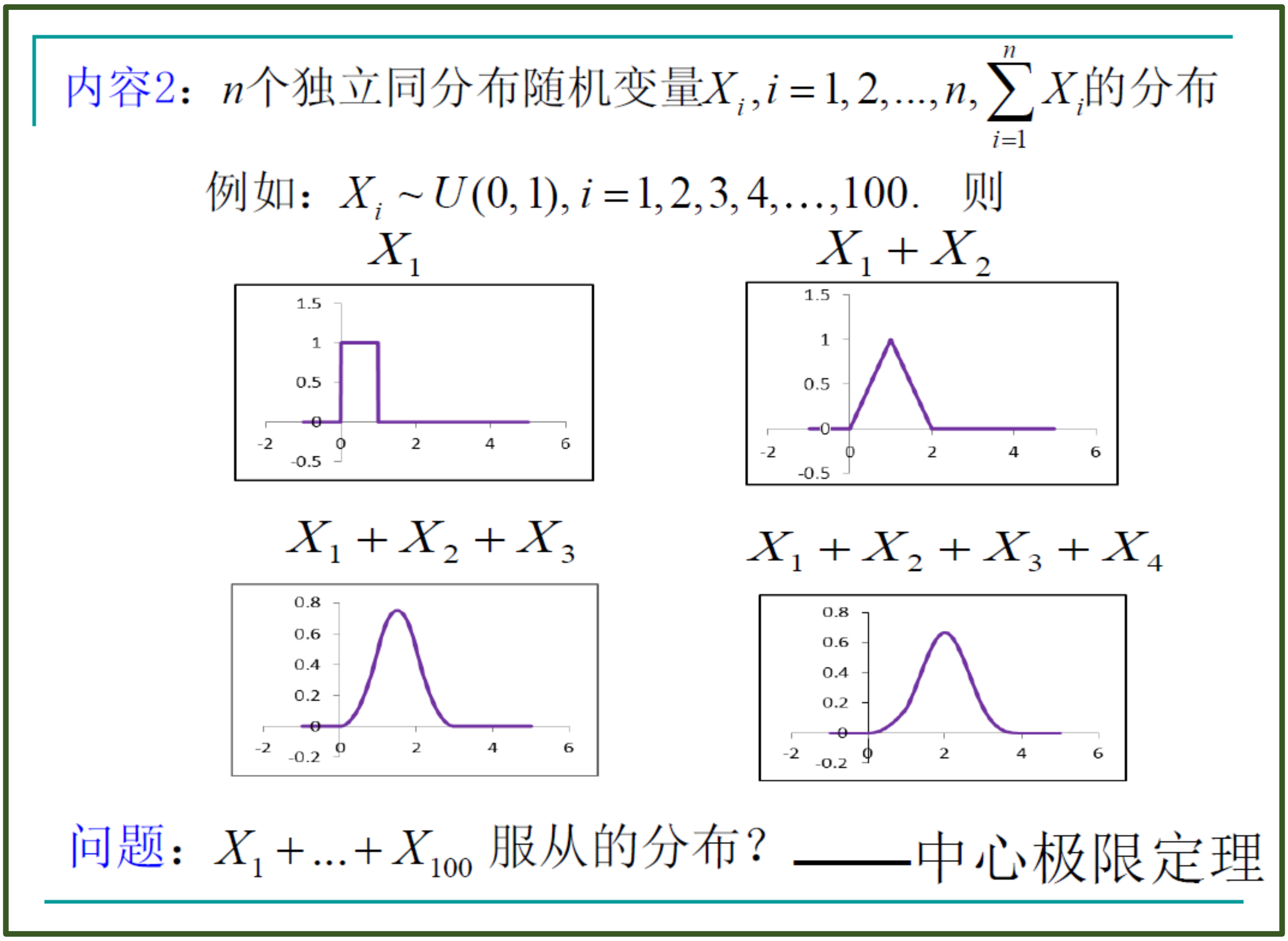
Numerical experiment aimed at showing that the distribution of the sum of independent random variables with the same distribution tends to a normal distribution



