ECE600 Computer Problem (Extra Credit)

@author: Zhankun Luo

@email: luo333@purdue.edu

Problem 1

Let $X_1, \cdots, X_n \overset{iid}{\sim} U[0,1]$, and

$$S_n = \sum_{i=1}^{n} X_i, \quad Z_n = \frac{S_n - n\mu}{\sqrt{n}\sigma}$$

Where $\mathbb{E}[X_n] = \mu$, $\mathrm{Var}[X_n] = \sigma^2$

(a) find μ, σ^2

(b) show that $\mathbb{E}[Z_n] = 0$, $\mathrm{Var}[Z_n] = 1$

(c) find and plot $f_{Z_3}(z)$

solution

(a) write PDF of X_n, $f_{X_n}(x) = I_{0<x<1}$

$$\mu = \mathbb{E}[X_n] = \int_{-\infty}^{\infty} f_{X_n}(x)\mathrm{d}x = \int_0^1 x\mathrm{d}x = \frac{1}{2}$$

$$\sigma^2 = \mathrm{Var}[X_n] = \mathbb{E}[X_n^2] - \mu^2 = \int_0^1 x^2\mathrm{d}x - \left(\frac{1}{2}\right)^2 = \frac{1}{12}$$

(b) notice for $X_1, \cdots, X_n \overset{iid}{\sim} U[0,1]$, $\mathbb{E}[X_i] = \mu, i = 1, \cdots, n$,

$$\mathbb{E}[Z_n] = \frac{\sum_{i=1}^{n} \mathbb{E}[X_i] - n\mu}{\sqrt{n}\sigma} = \frac{n\mu - n\mu}{\sqrt{n}\sigma} = 0$$

since $\mathrm{Var}[aX+b] = a^2\mathrm{Var}[X]$, let $a = \frac{1}{\sqrt{n}\sigma}, b = \sqrt{n}\frac{\mu}{\sigma}, X = S_n$

notice for independent random variables $X_1, \cdots, X_n$, $\mathrm{Var}[\sum_{i=1}^n X_i] = \sum_{i=1}^n \mathrm{Var}[X_i]$, and $\mathrm{Var}[X_i] = \sigma^2, i = 1, \cdots, n$

$$\mathrm{Var}[Z_n] = \frac{\mathrm{Var}[\sum_{i=1}^n X_i]}{n\sigma^2} = \frac{\sum_{i=1}^n \mathrm{Var}[X_i]}{n\sigma^2} = \frac{n\sigma^2}{n\sigma^2} = 1$$

(c) Actually, $S_n = \sum_{i=1}^n X_i$ follows **Irwin–Hall distribution** with PDF

see https://www.randomservices.org/random/special/IrwinHall.html for more details of this distribution

$$\begin{aligned} f_{S_n}(x) &= \frac{1}{(n-1)!} \sum_{k=0}^{n} (-1)^k \binom{n}{k} (x-k)^{n-1} I_{x \geq k} \\ &= \frac{1}{(n-1)!} \sum_{k=0}^{\lfloor x \rfloor} (-1)^k \binom{n}{k} (x-k)^{n-1} \end{aligned}$$

let's prove it for $\forall n \in \mathbb{N}^+$, check $n = 1$, it obviously holds

$$\begin{aligned} f_{S_1}(x) &= f_{X_1}(x) = I_{x \geq 0} - I_{x \geq 1} \\ &= \left[\frac{1}{(n-1)!} \sum_{k=0}^{n} (-1)^k \binom{n}{k} (x-k)^{n-1} I_{x \geq k} \right] \Big|_{n=1} \end{aligned}$$

If it holds for n, then we will show it also hold for $n+1$, notice $S_{n+1} = S_n + X_{n+1}$ and S_n, X_{n+1} are independent

$$f_{S_{n+1}}(x) = (f_{S_n} * f_{X_{n+1}})(x) = f_{S_n} * I_{0<x<1} \qquad = \int_{x-1}^{x} f_{S_n}(x') \mathrm{d}x'$$

notice $f_{S_n}(x) = 0$ when $x < 0$ or $x \geq n$, so we have $f_{S_{n+1}}(x) = 0$ when $x < 0$ or $x - 1 \geq n$

let's only consider $j = \lfloor x \rfloor, j \leq x < j+1, j \in \{0, \cdots, n\}$, take the integral into two parts

