IX1501 HT24 Project 1

Probability distribution of a sum of independent random variables

1. Implementation and Assessment

The course includes three mandatory project tasks on a total of 3.5 hp. They will be given a summary grade in Pass/Fail (G/U). In project 1 and 2, you will work in a group of two and solve computer-based tasks, write a report (English), and prepare a short oral presentation on your solution to the tasks (English). Carefully read the following instructions so that you know which rules apply and what is expected from you.

1.1. Report

The report should be written in English and contain the following:

- Title and authors of the report
- KTH emails of the authors
- Summary of the results and findings (maximum 250 words)
- Separate sections for each part containing e.g.,
    - Description of the methodology
    - Mathematical formulas and equations (if relevant)
    - Numerical results (including figures and/or tables)
    - Analysis and discussion
- Separate code section (do not mix the code and the main text in the report)
    - Code must contain comments detailed enough for an easy understanding.

The report must be uploaded before the deadline (see Section 1.4 below). The file type for the report is limited to pdf. Note that the **report is a group task**.

1.2. Oral presentation

You should prepare an oral presentation of your solution to the task. You should record your presentation, create a video file, and upload the file before the deadline. Instructions on how to create a recorded presentation can be found on a Canvas page under Projektuppgifter module. It is important that your face (video, not a still image) must be overlaid to the presentation material in your video. The video ***must not exceed five minutes***. Also, you should make sure that the uploaded file has adequate audio-visual quality. Carefully consider what is important, in what order and how it will be illustrated. English shall be used for the presentation. Note that the **video presentation is an individual task**.

1.3. Rules

Although it is a team project, **you, as an individual, must have full knowledge of all the material you submit and present**. To be approved, you must have solved the task and be able to explain the entire task

and solution. The project report will be prepared and uploaded per group of two students. On the other hand, the presentation material and video should be prepared and uploaded individually.

To account for the task that you do not have solved is considered cheating. It is also cheating to copy the whole or a part of a solution from others. If two solutions are deemed as (partial) copies, both will be rejected. If the solution contains parts that you do not have produced, e.g., background material, you must clearly indicate this and specify the source. Suspicion of cheating or misbehaving can be reported to the Disciplinary Board.

1.4. Examination

The project report and video should be uploaded before the deadline via Canvas.

If your report and/or video is not satisfactory, you will be requested to participate in an additional Zoom Q&A session on the presentation (redovisning) date. In addition, you can be randomly selected for the Q&A regardless of the quality of your work. Therefore, all students should be available for a Zoom meeting on the presentation (redovisning) date. The exact time will be announced a few hours before the session if you are selected. You may be requested to demonstrate and execute the code during the session.

Notice that the report and video should be uploaded in time even if you cannot attend the Zoom Q&A session. If you are not available on the redovisning date, you must inform the teacher, **Zhenyu Li** (zhenyuli@kth.se) with cc-ing Ki Won Sung (sungkw@kth.se), by email in advance.

If you do not fulfill any of the requirements without contacting the teachers in advance (in writing), you will have to wait for the next course for a new project exam.

2. Project Task

📣 Read Section 3 Background Knowledge carefully before delving into the task!

📣 As for the programming language, you are allowed to use Python or Mathematica for this project.

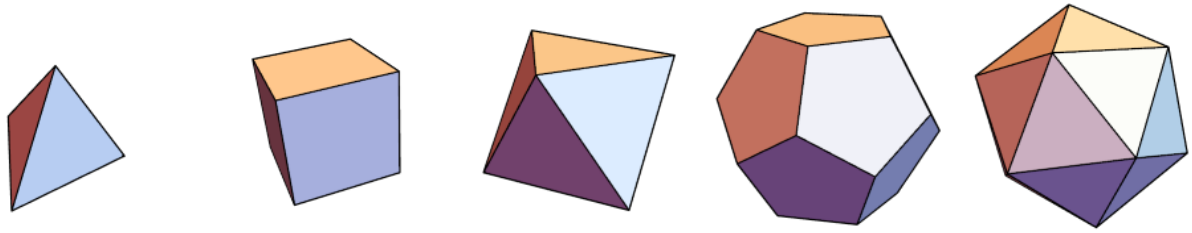


Figure 1: An illustration of platonic solids

At a dice game you throw five dice in the form of the Platonic solids as illustrated in Figure 1. The dice are numbered with integers $1, 2, \cdots, n$ where n is denotes the number of faces on each dice. The tetrahedron has its result number (1-4) written at the vertex pointing up. All other dice have the result number on the side facing up. You win the game if the sum of the numbers S satisfies $S \leq 10$ or $S \geq 45$. Note that S is an integer value ranging between 5 and 50.

