Projet de Calcul des probabilité et Théorie des erreurs

CASSART Justin - 180794

19 Décembre 2019

Table des matières

1 Introduction

Ce projet a pour but de répondre au problème suivant :

On considère une carte graphique dont on a testé la performance en changeant le processeur correspondant. Pour les tester, on a mesuré le temps CPU en ms nécessaire pour afficher plusieurs milliers de triangles. Cette procédure a été effectuée pour trois processeurs graphiques différents. Quel processeur choisiriez-vous?

Les données mises à disposition sont de l'ordre de 1000 mesures par nombre de triangles tracés. Ce nombre de triangles varie de 10 mille à 100 mille. Ce qui signifie que pour chaque processeur il y a 1000 * 10000 données, soit 100 mille données.

Afin de résoudre ce problème, il faudra suivre une certaine démarche :

- observation des données récoltées. Cette étape permettra d'avoir une vue d'ensemble du problème
- observation d'histogrammes par nombre de triangles et processeurs. Ceci permettra d'avoir une idée de la loi que suivent les données.
- détermination des paramètres de la loi.
- vérification de la loi. Un test de Kolmogorov sera nécessaire afin de déterminer si la loi avec les paramètres calculés est bien représentatrice des données mises à disposition.
-

Afin de résoudre ce problème, il faudra avant tout observer l'allure générale des données récoltes. Après quoi un série d'histogrammes sera créée pour chaque processeur et chaque tranche de nombre de triangles. Ceci permettra d'obtenir une idée de la loi régissant ces données. L'étape suivante sera d'estimer les paramètres de la dite loi. Après quoi des tests seront effectués pour vérifier que la loi supposée décrit bel et bien les données du problème. Finalement une comparaison sera effectuée entre les différents processeurs si leurs statistiques sont similaires. Et enfin une décision sera prise concernant le meilleur processeur.

2 Observation des données

La première étatape consiste à observer les données en les rapportant sur des graphiques. Cela permettra de comprendre plus facilement le comportement des nombreuses données qui sont à disposition. La première étape sera d'observer le comportement de chaque processeur individuellement. A cette observation on essayera déjà d'en ressortir une première pensée. Après quoi les trois processus seront affichés sur le même graphique. Cette démarche devrait nous permettre d'observer plus facilement la différence de comportement entre les trois processus.

2.1 Données brutes

Les données sont reportées en graphique grâce à fonction *plot* de la librairie *matplotlib.pyplot* de python. Les graphiques sont réalisés par la fonction *show* comprise dans le fichier *data_show.py*. Une explication de l'utilisation des fonctions de ce fichier est disponible en annexe avec quelques exemples.

Les trois premiers graphiques qui suivent représentent le comportement de la carte graphique avec le premier processeur, ensuite avec le second et enfin avec le troisième.

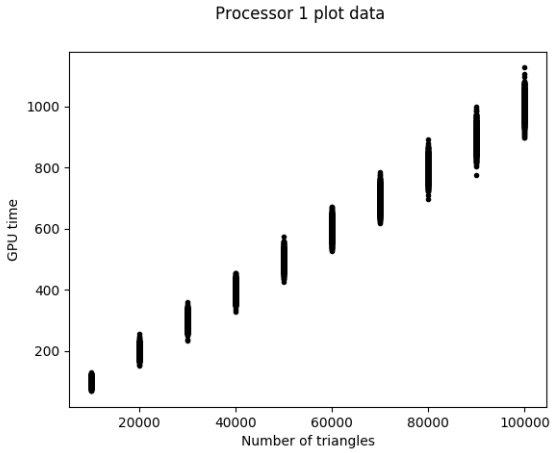

FIG. 1 – *Allure des données pour la carte graphique avec le processeur 1*

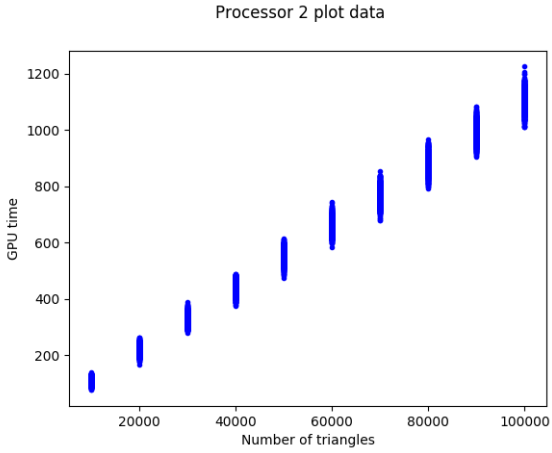

FIG. 2 – *Allure des données pour la carte graphique avec le processeur 2*

Il est dès lors possible de déterminer qu'avec l'utilisation des trois processeurs, dans chaqu'un des cas le temps d'exécution augmente avec le nombre de

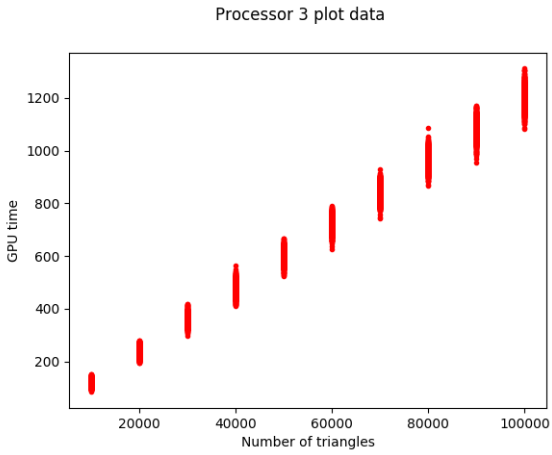

FIG. 3 – *Allure des données pour la carte graphique avec le processeur 3*

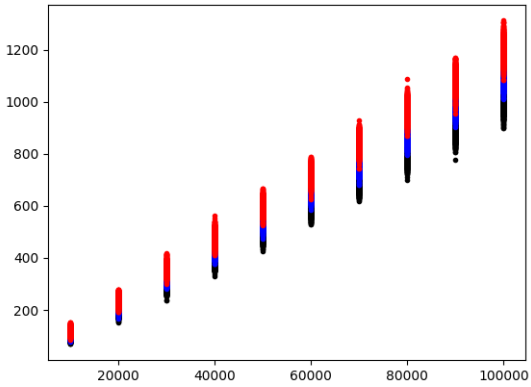

FIG. 4 – *Allure des données pour la carte graphique avec les trois processeurs*

triangles.

Maintenant, il serait intéressant d'observer les trois jeux de données sur un même graphique afin de se rendre compte plus facilement des différences de comportement ainsi que d'autres points communs. Pour faciliter les observations, le jeu de couleur a été respecté pour les trois processeurs.

Selon une première observation de ce dernier graphique, le temps d'exécution de la carte graphique avec le troisième processeur semble être dans l'ensemble plus long qu'avec les deux autres pour un nombre de triangles assez conséquents. Si le nombre de triangles est plus faible alors la carte graphique semble prendre un temps d'exécution de même grandeur peu importe le processeur utilisé. De plus, le premier processeur semble croître moins vite puisque une partie des points lui appartenant se situe plus bas que pour les deux autres.

Passons maintenant à une observation du comportement des données par nombre de triangles pour chaque processeur utilisé.

2.2 Observation des histogrammes

Dans cette section, il est question d'observer les histogrammes en fonction du nombre de triangles et du processeur utilisé pour la carte graphique. Seuls les histogrammes pour les tailles de triangles de 10, 50 et 100 mille sont utilisés. Les autres histogrammes sont disponibles en annexe.

La réalisation de ces histogrammes a été permise avec les fonctions *hist* et *xticks* de la librairie *matplotlib.pyplot*.

Passons dès lors à l'observation de ces histogrammes.

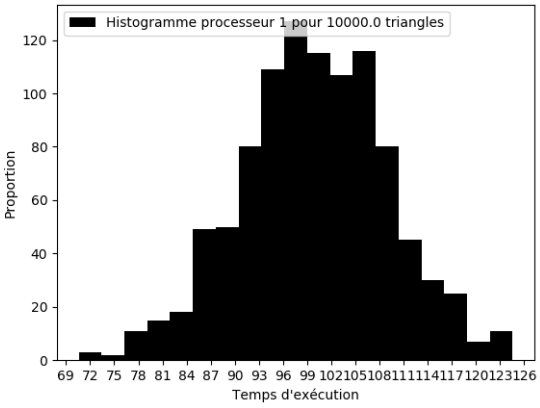

FIG. 5 – *Histogramme pour la carte graphique avec le processeur 1 pour 10.000 triangles*

Ces 9 histogrammes font aisément penser à des gaussiennes. De ce fait on peut dès lors supposer que la loi régissant ces données est une loi normale de paramètre μ et σ.

