1. Background and purpose of the process

1.1 Monte Carlo Basic Ideas

Computer simulations often use a stochastic simulation method or a statistical test method, which is the Monte Carlo method. It simulates the process by continuously generating a sequence of random numbers.

Monte Carlo methods are categorized into direct and indirect methods. Direct method in accordance with the actual problem followed by the statistical laws of probability, is also commonly referred to as “computer experiments”; indirect method is artificially constructed a suitable probability model, in accordance with a large number of statistical experiments on the model, so that some of its statistical parameters are exactly the solution of the problem to be solved, and here to solve the pi is the indirect method.

1.2 Basic Ideas for Solving Pi

According to the formula for solving the area of a circle $S = \pi r^2$, it can be deduced that the formula for solving the circumference of a circle is $\pi = \frac{S}{r^2}$, according to this formula, the circumference of a circle can be found by calculating the area of the circle and measuring the radius of the circle. The radius of the circle can be measured directly, but the area is not easy to find.

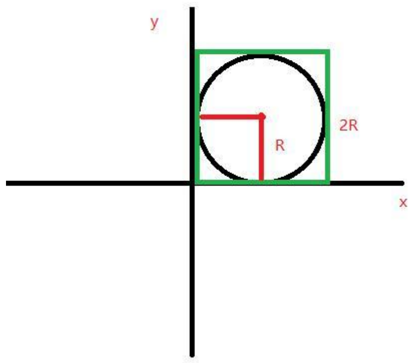


Fig 1 Sketch of the needle throwing experiment

In 1777, the French mathematician Buffon proposed the method of the needle-throwing experiment to find the circumference π, which is considered to be the origin of the Monte Carlo method. A square of side length 2 is drawn on a horizontal smooth tabletop, a unit circle is tangent to it, the origin of the Cartesian coordinate system is taken as the center of the unit circle, and the second axis is parallel to the square. Casually cast sewing needles on the top of the table, each cast, observe the location of the needle tip landing point, if the needle tip falls into the square, record $N_{正}$; if the needle tip falls into the circle, record $N_{圆}$; fall into the square outside of not record. In many times after the needle, you can count the number of times the tip of the needle fell into the square or circle, at this time through the following formula to get the circumference of the circle

