RAPPORT DU TP
'PROJETS INFORMATIQUES'

TP°3
'Monte Carlo Simulation
&
Confidence Intervals'

Eldis YMERAJ- GA- #22015179

Année universitaire 2021-2022

Exercice 1 :

```
// EXERCICE 1
double simPi(int nbPts){
    int compt=0;
    double xr,yr;
    for(int i=0;i<nbPts;i++){
        xr=genrand_real1();
        yr=genrand_real1();
        if((xr*xr+yr*yr)<1){
            compt++;
        }
    }
    return (double)compt/nbPts*4;

}
```

Dans cet exercice on va calculer simulation de la valeur de Pi avec la méthode de Monte Carlo. J'ai appelé notre fonction simPi qui prends en paramètres le nombre de simulations. Premièrement on a déclaré une variable de type int qui va nous compter le nombre de fois que notre condition est vraie. J'ai déclaré deux variables xr et yr qui prennent des valeurs dans intervalle [0,1]. Xr et yr sont les deux coordinats, on est entré dans une boucle pour avec un certain nombre de points qu'on va donner la valeur dans le main et on a généré pour chaque itération un valeur pour xr et yr dans [0,1] et si la valeur de (xr^2+yr^2) est inférieur à la surface de carré avec coté de taille 1, alors on compte combien de points tombent à l'intérieur du cercle. Dans le compteur on a le nombre de points qu'ils sont tombés dans le cercle et on le divise avec tous le nombre de points fois 4 car on est dans un carré. Donc juste en tirant des points aléatoires avec l'aide de fonction genrand_real1 , nous avons une estimation approximative de π.

Dans le main j'ai fait appel de notre fonction simPi avec 1'000, 1'000'000 et 1'000'000'000 nombre de points et on va voir combien de points on a besoin pour obtenir une précision au-dessous de 10^{-2}(3.14), au-dessous de 10^{-3} et au-dessous de 10^{-4}.

```
====================EXERCICE 1====================
For 1'000 points the value of Pi is: 3.124000
For 1'000'000 points the value of Pi is: 3.144720
For 1'000'000'000 points the value of Pi is:3.141541
```

Donc, pour une précision de 10^{-2} (3.14) on a besoin au moins 1'000'000 points. Pour un de 10^{-3} on a besoin de 1'000'000'000 points.

Exercice 2:

```c
// EXERCICE 2
double * mean(int nbSim,int nbPts ){
    double* arrPi=malloc(nbSim*sizeof(double));
    double absErr, relErr;
    double meanArr;
    double sum=0;
    for(int i=0;i<nbSim;i++){
        arrPi[i]=0;
    }
    for(int i=0;i<nbSim;i++){
        arrPi[i]=simPi(nbPts);
    }
    for(int i=0;i<nbSim;i++){
        sum+=arrPi[i];
    }
    meanArr=sum/nbSim;
    absErr=meanArr-M_PI;
    relErr=absErr/M_PI;
    printf("All the values of Pi for %d simulations and %d points: \n", nbSim, nbPts);
    for(int i=0;i<nbSim;i++){
        printf("%f ,", arrPi[i]);
    }
    printf("\nThe value of absolute error is: %f \n",absErr);
    printf("The value of relative error is: %f \n",relErr);
    return arrPi;
}
```

Dans cet exercice on va stocker toutes les estimations de π qui nous génère la fonction d'exercice 1 avec un certain nombre de points dans un tableau qu'on a appelé arrPi et après on calcule l'erreur absolue et erreur relative.

On a appelé notre fonction 'mean' qui prends pour paramètre le nombre de simulations et le nombre de points. On a déclaré un tableau nommé arrPi qui va nous stocker toutes les valeurs possibles de Pi qu'on a trouvé dans l'exercice 1. On a alloué une place dans la mémoire pour notre tableau et après j'ai déclaré deux variables absErr et relErr qu'ils vont contenir les valeurs d'erreur absolue et relative. La variable meanArr calcule la moyenne des éléments de notre tableau avec les estimations de Pi et somme vas nous aider pour calculer la moyenne. Premièrement on a initialisé à 0 notre tableau arrPi et après on a stocké dans le tableau toutes les estimations de π en appellent la simPi. Dans la variable sum on a calculé la somme de toutes les valeurs de tableau et avec l'aide de somme on a calculé la moyenne des éléments de tableau. Après j'ai calculé la valeur de l'erreur absolue et relative. On a imprimé le tableau avec 30 estimations de Pi et aussi la valeur d'erreur absolue et relative.