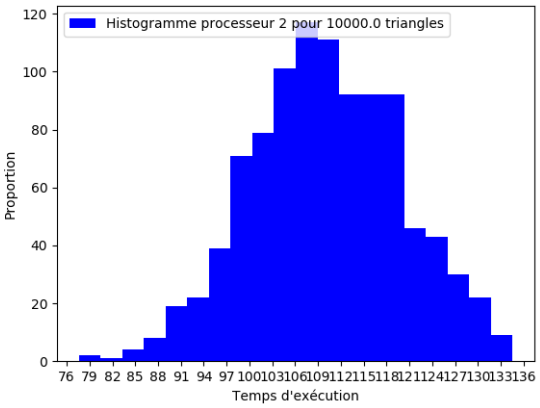

FIG. 6 – *Histogramme pour la carte graphique avec le processeur 2 pour 10.000 triangles*

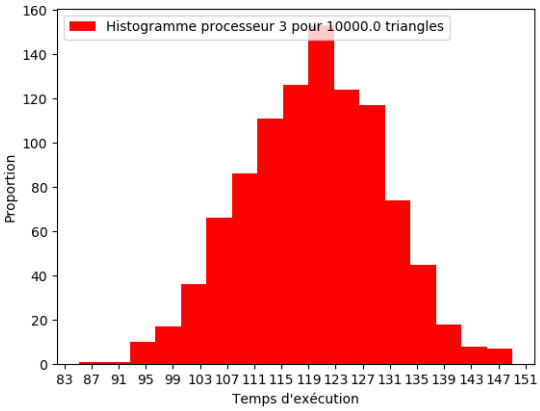

FIG. 7 – *Histogramme pour la carte graphique avec le processeur 3 pour 10.000 triangles*

3 Estimation des paramètres

Dans cette section nous allons estimer les paramètres μ et σ des lois normales pour chaque processeur utilisé et par nombre de triangles. Pour ce faire nous utilisons la méthode du maximum de vraissemblance. Les calculs des estimateurs sont les suivants :

$$\mu = \frac{1}{N} \sum_{i=1}^{N} x_i$$

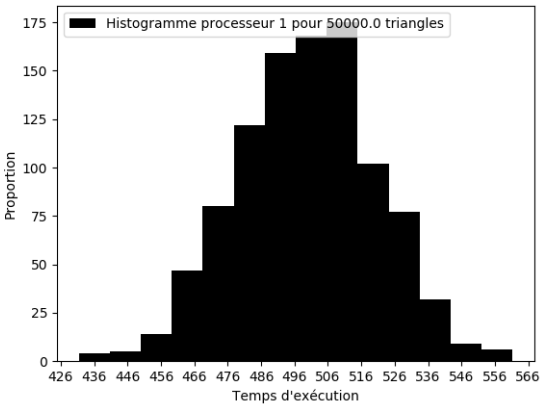

FIG. 8 – *Histogramme pour la carte graphique avec le processeur 1 pour 50.000 triangles*

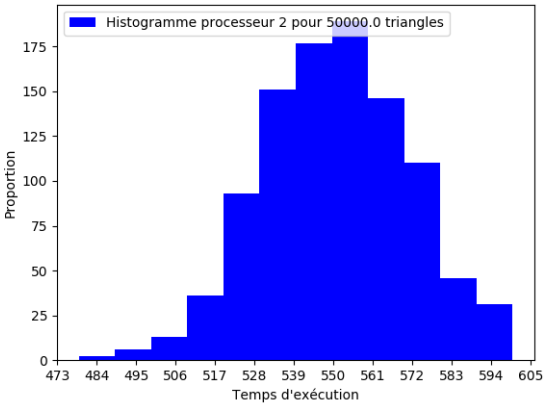

FIG. 9 – *Histogramme pour la carte graphique avec le processeur 2 pour 50.000 triangles*

$$S^2 = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - x)^2$$

Où N est le nombre de données dont nous disposons pour un certain processeur utilisé et un certain nombre de triangles. Dans notre cas ce nombre sera toujours 1000. De plus il est à noter que nous utilisons la moyenne empirique et la variance empirique corrigée qui sont restpectivement les estimateurs sans biais de la moyenne théorique et de la variance théorique.

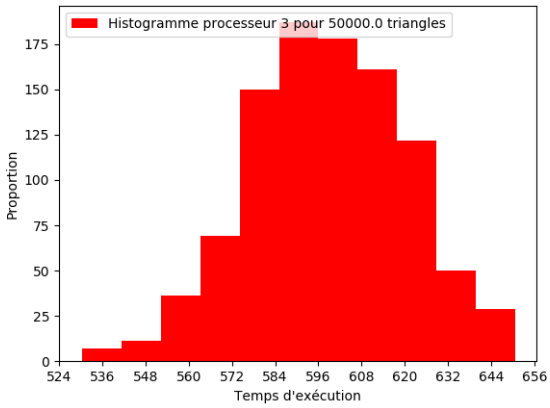

FIG. 10 – *Histogramme pour la carte graphique avec le processeur 3 pour 50.000 triangles*

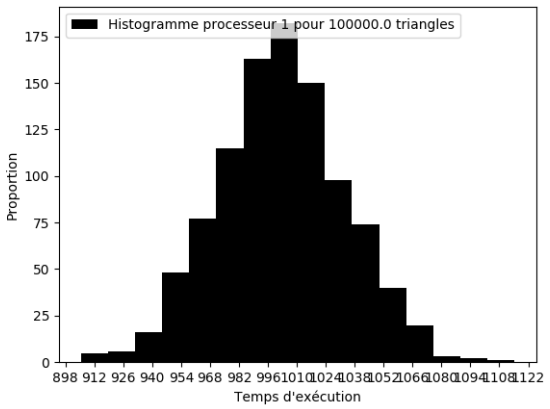

FIG. 11 – *Histogramme pour la carte graphique avec le processeur 1 pour 100.000 triangles*

Le tableau ci-dessous reprend les estimations des paramètres pour les trois processeurs et pour 10, 50 et 100 mille triangles. Tous les estimateurs sont repris dans les fichiers *processorx_para.txt*

Il est à noter que les valeurs trouvées ne sont que des estimations et non des valeurs exactes. Pour obtenir ces valeurs, il faudrait avoir une infinité de données. Voici pourquoi nos résultats ne sont que des estimations, certe proche dû au grand nombre de données à disposition, mais des estimations quand même.

Maintenant que nous disposons d'une estimation des paramètres des lois

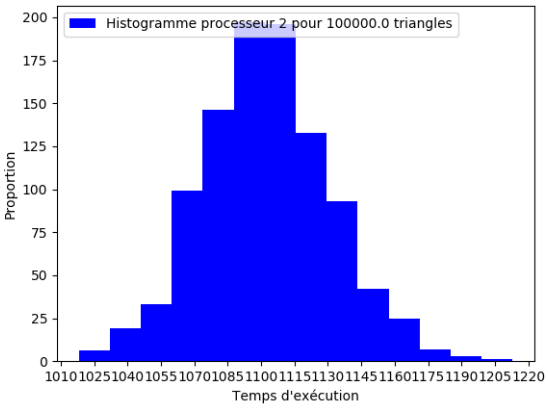

FIG. 12 – *Histogramme pour la carte graphique avec le processeur 2 pour 100.000 triangles*

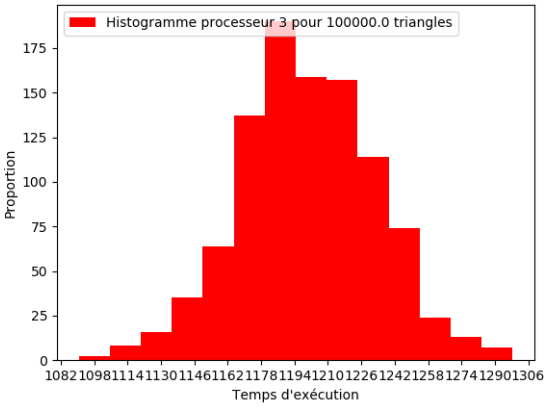

FIG. 13 – *Histogramme pour la carte graphique avec le processeur 3 pour 100.000 triangles*

Nombre de triangles	Processeur 1		Processeur 2		Processeur 3	
	μ	σ^2	μ	σ^2	μ	σ^2
10 000	99.9287	94.4632	110.2842	107.9117	119.7825	119.67235
50 000	499.45985	495.6565	551.20395	499.7602	599.06078	574.7486
100 000	1001.4492	1070.3130	1100.7769	966.81737	1199.80877	1245.2247

TAB. 1 – *Caption*

normales, il est nécessaire de s'assurer que ces lois sont bien représentatrices des données.

3.1 Test de Kolmogorov-Smirnov

Dans cette section nous allons nous assurer que les paramètres calculés précédement des lois normales permettent à ces lois d'être représentatrices des données. Pour ce faire nous allons utiliser le test de Kolomogorov-Smirnov car la loi théorique que nous avons supposée être représentatrice des données, c'est-à-dire la loi normale, est une loi théorique continue. De ce fait, le test de Kolmogorov-Smirnov est plus pertinant que celui du Khi-deux.

Le principe d'un test de Kolmogorov-Smirnov consiste à calculer la distance entre deux fonctions : la fonction théorique et la fonction de répartition empirique. Cette distance est définie tel que

$$d(F_n,F) = \underset{x \in R}{sup} |F_n(x) - F(x)|$$

. Pour que le test soit concluant, il faut que cette distance soit au maximum égale à un seuil défini comme étant le fractile compris dans la table de Kolmogorov-Smirnov. Dans notre cas, puisque nous disposons de 1000 données par test effectué, la table nous donne comme fractile :

- $1.36/\sqrt{1000}$ si on prend un α de 5%.
- $1.63/\sqrt{1000}$ si on pren un α de 1%.

Passons maintenant à la définition des fonctions à utiliser. La fonction théorique est la fonction correspondante à la loi normale. Ceci nous amène à devoir utiliser la relation suivant :

$$f(x) = \int_{-\infty}^{x} \frac{1}{\sigma * \sqrt{2\pi}} * \exp \frac{-(x-\mu)^2}{\sigma}$$

. Quant à la fonction de répartition empirique nous avons :

$$F_n(x) = \begin{cases} 0 & \text{si } x < x_1 \\ \frac{1}{N} & \text{si } x_1 \leq x < x_2 \\ \vdots & \\ \frac{N-1}{N} & \text{si } x_{N-1} \leq x < x_N \\ 1 & \text{si } x_N \leq x \end{cases}$$

A noter que les données ont été préalablement triées par ordre croissant avant de construire la fonction de répartition empirique.

Effectuons maintenant ce test pour les données mises à disposition et les estimateurs des paramètres calculés en amont pour chaque processeur utilisé et nombre de triangles. Ces tests sont réalisés grâce aux fonctions :

- *gauss* qui calcule les valeurs de la fonction théorique

– *repartition* qui calcule la fonciton de répartition des données
– *dist* qui calcul la distance entre les deux fonctions
– *kolmogorov* qui permet de vérifier la réussite ou non-réussite du test

Pour des raisons de clarification, le tableau suivant reprent seulement les tests des 3 processeurs pour 10, 50 et 100 mille triangles. Tous les autres tests sont repris dans les fichiers *processeurX_test.txt*.

