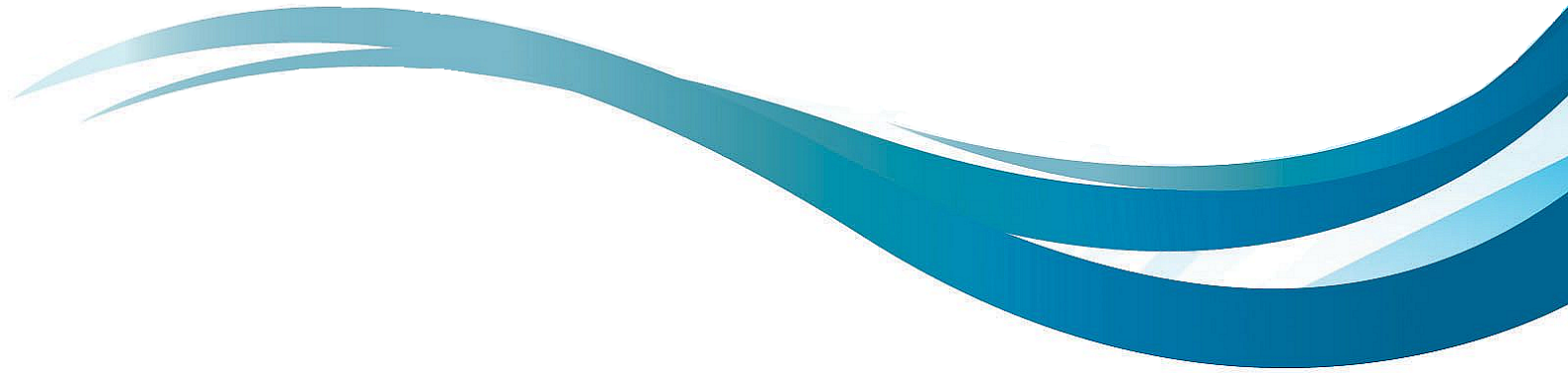
Rapport TP Simulation
Lab #3 - Simu PI & Conf Intervals

BALLEJOS Lilian
ISIMA INP Deuxième année
Spécialité F2
Année Universitaire 2022/2023



Enseignant référant : Mr David HILL
Date du rendu : 31 octobre 2022
Date de la dernière modification : 6 novembre 2022

ISIMA INP
1 rue de la Chebarde - TSA 60125 - CS 60026 - 63178 Aubière CEDEX
Tel : 04 73 40 50 00
Site web : isima.fr

Table des Matières

Introduction

Quelques remarques sur le fichier .c

Les commentaires /**/ sont voués à être supprimés Ils sont ici pour rendre l'exécution du main plus lisible et ne pas biaiser les tirages aléatoires en enchainant des tirages.

La compilation s'effectue via un **Makefile**.

Liste des fichiers fournit pour ce TP :
— mt.c
— mt.h
— makefile
— tp3_ballejos.c
— tp3_ballejos.pdf

J'ai réutilisé lors de ce TP les fonctions "MT" de génération de nombre aléatoire de Matsumoto que vous pouvez retrouver ici.

1 Exercice 1 : Création d'une fonction qui approche la valeur de PI

1.1 Explication

Nous allons tout d'abord commencer par créer une fonction nommée **simPi()** qui approche la valeur de π. Pour cela nous faisons la méthode de Monte-Carlo. La fonction prend en argument n nombre de point à tirer dans un quart de cercle du cercle d'unité et en comptant la proportion présente dans le cercle on arrive à retrouver une valeur approximative de π !

1.2 Résultats

On remarque pour le moment que plus on augmente le nombre de tirage de point plus on approche une meilleure valeur de π !

```
===== Sortie de simPI =====
Pour N = 10000   3.148000
Pour N = 100000  3.144640
Pour N = 1000000         3.144304
Pour N = 10000000        3.141122
Pour N = 100000000       3.141282
Pour N = 1000000000      3.141586
============================
```

FIGURE 1 – Sortie de simPi() avec différents nombre de tirage

On remarque tout de même qu'avec 1 milliard de tirage on arrive seulement à atteindre les 4 premiers décimales de π ! Nous pouvons surement trouver une meilleure méthode !

2 Exercice 2 : Obtention de PI par la moyenne

2.1 Explication

Une autre solution pourrait être d'appeler plusieurs fois la fonction simPi() et de faire la moyenne de toutes les sorties ! Ainsi on a maintenant 2 paramètres qui sont le nombre de point tiré ainsi que le nombre de fois qu'on appelle simPi(). Voyons maintenant si on obtient des résultats plus intéressant.

