PHYS 8601 – Problem 1

Yier Wan

February 1, 2018

1 Introduction

The main propose of this work is to apply simple sampling Monte Carlo algorithm to evaluate the value of π. The idea behind Monte Carlo method is to use random sampling and statistical analysis.

First, we consider a quadrant inscribed in a unit square and then uniformly scatter a given number of points over the square. After that, we count the number of points inside the quadrant (having a distance from the origin of less than 1). Finally, we calculate the ratio of the inside-count and the total-sample-count and multiply it by 4 to get an estimate of π.

2 Calculation details

2.1 Pseudo-code

The pseudo-code is written as:
Input: *length*, number of points simulated in a single run.
Output: *est*, estimated value of π.

```
1 for $i = 1$ to length
2     generate random numbers, $x[i]$, $y[i]$, between 0 and 1
3     calculate $r = \sqrt{x[i]^2 + y[i]^2}$
4      If $r \leq 1$ then $t1 = t1 + 1$
5      else $t2 = t2 + 1$
6 calculate $est = 4 * t1/(t1 + t2)$
```

The *length* is chosen to be different values (1000, 2000, ..., etc.) in order to study the changes in the mean value and errors as the sampling size increases.

2.2 Statistics

The unbiased estimator of the mean is calculated by,

$$\overline{A} = \frac{1}{n}\sum_{i=1}^{n} A_i$$

where n is the number of independent runs, A_i is the value of*est* in i^{th} run.

The variance is estimated by,

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n} A_i^2 - \overline{A}^2$$

and then the precision (or error bar) is evaluated by $\sqrt{\sigma^2}$.

The accuracy of the estimate, which represents how does the result compare to the correct answer, is presented by calculating percent error,

$$\frac{(A_{est} - A_{real})}{A_{real}} \times 100\%$$

where A_{est} is the value of $\overline{A}$ and A_{real} takes number 3.1415926535.

2.3 Random number

There are many useful functions available in C libraries which can generate random streams. In this work, we use `int rand(void)` function, declared in `stdlib.h`. It returns a pseudo-random number. We can get a random number between 0 and 1 with:`(rand()%1000000)/1000000.0`

However, we will always get the same sequence of stream, if we only use function `int rand(void)`. The `void srand(unsigned seed)` function sets the starting point for producing a series of pseudo-random integers.

If `srand()` is not called, the `rand()` seed is set as if `srand(1)` were called at the program start. The pseudo-random number generator should only be seeded once, before any calls to `rand()`. We adopt the common method to initialize a random number stream:`srand(time(0))`. It means current time is used as seed for this stream, so that we will get different and independent random streams when the program is executed each time.

3 Results and discussion

The simulated results are listed in Table. 1. The first column (N) and *length* mentioned in sec.2.2 are the same thing, meaning how many points generated in each independent run. For each N value, we run the program for 20 times.

Figure. 1 shows how the estimated value changes when the number of trials increases. It is noted that all the estimated values scattered around the black dash line, which represents the real value of π. The error bar becomes shorter as more simulations included. Although the estimated values fluctuate, the real value of π is within the first standard deviation of each data point.

Figure. 2 plots how the accuracy changes as N increases. Generally, the accuracy becomes closer to zero as N increases.

Table 1: Simulated results

N	Estimated value of π	Precision	Accuracy
1000	3.148000	0.053874	0.203952 %
2000	3.136300	0.035823	-0.168470 %
3000	3.152067	0.032134	0.333396 %
4000	3.137300	0.026978	-0.136639 %
5000	3.146080	0.025088	0.142837 %
6000	3.140367	0.020970	-0.039028 %
7000	3.139171	0.017987	-0.077069 %
8000	3.146025	0.018042	0.141086 %
9000	3.140756	0.016811	-0.026647 %
10000	3.138440	0.014235	-0.100352 %

4 Attachments

Other related materials are:
1. summary.log
2. p2.c (copy of the program)
3. samplerun.txt (sample output of a single run)
4. output.dat (all the simulation results)
5. post.pl (script to do the statistical calculation)