Nombres de triangles	processeur 1			processeur 2			processeur 3		
	distance	seuil	réussi	distance	seuil	réussi	distance	seuil	réussi
10 000	0.0177	0.05154	OUI	0.0162	0.05154	OUI	0.0242	0.05154	OUI
50 000	0.0186	0.05154	OUI	0.0137	0.05154	OUI	0.0224	0.05154	OUI
100 000	0.0167	0.05154	OUI	0.0226	0.05154	OUI	0.0246	0.05154	OUI

TAB. 2 – *Tableau des résultats du test de Kolomogorov-Smirnov*

Au total 30 tests ont été effectués, 10 par processeur utilisé. Sur ces 30 tests tous se sont déroulés avec succés. Par conséquent nous pouvons affirmer sans aucun doute que la loi normale, avec les paramètres que nous avons estimés, est bel et bien représentatrice des données mises à disposition.

4 Régression linéaire

A ce stade-ci de la recherche, nous savons que chaque paquet de nombre de triangles pour un processeur donné est représentable par une gaussienne. Avant de passer à la régression linéaire, nous allons comparer paquet par paquet les 3 processeurs sur un même schéma afin d'essayer de déjà établir des points communs et différences.

Pour ce faire, nous utilisons la fonction *show_gauss* du fichier *utils.py*. Comme d'habitude une explication plus détaillée de son utilisation est fournie en annexe. Encore une fois, pour des raisons de clareté du document, seules les gaussiennes pour 10, 50 et 100 mille triangles sont tacées ici. Les autres se trouvant en annexes.

Passons à l'observation de ces gaussiennes.

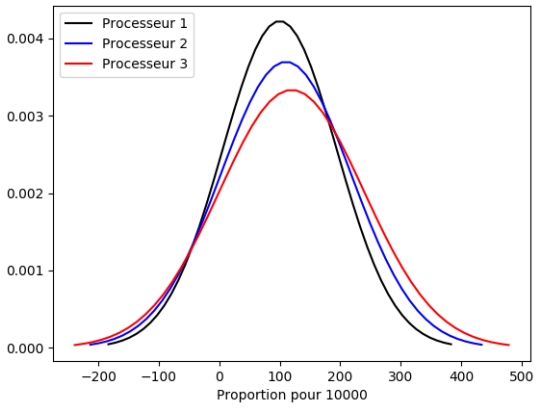

FIG. 14 – *Gaussiennes des lois normales pour 10.000 triangles*

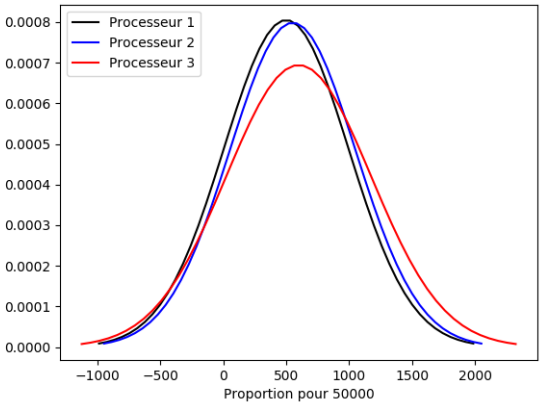

FIG. 15 – *Gaussiennes des lois normales pour 50.000 triangles*

A noter que le jeu de couleur est identique à celui utilisé précédement afin de pouvoir lire plus facilement les schémas et pouvoir en ressortir des observations plus claires.
En ce qui concerne les observations il parait évident que les trois lois normales n'ont pas l'air de se comporter de la même manière sur tous les nombres de triangles.

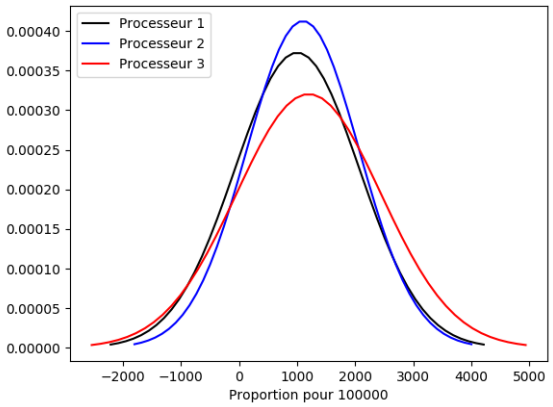

FIG. 16 – *Gaussiennes des lois normales pour 100.000 triangles*

4.1 Egalité des variances

Le test d'égalité des variances consiste à vérifier que les variances de deux gaussiennes sont identiques à un facteur près. Dans notre cas, puisque nous avont trois gaussiennes, nous testons l'égalité des moyennes entre la gaussienne du processeur 1 et 2 puis entre celle des processeurs 1 et 3 et finalement entre celle des processeurs 2 et 3. Ainsi, nous verrons quelles gaussiennes ont des variances très proches.

Afin de tester l'hypothèse H_0 selon laquelle les variances sont égales, il faut calculer :

$$\begin{cases} \frac{S_1^2}{S_2^2} \leq F_{\alpha/2} \text{ si } S_1^2 \leq S_2^2 \\ \frac{S_2^2}{S_1^2} \leq F_{\alpha/2} \text{ si } S_2^2 \leq S_1^2 \end{cases}$$

$F_{\alpha/2}$ et $F'_{\alpha/2}$ sont les fractiles de la loi de Fisher-Snedecor avec respectivement (n_1 - 1, n_2 - 1) et (n_2 - 1, n_1 - 1) degrés de libertés où n_1 est le nombre de données pour la première gaussienne et n_2 le nombre de données dans la seconde.

Dans notre cas nous prendrons un α de 5% ce qui implique que nous trouverons le résultat dans la table correspondante à un α de 2.5%. De plus nous savons que tous nos paquets de données ont une taille de 1000 données. Ceci nous amène au fait que $F_{\alpha/2}$ et $F'_{\alpha/2}$ car n_1 - 1 = n_2 - 1 = 999. Malheureusement la table mise à disposition dans le cours ne comprend pas le fractile pour (999, 999) degrés de libertés. Mais on sait que ce fractile est compris entre 1.16 et 1.11 qui sont respectivement les fractiles pour (500, 1000) et (2000, 1000). Ceci n'est clairement pas une donnée précise. Pour palier à ce problème, nous allons utiliser la table disponible à l'adresse `https://fr.wikipedia.org/wiki/Loi_de_Fisher` qui nous donne $F_{\alpha/2} = F'_{\alpha/2} = 1{,}11$. Au final, cette valeur est identique au

fractile de (2000, 1000) mais cette fois-ci nous somme certain de la valeur. Pour être le plus précis possible nous devrions prendre la valeur pour (999, 999) mais puisque cette valeur n'est pas reprise dans la table et que 999 est assez proche de 1000 nous pouvons prendre cette valeur.

Comme pour les points précédents seuls les résultats pour un nombre de triangles de 10, 50 et 100 mille sont repris. Tous les résultats étant disponibles dans les fichiers *processorXY_test_variances.txt*.

Nombre de triangles	Processeur A	Processeur B	Valeur	Réussi?
10 000	processeur 1	processeur 2	1.1423	NON
50 000	processeur 1	processeur 2	1.0082	OUI
100 000	processeur 1	processeur 2	1.107	OUI
10 000	processeur 1	processeur 3	1.2668	NON
50 000	processeur 1	processeur 3	1.1595	NON
100 000	processeur 1	processeur 3	1.1634	NON
10 000	processeur 2	processeur 3	1.1089	OUI
50 000	processeur 2	processeur 3	1.15	NON
100 000	processeur 2	processeur 3	1.2879	NON

TAB. 3 – *Tableau des résultats du test de Kolomogorov-Smirnov*

Pour chaque comparaison entre deux processeurs, nous effectuons 10 tests, un par nombre de triangles. Une fois tous ces tests réalisés, nous avons obtenus les pourcentages de réussite suivants :

- 80% de réussite entre les processeurs 1 et 2.
- 40% de réussite entre les processeurs 2 et 3.
- 10% de réussite entre les processeurs 1 et 3.

Nous pouvons en conclure que l'égalité des variances est seulement vérifiée entre les processeurs 1 et 2. Ceci signifie que ces deux processeurs ont des variances très proches. Quant au processeur 3, il possède une variance extrèmement différente à celle des deux autres.

Passons maintenant au test sur les moyennes.

4.2 Egalité des moyennes

Grâce au point précédent, nous savons que les variances sont égales seulement entre les deux premiers processeurs. Néanmoins nous ne les connaissons pas. Nous ne les avons qu'estimées. Par conséquent nous devons réaliser le test suivant :

$$\left| \frac{\overline{x} - \overline{y}}{\sqrt{\frac{(n_1-1)*S_1^2+(n_2-1)*S_2^2}{n_1+n_2-2}}} * \frac{1}{\sqrt{\frac{1}{n_1}+\frac{1}{n_2}}} \right| < S_{\alpha/2}$$

avec comme caractéristiques que

- $\overline{x}$ et $\overline{y}$ sont respectivement les moyennes de la première et de la seconde gaussienne.

– n_1 et n_2 sont les tailles des échantillons.

– S_1^2 et S_2^2 sont les variances des deux lois normales.

Quant à $S_{\alpha/2}$, c'est le fractile de la loi de Student avec n_1 + n_2 - 2 degrés de libertés. Dans notre cas, ce degré de liberté vaut 1000 + 1000 - 2 = 1998. Prenons un α de 5%. Notre seuil à ne pas dépasser est donc de 1.96 puisque comme nous avons 1998 degrés de libertés, ce nombre étant très grand, nous devons prendre les valeurs de la table pour l'infini. Ce qui nous mène à un seuil de 1.96.