Your task is as follows:

By means of the **discrete convolution method**,

[**Task 1**] Determine the probability function of S. Express it as a form of a table below.

s	$P(S=s)$
5	
6	
…	
49	
50	

[**Task 2**] Determine the probability of winning the game.

By means of the **Monte Carlo simulation**,

[**Task 3**] Obtain the probability of winning the game with 1000 trials.

[**Task 4**] Discuss how the probability changes when the number of trials changes from a small number (e.g., 10) to a large number (e.g., 100000). Provide a figure with the discussion.

[**Task 5**] How many trials are needed to keep the relative error between the exact value and the simulation result less than 10%? Provide your reasoning.

Note that the discrete convolution shall be used for tasks 1 and 2, while Monte Carlo simulation shall be used for tasks 3-5.

3. Background Knowledge

3.1 Discrete convolution

The distribution of a sum of two independent discrete random variables $S = X + Y$ can be determined by the following reasoning. If $S = s$ and we have $X = i$, then clearly $Y = s - i$. Since X and Y are independent, the probability is given by

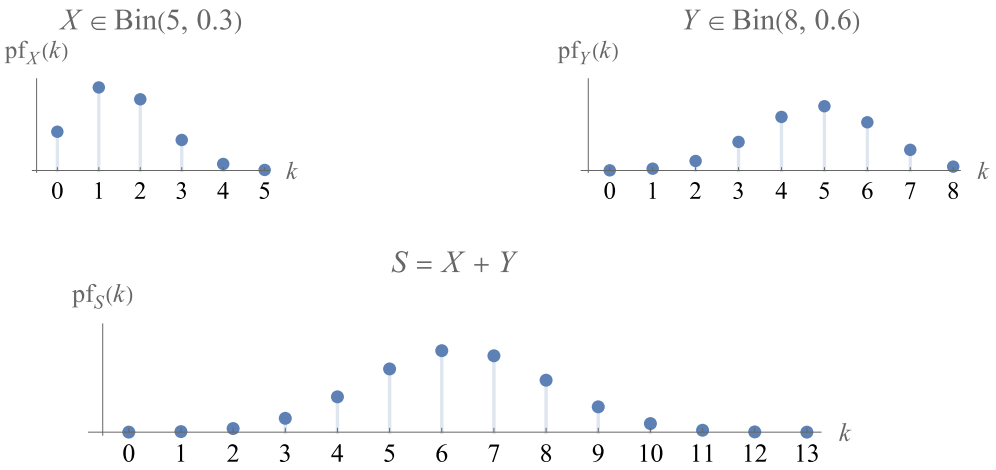
$$P(X = i \cap Y = s - i) = P(X = i)P(Y = s - i) = \mathrm{pf}_X(i)\mathrm{pf}_Y(s - i).$$

By adding up all the possible values for i, we obtain the probability of $S = s$ as follows:

$$P(S = s) = \mathrm{pf}_S(s) = \sum_{\forall i} \mathrm{pf}_X(s - i)\mathrm{pf}_Y(i).$$

This sum is called a **convolution sum**. The operation of convolution is usually denoted by an asterisk such that $\mathrm{pf}_S(s) = \left(\mathrm{pf}_X * \mathrm{pf}_Y\right)(s)$.

The example below illustrates the convolution of two independent random variables, X and Y, following binomial distributions.



k	0	1	2	3	4	5
$\text{pf}_X(k)$	0.168	0.360	0.309	0.132	0.028	0.002

k	0	1	2	3	4	5	6	7	8
$\text{pf}_Y(k)$	0.001	0.008	0.041	0.124	0.232	0.279	0.209	0.090	0.017

k	0	1	2	3	4	5	6
$\text{pf}_S(k)$	$1.1*10^{-4}$	$1.6*10^{-3}$	0.010	0.038	0.097	0.174	0.225
k	7	8	9	10	11	12	13
$\text{pf}_S(k)$	0.211	0.143	0.070	0.024	$5.3*10^{-3}$	$6.9*10^{-4}$	$4.1*10^{-5}$

If we have more than two random variables, for example $V = X + Y + Z$, we can obtain the probability distribution of $S = X + Y$ first, and then apply the convolution for $V = S + Z$.

