Dodeman Léna

UNIVERSITÉ Clermont Auvergne

PROJET INFORMATIQUE
TP3

SIMULATION DE MONTE CARLO & INTERVALLES DE CONFIANCE POUR DES VALEURS DE PI

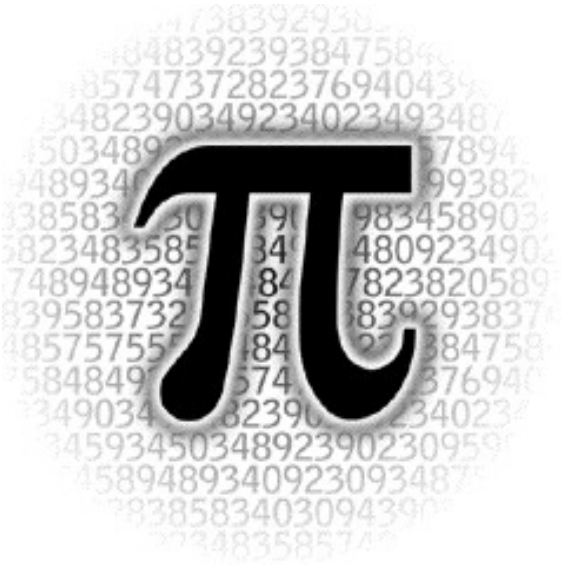

TABLE DES MATIERES

TABLE DES ILLUSTRATIONS

Introduction :

Le Mersen Twister que nous utilisons dans ce TP est un générateur de nombre pseudo-aléatoires. Il a été développé par Makoto Mastumoto et Takuji Nishimura. C'est avec ces méthodes que l'on initialise notre programme. Avec ce générateur on peut réaliser des expériences répétable, pratique pour le débogage et les simulations dans plusieurs domaines notamment des simulations de Monte-Carlo.

Dans ce TP nous allons manipuler une fonction de générations de nombres aléatoires de Matsumoto, le Mersenne Twister, afin d'obtenir des valeurs de π . A travers plusieurs exercices nous allons voir des méthodes différentes pour évaluer la valeur de π. Avec un tirage pseudo-aléatoire de nombres situés entre 0 et 1 nous parviendrons à calculer plusieurs valeurs de π. Pour ce TP nous utilisons la fonction genrand_real1().

Exercice 1 : Calcul de π avec la méthode de Monte Carlo

Dans cet exercice nous allons tirer un point aléatoire qui a pour coordonnées x et y ayant une valeur pseudo aléatoire située entre 0 et 1. Comme on sait que la surface d'un quart de cercle est égale à π/4 alors un calcul nous permet de savoir si le point tiré est situé à l'intérieur de notre quart de cercle ou non. Ce calcul est le suivant : $(x^2 + y^2 < R^2)$, R étant le rayon du cercle, R = 1 dans notre simulation.
Nous tirons N nombres pseudo-aléatoires et nous comptons ceux d'entre eux qui sont à l'intérieur du cercle en comparant leur coordonnées avec le rayon comme dans le calcul ci-dessus. Ce compte est noté M. Ainsi le rapport du nombre de points qui sont à l'intérieur du cercle par le nombre de ceux qui ne le sont pas va nous servir à calculer π :
M / N converge vers π / 4 → 4 x M / N converge vers π
On obtient ainsi une estimation de la valeur de π. Plus on fera de tirages et plus la valeur obtenue se rapprochera de la valeur réelle de π.

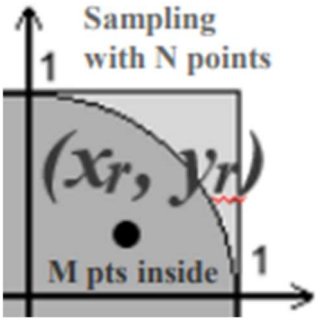

Figure 1 : Illustration d'un point à l'intérieur du disque

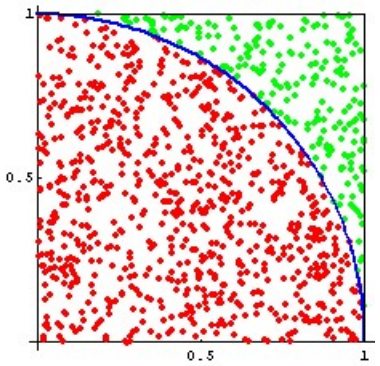

Figure 2 : Répartition de points sur un quart de cercle avec la méthode Monte Carlo

Nous avons programmé une fonction qui permet de réaliser cette simulation :

La fonction simuPiDisk prend en paramètre le nombre de points que l'on veut tirer.
On déclare tout d'abord les variables utiles au tirage :

