Introduction to Computational Physics – Exercise 10

Simon Groß-Bölting, 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:

$$\frac{A_\mathrm{c}}{A_\mathrm{s}} = \frac{\pi r^2}{4r^2} = \frac{\pi}{4} \tag{1}$$

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 $i = 1,\ \ldots,\ N$ (random coordinates from the square) and checks whether the coordinate fell into the quarter circle or not (rejection method):

$$\left[\text{if} \quad y_i < \sqrt{1-x_i^2} \quad \Rightarrow \quad \text{“take”}\right] \qquad \dot{\vee} \qquad \left[\text{if} \quad y_i > \sqrt{1-x_i^2} \quad \Rightarrow \quad \text{“reject”}\right] \tag{2}$$

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:

$$\pi \approx 4\,\frac{N_\mathrm{take}}{N_\mathrm{total}} \tag{3}$$

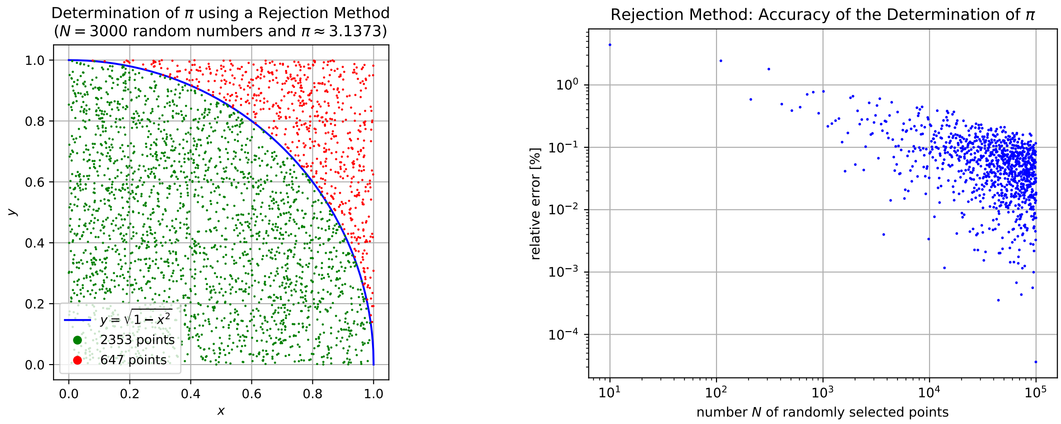
To generate the random numbers, we use the function `numpy.random.rand`, which creates an array of the given shape and populates it with random samples from a uniform distribution over $[0,\ 1)$. Fig. 1a shows the idea/principle of the rejection method. The accuracy (relative error compared to the “true” value of π) of the rejection method is shown in Fig. 1b. Because we work with random numbers, the accuracy can sometimes be better for a smaller N than for a large N for statistical reasons: Therefore, in Python-Code 1 we used the mean of ten π calculations to determine the accuracy for a given N. Regardless of the statistical fluctuations, the relative error in the log-log-plot decreases linearly (see Fig. 1b).
Using the rejection method, we get (for example) the following value for π:

$$\boxed{\text{Rejection Method with } N = 6 \cdot 10^7\text{:} \qquad \pi \approx 3.141\,445} \tag{4}$$

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 $(i = 1,\ \ldots,\ N)$ (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 $f(x_i)$:

$$\int_a^b f(x)\,\mathrm{d}x \approx \frac{b-a}{N}\sum_{i=1}^{N} f(x_i) \tag{5}$$

The fact that this method works is due to the law of large numbers, i.e. $N \to \infty$. 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.



(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)

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:

$$\frac{\pi}{4} = \int_0^1 \sqrt{1-x^2}\,\mathrm{d}x \approx \frac{1}{N}\sum_{i=1}^{N}\sqrt{1-x_i^2} \qquad \Longrightarrow \qquad \pi \approx \frac{4}{N}\sum_{i=1}^{N}\sqrt{1-x_i^2} \tag{6}$$

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

$$\boxed{\text{Monte Carlo Integration with } N = 6\cdot 10^7: \qquad \pi \approx 3.141\,557\,094} \tag{7}$$

It should be noted that the rejection method takes much more time than the Monte Carlo integration.