Encore une fois nous avons effectués 10 tests pour chaque comparaison entre deux processeurs et seuls les résultats pour 10, 50 et 100 mille sont repris dans le tableau suivant. Les autres résultats se trouvent dans les fichiers *processeurXY_test_moyennes.txt*

Nombre de triangles	Processeur A	Processeur B	Valeur	Réussi?
10 000	processeur 1	processeur 2	23.0191	NON
50 000	processeur 1	processeur 2	51.863	NON
100 000	processeur 1	processeur 2	69.5922	NON
10 000	processeur 1	processeur 3	42.9039	NON
50 000	processeur 1	processeur 3	96.2696	NON
100 000	processeur 1	processeur 3	130.3548	NON
10 000	processeur 2	processeur 3	19.9101	NON
50 000	processeur 2	processeur 3	46.1677	NON
100 000	processeur 2	processeur 3	66.5852	NON

TAB. 4 – *Tableau des résultats du test de Kolomogorov-Smirnov*

Puisque aucun résultat n'est correct, nous en déduisons que les lois normales ont des moyennes très différentes.

4.3 Conclusion sur les deux tests

Les deux tests réalisés précédemment nous confirment que les populations de données ne se comportent pas de la même manière. Par conséquent nous savons déjà qu'un des trois processeurs est meilleur que les deux autres. De plus on sait que le processeur 3 s'écarte rapidement des deux autres puisque les tests sont négatifs avec celui-ci. Quant aux deux autres, ils sont plus proches mais finissent aussi par différer.

Passons sans plus attendre à la régression linéaire elle-même afin d'observer les différences entre les trois droites.

4.4 Régression

Grâce aux tests des égalités des variances et des moyennes, nous savons déjà que les droites vont diverger. Puisque nous cherchons le processeur le plus performant pour notre carte graphique nous allons prendre le processeur avec la

pente la plus faible. Cette faible pente signifirait que le temps d'exécution croit moins vite en fonction du nombre de triangles que les deux autres.

4.5 Calcul de la régression

Nous allons essayer de voir s'il existe une relation linéaire entre le nombre de triangles tracés et le temps d'exécution de la carte graphique en fonction du processeur utilisé. Nous devons donc chercher la droite f(x) = α * x + β où α et β sont deux paramètres à déterminer.

Ces deux paramètres sont tels que la fonction

$$S(\alpha,\beta) = \sum_{i=1}^{n} \Big(\frac{\overline{y}_i - \alpha * x_i + \beta}{\sigma_i}\Big)^2 = \sum_{i=1}^{n} w_i * (\overline{y}_i - \alpha * x_i + \beta)^2$$

se trouve minimisée. Ceci nous conduit à résoudre :

$$\begin{pmatrix} \sum_{i=1}^{n} w_i * w_i^2 & \sum_{i=1}^{n} w_i * x_i \\ \sum_{i=1}^{n} w_i * x_i & \sum_{i=1}^{n} w_i \end{pmatrix} \cdot \begin{pmatrix} \hat{\alpha} \\ \hat{\beta} \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^{n} w_i * \overline{y}_i * x_i \\ \sum_{i=1}^{n} w_i * \overline{y}_i \end{pmatrix}$$

Le graphique suivant reprent les droites des régressions calculées pour chaque processeur.

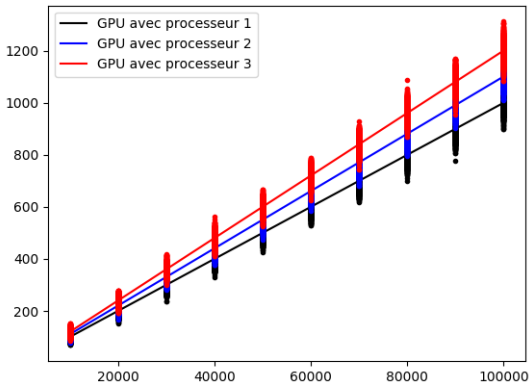

FIG. 17 – *Schéma de la régression des processeurs*

Nous pouvons observer que l'hypothèse que nous avons émise au début selon laquelle le processeur 1 est le plus adapté se vérifie. En effet, en tout point la droite de la régression de ce processeur se situe plus bas que les deux autres. Ce qui signifie que pour un nombre de triangles donnés la carte graphique mettra moins de temps si nous utilisons le premier processeur. Passons maintenant au coëfficient de détermination R^2. Ce coëfficient permet de vérifier que la régression que nous venons de calculer pour chaque processeur explique correctement la dispersion des données. Nous devons donc calculer

$$R^2 = \frac{\sum_{i=1}^{n} w_i * (\tilde{y} - \overline{y})^2}{\sum_{i=1}^{n} w_i * (\overline{y}_i - \overline{y})^2}$$

. Plus cette valeur est proche de 1 meilleure est la régression. Pour ce faire nous allons calculer trois autres paramètres :

- variance total des données ou SST : $\sum_{i=1}^{n} w_i * (\overline{y}_i - \overline{y})^2$
- variance expliquée par la régression ou SSE : $\sum_{i=1}^{n} w_i * (\tilde{y} - \overline{y})^2$
- somme des carrés des résidus SSR : $\sum_{i=1}^{n} w_i * (\overline{y} - \tilde{y})^2$

Puisque nous avons comme relation SST = SSE + SSR, nous pouvons déjà d'une part vérifier nos calculs afin de s'assurer qu'ils soient corrects et ensuite nous pouvons calculer notre coefficient comme étant :

$$\begin{cases} R^2 = \frac{\text{SSE}}{\text{SST}} \\ R^2 = 1 - \frac{\text{SSR}}{\text{SST}} \end{cases}$$

A noter que la variable $\overline{y}$ équivaut à la somme des y moyens. Autrement dit $\overline{y} = \sum_{i=1}^{n} \overline{y}_i$. Quant à la variable $\tilde{y}_i$, celle-ci suit la relation $\tilde{y}_i = \hat{\alpha} * x_i + \hat{\beta}$. Nous obtenons les résultats suivants :

Processeur	R^2
Processeur 1	0.99
Processeur 2	0.99
Processeur 3	0.99

TAB. 5 – *Table des coëfficients de corrélation*

Ces données étant extrêmement proche de 1, nous pouvons amplement nous fier à nos régressions afin d'expliquer le comportement de chaque processeur.

5 Bootstrap

Passons maintenant à la méthode du *bootstrap* afin de s'assurer que nos observations sur les régressions soient bonnes. Nous devrions obtenir les mêmes conclusions. Avec la méthode de la régression linéaire, nous avons obtenu les estimateurs pour $\hat{\alpha}$ et $\hat{\beta}$. Cependant, puisque les données ayant permises de les obtenir sont dues à la réalisation particulière d'échantillons aléatoires, les résultats obtenus auraient été différents avec une autre réalisation de tests même si le même nombre de mesures aurait été pris. Par conséquent $\hat{\alpha}$ et $\hat{\beta}$ sont deux variables aléatoires. La méthode du *bootstrap* nous permet d'obtenir les meilleurs estimateurs possibles pour ces deux variables.

Pour réaliser cette méthode nous commençons par calculer les résidus de la première régression $\hat{\epsilon}_i = \overline{y}_i - (\hat{\alpha} * x_i + \hat{\beta})$. Nous obtenons ainsi n résidus, soit dans notre cas 10 résidus. Ceci fait, nous effectuons les trois étapes suivantes un nombre arbitraire de fois B. Premièrement, il faut définir une nouvelle population ϵ_i^b grâce à un tirage avec remise des $\hat{\epsilon}_i$. Cette nouvelle population possède aussi n élément. La seconde étape consite à calculer les variables dépendantes $y_i^b = \hat{\alpha} * x_i + \hat{\beta} + \epsilon_i^b$. Et finalement, nous calculons les paramètres optimaux $\hat{\alpha}^b$ et $\hat{\beta}^b$. Une fois ce cycle terminé, nous obtenons enfin nos paramètres optimaux grâce aux relations :

$$\begin{cases} \overline{\hat{\alpha}} = \frac{1}{B} \sum_{b=1}^{B} \hat{\alpha}^b \\ \overline{\hat{\beta}} = \frac{1}{B} \sum_{b=1}^{B} \hat{\beta}^b \end{cases}$$

Observons maintenant graphique les résultats.

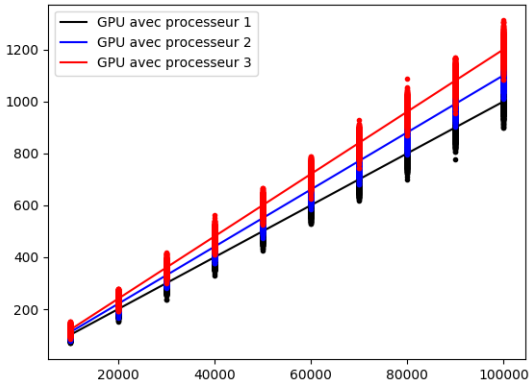

FIG. 18 – *Schéma du bootstrap*

Il parait évident que ce schéma ressemble fortement à celui de la régression linéaire. Par conséquent, nous sommes certains que notre régression était bonne.