- xr et yr qui sont les coordonnées de chaque point, leur valeur est obtenue grâce à genrand_real1()

```
double simuPiDisk(long nbOfPoints)
{
        double xr = 0.;
        double yr = 0.;
        double pi = 0.;
        long i;
        long intDisk = 0;
```

- pi car c'est cette valeur que l'ont cherche
- i pour la boucle qui réalise le nombre de tirages voulus
- une variable intDisk qui compte le nombre de points qui se situent à l'intérieur du disque

Nous avons ensuite une boucle qui , à chaque itérations, affecte à xr et yr des coordonnées entre 0 et 1 et vérifie que le point obtenue se situe à l'intérieur du disque ou pas, on les compte si c'est le cas.
Enfin on calcule notre convergence de π avec la formule vue plus haut.

```
	for(i=0 ; i < nbOfPoints ; i++)
	{
		xr = genrand_real1();
		yr = genrand_real1();

		if( (xr * xr + yr * yr) <= 1)
		{
			intDisk++;
		}
	}

	pi = ((double) intDisk / (double) nbOfPoints ) * 4.;

	return pi;
}
```

Nous avons obtenu les valeurs suivantes :

Nombre de points par expérience	Valeurs de π
1000	3,1240
100 000	3,14592
1 000 000	3,14476
100 000 000	3,141245
1 000 000 000	3,141546
10 000 000 000	3,141563

Figure 3 : Tableau de référence des valeurs de pi obtenues

Nous obtenons une précision de 10^{-4} sur π à partir d'un milliard de points tirés.

Nous obtenons le graphique suivant des valeurs de π :

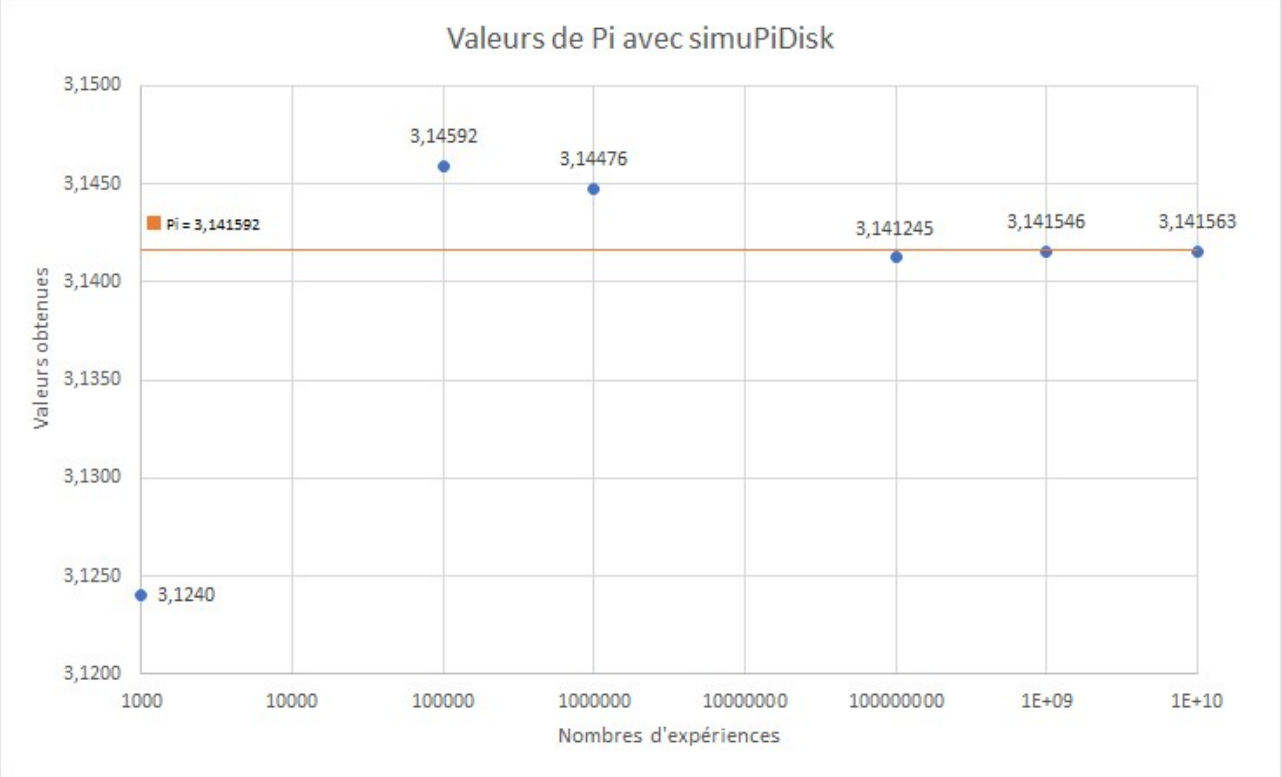

Figure 4 : Valeurs de pi en fonction du nombre de tirages

On observe bien que plus le nombre de points est élevé, plus la valeur de π obtenue converge vers la vraie valeur : π = 3.14159265359…