$$f_{S_{n+1}}(x) = \int_{x-1}^{j} f_{S_n}(x') \mathrm{d}x' + \int_{j}^{x} f_{S_n}(x') \mathrm{d}x'$$

since, the PDF expression for S_n holds for n, notice $\lfloor x' \rfloor = j - 1, x' \in [x-1, j)$

let $z = x' - (j-1)$

$$
\begin{aligned}
\int_{x-1}^{j} f_{S_n}(x')\mathrm{d}x' &= \int_{x-1}^{j} \frac{1}{(n-1)!}\sum_{k=0}^{j-1}(-1)^k\binom{n}{k}(x'-k)^{n-1}\mathrm{d}x' \\
&= \int_{x-j}^{1} \frac{1}{(n-1)!}\sum_{k=0}^{j-1}(-1)^k\binom{n}{k}(z+(j-k-1))^{n-1}\mathrm{d}z \\
&= \frac{1}{n!}\sum_{k=0}^{j-1}(-1)^k\binom{n}{k}(j-k)^n - \frac{1}{n!}\sum_{k=0}^{j-1}(-1)^k\binom{n}{k}(x-1-k)^n
\end{aligned}
$$

Similarly, $\lfloor x' \rfloor = j, x' \in [j, x)$, let $z = x' - j$

$$
\begin{aligned}
\int_{j}^{x} f_{S_n}(x')\mathrm{d}x' &= \int_{j}^{x} \frac{1}{(n-1)!}\sum_{k=0}^{j}(-1)^k\binom{n}{k}(x'-k)^{n-1}\mathrm{d}x' \\
&= \int_{0}^{x-j} \frac{1}{(n-1)!}\sum_{k=0}^{j}(-1)^k\binom{n}{k}(z+(j-k))^{n-1}\mathrm{d}z \\
&= \frac{1}{n!}\sum_{k=0}^{j}(-1)^k\binom{n}{k}(x-k)^n - \frac{1}{n!}\sum_{k=0}^{j}(-1)^k\binom{n}{k}(j-k)^n
\end{aligned}
$$

notice $\binom{n}{k-1} + \binom{n}{k} = \binom{n+1}{k}$ holds for $k = 1, \cdots, n$, and notice $j = \lfloor x \rfloor$

$$
\begin{aligned}
f_{S_{n+1}}(x) &= -\frac{1}{n!}\sum_{k=0}^{j-1}(-1)^k\binom{n}{k}(x-1-k)^n \\
&+ \frac{1}{n!}\sum_{k=0}^{j}(-1)^k\binom{n}{k}(x-k)^n \\
&= \frac{1}{n!}\left[\sum_{k=1}^{j}(-1)^k\left[\binom{n}{k-1}+\binom{n}{k}\right](x-k)^k + x^n\right] \\
&= \frac{1}{(n)!}\sum_{k=0}^{\lfloor x \rfloor}(-1)^k\binom{n+1}{k}(x-k)^n
\end{aligned}
$$