6 Conclusion

En conclusion, nous avions au départ des données brutes représentant le temps d'exécution d'une carte graphique pour un certain nombre de triangles et un certain processeur utilisé. Le but étant de trouver le meilleur processeur à utiliser avec cette carte graphique, nous devions trouver le processeur pour lequel le temps d'exécution croit moins vite en fonction du nombre de triangles. Pour ce faire, nous avons d'abord tracé les données pour chaque processeur et nous avons pu remarquer que dans les trois cas elles grandissaient. Ensuite, nous avons tracé les gaussiennes afin de trouver une loi régissant ces données. Une fois cette loi supposées, la loi normale, nous avons estimés ses paramètres. Pour s'assurer que ces lois régissaient bien les données, nous avons effectué un test de Kolmogorov-Smirnov. Ce test effectué, avec succès nous avons tracé les gaussiennes correspondantes. Ceci nous a donné comme hypothèse que les trois jeux de données ne se comportent pas exactement de la même manière. Nous avions plus précisement supposé que le processeur 3 n'agit pas du tout comme les deux autres mais que les deux premiers étaient déjà plus proches. Nous avons vérifié nos hypothèses grâce aux tests de variances et des moyennes qui a bien révélé ce que nous supposions. Nous savions donc que les droites des régressions allaient être différentes. Pour s'en assurer, nous avons calculé la régression des trois processeurs et nous avons bel et bien vu que la droite correspondante au premier processeur était plus basse entre que les deux autres droites. Par conséquent, nous en avons conclu que le meilleur processeur à utliser

entre les trois proposés pour cette carte graphique était le premier. Finalement, nous avons effectué le test du bootstrap pour être certain que notre régression était bonne. Puisque nous avons obtenu les mêmes observations nous avons pu conclure que le premier processeur était bien le meilleur à utiliser.

7 Annexe

7.1 Explication du code

Cette section explique l'utilité générale de chaqu'un des fichiers python contenant dans le dossier *Source*

7.1.1 data_extractor.py

Le fichier *data_extractor* permet de récupérer les données comprises dans un fichier. Le programme les reclasse par système, dans notre cas par processeur, et par unité de valeur mesure, soit le nombre de triangles dans notre cas.

Pour utiliser se fichier il suffit de faire appel à la fonction *extract* pour laquelle le seul argument nécessaire est le nom du fichier. De plus, avant de retourner l'array contenant les données, cette fonction crée un fichier par système contenant ses données.

7.1.2 utils.py

Le fichier *utils* contenient des fonctions utiles pour les autres fichiers. Ce fichier contient entre autre les fonctions permettant de réaliser les tests de Kolmogorov-Smirnov, d'égalité des moyennes et d'égalité des variances. Mais aussi la fonction permet de calculer les moyennes et variances, ...

7.1.3 data_show.py

Ce fichier comprend toutes les fonctions permettant d'afficher les différents graphiques telque les gaussiennes et les histogrammes.

7.1.4 Regression.py

Ce fichier contenant les fonctions spécifiquemet utiles afin d'effectuer la régression linéaire. Nous y retrouvons la fonction permettant de calculer les α et β ainsi que la qualité de la régression avec les SST, SSR et SSE.

7.1.5 bootstrap.py

Ce fichier est le dernier fichier utile afin de réaliser le projet. Il permet d'effectuer les calculs relatifs au bootstrap. De ce fait ce fichier ne contenant que la fonction du même nom.

7.1.6 main.py

Quant à ce fichier ci il n'est pas utiles dans la réalisation du projet à proprement parler mais il montre comment utiliser la plus part des fonctions des autres fichiers avec des explications.

```
import sys
sys.path.append("../")
import Source.regression as regression
import Source.data_show as data_show
import Source.data_extractor as data_extractor
import numpy
import Source.utils as utils
import math
import Source.bootstrap as bootstrap

if __name__ == "__main__":
    """
    La list x ci-dessous n'est utile que pour certaines fonctions
    Elle est cr   e seulement pour limiter le code et se
    concentrer sur les fonctions
    """
    x =
    [10000,20000,30000,40000,50000,60000,70000,80000,90000,100000]

    """
    Avant tout il faut extraire les donn es du fichier datas.dat
    """
    data = data_extractor.extract("datas.dat") # Nous extratons les
     donn es
    """
    Pour faire un afficher toutes les donn es brutes sur un m me
    graphique
    il suffit d'appeller la fonction data_show du module data_show
    """
    data_show.data_show(data)

    """
    Pour afficher les donner des processeurs s paremment il faut
    1. donner les donn es du processeur
    2. le num ro du proesseur (1, 2 ou 3)
    """
    # Avec ce code ci les trois processeurs sont affich s
    successivement
    for i in range(len(data)):
        data_show.show_processor(data[i], i+1)

    """
    Pour afficher tous les histogrammes sur un m me grahique
    il faut appeller la fonction histogram_show en lui donnant le
    nombre de triangles
    """
    # Nous demandons d'afficher les histogrammes pour 10 mille
    triangles
    data_show.histogram_show(data, 100000)

    """
    Pour afficher histogramme par histogramme
    il faut utiliser la fonction show_hist
    """
    for i in range(len(data)):
        # Nous allons parcourir chaque ligne des processeurs
        for j in data[0][:,0]:
```

```
            data_show.show_hist(data[i], j, i+1)

    """
    Pour afficher les gaussiennes correspondantes
    il faut utiliser la fonction show_gauss en lui donnant
    1. une liste des mu
    2. une liste des sigma
    """
    for v in [0,1,2,3,4,5,6,7,8,9]:
        mu = []
        sigma = []
        for i in range(len(data)): # Nous parcourons les trois
    processeurs
            y = data[i][v][1:]
            m = utils.moyennes(y) # Nous calculons la moyenne du
    processeur
            s = utils.variances(y, m) # Nous calculons la variance
    du processeur
            mu.append(m)
            sigma.append(s)
        data_show.show_gauss(mu, sigma, x[v])

    """
    Pour afficher les r sultats du bootstrap il faut
    calculer les y_mean, les alpha et les beta
    """
    d = data[:,:,1:]
    a = []
    b = []
    for i in range(len(data)):
        syst = data[i]
        y_mean, _ = utils.mean_error_calculator(syst)
        alpha, beta = regression.alpha_beta_calculator(syst)
        f = lambda x : alpha * x + beta
        x = numpy.array(x)
        y_hat = f(x)
        n_a, n_b = bootstrap.bootstrap(x,y_hat, y_mean)
        a.append(n_a)
        b.append(n_b)
    data_show.show_line(numpy.array(x), numpy.array(a), numpy.array
    (b), processeur=d)
```

7.2 Images

7.2.1 Histogrammes

Cette section reprend tous les histogrammes pour chaque processeur et chaque quantité de triangles. La première série de 10 images correspont au processeur 1.

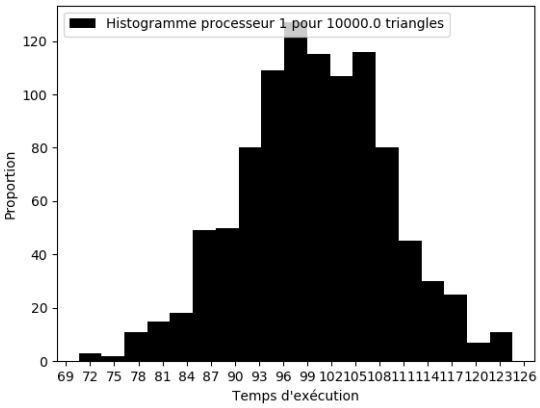

FIG. 19 – *Histogramme pour 10000 triangles pour le processeur 1*

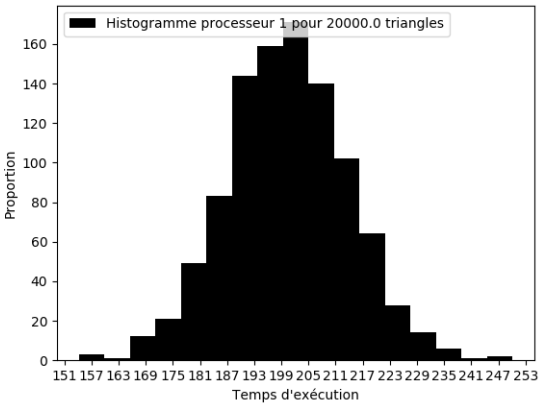

FIG. 20 – *Histogramme pour 20000 triangles pour le processeur 1*

Voici celles pour le processeur 2.

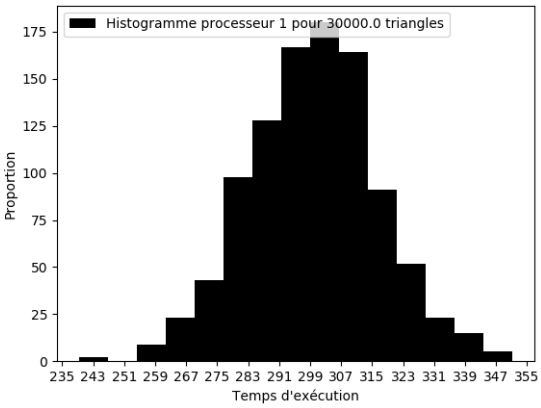

FIG. 21 – *Histogramme pour 30000 triangles pour le processeur 1*

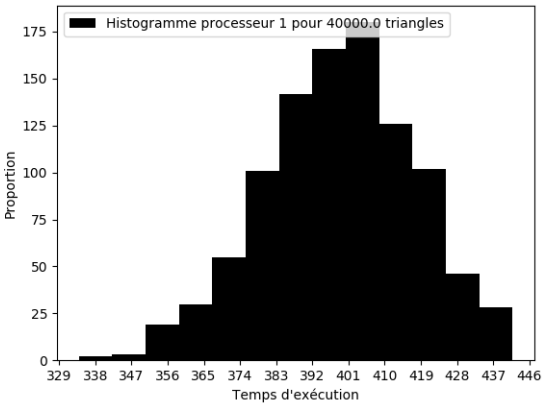

FIG. 22 – *Histogramme pour 40000 triangles pour le processeur 1*

Et finallement les histogrammes pour le troisième processeur.

7.2.2 Gaussiennes

Cette section reprend tous les graphiques des gaussiennes. Sur chaque graphique se trouve les trois gaussienne afin de pouvoir observer leurs différences et points communs.