Dans le main on a appelé cet fonction pour un nombre d'expérimentes de 10 jusqu'à 30 et pour nombre de points de 1'000, 1'000'000 et 1'000'000'000 mais pour la dernière j'ai pas affiché car ca prends beaucoup de temps pour calculer 1'000'0000'000 points. Et pour chaque nombre d'opérations et nombre de points vous allez voir dans la console toutes les valeurs de π et les valeurs d'erreur absolue et relative.

```
====================EXERCICE 2====================
All the values of Pi for 10 simulations and 1000 points:
3.124000 ,3.152000 ,3.120000 ,3.112000 ,3.196000 ,3.136000 ,3.204000 ,3.212000 ,3.100000 ,3.124000 ,
The value of absolute error is: 0.006407
The value of relative error is: 0.002040
All the values of Pi for 10 simulations and 1000000 points:
3.144408 ,3.140536 ,3.142088 ,3.142056 ,3.142132 ,3.139992 ,3.139656 ,3.140540 ,3.138920 ,3.142216 ,
The value of absolute error is: -0.000338
The value of relative error is: -0.000108
All the values of Pi for 20 simulations and 1000 points:
3.144000 ,3.120000 ,3.164000 ,3.088000 ,3.136000 ,3.108000 ,3.136000 ,3.048000 ,3.184000 ,3.100000 ,3.076000 ,3.048000 ,3.144000 ,3.152000 ,3.244000 ,3
.148000 ,3.216000 ,3.180000 ,3.112000 ,3.152000 ,
The value of absolute error is: -0.006593
The value of relative error is: -0.002099
All the values of Pi for 20 simulations and 1000000 points:
3.144064 ,3.141716 ,3.140752 ,3.138876 ,3.140004 ,3.142468 ,3.142244 ,3.140348 ,3.140856 ,3.139240 ,3.140548 ,3.144340 ,3.138308 ,3.143104 ,3.141776 ,3
.142556 ,3.141292 ,3.141100 ,3.138528 ,3.139764 ,
The value of absolute error is: -0.000498
The value of relative error is: -0.000159
All the values of Pi for 30 simulations and 1000 points:
3.164000 ,3.228000 ,3.248000 ,3.176000 ,3.144000 ,3.132000 ,3.136000 ,3.204000 ,3.056000 ,3.128000 ,3.048000 ,3.168000 ,3.168000 ,3.244000 ,3.160000 ,3
.124000 ,3.108000 ,3.108000 ,3.080000 ,3.148000 ,3.168000 ,3.044000 ,3.188000 ,3.136000 ,3.108000 ,3.196000 ,3.132000 ,3.236000 ,3.160000 ,3.164000 ,
The value of absolute error is: 0.008541
The value of relative error is: 0.002719
All the values of Pi for 30 simulations and 1000000 points:
3.141504 ,3.143980 ,3.139192 ,3.142648 ,3.139928 ,3.142556 ,3.140928 ,3.139616 ,3.139844 ,3.142812 ,3.141044 ,3.141472 ,3.142872 ,3.141392 ,3.140556 ,3
.140944 ,3.136796 ,3.139340 ,3.138064 ,3.140696 ,3.142392 ,3.140660 ,3.138100 ,3.140280 ,3.141272 ,3.137524 ,3.141976 ,3.140244 ,3.142312 ,3.143328 ,
The value of absolute error is: -0.000784
The value of relative error is: -0.000249
```

Exercice 3 :