Thus, we prove that the PDF expression of S_{n+1} also holds for $n+1$

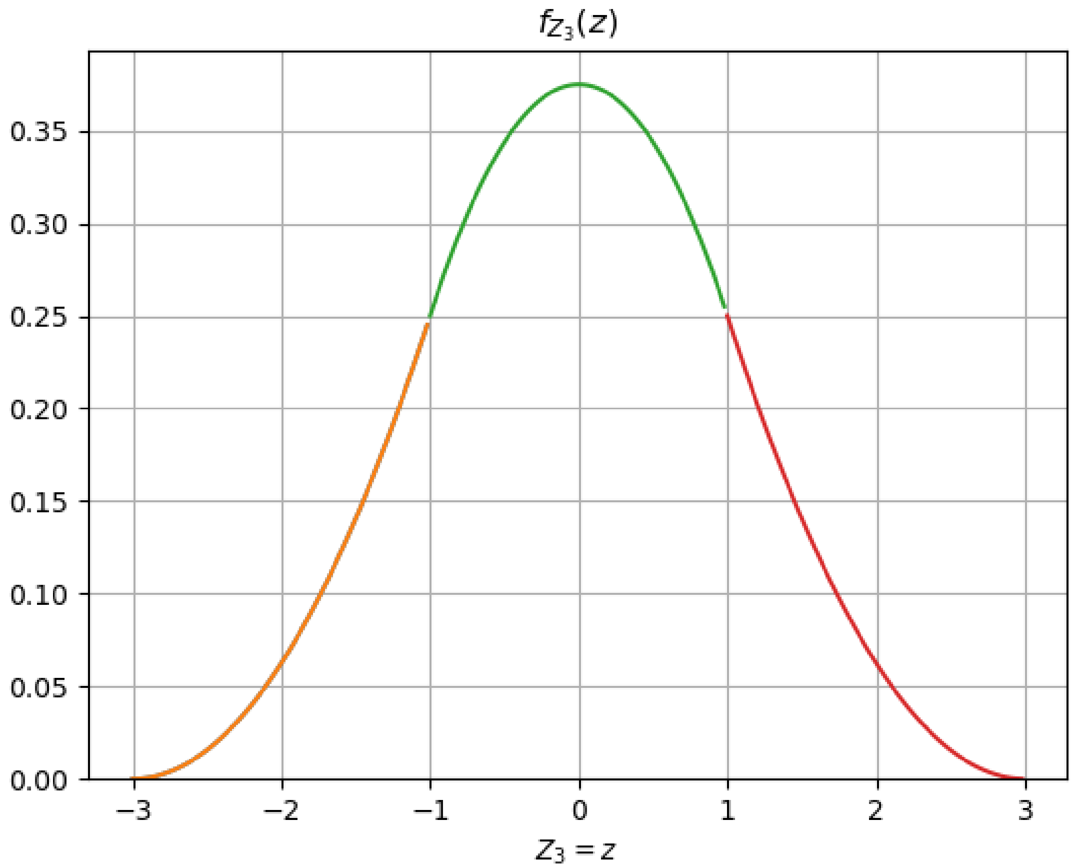
By set $n = 3$, we obtain the close form of $f_{S_3}(x)$

$$
\begin{aligned}
f_{S_3}(x) &= \frac{1}{(3-1)!}\sum_{k=0}^{\lfloor x \rfloor}(-1)^k\binom{3}{k}(x-k)^{3-1} \\
&= \begin{cases}
\frac{1}{2}x^2, & 0 \le x \le 1 \\
\frac{1}{2}x^2 - \frac{3}{2}(x-1)^2, & 1 \le x \le 2 \\
\frac{1}{2}x^2 - \frac{3}{2}(x-1)^2 + \frac{3}{2}(x-2)^2, & 2 \le x \le 3
\end{cases}
\end{aligned}
$$

So, $\quad Z_3 = \frac{S_3 - 3\mu}{\sqrt{3}\sigma} = 2(S_3 - \frac{3}{2}) = 2S_3 - 3, z = 2x - 3$ where $\mu = \frac{1}{2}, \sigma = \frac{1}{2\sqrt{3}}$

thus $S_3 = \frac{Z_3+3}{2}, x = \frac{z+3}{2}$, the PDF of Z_3 would be

$$
\begin{aligned}
f_{Z_3}(z) &= f_{S_3}(x)|_{x=\frac{z+3}{2}} \cdot |\frac{\mathrm{d}x}{\mathrm{d}z}| = \frac{1}{2} f_{S_3}(x)|_{x=\frac{z+3}{2}} \\
&= \begin{cases} \frac{1}{4}\left(\frac{z+3}{2}\right)^2, & -3 \le z \le -1 \\ \frac{1}{4}\left(\frac{z+3}{2}\right)^2 - \frac{3}{4}((\frac{z+3}{2}) - 1)^2, & -1 \le z \le 1 \\ \frac{1}{4}\left(\frac{z+3}{2}\right)^2 - \frac{3}{4}((\frac{z+3}{2}) - 1)^2 + \frac{3}{4}((\frac{z+3}{2}) - 2)^2, & 1 \le z \le 3 \end{cases} \\
&= \begin{cases} \frac{z^2}{16} + \frac{3z}{8} + \frac{9}{16}, & -3 \le z \le -1 \\ \frac{3}{8} - \frac{z^2}{8}, & -1 \le z \le 1 \\ \frac{z^2}{16} - \frac{3z}{8} + \frac{9}{16}, & 1 \le z \le 3 \end{cases}
\end{aligned}
$$