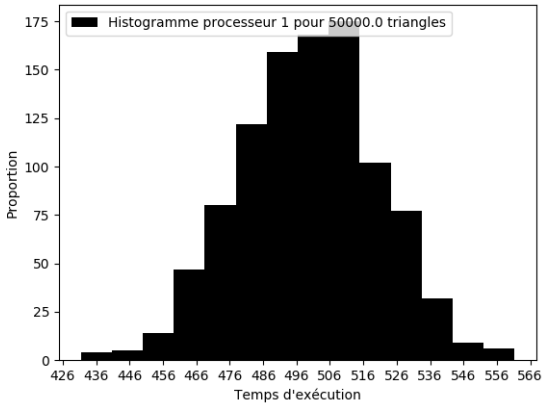

FIG. 23 – *Histogramme pour 50000 triangles pour le processeur 1*

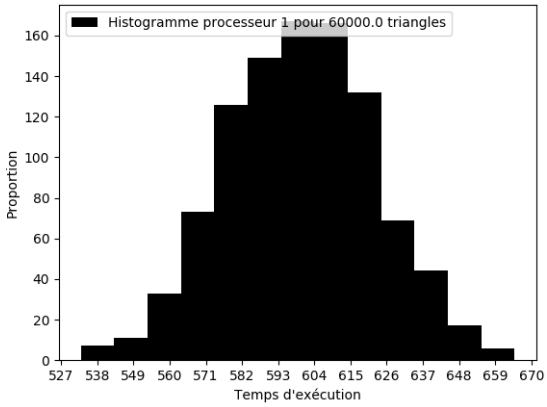

FIG. 24 – *Histogramme pour 60000 triangles pour le processeur 1*

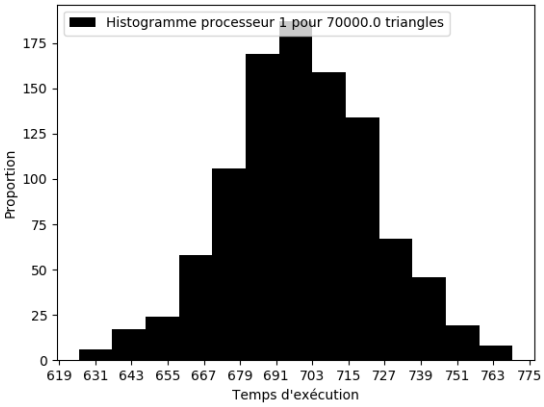

FIG. 25 – *Histogramme pour 70000 triangles pour le processeur 1*

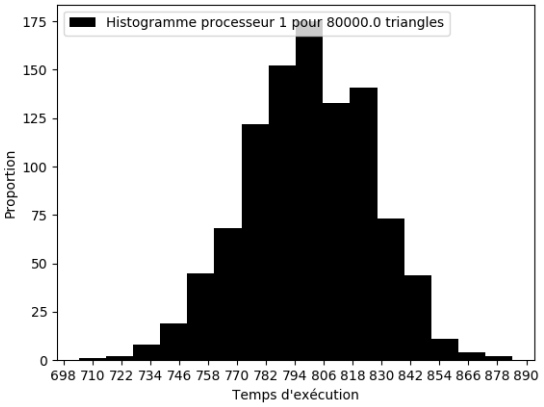

FIG. 26 – *Histogramme pour 80000 triangles pour le processeur 1*

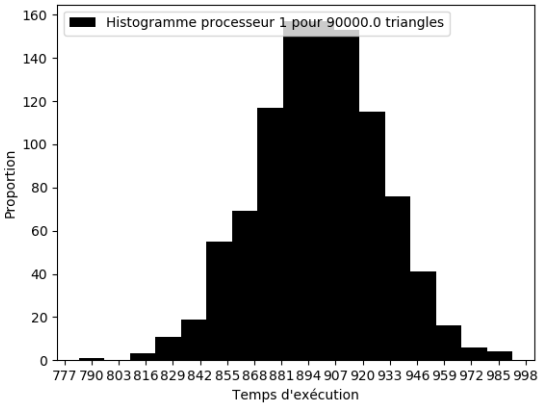

FIG. 27 – *Histogramme pour 90000 triangles pour le processeur 1*

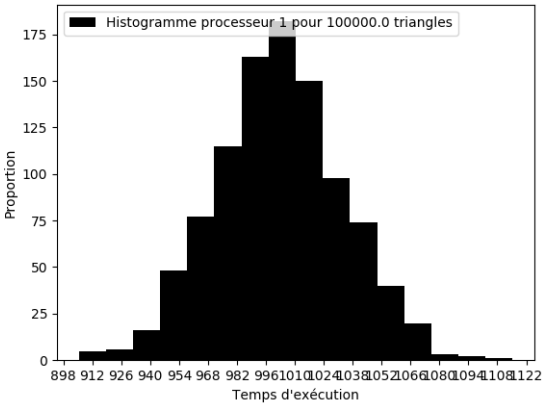

FIG. 28 – *Histogramme pour 100000 triangles pour le processeur 1*

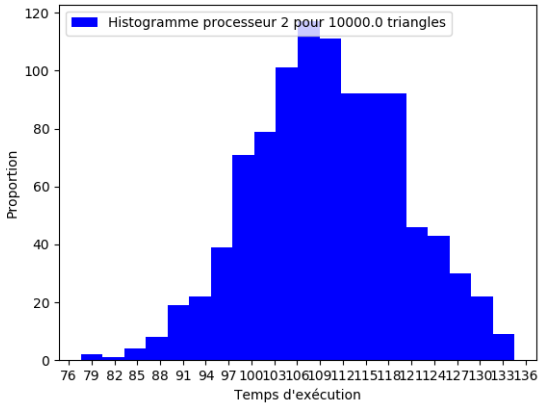

FIG. 29 – *Histogramme pour 10000 triangles pour le processeur 2*

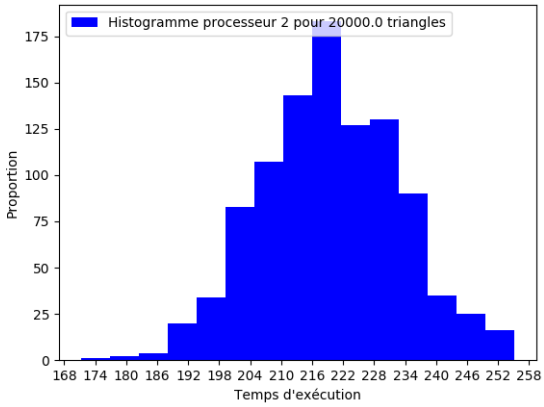

FIG. 30 – *Histogramme pour 20000 triangles pour le processeur 2*

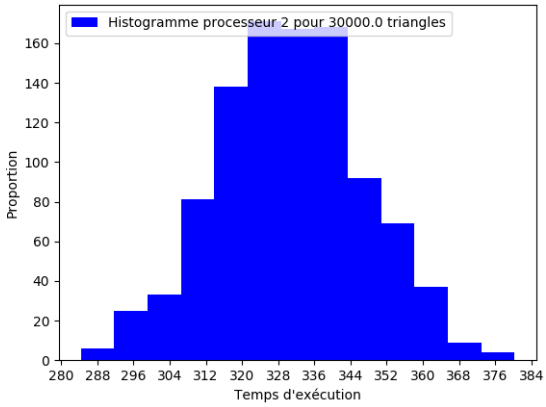

FIG. 31 – *Histogramme pour 30000 triangles pour le processeur 2*

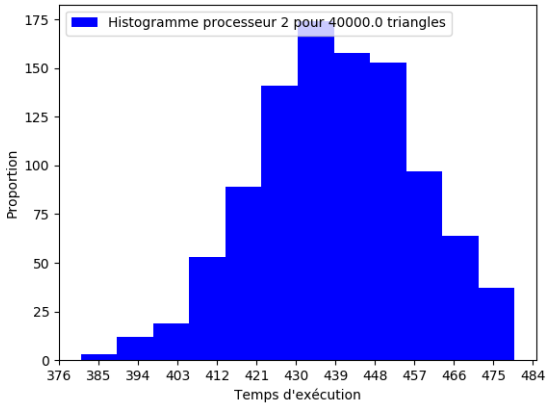

FIG. 32 – *Histogramme pour 40000 triangles pour le processeur 2*

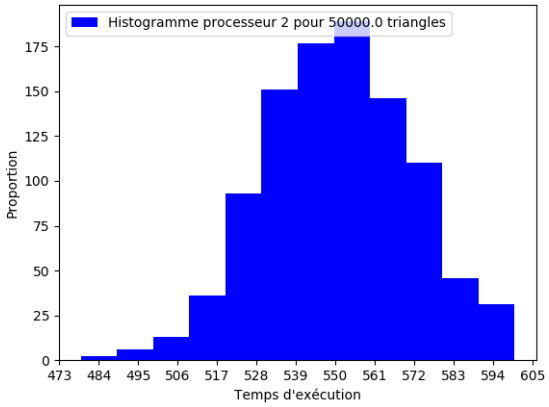

FIG. 33 – *Histogramme pour 50000 triangles pour le processeur 2*

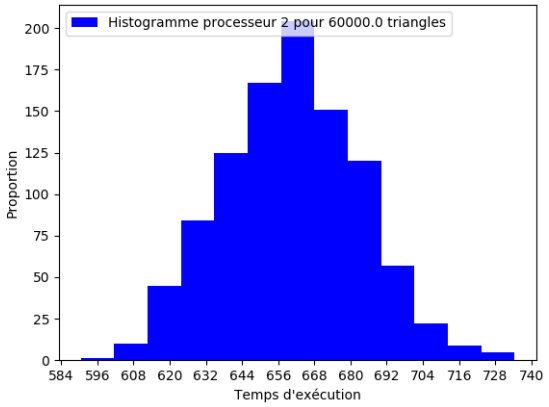