X_i are random variables with the uniform distribution $U(x)=\begin{cases}1, & x\in[0,1]\\ 0, & x\notin[0,1]\end{cases}$ in the interval $[0,1]$. The sum $Z=\sum_{i=1}^{n} X_i$ has the distribution $U_Z=\underbrace{U*U*\dots*U}_{n}$ composed of the convolutions. To find U_Z, we will use the direct and inverse Fourier transforms:

$$\widehat{U}_Z(p)=\int_{-\infty}^{+\infty} U_Z(x)\,exp(-ipx)dx=\int_{-\infty}^{+\infty} U*U*\dots*U(x)\,exp(-ipx)dx=\left(\int_{-\infty}^{+\infty} U(x)\,exp(-ipx)dx\right)^n=$$

$$=\left(\int_0^1 exp(-ipx)dx\right)^n=\left(\frac{exp(-ipx)}{-ip}\Big|_0^1\right)^n=\frac{i^n}{p^n}(exp(-ip)-1)^n \qquad (1)$$

Therefore:

$$U_Z(x)=\frac{1}{2\pi}\int_{-\infty}^{+\infty}\widehat{U}_Z(p)\,exp(ipx)dp=\frac{i^n}{2\pi}\int_{-\infty}^{+\infty}\frac{(exp(-ip)-1)^n}{p^n}exp(ipx)dp \qquad (2)$$

Since the final function must be real, for numerical calculations we can use only the real part of the integral kernel:

$$U_Z(x) = \frac{1}{2\pi}\int_{-\infty}^{+\infty} Re\left[\frac{i^n(exp(-ip)-1)^n}{p^n} exp(ipx)\right] dp \tag{3}$$

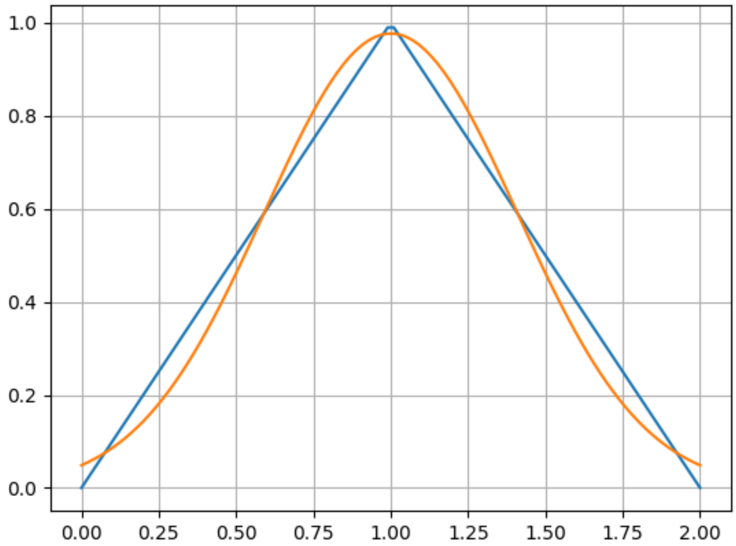
This integration was programmed in Python using the “integrate.quad” function from the “scipy” library. For operations with complex numbers, we used the “cmath” library. The distribution $U_Z(x)$ was calculated for different $n \geq 2$ and compared with the normal distribution:

$$N(x) = \frac{1}{\sigma\sqrt{2\pi}} exp\left(-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}\right) \tag{4}$$