Programming languages have built-in functions for discrete convolution. For example,

- Python has *numpy.convolve*. See this link for details.
- Mathematica has *DiscreteConvolve*. See this link for details.

3.2. Monte Carlo simulation

Monte Carlo simulation is a method of obtaining numerical results by means of repeated independent random trials. It is useful when the system can be modeled by a function of random variables, e.g., queueing systems. Interested readers are referred to the following Wikipedia articles for further information: Monte Carlo method and Queueing theory. Here, we will consider two simple examples to learn about the concept of the Monte Carlo simulation.

In the first example, consider a random variable X following the normal distribution with $\mu = 2$ and $\sigma = 3$, i.e., $X \in N(2,3)$. We want to obtain $P(X > 2.7)$. You will learn how to calculate the probability in F7 (Chapter 6 in Blom) and the answer is approximately **0.409**. However, you should rely on Monte Carlo simulation for now. The simulation can be conducted in the following manner:

- Generate n independent random values from $N(2,3)$
- Count the number of trials where the value is greater than 2.7. Let c denote the number.
- $P(X > 2.7) = c/n$.

Below is a Python code for this example. Note that the result changes every time you run the simulation unless you fix the seed of the random number generator.

```
from numpy import random
import numpy as np

# Set the number of independent trials (n)
NumberOfTrials = 1000
# Variable recording number of trials greater than 2.7 (c)
RecordResult = 0

for j in range (NumberOfTrials):
    # generate random number
    X = random.normal(2,3)
    if X>2.7:
        RecordResult = RecordResult + 1

# P (X>2.7) = c/n
Probability = RecordResult/NumberOfTrials

print('P(X>2.7)=', Probability)
```

```
P(X>2.7)= 0.405
```

In the second example, you throw two fair dice. For each dice, you get one point if an odd number comes out and zero point for even numbers. Let X and Y be the results of the first and second dice, respectively. We want to obtain the probability function of the product of X and Y, i.e., $Z = XY$.

Note that X and Y are independent and have the same probability function as follows:

$$\mathrm{pf}_X(k) = \mathrm{pf}_Y(k) = \begin{cases} 1/2, & k = 0 \\ 1/2, & k = 1 \end{cases}$$

Thus, we can easily determine that Z has the following probability function:

$$\mathrm{pf}_Z(k) = \begin{cases} 3/4, & k = 0 \\ 1/4, & k = 1 \end{cases}.$$

Assume that you don't know the probability function and use Monte Carlo simulation to obtain it. It can be done by the following steps:

- For each n independent trials,
  - Generate two discrete uniform random numbers between 1 and 6.
  - Calculate X, Y, and Z.
- Count the number of trials ending up with $Z=0$ and $Z=1$, denoted by c_1 and c_2, respectively.
- $P(Z=0)=c_1/n$ and $P(Z=1)=c_2/n$.

Below is a Python code for the second example.

```
from numpy import random
import numpy as np

# Set the number of independent trials (n)
NumberOfTrials = 10
# This is the list storing the result Z
RecordResult = []

for j in range (NumberOfTrials):
    # randint(a,b) generates discrete uniform random number between a and b-1
    # % is the modulo operator, i.e., it returns the remainder of the division
    X = random.randint(1,7) % 2
    Y = random.randint(1,7) % 2
    Z = X*Y
    # record the result Z onto the list
    RecordResult.append(Z)

# P(Z=0) = c1/n, P(Z=1)=c2/n
Probability = [RecordResult.count(0)/NumberOfTrials, RecordResult.count(1)/
 ↪NumberOfTrials]

print('P(Z=0)=', Probability[0], 'P(Z=1)=', Probability[1])
```

```
P(Z=0)= 0.6 P(Z=1)= 0.4
```

It is observed that $n=10$ is not sufficient to obtain a reliable result. By increasing n to 1000, we obtain $P(Z=0)=0.76$ and $P(Z=1)=0.24$, which is much closer to the exact result.

In Monte Carlo simulation, **it is important to select sufficiently large number of trials**. You will learn how to determine the required number of trials later in this course. In this project, you would need to find the suitable n by multiple experiments and observation.