Exercice 2 : Calcul de π à l'aide d'expériences indépendantes

Dans cet exercice nous allons voir une seconde méthode pour calculer π. Nous allons utiliser la fonction précédente afin d'avoir plusieurs simulations indépendantes de la valeur de π. Toutes ces valeurs de π obtenues seront stockées dans un tableau de grandeur correspondante au nombre d'expériences voulues. Nous pourrons alors faire la moyenne des valeurs du tableau pour calculer notre π.
Pour calculer notre moyenne on fera le calcul suivant :

$\bar{X} = \frac{\Sigma_{i=1}^{n} X_i}{n}$ Avec n qui représente le nombre total d'expériences réalisées.

Nous avons donc fait une fonction pour réaliser cet exercice :

La fonction simuAvgPi prend en paramètre le tableau du nombre d'expériences que l'on veut réaliser ainsi que le nombre de points que l'on veut tirer.
 	
On déclare les variables utiles :
• une variable intermédiaire addPi qui fait la somme de toutes les valeurs tirées de π du tableau
	double simuAvgPi(double tabPi[], long nbOfPoints)
	{
	double addPi = 0.;
	long i;
	
	for(i = 0 ; i < nbExp ; i++)
	{

• i pour la boucle
 	
Après avoir additionné toutes les valeurs, on peut retourner la moyenne obtenue avec le calcul précisé ci-dessus.
	tabPi[i]= simuPiDisk(nbOfPoints);
	addPi += tabPi[i];
	}
	
	return addPi / (double) nbExp;
	}

Pour notre calcul nous avons pris n=30 et 1000 points tirés pour la simulation de la valeur de π.

On peut voir notre résultat sur le graphique suivant :

