Skr2-Oppg2

September 12, 2023

1 Skriftlig innlevering 2, Oppgave 2 (av 4)

1.1 Oppgave 2 *

La X være en stokastisk variabel som beskriver hvor lang tid en komponent har fungert i det den svikter. Vi kaller da X for levetiden for komponenten.

Levetiden X (målt i antall år) til en bestemt type mekaniske komponenter har vist seg å følge en fordeling med kumulativ fordelingsfunksjon gitt ved

$$F_X(x) = 1 - \exp\left\{-\frac{x^2}{\alpha}\right\}; x \geq 0,$$

der α er en parameter som beskriver kvaliteten til komponentene.

1.1.1 Deloppgave a)

- Finn sannsynlighetstettheten til X, $f_X(x)$. Eventuelt hent denne fra din besvarelse av Skriftlig innlevering 1.
- Finn en formel for $E[X]$ (som funksjon av α). Du kan her uten bevis benytte at

$$\int_{-\infty}^{\infty} e^{-x^2} dx = \sqrt{\pi}.$$

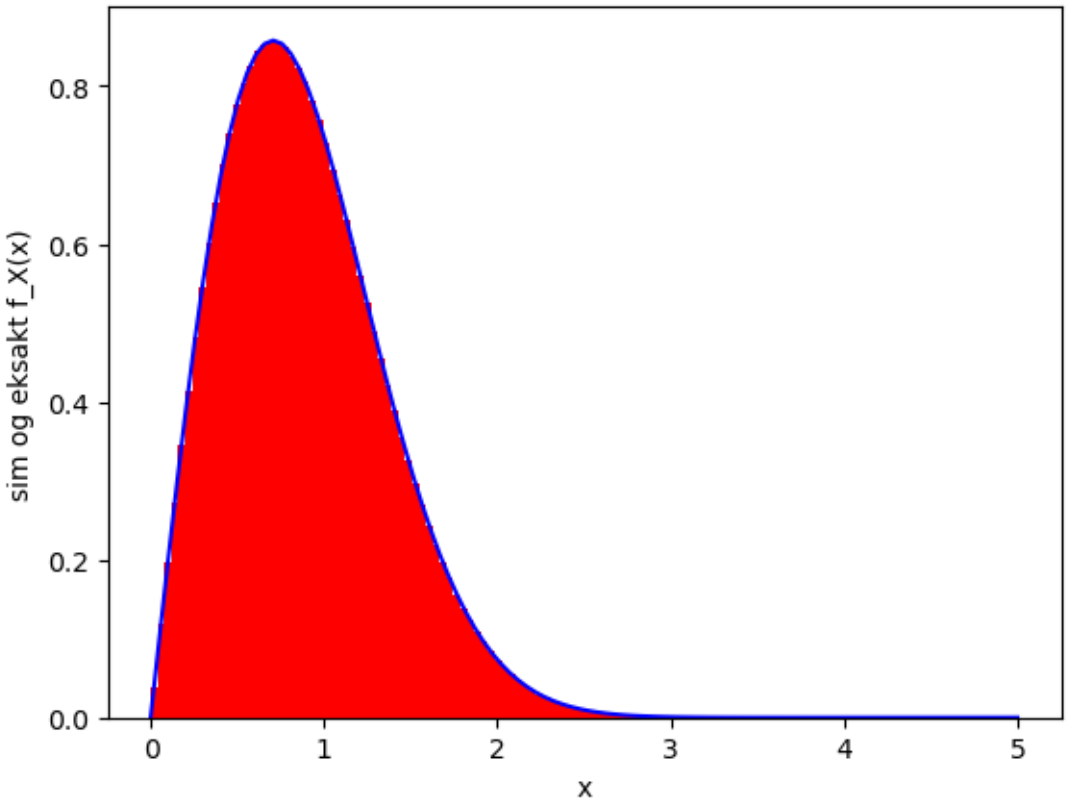
1.1.2 Deloppgave b)

La $U \sim \text{Unif}[0, 1]$.

- Finn en formel for hvordan man fra U kan definere X slik at kumulativ fordeling for X blir som angitt over.
- Skriv en python-funksjon som genererer n realisasjoner av X. La funksjonen ha to input-parametre, antall realisasjoner n og verdien til kvalitetsparameteren α. Benytt funksjonen til å generere (for eksempel) $n = 10\ 000\ 000$ realisasjoner av X med (for eksempel) $\alpha = 1$, og lag et sannsynlighetshistogram for de genererte verdiene. Spesifiser at histogrammet skal ha 100 intervaller, se kode under. Plott også sannsynlighetstettheten $f_X(x)$ i samme plott som sannsynlighetshistogrammet. Ser det ut til at du har generert realisasjoner av X på korrekt måte?

[2]:
```
import numpy as np
import matplotlib.pyplot as plt
```

```python
def simX(n,alpha):
    u = np.random.uniform(size=n) #array med n elementer.
    x = np.sqrt(-alpha*np.log(1-u))  # fyll inn formelen du fant for x

    return x



# Sett antall realisasjoner og verdien til alpha
n = 10000000
alpha = 1

# simuler realisasjoner av X
x = simX(n, alpha)

# Lag sannsynlighetshistogram for de simulerte verdiene
plt.hist(x, density=True,bins=100, color="red")  #density=True gjør at vi får␣
 ↪et sannsynlighetshistogram

# Angi navn på aksene
plt.xlabel("x")
plt.ylabel("sim og eksakt f_X(x)")


# Regn ut og plott sannsynlighetstettheten til X på samme plott
u = np.linspace(0,5,100)
f_x = lambda x : ((2*x)/alpha)*np.exp(-x**2/alpha)
plt.plot(u, f_x(u), color= "blue")

# Avslutt med å generere alle elementene du har plottet
plt.show()
```