the script to generate this figure is

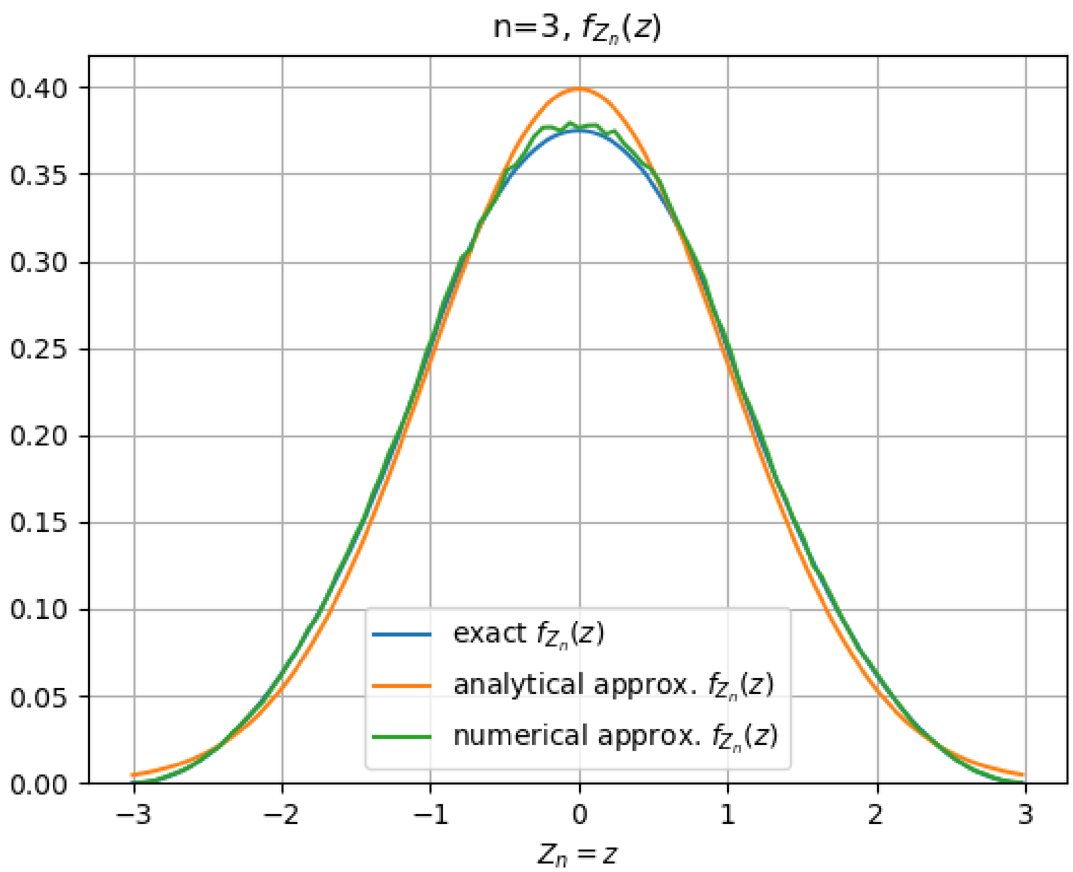
```
import matplotlib.pyplot as plt
import numpy as np
if __name__ == "__main__":
    f1 = lambda z: z*z/16.0 + 3*z/8.0 + 9/16.0
    f2 = lambda z: 3/8.0 - z*z/8.0
    f3 = lambda z: z*z/16.0 - 3*z/8.0 + 9/16.0
```

```
    z1 = np.linspace(-3, -1, num=100, endpoint=False)
    z2 = np.linspace(-1, 1, num=100, endpoint=False)
    z3 = np.linspace(1, 3, num=100, endpoint=False)
    y1, y2, y3 = [list(map(f, z)) for (f, z) in list(zip([f1, f2,
f3], [z1, z2, z3]))]
    plt.plot(z1, y1)
    list(plt.plot(z, y) for z, y in list(zip([z1, z2, z3], [y1, y2,
y3])))
    plt.ylim([0, None])
    plt.grid()
    plt.xlabel(r"$Z_3=z$")
    plt.title(r"$f_{Z_3}(z)$")
    plt.savefig("./p1_c.png")
    plt.show()
```