FIG. 34 – *Histogramme pour 60000 triangles pour le processeur 2*

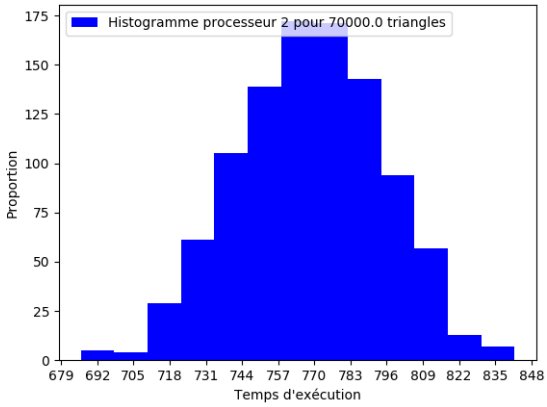

FIG. 35 – *Histogramme pour 70000 triangles pour le processeur 2*

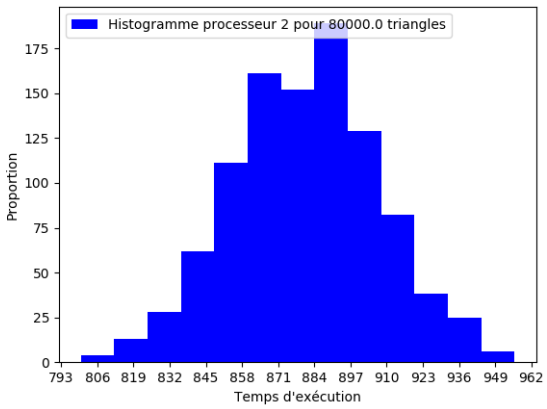

FIG. 36 – *Caption*

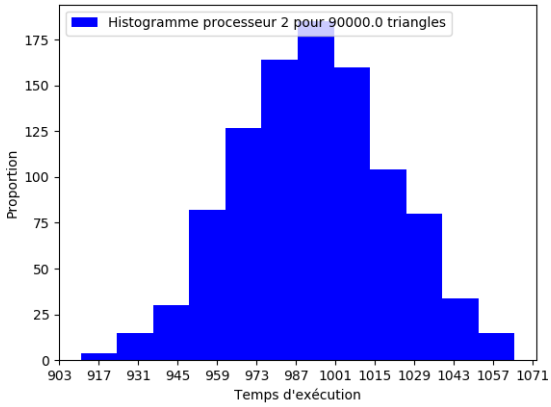

FIG. 37 – *Histogramme pour 90000 triangles pour le processeur 2*

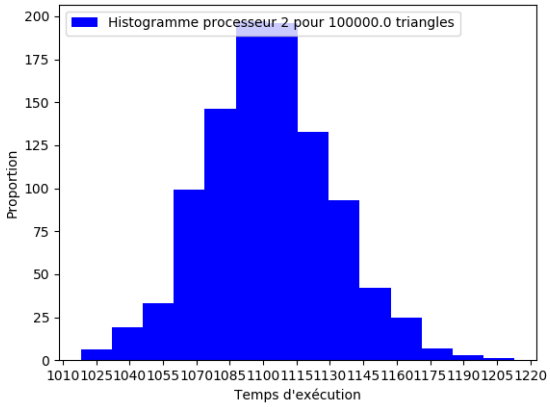

FIG. 38 – *Histogramme pour 100000 triangles pour le processeur 2*

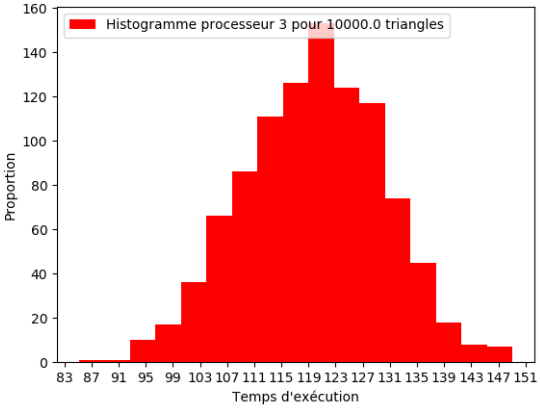

FIG. 39 – *Histogramme pour 10000 triangles pour le processeur 3*

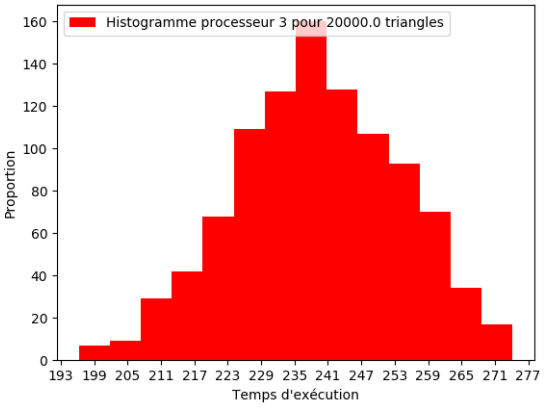

FIG. 40 – *Histogramme pour 20000 triangles pour le processeur 3*

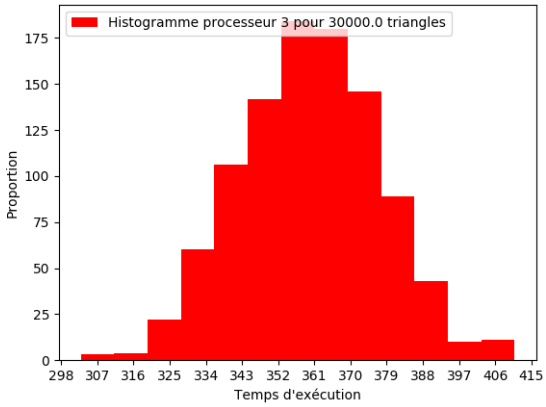

FIG. 41 – *Histogramme pour 30000 triangles pour le processeur 3*

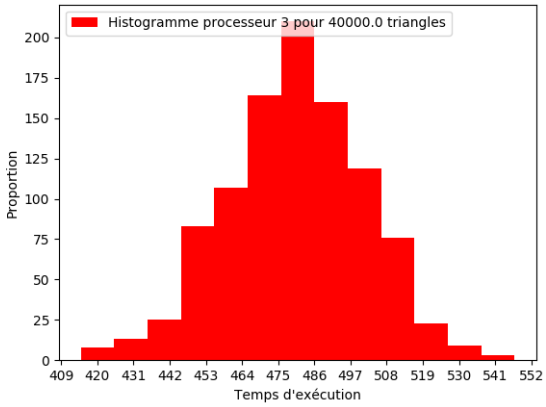

FIG. 42 – *Histogramme pour 40000 triangles pour le processeur 3*

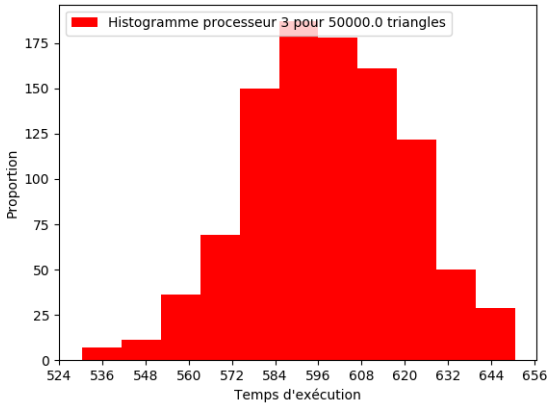

FIG. 43 – *Histogramme pour 50000 triangles pour le processeur 3*

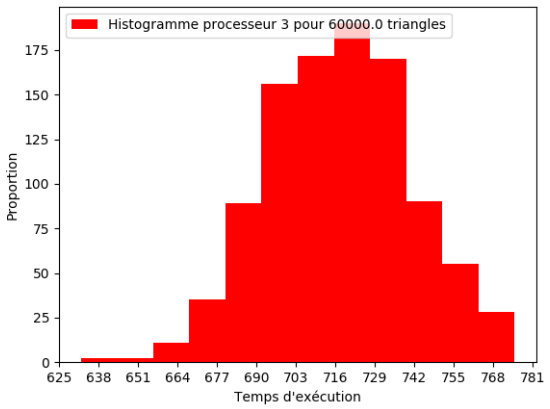

FIG. 44 – *Histogramme pour 60000 triangles pour le processeur 3*

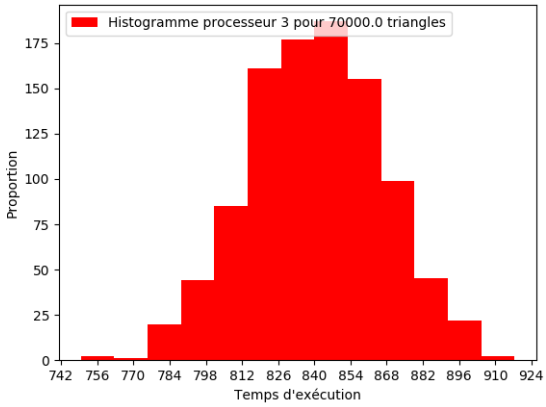

FIG. 45 – *Histogramme pour 70000 triangles pour le processeur 3*

