TP 3 : Simulations de Monte Carlo et intervalles de confiance

Coline Trehout

6 mars 2021

Table des matières

1 Introduction

L'objectif de ce TP est d'obtenir une estimation de la valeur de π grâce à une simulation de Monte Carlo. Dans la première partie, le principe de calcul est expliqué puis plusieurs simulations sont réalisées avec différents nombres de points. Puis, la moyenne est calculée pour un certain nombre d'expériences. Enfin, l'estimation sans biais de la variance, le rayon et l'intervalle de confiance correspondants aux valeurs obtenues sont calculés puis comparés pour différentes simulations.

2 Simulation de la valeur π par la méthode de Monte Carlo

2.1 Principe de calcul

Nous allons calculer la valeur π grâce à une méthode de Monte Carlo. En effet la valeur de π peut être obtenue par des tirages de valeurs aléatoires. Comme π correspond à la surface d'un disque de rayon 1, nous allons tirer des points aléatoires dans un carré de côté 2 (donc de surface 4), contenant ce disque (voir figure 1). Il suffit de compter le nombre de points contenus à l'intérieur du disque et le nombre total de points tirés. La proportion de points tirés à l'intérieur de ce disque tend vers $\pi/4$. En multipliant ce résultat par 4 nous obtenons alors une approximation de π. La qualité de cette estimation dépend évidemment du nombre de tirages aléatoires effectués.

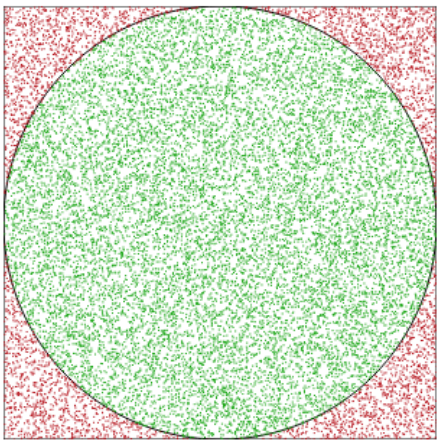

FIGURE 1 – Exemple de simulation de π avec 20 000 points aléatoires

2.2 Implémentation en langage C

La fonction `simPi` calcule une estimation de la valeur π pour un nombre de points aléatoires déterminé en entrée. Un générateur de nombres pseudos aléatoires de qualité est nécessaire pour réaliser les tirages aléatoires. C'est pourquoi nous utiliserons la version de 2002 du Mersenne Twister de Makoto Matsumoto. Il est initialisé au début du programme principal puis la fonction `genrand_real1()` est appelée pour tirer un nombre réel dans l'intervalle [0 ;1]. Chaque point nécessite le tirage de deux valeurs aléatoires, une pour l'abscisse et une autre pour l'ordonnée. Afin de simplifier les calculs par rapport à l'exemple de la figure 1, la simulation est réalisée dans le quart supérieur droit du carré. La somme des points tirés à l'intérieur du disque est comptée. Puis la proportion de points à l'intérieur du disque est calculée puis multipliée par 4 pour obtenir une approximation de la valeur π.

```
1  double simPi (long nbpoints)
2  {
3    long inDisk = 0;
4    double tiragex;
5    double tiragey;
6    long i;
7
8    for ( i = 0; i < nbpoints; i ++)
9    {
10     tiragex = genrand_real1();
11     tiragey = genrand_real1();
12
13     if ( (tiragex * tiragex + tiragey * tiragey) <= 1)
14     {
15       inDisk ++;
16     }
17   }
18
19   return ( (double ) inDisk / (double) nbpoints * 4);
20 }
```

L'estimation de π est faite avec différentes tailles d'échantillons afin de tester la précision de la méthode. Quatre simulations sont effectuées avec respectivement mille, un million, 100 millions et un milliard de points. Les valeurs sont aussi comparées avec la valeur théorique de π contenue dans la bibliothèque `math.h` : $\pi = 3.14159265$. Les résultats sont présentés dans le tableau 1.

nombre de points	valeur de π calculée	$\lvert$écart avec π théorique$\rvert$
10^3	3.12400000	$1.759 * 10^{-2}$
10^6	3.14472000	$3.127 * 10^{-3}$
10^8	3.14124388	$3.488 * 10^{-4}$
10^9	3.14158220	$1.046 * 10^{-5}$

TABLE 1 – Comparaison des valeurs de π calculées pour différents nombres de points

On constate que pour obtenir une valeur satisfaisante de π, il est indispensable de choisir un nombre de points conséquent. En effet, avec mille points, on a seulement la première décimale de π correcte. Avec un million de points, on obtient 2 décimales correctes et avec 100 millions de points, on a 3 décimales correctes. Enfin, le tirage d'un milliard de points est nécessaire pour avoir un résultat correct à la quatrième décimale.

