95193236 -29.26119509 42.43004516
45.50187908 -86.73548634 -8.21591806 -0.87059528 20.77758517
5.67483683 -64.56050680 -68.61955594 -79.98533354 -64.90679725
22.83301262 -44.25887133 54.47876092 -41.80876688 37.70161533
-17.80432483 37.51783850 11.02622623 -32.74452955 9.16281101
-8.67374693 -55.09099866 -12.86614632 -17.48111553 20.77258760
49.47992920 -81.73390169 -5.83628772 36.81449758 -60.81446251
-7.26608478 3.71488917 -63.50437742 -15.30814617 -25.54479026
4.64326204 52.17657210 24.63681240 -83.04701686 21.87819639
-62.83827283 -13.84905784 12.76927619 -25.78394170 -16.72068546
-88.01571024 -48.70999203 16.18548714 9.87408080 -54.80868577
-14.98130008 55.42381758 9.29589035 40.50065217 13.75477745
-45.64497989 -74.93247764 -10.17148383 3.92722494 -45.78564819
37.98974542 -79.96773457 0.01640616 -2.21766509 14.61153340
54.61288997 -47.47134132 32.94179482 -52.58323553 -7.81021556
-26.29004651 13.17480007 -21.26097148 12.94398948 -53.53892927
hugo@LAPTOP-MGSVGLLC:/mnt/c/Users/hugod/Downloads/Tp2$
```

Le résultat retourné par la fonction est bien celui attendu puisque ce sont des températures entre  $-89.2^{\circ}\text{C}$  et  $56.7^{\circ}\text{C}$ . Il n'y a que 100 répétitions mais on peut voir que ça fonctionne.

- 3) A) Le but de cette fonction est de partir de 3 classes contenant chacune une probabilité. On va effectuer un nombre de tirages de réels aléatoires et répartir ces réels selon les probabilités cumulées. A la fin si on compare les fréquences de bases de chaque classe et les fréquences finales on peut voir qu'il n'y a pas beaucoup de différences. Plus le nombre de tirages de réels est important plus le résultat se rapprochera des classes de bases. Voici les résultats obtenus à partir des classes A=35%, B=45% et C=20%.

```
hugo@LAPTOP-MGSVGLLC: /mnt/c/Users/hugod/Downloads/Tp2
hugo@LAPTOP-MGSVGLLC:/mnt/c/Users/hugod/Downloads/Tp2$ ./tp2
Pour 1 000 tirages il y a 0.343000 de A, 0.455000 de B, 0.202000 de C
Pour 100 000 tirages il y a 0.350670 de A, 0.449930 de B, 0.199400 de C
Pour 1 000 000 tirages il y a 0.350206 de A, 0.449636 de B, 0.200158 de C
hugo@LAPTOP-MGSVGLLC:/mnt/c/Users/hugod/Downloads/Tp2$
```

On peut remarquer que plus le nombre de tirages est important plus les pourcentages des classes de bases se retrouvent.

B) Cette question reprend le même principe que la question précédente mais de façon plus générique car on va passer en paramètre un tableau contenant des observations. Le principe reste quand même le même, plus il y a de tirages et plus nous retrouvons les pourcentages de base.

```
hugo@LAPTOP-MGSVGLLC: /mnt/c/Users/hugod/Downloads/Tp2
hugo@LAPTOP-MGSVGLLC:/mnt/c/Users/hugod/Downloads/Tp2$ ./tp2
Liste des probas des 6 classes :
    0.055556
    0.222222
    0.333333
    0.222222
    0.055556
    0.111111
Liste des probas cumulées des 6 classes :
    0.055556
    0.277778
    0.611111
    0.833333
    0.888889
    1.000000
Pour 1 000 000 tirages :
    Il y a 0.056000 de 0
    Il y a 0.225000 de 1
    Il y a 0.314000 de 2
    Il y a 0.236000 de 3
    Il y a 0.061000 de 4
    Il y a 0.108000 de 5
Pour 1 000 000 tirages :
    Il y a 0.055483 de 0
    Il y a 0.222453 de 1
    Il y a 0.333398 de 2
    Il y a 0.221980 de 3
    Il y a 0.055519 de 4
    Il y a 0.111167 de 5
hugo@LAPTOP-MGSVGLLC:/mnt/c/Users/hugod/Downloads/Tp2$
```

4) A) Pour cette question j'ai repris le code de la loi exponentielle négative déjà présent dans le cours.

B) Le but de cette question est de montrer qu'en donnant un paramètre un mean de 11, la moyenne de tous les résultats obtenus fait 11. Comme précédemment plus il y a de tirages plus la moyenne se rapprochera du mean.