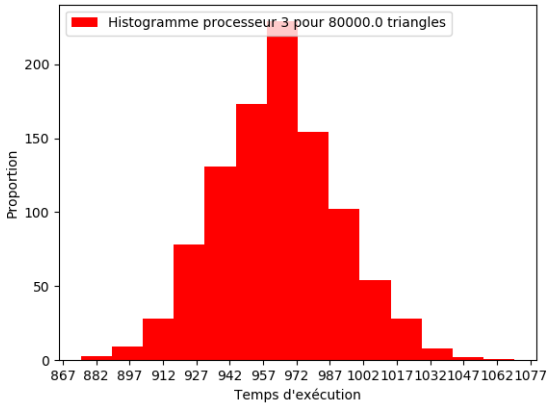

FIG. 46 – *Histogramme pour 80000 triangles pour le processeur 3*

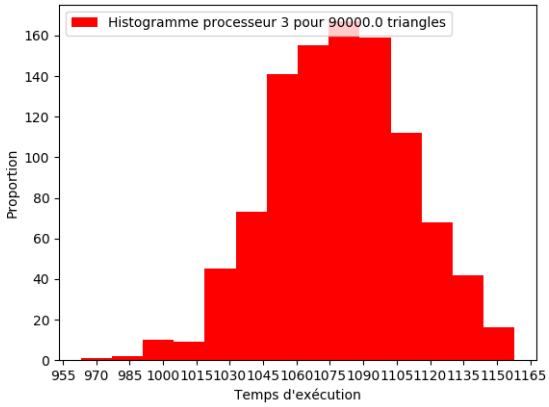

FIG. 47 – *Histogramme pour 90000 triangles pour le processeur 3*

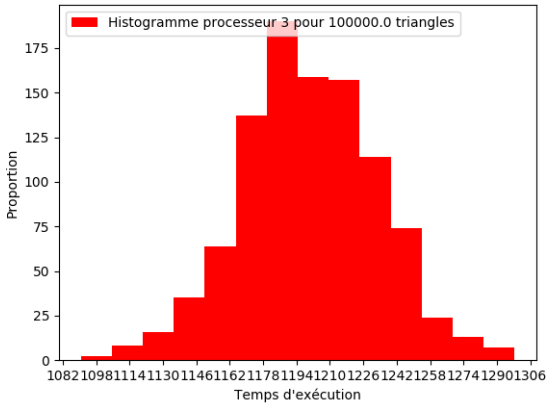

FIG. 48 – *Histogramme pour 100000 triangles pour le processeur 3*

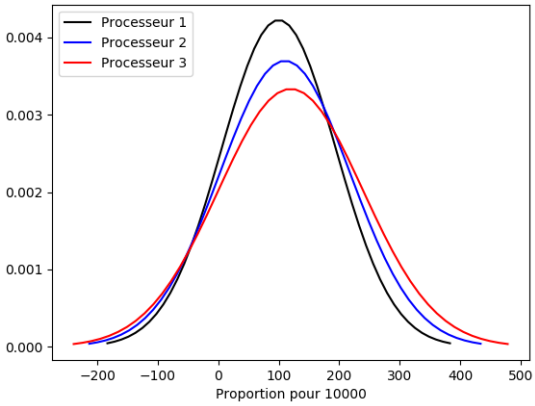

FIG. 49 – *Gaussiennes pour 10.000 triangles*

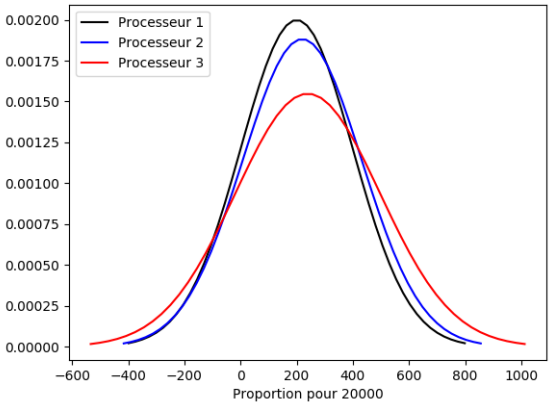

FIG. 50 – *Gaussiennes pour 20.000 triangles*

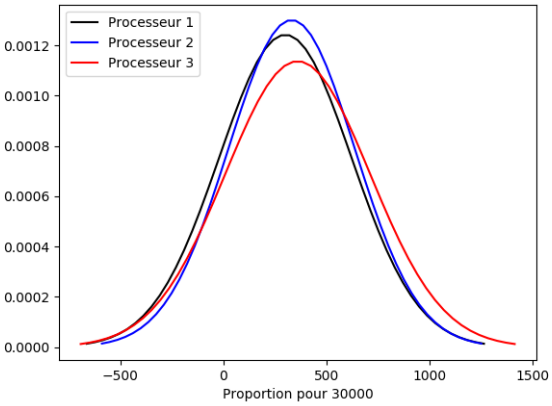

FIG. 51 – *Gaussiennes pour 30.000 triangles*

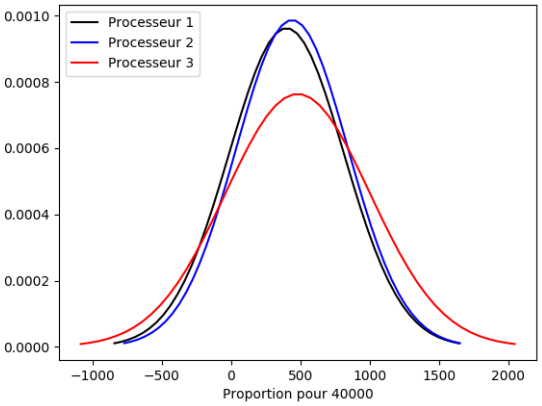

FIG. 52 – *Gaussiennes pour 40.000 triangles*

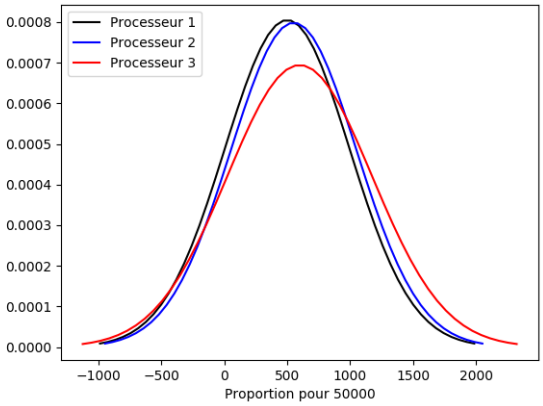

FIG. 53 – *Gaussiennes pour 50.000 triangles*

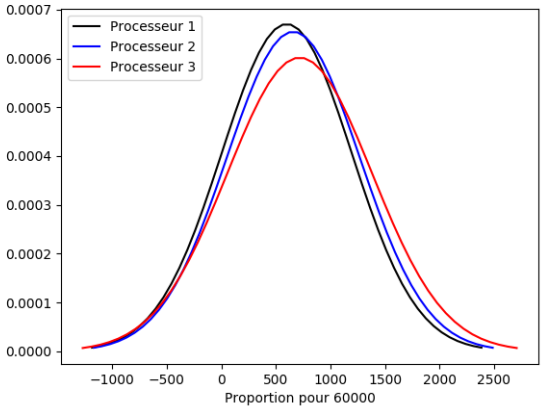

FIG. 54 – *Gaussiennes pour 60.000 triangles*

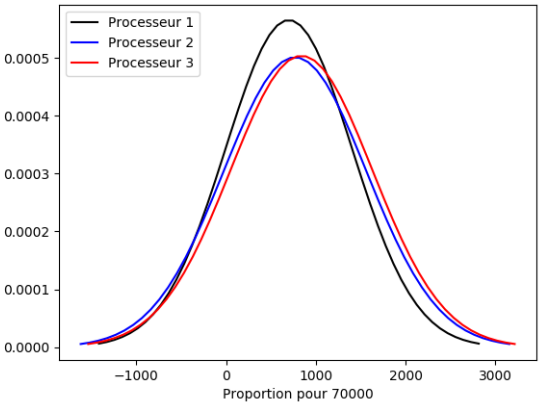

FIG. 55 – *Gaussiennes pour 70.000 triangles*

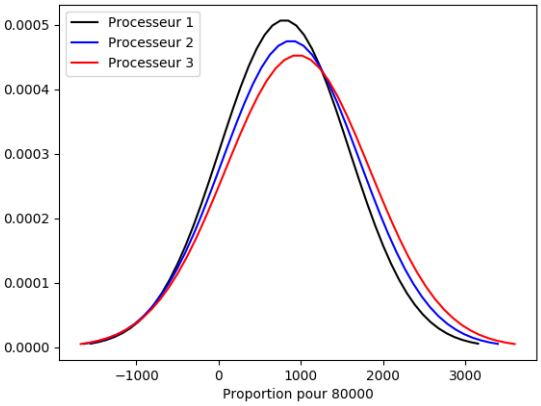

FIG. 56 – *Gaussiennes pour 80.000 triangles*

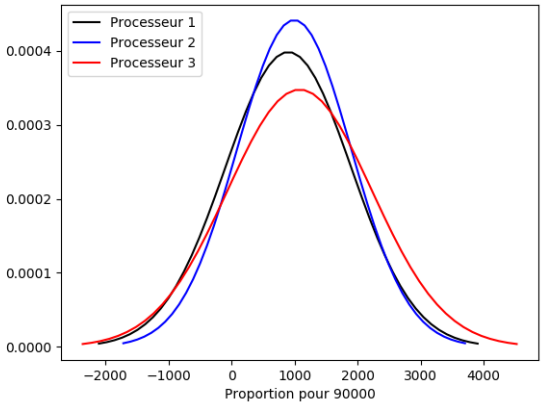

FIG. 57 – *Gaussiennes pour 90.000 triangles*

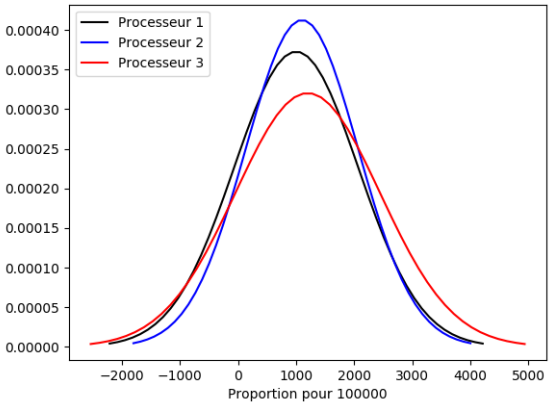

FIG. 58 – *Gaussiennes pour 100.000 triangles*