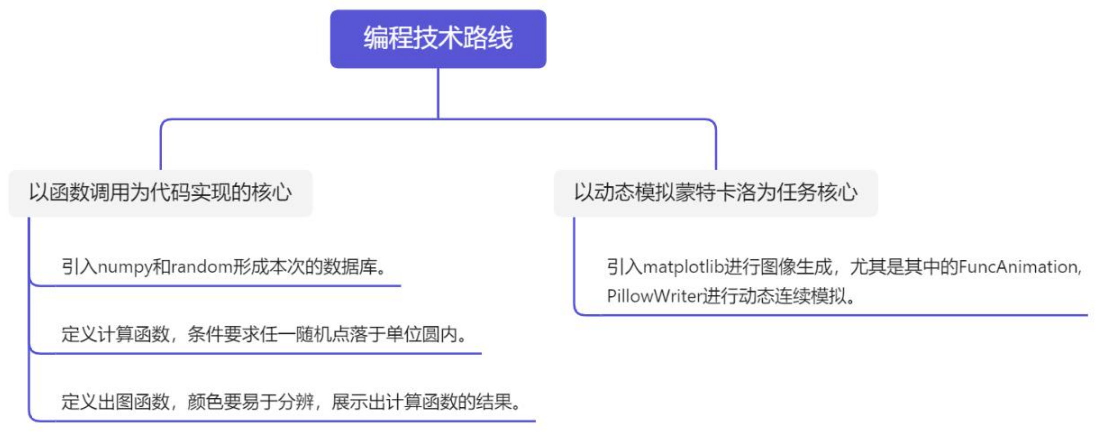
$$\pi = 4 \times \frac{N_{圆}}{N_{正}}$$

1.3 Solving for Circumference by Monte Carlo Method (Program Purpose)

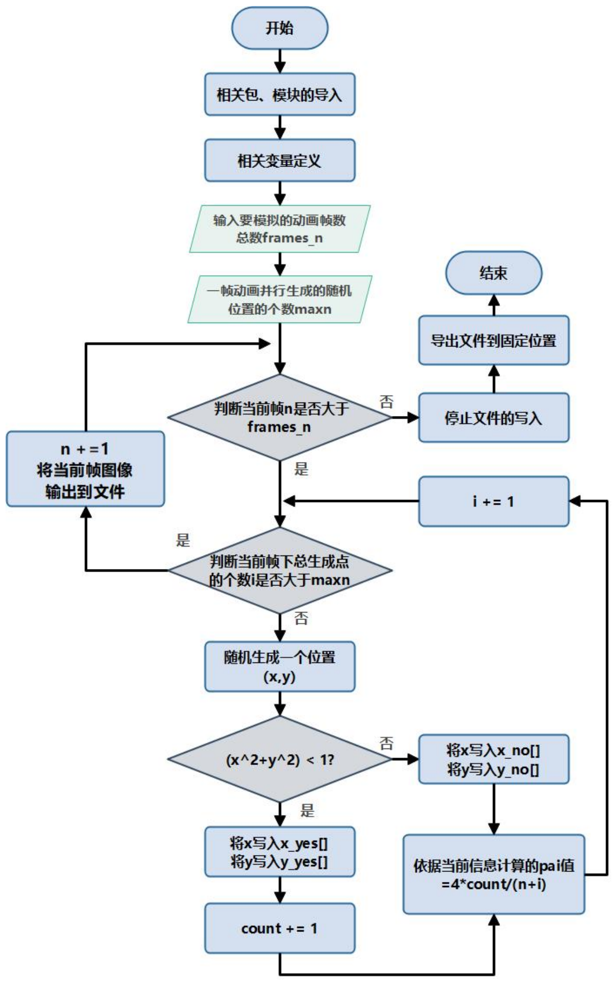
Take a set of numbers from 0 to 2, each element of this set of numbers contains two variables x and y, x represents the horizontal coordinates of the point taken, y represents the vertical coordinates of the point taken, x and y are randomly selected by the computer, and then edit the formula to screen the points that meet the conditions, the screening conditions are: x and y are included in the circle, and then set a counter i, whenever there is a point that meets the conditions, then i plus 1, and so on and so forth. Cycle to take enough points set, and then divide the i value by the number of points set to get a value, and then multiplied by 4, the value obtained is the area of the circle, because the radius is 1 by default, so at this time also get the circumference.

This program two aspects of the realization of Monte Carlo method to solve the circumference, one is the direct acquisition, high precision but not intuitive; the second is a dynamic simulation, precision is limited but easy for users to understand the process, very intuitive.

2. Programming technical lines


编程技术路线
以函数调用为代码实现的核心
引入numpy和random形成本次的数据库。
定义计算函数，条件要求任一随机点落于单位圆内。
定义出图函数，颜色要易于分辨，展示出计算函数的结果。
以动态模拟蒙特卡洛为任务核心
引入matplotlib进行图像生成，尤其是其中的FuncAnimation, PillowWriter进行动态连续模拟。

3. Flowchart of the program framework



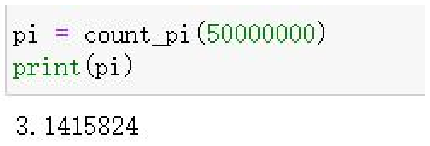
4. Explanation of how the program works and the results

4.1 Running mode

This program ‘Monte Carlo method for pi.ipynb’, the operating environment requires the program in the Jupter-Notebook to upload the file and run, the premise of running in the operating environment is to install a good Numpy library, Matplotlib library, the process of running Matplotlib should be called! software package in the sub-package pyplot, animation (for animation, the main use of the package FuncAnimation, PillowWriter two functions) and python comes with the random function.

4.2 Interpretation of results

4.2.1 Directly calculated results



```
pi = count_pi(50000000)
print(pi)
```

```
3.1415824
```

Fig 2 Circumference values calculated at 50,000,000 experimental points

The direct solution is strongly influenced by the number of points, intuitively speaking, the more points are selected, the closer the result of this step is to the real value of pi. Unlike other cases where multiple points are needed for calculation, this part of the program runs very fast, even though 50,000,000 experimental points are selected for this example, due to the clarity and simplicity of the calculation process.

4.2.2 Results of dynamic simulations

The direct result of this process is the generation of a simulation of the moving picture “test-1.gif” file, subject to word limitations can only be attached to a single state of the static picture, the specific file has been included in the job folder, see the figure below.

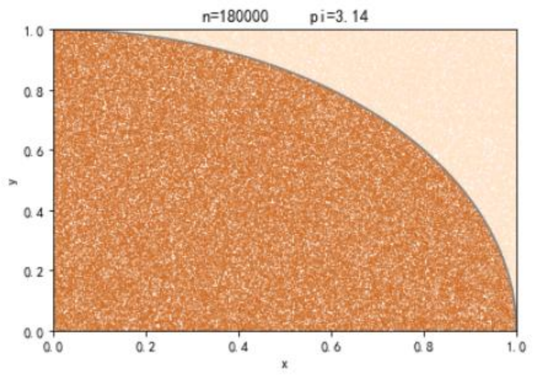


Fig 3 The static case of taking 180,000 experimental points, at which point the value of the calculated circumference = 3.14

In the paper, the needle casts from n=2000 to n=400000 and the corresponding pi values are shown with an interval of 2000. it can be seen that along with the increase of the experimental points, the number of points falling into the quarter-unit circle is increasing in synchronization, and is always more than the number of points in the blank part of the square; at the same time, along with the increase of the experimental points, the pi value is approaching to the exact pi value, but the there are occasional errors, which need to be reflected by specific iterative plots.

4.2.3 Results of the iterative convergence process

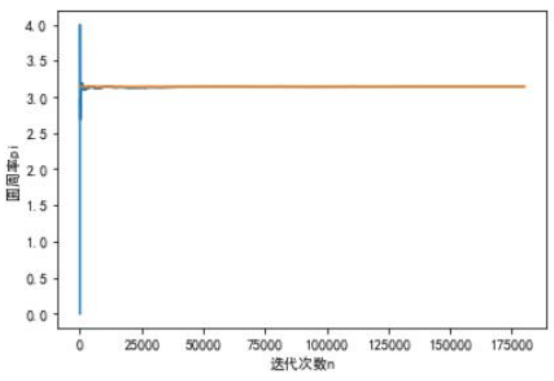


Fig 4 Specifics of the iterative convergence process at n=180,000

In the process of iteration, the calculated pi value converges continuously, it can be seen that in the iteration number of 0 near the smaller interval, the error is very large, at this time by the probability of accuracy of the error; but along with the number of iterations, that is, the number of points of the experiment continues to increase, the calculated pi is gradually close to the true value of pi, around the true value of doing the oscillation of the small interval.

4.3 Analyze program strengths and weaknesses

The advantages of this program are:

1. the Monte Carlo method is more accurate to calculate pi directly, and also uses moving pictures to visualize the algorithm and better understand the connotation of the Monte Carlo method.

2. the dynamic diagrams generated in the second part are not restricted by directories and can be generated directly in the working

directory.

3. The contents of the Numpy library and Matplotlib library that we have learned in this semester are directly utilized to review and consolidate our understanding of the library and the corresponding functions.

4. Since the program is run under the Jupter-Notebook environment, the variables in the two and three parts are the same, so there is no need to write multiple .py files and redefine the variables to get the results in the same situation, which better corresponds to the experimental conclusions.

The disadvantages and shortcomings are:

1. Influenced by the Monte Carlo method itself, there is no way to accurately find the value of circumference because it is an estimation method.

2. The generation of moving pictures is not only limited by the number of experimental points, but also by the number of frames, and there is no specific consideration of how the number of frames is affected and what different results there will be, but it does not affect the formation of conclusions.

3. In Jupter-Notebook, you cannot see the moving pictures directly after calling the program, only static pictures are presented for that

moment in time, and you need to be in the catalog to observe the moving pictures properly.

Other ways can be used to form the dynamic simulation, not only limited to the function called in this program, can be used in the future study and practice in more ways of implementation.