```
hugo@LAPTOP-MGSVGLLC: /mnt/c/Users/hugod/Downloads/Tp2
hugo@LAPTOP-MGSVGLLC:/mnt/c/Users/hugod/Downloads/Tp2$ ./tp2
Pour 1 000 tirages la moyenne est de 11.099904
Pour 1 000 000 tirages la moyenne est de 11.000834
hugo@LAPTOP-MGSVGLLC:/mnt/c/Users/hugod/Downloads/Tp2$
```

C) Dans cette fonction on va tirer plusieurs nombres grâce à la loi exponentielle négative et on va les mettre dans des classes afin de montrer qu'il y a plus de réels compris entre 0-1 que de réels compris entre 1-2 et ainsi de suite.

```
hugo@LAPTOP-MGSVGLLC:/mnt/c/Users/hugod/Downloa
Pour 1 000 tirages :
    classe 0 à une fréquence de 0.081000
    classe 1 à une fréquence de 0.070000
    classe 2 à une fréquence de 0.087000
    classe 3 à une fréquence de 0.065000
    classe 4 à une fréquence de 0.053000
    classe 5 à une fréquence de 0.062000
    classe 6 à une fréquence de 0.049000
    classe 7 à une fréquence de 0.042000
    classe 8 à une fréquence de 0.039000
    classe 9 à une fréquence de 0.028000
    classe 10 à une fréquence de 0.045000
    classe 11 à une fréquence de 0.038000
    classe 12 à une fréquence de 0.033000
    classe 13 à une fréquence de 0.030000
    classe 14 à une fréquence de 0.027000
    classe 15 à une fréquence de 0.022000
    classe 16 à une fréquence de 0.023000
    classe 17 à une fréquence de 0.006000
    classe 18 à une fréquence de 0.022000
    classe 19 à une fréquence de 0.014000
    classe 20 à une fréquence de 0.016000
    classe 21 à une fréquence de 0.013000
    classe 22 à une fréquence de 0.135000
```

```
Pour 1 000 000 tirages :
    classe 0 à une fréquence de 0.087090
    classe 1 à une fréquence de 0.079665
    classe 2 à une fréquence de 0.072242
    classe 3 à une fréquence de 0.066156
    classe 4 à une fréquence de 0.060068
    classe 5 à une fréquence de 0.054860
    classe 6 à une fréquence de 0.050612
    classe 7 à une fréquence de 0.045941
    classe 8 à une fréquence de 0.042415
    classe 9 à une fréquence de 0.038449
    classe 10 à une fréquence de 0.034833
    classe 11 à une fréquence de 0.031869
    classe 12 à une fréquence de 0.029055
    classe 13 à une fréquence de 0.026853
    classe 14 à une fréquence de 0.024324
    classe 15 à une fréquence de 0.021897
    classe 16 à une fréquence de 0.020294
    classe 17 à une fréquence de 0.018703
    classe 18 à une fréquence de 0.016841
    classe 19 à une fréquence de 0.015383
    classe 20 à une fréquence de 0.014154
    classe 21 à une fréquence de 0.012798
    classe 22 à une fréquence de 0.135498
```

On peut voir encore une fois que plus le nombre de tirages est important plus ce qu'on cherche à montrer se précise. La classe 22 contient tout les réels compris entre 22 et + l'infini c'est pour cela qu'elle a une forte fréquence.

- 5) A) Pour cette fonction il fallait simuler le résultat de différentes sommes sur 30 lancers de dés à 6 faces. Ce résultat doit suivre une loi Gaussienne. J'ai réussi à recréer la simulation et a obtenir des résultats qui semblent cohérents avec une loi Gaussienne cependant je n'ai pas réussi à les afficher en courbe avec Excel.