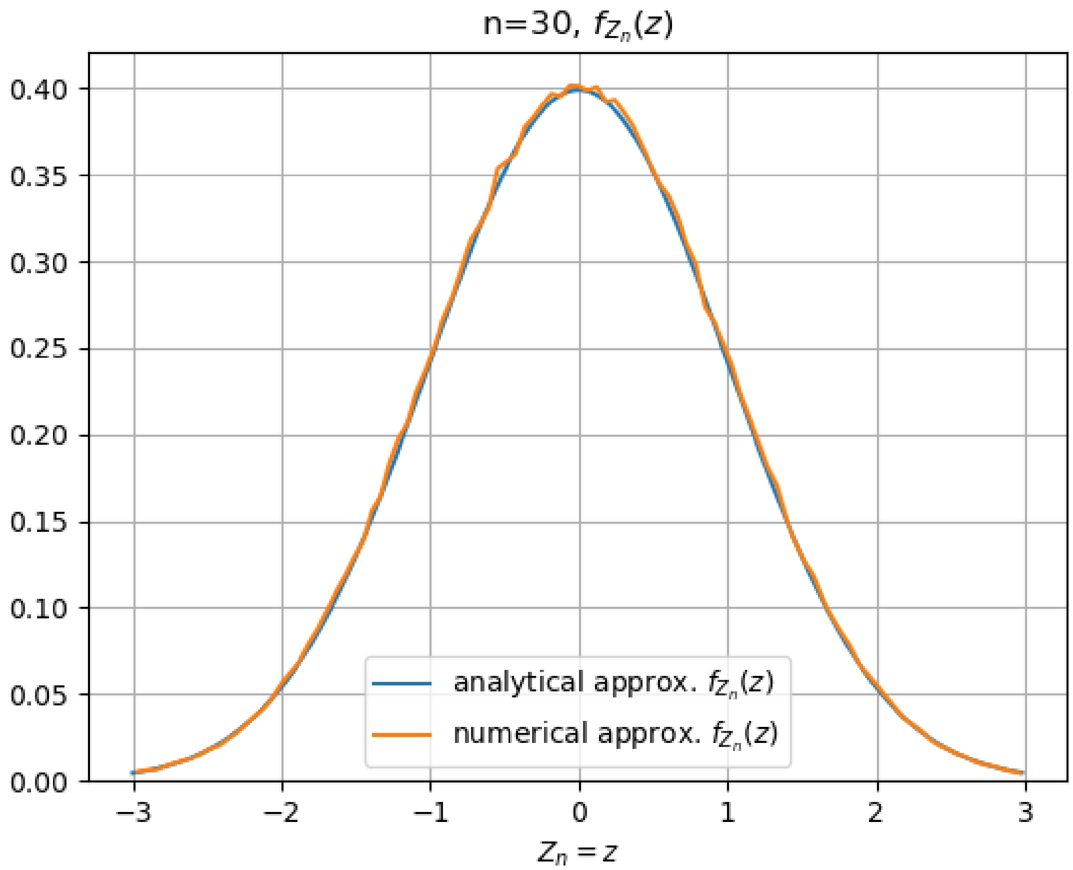
Problem 2

solution

we can plot the figures for $n = 3$ as below



we can plot the figures for $n = 30$ as below



the script to generate these figures is

```
import numpy as np
import matplotlib.pyplot as plt
from numba import njit # with gpu to accelerate computing

@njit
def runs(K, N=3):
    S = np.random.uniform(0, 1, size=(K, N))
    S = np.sum(S, axis=1)
    mu, sigma = 0.5, 0.5/np.sqrt(3)
    Z = (S - N*mu) / (np.sqrt(N)*sigma)
    return Z

def plot_approx(N=3):
    f_norm = lambda z: np.exp(-z*z/2.0) / np.sqrt(2*np.pi)
    y_approx_analy = f_norm(z_actual)
    plt.plot(z_actual, y_approx_analy, label=r"analytical approx.
$f_{Z_n}(z)$")
    # simulation
    K = int(1e6)
```

```
    Z = runs(K, N=N)
    num_bin = 100
    z_bin = np.linspace(-3, 3, num=num_bin, endpoint=True)
    hist, _ = np.histogram(Z, bins=z_bin)
    z_bin = 0.5 * (z_bin[:-1] + z_bin[1:])
    y_approx_numer = num_bin * hist / (6*K)
    plt.plot(z_bin, y_approx_numer, label=r"numerical approx.
$f_{Z_n}(z)$")
    plt.legend()
    plt.ylim([0, None])
    plt.grid()
    plt.xlabel(r"$Z_n=z$")
    plt.title(f"n={N}, " + r"$f_{Z_n}(z)$")
    plt.savefig(f"./p2_n={N}.png")
    plt.show()

if __name__ == "__main__":
    """n = 3"""
    f1 = lambda z: z*z/16.0 + 3*z/8.0 + 9/16.0
    f2 = lambda z: 3/8.0 - z*z/8.0
    f3 = lambda z: z*z/16.0 - 3*z/8.0 + 9/16.0
    z1 = np.linspace(-3, -1, num=100, endpoint=False)
    z2 = np.linspace(-1, 1, num=100, endpoint=False)
    z3 = np.linspace(1, 3, num=100, endpoint=False)
    y1, y2, y3 = [list(map(f, z)) for (f, z) in list(zip([f1, f2,
f3], [z1, z2, z3]))]
    y_actual = y1 + y2 + y3
    z_actual = np.concatenate((z1, z2, z3))
    plt.plot(z_actual, y_actual, label=r"exact $f_{Z_n}(z)$")
    plot_approx(N=3)
    plot_approx(N=30)
```

Problem 3

To see how the approximations might be used, consider the following problem.

Suppose an messages arrive at a node where the message lengths are independent uniform random variables between 0 and 1 MB (this is an approximations as bits are of course discrete). These messages are to be stored on a hard drive or transmitted over a network.

For $n = 3$ find the probability that the total message length exceeds 2 MB using the analytical and numerical
approximations in part 2, and also the exact probability using part 1.

For $n = 30$ find the probability that the total message length exceeds 20 MB using the analytical and numerical approximations in part 2.

solution

For $n = 3$, $Z_3 = 2S_3 - 3$

$$\begin{aligned}
&\Pr(\text{total length} > 2\text{M}) = \Pr(S_3 > 2) = \Pr(Z_3 > 1) = \int_1^\infty f_{Z_3}(z)\mathrm{d}z \\
&= \int_1^3 \left(\frac{z^2}{16} - \frac{3z}{8} + \frac{9}{16}\right)\mathrm{d}z \\
&= \frac{z^3}{48} - \frac{3}{16}z^2 + \frac{9}{16}z\Big|_1^3 = \frac{26}{48} - \frac{3}{16}\cdot 8 + \frac{9}{16}\cdot 2 \\
&= \frac{13}{24} - \frac{3}{2} + \frac{9}{8} = \frac{13-36+27}{24} = \frac{1}{6}
\end{aligned}$$

The analytical approximation is

$$\begin{aligned}
&\Pr(\text{total length} > 2\text{M}) = \Pr(S_3 > 2) = \Pr(Z_3 > 1) \approx 1 - \Phi(1) = 1 - 0.841345 \\
&= 0.158655
\end{aligned}$$

The numerical approximation is

```
0.166833
```

For $n = 30$, $Z_{30} = \frac{(S_{30} - 30/2)}{\sqrt{30}\cdot\frac{1}{2\sqrt{3}}} = (2S_{30} - 30)/\sqrt{10}$

The analytical approximation is

$$\begin{aligned}
&\Pr(\text{total length} > 20\text{M}) = \Pr(S_{30} > 20) = \Pr(Z_{30} > \sqrt{10}) \approx 1 - \Phi(\sqrt{10}) = 1 - 0.99921 \\
&= 0.000783
\end{aligned}$$

The numerical approximation is

```
0.000688
```

The script to compute the numerical approximation is

```python
import numpy as np
import matplotlib.pyplot as plt
from numba import njit # with gpu to accelerate computing

@njit
def runs(K, N=3):
    S = np.random.uniform(0, 1, size=(K, N))
    S = np.sum(S, axis=1)
    return S

def compute_prob(N=3, threshold=2):
    K = int(1e6)
    S = runs(K, N)
    count = np.count_nonzero(S > threshold)
    return count / float(K)

if __name__ == "__main__":
    print(compute_prob(N=3, threshold=2))
    print(compute_prob(N=30, threshold=20))
```