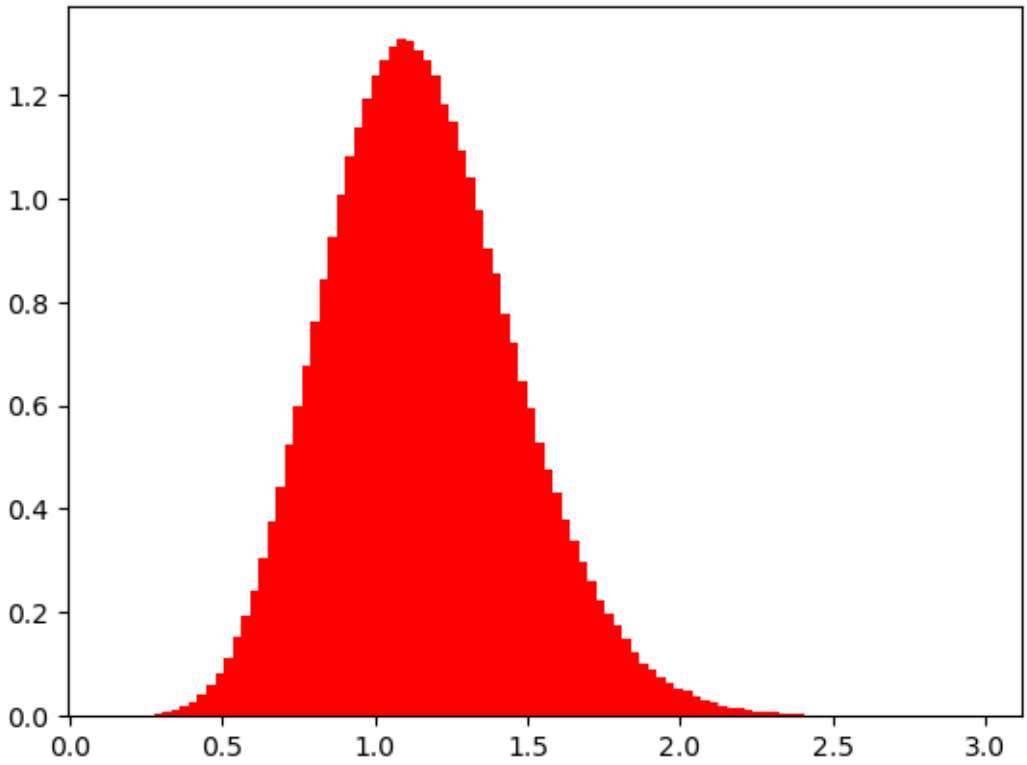
Her er deloppgave b) slutt. Et instrument inneholder fem komponenter av denne typen, to av disse komponentene har kvalitetsparameter $\alpha = 1$ og de andre tre komponentene har $\alpha = 1.2$. De fem komponentene svikter uavhengig av hverandre og instrumentet fungerer så lenge minst tre av de fem komponentene fungerer. La Y betegne levetiden til instrumentet.

2 Deloppgave c)

Skriv en python-funksjon som genererer n realisasjoner av Y. Funksjonen skal ha en input-parameter, nemlig antall realisasjoner n. Benytt funksjonen til å generere (for eksempel) $n = 10\ 000$ realisasjoner av Y, og lag et sannsynlighetsistogram for de genererte verdiene.

[36]:
```
import numpy as np
import matplotlib.pyplot as plt

# definer funksjonen og benytt denne som angitt i oppgaven
def simY(n):
    # generate list of 5 komponents of simX
    # after, iterate through them and find the time it fails, the second␣
 ↪highest X
    alphas = np.array([1]*2 + [1.2]*3)
```

```
    comps = []
    y = []
    for alpha in alphas:
        comps.append(simX(n, alpha))
    for i in range(n):
        largest = second = 0
        if (len(alphas) < 2):
            print("Invalid Input");
            return;

        # Find the largest element
        for compIndex in range(len(alphas)):
            largest = max(largest, comps[compIndex][i]);

        # Find the second largest element
        for compIndex in range(len(alphas)):
            if (comps[compIndex][i] != largest):
                second = max(second, comps[compIndex][i]);

        y.append(second)
    return y

# Sett antall realisasjoner
n = 1000000

# simuler realisasjoner av Y
x = simY(n)

# Lag sannsynlighetshistogram for de simulerte verdiene
plt.hist(x, density=True,bins=100, color="red")  #density=True gjør at vi får␣
 ↪et sannsynlighetshistogram

# Avslutt med å generere alle elementene du har plottet
plt.show()
```



3 Deloppgave d)

Benytt python-funksjonen du har implementert over til å finne tilnærmede verdier for $E[Y]$, $\mathrm{SD}[Y]$, $P(Y \geq 2)$ og $P(Y \geq 2|Y \geq 1)$.

[46]:
```
# Fra nå av må du selv huske på å inkludere pakkene du skal bruke selv

E_Y = sum(x)/n
print(E_Y)

# SD[Y] = sqrt(Var[Y]). Var[Y] = E[(Y-E[Y])^2]
# SD[Y] = |gjennomsnitt av avvik mellom simulert verdi og E[Y]|
SD_Y = abs(sum((x-E_Y))/n)
print(SD_Y)

Y_ge_2 = sum(1 for value in x if value >= 2)
Y_ge_1 = sum(1 for value in x if value >= 1)

P_Y_ge_2 = Y_ge_2/n
print(P_Y_ge_2)
```

```
P_Y_ge_2_given_Y_ge_1 = Y_ge_2/Y_ge_1
print(P_Y_ge_2_given_Y_ge_1)
```

```
1.156914522351635
3.207762167001249e-15
0.007718
0.011410340565697423
```

[ ]: