

Introduction to Computational Physics – Exercise 10

Simon Groß-Böltig, Lorenz Vogel, Sebastian Willenberg

July 3, 2020

Determination of π with Random Numbers

Task: Compute the number π using a rejection method with the function $f(x) = \sqrt{1 - x^2}$, for $0 \leq x \leq 1$.

Hint: It is enough to use only one quadrant $x, f(x) > 0$. Vary the number of random numbers (RNs) widely (orders of magnitude) and plot the accuracy of the result as a function of the number of RNs. Use logarithmic variables for the plot.

To determine π numerically, we first use a so-called *rejection method*: To do this, consider a quadrant $f(x) = \sqrt{1 - x^2}$ ($0 \leq x \leq 1$) with radius $r = 1$ that lies within a square with the side lengths $r = 1$. The following applies to the ratio of the areas of the quadrant and the square:

$$\frac{A_c}{A_s} = \frac{\pi r^2}{4r^2} = \frac{\pi}{4} \quad (1)$$

The goal is now to estimate the ratio of the areas in order to calculate π . We can devise an algorithm that generates two random numbers $x_i, y_i \in [0, 1]$ with $i = 1, \dots, N$ (random coordinates from the square) and checks whether the coordinate fell into the quarter circle or not (rejection method):

$$\left[\text{if } y_i < \sqrt{1 - x_i^2} \Rightarrow \text{"take"} \right] \quad \dot{\vee} \quad \left[\text{if } y_i > \sqrt{1 - x_i^2} \Rightarrow \text{"reject"} \right] \quad (2)$$

The value of π is then approximately given by the ratio of the number N_{take} of points within the quadrant to the total number N_{total} of points:

$$\pi \approx 4 \frac{N_{\text{take}}}{N_{\text{total}}} \quad (3)$$

To generate the random numbers, we use the function `numpy.random.rand`, which creates an array of the given shape and populates it with random samples from a uniform distribution over $[0, 1]$. Fig. 1a shows the idea/principle of the rejection method. The accuracy (relative error compared to the “true” value of π) of the rejection method is shown in Fig. 1b.

Another possibility for the numerical determination of π is the so-called *Monte Carlo integration*: Monte Carlo integration is a technique for numerical integration using random numbers. We want to integrate the function $f(x)$ on the interval $[a, b]$. The idea with Monte Carlo integration is to evaluate the function $f(x)$ on N randomly selected points x_i ($i = 1, \dots, N$) (random sampling) in the interval $[a, b]$. In the case of equally distributed random numbers x_i , the value of the integral is simply the average of the function values $f(x_i)$:

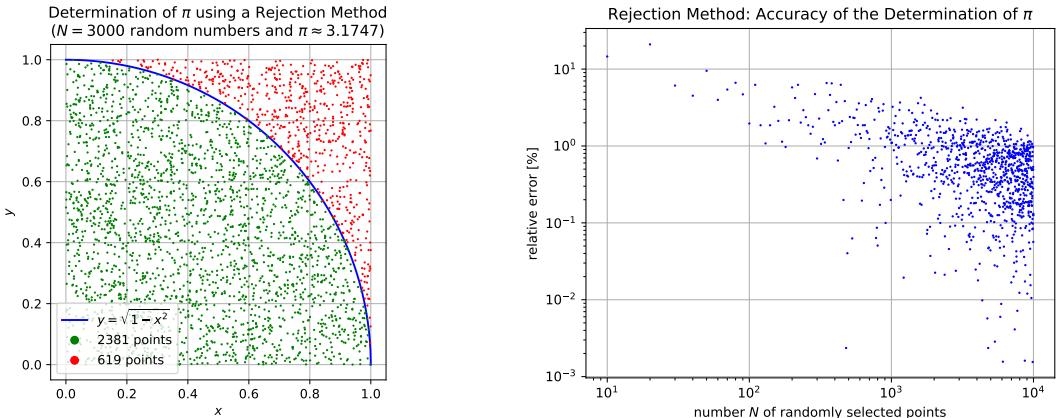
$$\int_a^b f(x) dx \approx \frac{b-a}{N} \sum_{i=1}^N f(x_i) \quad (4)$$

The fact that this method works is due to the law of large numbers, i.e. $N \rightarrow \infty$. We know that the integral of $f(x) = \sqrt{1 - x^2}$ on the interval $0 \leq x \leq 1$ gives a quarter of the area of the unit circle. The area of the unit circle (radius $r = 1$) is $A_c = \pi r^2 = \pi$. With N randomly selected values $x_i \in [0, 1]$, we obtain the following approximation formula for π using Monte Carlo integration:

$$\frac{\pi}{4} = \int_0^1 \sqrt{1 - x^2} dx \approx \frac{1}{N} \sum_{i=1}^N \sqrt{1 - x_i^2} \quad \Rightarrow \quad \pi \approx \frac{4}{N} \sum_{i=1}^N \sqrt{1 - x_i^2} \quad (5)$$