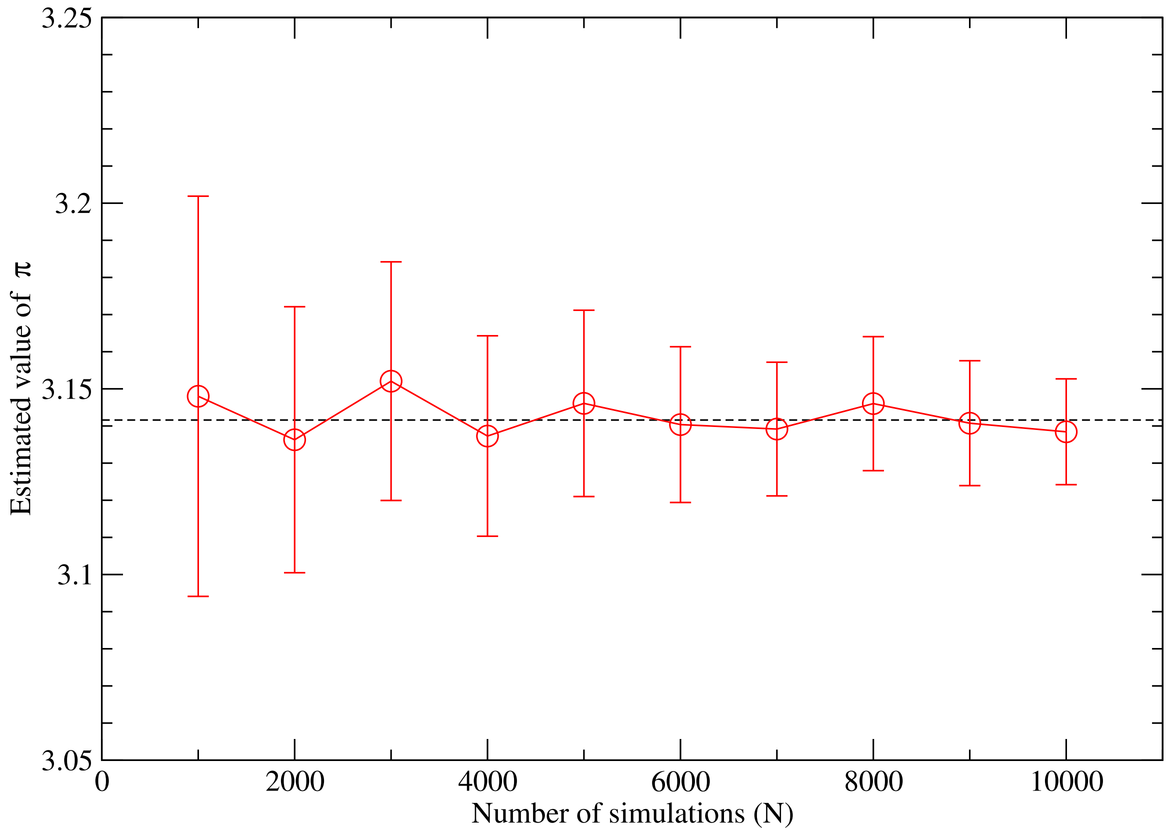

Figure 1: Estimated value of π as a function of number of simulations.

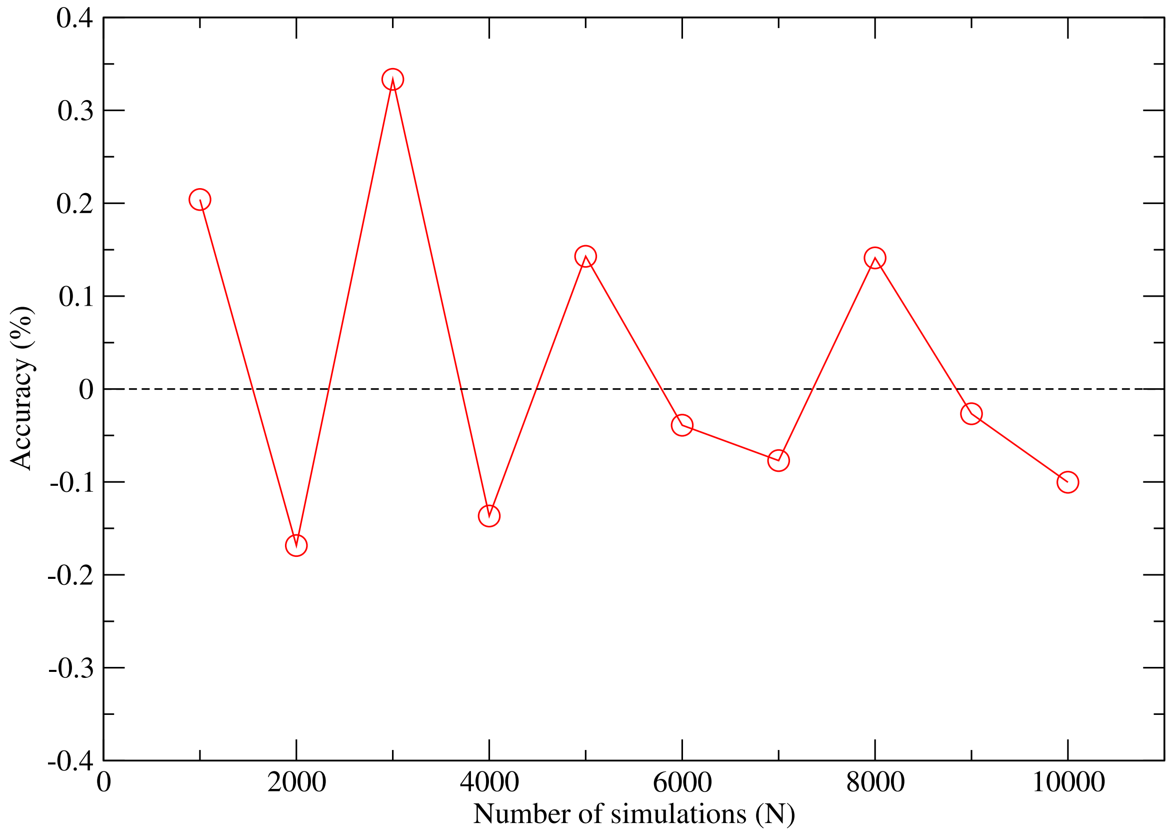

Figure 2: Accuracy as a function of number simulations.