3 Expériences indépendantes

Nous allons à présent réaliser un certain nombre n d'expériences indépendantes de la simulation présentée dans la partie précédente et calculer la moyenne de ces valeurs. Afin d'avoir une distribution qui se rapproche le plus possible de la loi normale, il faut un nombre suffisant d'expériences, les simulations seront donc réalisées pour n=30.

Dans le programme principal les tableaux `valpi` et `moypi` sont déclarés et initialisés à zéro. Les valeurs calculées seront stockées dans ces tableaux. Il y a 3 tableaux `valpi` de 30 cases (un pour chaque nombre de points choisis pour la simulation, une case par expérience) et un tableau `moypi` de 3 cases (une case contenant la moyenne calculée pour chaque simulation, c'est-à-dire pour mille, 1 million et 10 milliards de points). La fonction `calculMoyenne` contient une boucle qui fait appel à la fonction `simPi` à chaque itération. Aucune valeur n'est renvoyée par cette fonction puisque les valeurs de π et leur moyenne calculée sont directement stockées dans les tableaux correspondants qui pourront ensuite être utilisés dans le programme principal.

```
void calculMoyenne (int nbexp, long nbpoints, double * valpi,
    double * moypi, int indiceMoy)
{
  long i;
  double pi;

  for ( i = 0; i < nbexp; i ++)
  {
    pi = simPi( nbpoints );
    valpi[i] = pi;
    moypi[indiceMoy] = moypi[indiceMoy] + pi;
  }
  moypi[indiceMoy] = moypi[indiceMoy] / nbexp;
}
```

D'après la formule 1, la moyenne est calculée en faisant la somme des résultats de chaque expérience et en divisant par n, le nombre d'expériences. Dans notre cas, n=30.

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

Les résultats du calcul de la moyenne seront présentés dans la partie suivante.

4 Calcul des intervalles de confiance autour de la moyenne calculée

4.1 Calcul de l'estimation de la variance $S^2(n)$

À partir de la moyenne calculée, nous pouvons calculer une estimation sans biais de la variance d'après la formule 2.

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

Afin de calculer $S^2(n)$, nous allons d'abord calculer les carrés des écarts entre les valeurs calculées de π et la moyenne calculée. Il y a autant d'écarts à calculer que d'expériences réalisées, soit n=30. Donc ces valeurs sont stockées dans un tableau de 30 cases `sommeEcartsCarres` préalablement déclaré et initialisé. C'est le rôle de la fonction `calculCarresEcarts` ci-dessous.

```c
void calculCarresEcarts (int nbexp, double *
    sommeEcartsCarres, double * valpi, double * moypi, int
    indiceMoy)
{
  int i;
  for ( i = 0; i < 30; i ++)
  {
    sommeEcartsCarres[i] = (valpi[i] - moypi[indiceMoy]) *
   (valpi[i] - moypi[indiceMoy]);
  }
}
```

Puis à partir de ces écarts, nous pouvons calculer une estimation sans biais de la variance. Comme nous ne travaillons pas sur toute la population mais seulement sur un échantillon, nous ne pouvons pas connaître la variance théorique σ^2. L'estimation sans biais de la variance S^2 est donc calculée grâce à la formule 2 dans la fonction `calculVarianceEstimee`.

```c
void calculVarianceEstimee (int nbexp, double * S2, int
    indiceS2, double * sommeEcartsCarres)
{
  int i;
  for ( i = 0; i < 30; i ++)
  {
    S2[indiceS2] =  S2[indiceS2] + sommeEcartsCarres[i];
  }
  S2[indiceS2] = S2[indiceS2] / (nbexp - 1);
}
```