Using Monte Carlo integration, we get the following value for π :

Monte Carlo integration with $N = 6 \cdot 10^7$:	$\pi \approx 3.141\,557\,094$	(6)
---	-------------------------------	-----



(a) Illustration of the determination of π using a rejection method with randomly selected points

(b) Accuracy analysis of the determination of π using a rejection method with randomly selected points

Figure 1: Determination of π with random numbers (rejection method)

Python-Code 1: Determination of π using a rejection method and Monte Carlo integration

```

1 # -*- coding: utf-8 -*-
2 """
3 Introduction to Computational Physics
4 - Exercise 10: Determination of Pi with Random Numbers
5 - Group: Simon Groß-Böltling, Lorenz Vogel, Sebastian Willenberg
6 """
7
8 import numpy as np; import matplotlib.pyplot as plt; import scipy.constants as const
9
10 def quadrant_function(x):
11     ''' Function that describes the unit quarter circle (quadrant) '''
12     return np.sqrt(1-x**2)
13
14 def pi_Monte_Carlo(N,quadrant):
15     ''' Function to determine pi using Monte Carlo integration '''
16     random_x = np.random.rand(1,N) # random numbers (random sampling)
17     return (4./N)*np.sum(quadrant(random_x))
18
19 def pi_rejection_method(N,quadrant):
20     ''' Function to determine pi using a rejection method '''
21
22     # create random coordinates within the square
23     random_x = np.random.rand(1,N); random_y = np.random.rand(1,N)
24
25     # check whether the coordinate fell into the quadrant
26     x_take = random_x[random_y < quadrant(random_x)]
27     y_take = random_y[random_y < quadrant(random_x)]
28
29     # check whether the coordinate fell not into the quadrant
30     x_reject = random_x[random_y > quadrant(random_x)]
31     y_reject = random_y[random_y > quadrant(random_x)]
32
33     return (4.*len(x_take)/N, x_take, y_take, x_reject, y_reject)
34
35
36 ## Determination of Pi using Monte Carlo Integration
37 N = int(6e7) # number of randomly selected points
38 print('Monte Carlo Integration: {}'.format(pi_Monte_Carlo(N,quadrant_function)))
39
40 ## Determination of Pi using a Rejection Method
41 N = int(3e3) # number of randomly selected points
42 out = pi_rejection_method(N,quadrant_function)
43
44 fig, ax = plt.subplots() # plot to illustrate the rejection method
45 ax.set_title(r'Determination of $\pi$ using a Rejection Method'+r'\n'+r'($N={}$ random numbers and $\pi\approx{}$)'.format(N,round(out[0],4)))
46 ax.set_xlabel(r'$x$'); ax.set_ylabel(r'$y$')
47

```

```

48
49 x = np.linspace(0,1,1000)
50 ax.plot(x,quadrant_function(x), 'b-', linewidth=1.5, label=r'$y=\sqrt{1-x^2}$')
51 ax.plot(out[1], out[2], 'g.', markersize=1.5, label=r'$\{\} points'.format(len(out[1])))
52 ax.plot(out[3], out[4], 'r.', markersize=1.5, label=r'$\{\} points'.format(len(out[3])))
53
54 ax.grid(); ax.legend(loc='lower left', markerscale=8)
55 ax.set(xlim=(-0.05,1.05), ylim=(-0.05,1.05)); ax.set_aspect('equal', 'box')
56 fig.savefig('figures/RN-Rejection-Method_Pi-Determination.pdf', format='pdf')
57
58 # accuracy of the result as a function of the number of randomly selected points
59 N = np.linspace(10,int(1e4),1000) # number of randomly selected points
60 rel_error = np.zeros(len(N)) # array for the accuracy of the result
61
62 for i in range(0,len(N)):
63     N[i] = int(N[i]); pi = pi_rejection_method(int(N[i]),quadrant_function)[0]
64     rel_error[i] = abs(pi-const.pi)/abs(const.pi)
65
66 fig, ax = plt.subplots()
67 ax.set_title(r'Rejection Method: Accuracy of the Determination of $\pi$')
68 ax.set_xlabel(r'number $N$ of randomly selected points')
69 ax.set_ylabel(r'relative error [%]')
70 ax.plot(N, 100*rel_error, 'b.', markersize=1.5)
71 ax.grid(); ax.set_xscale('log'); ax.set_yscale('log')
72 #ax.set(xlim=(-0.05,1.05), ylim=(-0.05,1.05))
73 fig.savefig('figures/RN-Rejection-Method_Accuracy-Analysis.pdf', format='pdf')
74 plt.show(); plt.clf(); plt.close()

```