Dans cet exercice on a essayé de calculer les intervalles de confiance. On a déclaré n qui est le nombre d'opérations égal à 30, une variable sum qui va nous calculer la somme des éléments de tableau qu'on va utiliser après, Xbarre→ Si X est un résultat de simulation, (X1 , ... ,,Xn) est l'ensemble obtenu avec n répétitions indépendantes de cette simulation stochastique. Cela signifie que chaque expérience de simulation stochastique indépendante est exécutée mais avec un flux aléatoire différent. Le calcul des intervalles de confiance est réalisé sur les moyennes arithmétiques observées. Et par conséquent la valeur de Xbarre va être calculé par la formule :

```c
// EXERCICE 3
void confIntervals(){
    int n=30;
    double sum=0;
    double Xbarre;
    double simVar=0;
    double R=0;
    double t=2.042;
    double *arr=malloc(n*sizeof(double));
    arr=mean(30,10000000);
    for(int i=0;i<n;i++){
        sum+=arr[i];
    }
    Xbarre=sum/n;
    for(int i=0;i<n;i++){
    simVar+=pow(arr[i]-Xbarre,2);
    }
    simVar/=n-1;
    R=t*sqrt(simVar/n);
    printf("The value of error margin is :%f \n",R);
    printf("The mean Xbarre = %f\n", Xbarre);
    printf("Value of variance = %f\n", simVar);
    printf("The interval which which takes 95b percent of values is: [%f,[%f]\n", Xbarre-R,Xbarre+R);
}
```

$$\text{Xbarre(n)} = \sum_{i=1}^{n} \text{Xi/n}$$

La variable simVar c'est la variance qui décrit dans quelle mesure les valeurs échantillonnées : les différents résultats de simulation varient les uns des autres. La variable R c'est le rayon de confiance et variable t c'est une valeur fixe pour une certain nombre d'itérations et dans notre cas on a pris n=30 et la valeur de t=2.042 qu'on va l'utiliser pour trouver R. On a déclaré un tableau arr qui prends la valeur qui donne la fonction de l'exercice 2 avec nombre de simulations=30 et 1'000'000 points et qui stocke les valeurs de π. On a calculé la somme des éléments de tableau pour trouver Xbarre.

On a vu que la valeur de $simVar(n) = \frac{\sum_{i=1}^{n}[Xi-Xbarre(n)]^2}{n-1}$ et on a calculé la valeur finale de simVar. Je précise que simVar(n) est une estimation sans biais de σ^2 variance. Comme nous ne connaissons pas l'écart-type théorique σ, nous estimons la variance avec les résultats que nous avons et utilisons donc cette approximation. Nous calculons simVar(n) et l'utilisons pour obtenir le rayon de confiance R dans le niveau 1-α avec α=0.05. On sait que

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

L'intervalle de confiance est dit à 95%. Le calcul de R donne l'intervalle suivant [Xbarre-R,Xbarre+R], avec une confiance de 1-α(95%).

Donc on fait les calcules en utilisant les formules données et on affiche le résultat.

Quand on va appeler le fonction on va avoir :

```
====================EXERCICE 3====================
All the values of Pi for 30 simulations and 1000000 points:
3.140044 ,3.142404 ,3.142128 ,3.142188 ,3.139760 ,3.139872 ,3.140236 ,3.139288 ,3.142300 ,3.143456 ,3.142272 ,3.140208 ,3.138976 ,3.140292 ,3.141908 ,3.142532
38956 ,3.140652 ,3.144988 ,3.137976 ,3.143032 ,3.141652 ,3.142856 ,3.140776 ,3.141508 ,3.138528 ,3.139604 ,3.142172 ,
The value of absolute error is: -0.000537
The value of relative error is: -0.000171
The value of error margin is :0.000608
The mean Xbarre = 3.141055
Value of variance = 0.000003
The interval which which takes 95b percent of values is: [3.140448,3.141663]
```

CONCLUSION

Dans cet TP on a utilisé la simulation de Monte Carlo et les Intervalles de confiance pour réaliser le devoir. La méthode de Monte Carlo fait référence à toute méthode utilisant des nombres aléatoires et un échantillonnage statistique pour simuler des phénomènes naturels, analyse numérique pour intégration complexe et des autres applications. Avec MC simulation on peut bien préciser les valeurs de plusieurs constants scientifiques. La simulation de Monte Carlo mene à un ensemble d'expériences d'échantillonnage statistique qui utilisent intelligemment le caractère aléatoire - "stochastique" qui est basé sur un échantillonnage aléatoire. On arrive à la conclusion que on peut accélérer les méthodes de Monte Carlo avec les techniques de réduction de la variance.

On peut dire que l'une des avantages de cet méthode est qui rocade la complexité de la résolution de problèmes en méthodes analytiques. Et pour les désavantages on peut dire que la solution est statistique par nature et la haute précision a un cout de calcul élevé.