2.2 Résultats

```
===== Sortie de meanPI avec 10 appels de simPi =====
Pour N = 1000    3.148000 | Erreur relative: 1.002040 | Erreur absolue: -0.006407
Pour N = 1 000 000       3.141254 | Erreur relative: 0.999892 | Erreur absolue: 0.000338
Pour N = 1 000 000 000   3.141564 | Erreur relative: 0.999991 | Erreur absolue: 0.000028
=================================================
```

FIGURE 2 – Sortie de meanPI() avec 10 appels de simPi()

On remarque que l'erreur absolue tend de plus en plus vers 0 alors que l'erreur relative elle tend de plus en plus vers 1 ce qui est très bon signe. Testons maintenant avec 40 appels de simPi()

```
===== Sortie de meanPI avec 40 appels de simPi =====
Pour N = 1000    3.142400 | Erreur relative: 1.000257 | Erreur absolue: -0.000807
Pour N = 1 000 000       3.141189 | Erreur relative: 0.999872 | Erreur absolue: 0.000403
Pour N = 1 000 000 000   3.141586 | Erreur relative: 0.999998 | Erreur absolue: 0.000006
=====================================================
```

FIGURE 3 – Sortie de meanPI() avec 40 appels de simPi()

En plus de remarquer que mon ordinateur souffre de tirer 40 fois 1 milliards de point on peut aussi voir que les erreurs sont de plus proche de 0 et de 1 ce qui implique un bien meilleur pseudo PI approché.

3 Exercice 3 : Intervalle de confiance avec la moyenne de PI

3.1 Explication

Nous allons maintenant maintenant effectuer un intervalle de confiance de notre π simulé en calculant tout d'abord un estimateur non biaisé puis un l'intervalle.

3.2 L'estimateur

J'ai d'abord crée une nouvelle fonction **meanPiWithArray** qui effectue le même comportement que **meanPi** mais stocke aussi dans un tableau en argument toutes les sorties de π ayant servi pour la moyenne. Ces valeurs vont nous servir à calculer l'estimateur à l'aide de cette formule :

$$\overline{X}(n) = \sum_{i=1}^{n} X_i/n \tag{1}$$

$$S^2(n) = \frac{\sum_{i=1}^{n}[X_i - \overline{X}(n)]^2}{n-1} \tag{2}$$

FIGURE 4 – Formule pour calculer l'estimateur

C'est la fonction **EstimateWithoutBiasPi** qui retourne la valeur de l'estimateur.

3.3 Calcul de R

Après avoir calculé l'estimateur de notre pseudo π il faut calculer R et pour cela la fonction **CalculR** entre en jeu.

Cette fonction retourne la valeur de R avec la formule suivante :

$$R = t_{n-1,1-\frac{\alpha}{2}} * \frac{\sqrt{S^2(n)}}{n}$$

FIGURE 5 – Formule pour calculer R

Il faut remarquer que j'ai fourni dans mon code seulement les 30 premières valeurs de $t_{n-1,1-\frac{\alpha}{2}}$. Ainsi il ne sera pas possible de calculer un intervalle avec plus de 30 appels de simPi().

3.4 Calcul de l'intervalle de confiance

Enfin avec la fonction **showConfidenceInterval** il nous suffit d'appeler dans l'ordre la fonction meanPiWithArray puis EstimateWithoutBiasPi et enfin CalculR pour avoir notre intervalle de confiance. J'ai juste à afficher dans le terminal l'intervalle en soustrayant et additionnant R à la valeur du pseudo π calculé.

3.5 Résultat

Voyons maintenant les résultats obtenus

```
Estimateur non biaisé: 0.000003
Pour un nombre d'experience de 10 avec 1000000 points on a:
Intervalle de confiance à 95% : [3.140060 - 3.142511]
```

FIGURE 6 – Sortie de showConfidenceInterval avec 1 million de point et 10 appels de simPi

Avec 1 million de point et 10 appels de simPi on remarque l'intervalle est concluant mais peut être mieux

```
Estimateur non biaisé: 0.000000
Pour un nombre d'experience de 10 avec 1000000000 points on a:
Intervalle de confiance à 95% : [3.141537 - 3.141588]
```

FIGURE 7 – Sortie de showConfidenceInterval avec 1 milliard de point et 10 appels de simPi

Avec 1 milliard de point et 10 appels de simPi le "printf lf" de *stdlib.h* n'est pas suffisant pour afficheur l'estimateur tant il est petit ! L'intervalle est lui bien plus précis !

```
Estimateur non biaisé: 0.000000
Pour un nombre d'experience de 30 avec 1000000000 points on a:
Intervalle de confiance à 95% : [3.141565 - 3.141604]
```

FIGURE 8 – Sortie de showConfidenceInterval avec 1 milliard de point et 30 appels de simPi

Enfin, avec 1 milliard de point et 30 appels de simPi on obtient des résultats moins précis que pour 10 appels ce qui n'est pas un résultat attendu.

Il semble que faire la moyenne avec beaucoup de valeur n'est la meilleure stratégie pour obtenir un meilleur intervalle ce confiance.

Table des figures