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

```python
# -*- coding: utf-8 -*-
"""
Introduction to Computational Physics
- Exercise 10: Determination of Pi with Random Numbers
- Group: Simon Groß-Bölting, Lorenz Vogel, Sebastian Willenberg
"""

import numpy as np; import matplotlib.pyplot as plt; import scipy.constants as const

def quadrant_function(x):
    ''' Function that describes the unit quarter circle (quadrant) '''
    return np.sqrt(1-x**2)

def pi_Monte_Carlo(N,quadrant):
    ''' Function to determine pi using Monte Carlo integration '''
    random_x = np.random.rand(1,N) # random numbers (random sampling)
    return (4./N)*np.sum(quadrant(random_x))

def pi_rejection_method(N,quadrant):
    ''' Function to determine pi using a rejection method '''

    # create random coordinates within the square
    random_x = np.random.rand(1,N); random_y = np.random.rand(1,N)

    # check whether the coordinate fell into the quadrant
    x_take = random_x[random_y < quadrant(random_x)]
    y_take = random_y[random_y < quadrant(random_x)]

```

```
    # check whether the coordinate fell not into the quadrant
    x_reject = random_x[random_y > quadrant(random_x)]
    y_reject = random_y[random_y > quadrant(random_x)]

    return (4.*len(x_take)/N, x_take, y_take, x_reject, y_reject)


## Determination of Pi using Monte Carlo Integration
N = int(6e7) # number of randomly selected points
print('Monte Carlo Integration: {}'.format(pi_Monte_Carlo(N,quadrant_function)))

## Determination of Pi using a Rejection Method
N = int(3e3) # number of randomly selected points
out = pi_rejection_method(N,quadrant_function)
print('Rejection Method: {}'.format(out[0]))

fig, ax = plt.subplots() # plot to illustrate the rejection method
ax.set_title(r'Determination of $\pi$ using a Rejection Method'+'\n'
             +r'($N={}$ random numbers and $\pi\approx{}$)'.format(N,round(out[0],4)))
ax.set_xlabel(r'$x$'); ax.set_ylabel(r'$y$')

x = np.linspace(0,1,1000)
ax.plot(x,quadrant_function(x), 'b-', linewidth=1.5, label=r'$y=\sqrt{1-x^2}$')
ax.plot(out[1], out[2], 'g.', markersize=1.5, label=r'${}$ points'.format(len(out[1])))
ax.plot(out[3], out[4], 'r.', markersize=1.5, label=r'${}$ points'.format(len(out[3])))

ax.grid(); ax.legend(loc='lower left', markerscale=8)
ax.set(xlim=(-0.05,1.05), ylim=(-0.05,1.05)); ax.set_aspect('equal', 'box')
fig.savefig('figures/RN-Rejection-Method_Pi-Determination.pdf', format='pdf')

# accuracy of the result as a function of the number of randomly selected points
N = np.linspace(10,int(1e5),1000)   # number of randomly selected points
rel_error = np.zeros(len(N))        # array for the accuracy of the result

for i in range(0,len(N)):
    N[i] = int(N[i]); pi = 0.
    for _ in range(0,10):
        pi += pi_rejection_method(int(N[i]),quadrant_function)[0]
    rel_error[i] = abs((pi/10.)-const.pi)/abs(const.pi)

fig, ax = plt.subplots()
ax.set_title(r'Rejection Method: Accuracy of the Determination of $\pi$')
ax.set_xlabel(r'number $N$ of randomly selected points')
ax.set_ylabel(r'relative error [$\%$]')
ax.plot(N, 100*rel_error, 'b.', markersize=2)
ax.grid(); ax.set_xscale('log'); ax.set_yscale('log')
fig.savefig('figures/RN-Rejection-Method_Accuracy-Analysis.pdf', format='pdf')
plt.show(); plt.clf(); plt.close()
```