4.2 Calcul du rayon de confiance R

Nous pouvons à présent calculer le rayon de confiance R à partir de la formule 3. Comme la distribution des valeurs ne suit pas une loi normale parfaite, il faut appliquer un coefficient correctif $t_{n-1,1-\alpha/2}$ d'après la loi de Student. Ce coefficient permet de corriger le biais induit par la taille réduite de l'échantillon considéré. Bien sûr, plus l'échantillon est grand et moins il y a de biais à corriger et donc plus la distribution ressemble à la loi normale. La valeur de $t_{n-1,1-\alpha/2}$ dépend donc du nombre d'expériences n choisi ainsi que du seuil de confiance α. Ici, la valeur de R est calculée pour n=30 et un seuil de confiance de 95%, c'est-à-dire pour $\alpha = 0.05$. D'après la table des valeurs de t, on a $t_{30,0.025} = 2.042$.

$$R = t_{n-1,1-\alpha/2} \times \sqrt{\frac{S^2(n)}{n}} \tag{3}$$

La valeur de t est déclarée comme constante de pré processeur au début du programme puis est utilisée dans la fonction `calculR` pour calculer le rayon de confiance R. La valeur de R est ensuite stockée dans le tableau `rayon`.

```
void calculR (int nbexp,  double * S2, int indiceS2, double *
    rayon, int indicerayon)
{
  rayon[indicerayon] = T * sqrt ( S2[indiceS2] / nbexp );
}
```

4.3 Comparaison des résultats de différentes simulations

La simulation a été réalisée pour différents nombres de points ; mille points, un million de points et enfin 10 milliards de points. Les résultats finaux sont présentés dans les tableaux 2 et 3.

nombre de points	$\overline{X}(n)$	|écart|	$S^2(n)$
10^3	3.14680000	$5.207.10^{-3}$	$3.605.10^{-3}$
10^6	3.14162987	$3.721.10^{-5}$	$1.864.10^{-6}$
10^{10}	3.14159173	$9.275.10^{-7}$	$3.020.10^{-10}$

TABLE 2 – Moyennes, écarts et S^2 obtenus pour différentes simulations des valeurs de π avec n=30

nombre de points	R	$[\overline{X} - R\,;\, \overline{X} + R]$
10^3	$2.238.10^{-2}$	[3.1244151 ; 3.1691849]
10^6	$5.090.10^{-4}$	[3.1411209 ; 3.1421389]
10^{10}	$6.479.10^{-6}$	[3.1415852 ; 3.1415982]

TABLE 3 – Rayons et intervalles de confiance obtenus pour différentes simulations des valeurs de π avec $n = 30$ et $\alpha = 0.05$

Plus le nombre de points tirés augmente et plus le rayon de confiance diminue et par conséquent l'intervalle de confiance est réduit.

On constate que pour obtenir une valeur satisfaisante de π, il est nécessaire de choisir un nombre de points très conséquent. En effet d'après

la formule 3, le rayon de confiance est inversement proportionnel à $\sqrt{n}$, le nombre d'expériences réalisées, donc la convergence est très lente. Pour obtenir un résultat 2 fois plus précis, il faut un échantillon 4 fois plus grand. Pour avoir un résultat 10 fois plus précis, il faut donc un échantillon 100 fois plus grand.

C'est pourquoi avec mille points, nous n'avons que les deux premières décimales de π correctes. Avec un million de points, nous obtenons 3 décimales correctes. Enfin, le nombre de points nécessaire pour avoir les 5 premières décimales correctes est de 10 milliards.

5 Conclusion

La méthode de Monte Carlo permet de calculer une valeur approchée de la valeur du nombre π avec une précision dépendant du nombre de points et du nombre d'expériences réalisés. Afin d'obtenir une valeur de π avec une précision satisfaisante, le nombre de points nécessaire est très élevé. En effet la convergence de cette méthode est très lente, ce qui entraîne un temps de calcul parfois long. Les méthodes analytiques sont donc souvent plus efficaces et plus rapides pour calculer la valeur de π. Cependant cet exemple permet de mettre en œuvre une application simple de la méthode de Monte Carlo.

Table des figures

Liste des tableaux