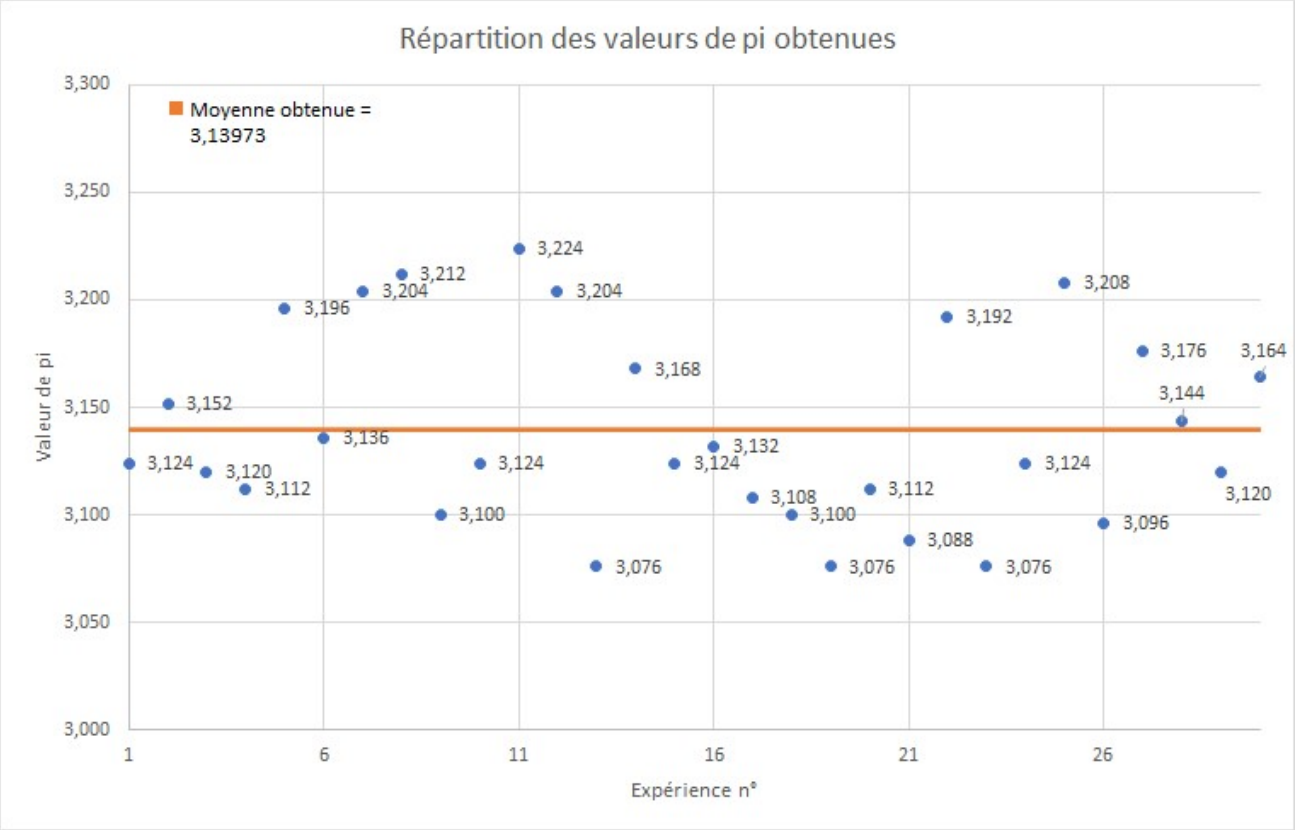

Figure 5 : Représentation des 30 valeurs obtenues de pi ainsi que la moyenne

Nous pouvons observer que nous obtenons une valeur de π = 3,13973333.

Au premier exercice, pour 1000 points nous avons obtenu une valeur moins précise de π qui était de 3,1240. Cette fois en répétant le tirage 30 fois puis en faisant la moyenne, nous avons obtenu une valeur de π plus précise. Cette méthode est donc meilleure que la dernière et c'est normal étant donné que l'on réalise la simulation 30 fois au lieu d'une seule.
Si l'on refait le test avec n=30 mais cette fois ci un nombre de points plus élevé, nous obtiendrons aussi une valeur de π plus proche de la réalité que lors du 1er exercice. Par exemple pour 1 000 000 de points et toujours 30 expériences, nous obtiendrons π = 3,14114 au lieu de π = 3,14476 pour le 1er exercice.

Nous avons donc trouvé ici une méthode un peu plus précise pour calculer π.

Exercice 3 : Intervalle de π avec une intervalle de confiance

Dans cet exercice nous allons calculer une intervalle de confiance de π centrée sur la simulation de la moyenne calculée à l'exercice 2. On doit calculer la marge d'erreur qui dépend de notre nombre d'expérience.
Ici nous avons toujours n = 30, on choisira donc $t_{n-1,1-\alpha/2}$ = 2,042. Cette valeur sera utile pour calculer la marge d'erreur. Nous devons tout d'abord calculer $S^2(n)$ qui est une approximation de la variance, avec la formule suivante :

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

Une fois que cela est fait, on peut calculer le radius avec notre $t_{n-1,1-\alpha/2}$ déterminé plus tôt, et $S^2(n)$:

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

On obtient alors une marge d'erreur qui varie selon le nombre d'expériences réalisées. Il nous suffit de faire notre intervalle de confiance à 95 % : $[\bar{X} - radius, \bar{X} + radius]$

La fonction piComputing prend en paramètres la moyenne calculée au préalable ainsi que le tableau des simulations indépendantes de pi.

On déclare les variables utiles :

- i pour la boucle pour le calcul de $S^2(n)$
- une variable intermédiaire au calcul de $S^2(n)$ sum
- une variable inter pour la bonne réalisation de la boucle

Après la boucle qui fait la somme des valeurs utiles à calculer le nominateur de $S^2(n)$, on obtient la valeur de $S^2(n)$ avec le calcul précisé ci-dessus, puis la valeur du radius.

```
double piComputing(double avgPi, double
xPi[nbExp])
{
        long i;
        double inter = 0.;
        double sum = 0.;
        for(i = 0 ; i < nbExp ; i++)
        {
                inter = (xPi[i] - avgPi) * (xPi[i] -
avgPi);
                sum = sum + inter;
        }

        double squaredS = (sum / (((double)
nbExp) - 1));
        printf("S²(n) = %.10f \n", squaredS);
        double radius = 2.042 * sqrt((squaredS) /
(double) nbExp);
        return radius;
}
```

Lors de notre réalisation de l'exercice nous avons pris n = 30 et 1000 points tirés pour chaque valeur de π. Nous avons obtenu :

- $S^2(n)$ = 0,0020115126
- R = 0,0167207785
- Une intervalle de confiance de π sur [3,1230125548 , 3,1564541118]

Avec un affichage de 10 décimales pour plus de précision.

Conclusion :

A travers ce TP nous avons vu plusieurs méthodes de calcul de la valeur de π. Nous avons pu utiliser la méthode du Mersen Twister afin de générer des nombres pseudo-aléatoire pour ensuite les utiliser afin de calculer π. Nous avons vu qu'il existait plusieurs type de générateurs aléatoires et que certains sont bien meilleurs que d'autre au niveau de la répétabilité de l'algorithme de chacun par exemple. Ainsi avec le générateur que nous avons utilisé, nous pouvions répéter nos expériences afin de savoir si nos valeurs étaient correctes. Ces 3 exercices nous montrent aussi qu'il existe plusieurs méthode de calcul pour une variable voulue et que certaines de ces méthodes peuvent être plus efficaces ou plus précises que d'autres. Un résultat peut donc être obtenu de plusieurs manières et avec une précision variable.