```

hugo@LAPTOP-MGSVGLLC: /mnt/c/Users/hugod/Downloads/Tp2
hugo@LAPTOP-MGSVGLLC:/mnt/c/Users/hugod/Downloads/Tp2$ ./tp2
0.000000      0.000000      0.000000      0.000000      0.000000
0.000000      0.000000      0.000000      0.000000      0.000000
0.000000      0.000000      0.000000      0.000000      0.000000
0.000000      0.000000      0.000000      0.000000      0.000000
0.000000      0.000000      0.000000      0.000000      0.000000
0.000001      0.000003      0.000004      0.000010      0.000011
0.000016      0.000045      0.000057      0.000099      0.000153
0.000248      0.000365      0.000603      0.000909      0.001239
0.001782      0.002593      0.003431      0.004583      0.006196
0.008028      0.010132      0.012815      0.015570      0.019169
0.022681      0.026473      0.030379      0.033917      0.038142
0.041686      0.044833      0.047548      0.050037      0.050565
0.051206      0.050729      0.049684      0.047690      0.044905
0.041750      0.038169      0.034336      0.030165      0.026517
0.022568      0.018980      0.015524      0.012832      0.010315
0.007885      0.006147      0.004670      0.003432      0.002546
0.001859      0.001281      0.000860      0.000578      0.000416
0.000247      0.000148      0.000081      0.000062      0.000047
0.000024      0.000011      0.000005      0.000004      0.000002
0.000002      0.000000      0.000000      0.000000      0.000000
0.000000      0.000000      0.000000      0.000000      0.000000
0.000000      0.000000      0.000000      0.000000      0.000000
0.000000      0.000000      0.000000      0.000000      0.000000
0.000000      0.000000      0.000000      0.000000      0.000000
0.000000      0.000000      0.000000      0.000000      0.000000
0.000000      0.000000      0.000000      0.000000      0.000000
0.000000      0.000000      0.000000      0.000000      0.000000
0.000000      0.000000      0.000000      0.000000      0.000000
0.000000      0.000000      0.000000      0.000000      0.000000
hugo@LAPTOP-MGSVGLLC:/mnt/c/Users/hugod/Downloads/Tp2$

```

B) Pour cette fonction il fallait à partir de la méthode de Box and Muller recréer une courbe de Gausse en tirant des réels aléatoires et en appliquant les équations de Box and Muller. Après avoir obtenus des résultats il fallait trier ces résultats dans des classes et les afficher afin d'avoir un aperçu d'une courbe Gaussienne. J'ai fait cette partie avec beaucoup de if mais il y avait sûrement moyen de faire un code plus propre mais je n'ai pas réussi à trouver comment faire. Idem que pour la question précédente je n'ai pas réussi à afficher la courbe sur excel cependant les résultats que j'ai obtenu semblent en cohérence avec une courbe Gaussienne.

```

hugo@LAPTOP-MGSVGLLC: /mnt/c/Users/hugod/Downloads/Tp2
hugo@LAPTOP-MGSVGLLC:/mnt/c/Users/hugod/Downloads/Tp2$ ./tp2
Classe 0 : 0.00Classe 1 : 0.00Classe 2 : 0.00Classe 3 : 0.00Classe 4 : 0.00
Classe 5 : 0.00Classe 6 : 0.00Classe 7 : 0.00Classe 8 : 0.00Classe 9 : 0.00
Classe 10 : 0.00Classe 11 : 0.25Classe 12 : 0.33Classe 13 : 0.22Classe 14 : 0.12
Classe 15 : 0.05Classe 16 : 0.01Classe 17 : 0.00Classe 18 : 0.00Classe 19 : 0.00
Classe 20 : 0.00
hugo@LAPTOP-MGSVGLLC:/mnt/c/Users/hugod/Downloads/Tp2$

```

- 6) En cherchant sur internet j'ai trouvé les bibliothèques Splitmix, Xorshift et PCG pour le C++, pour le java j'ai trouvé PRNGine.