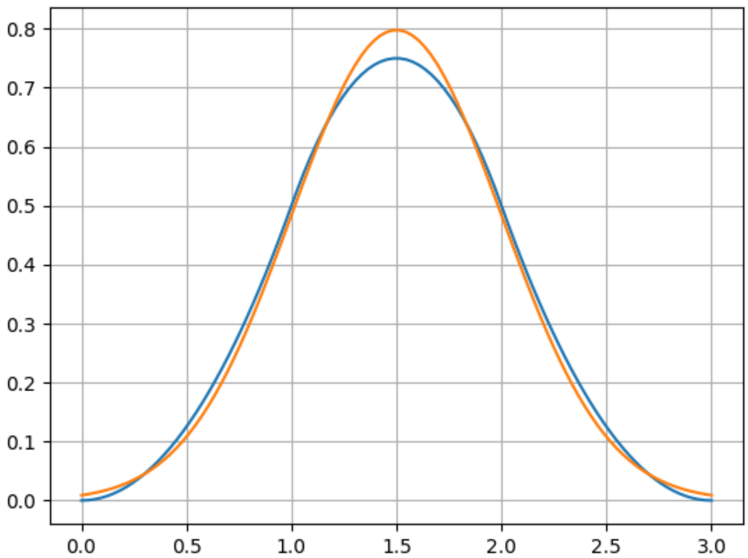
where $\mu = n\mu_U$, $\sigma = \sqrt{n\sigma_U^2}$, $\mu_U = 0.5$ and $\sigma_U = \sqrt{1/12}$ are [the mean and standard deviation]() of the uniform distribution on the interval $[0,1]$.

```
from scipy import integrate
from cmath import exp
import numpy as np
import matplotlib.pyplot as plt

# 2 <= n is the number of independent uniformly distributed variables
n = 4

j = complex(0.0, 1.0)  # imaginary unit
mu = 0.5 * n  # effective mu of the sum of independent uniformly distributed variables
sigma = ((1.0 / 12.0) * n)**0.5  # effective standard deviation of independent uniformly distributed variables

# Kernel for the inverse Fourier transform
def imag(p,x):
    return np.real((j**n / (2.0 * np.pi * p**n)) * (exp(-j * p) - 1.0)**n * exp(j * p * x))

x = np.linspace(0, n, num=100)  # x-points for the graphs
y1 = []
y2 = []
for i in range(0, 100):
    val1, error = integrate.quad(imag, -np.inf, np.inf, args=(x[i],), limit=5000, limlst=5000,
                                 maxp1=5000, epsabs=1.5e-5, epsrel=1.5e-5)  # real part of the inverse Fourier transform
    # +/-np.inf are the infinite integration limits
    val2 = (1.0 / (sigma * (2.0 * np.pi)**0.5)) * np.real(exp(-0.5 * (x[i] - mu)**2 / sigma**2))
    y1.append(val1)
    y2.append(val2)

plt.plot(x, y1)  # distribution of the sum of independent uniformly distributed variables
plt.plot(x, y2)  # normal distribution
plt.grid()
plt.show()

```

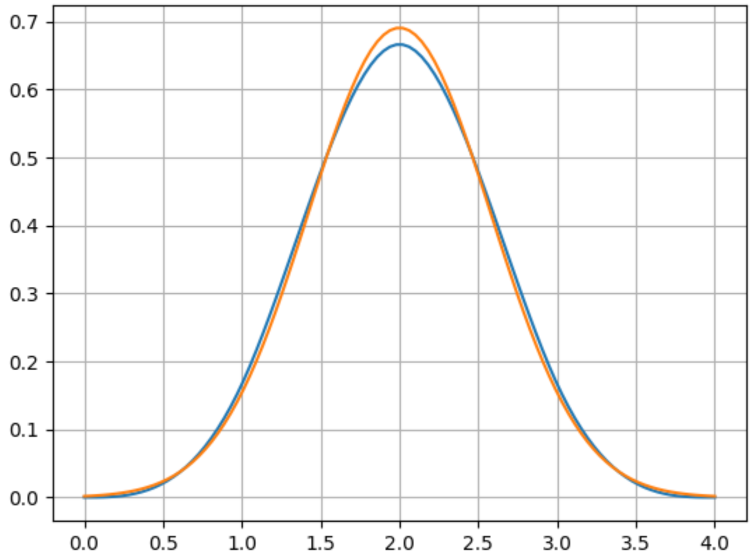
$U_Z(x)$ and $N(x)$ for $n = 2$



$U_Z(x)$ and $N(x)$ for $n = 3$



$U_Z(x)$ and $N(x)$ for $n = 4$



$U_Z(x)$ and $N(x)$ for $n = 10$