Maja Szymajda 254313

Sprawozdanie 5

Do realizacji listy 5 użyłam środowiska Python3, napisałam w nim trzy programy: za1.py, zad2.py, zad3.py korzystające następyjących bibliotek: math oraz time.

Zadanie 1

Celem zadania 1 było oszacowanie za pomocą metody Monte Carlo wartości całkek z poniższych funkcji:
a) f(x) = exp[-(x^2)/2] na przedziale [0, 1]
b) f(x) = (1-x^2) na przedziale [0,2]

Aby wykonać to zadanie musiałam wygenerować dużą ilość punktów podziałowych, do tego wykorzystałam stworzony przeze mnie w poprzednich listach generator:

```
def generator(param, xp, xk):
    ArrayOfData = []
    mGet = 134456
    aGet = 24091
    cGet = 90821
    xGet = time.time()
    dol = xp
    gora = xk
    ArrayOfData.append(xGet)

    for index in range(param):
        xi = ((aGet * ArrayOfData[len(ArrayOfData)-1]) + cGet) % mGet
        finish = (dol + (xi) % (gora - dol))
        if index == 0:
            ArrayOfData.clear()
        ArrayOfData.append(finish)
    return ArrayOfData
```

Ten generator umożliwia on wygenerowanie liczby z wybranego zakresu.

W tym przypadku interesują nas dane z zakresu [0,1] oraz [0,2].

Następnie stworzyłam funkcje, która zawierały wzory podane w zadaniu 1:

```
def funkcje(x, ktora_funkcja):
    if (ktora_funkcja == 1):
        f1 = math.exp(-(x**2)/2)
        return f1
    else:
        f2 = 1-x**2
        return f2
```

Jako argument przyjmuje ona wylosowany punkt z zakresu podanego dla danej funkcji w zadaniu oraz jej górną granice, która posłuży do rozróżnienia którego wzoru funkcji używamy.

Aby oszacować ich wyniki użyłam metody Monte Carlo, którą stosuje się do obliczania całek oraz złożonych procesów statystycznych.

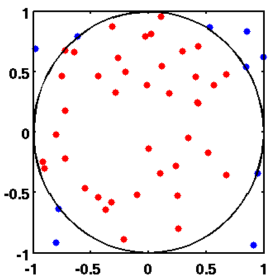
Przykładem takiego zastosowania może być obliczenie pola figur, trudnego do obliczenia jak koła o promieniu R i środku w punkcie (0,0).

1. Losuje się punktów z opisanego na tym kole kwadratu – dla koła o współrzędne wierzchołków (−1,−1), (−1,1), (1,1), (1,−1).
2. Po wylosowaniu każdego z tych punktów trzeba sprawdzić czy jego współrzędne spełniają nierówność (tj. czy punkt należy do koła): $x^2 + y^2 <= R$

Wynikiem losowania jest informacja, że z wszystkich prób było trafionych, zatem pole koła wynosi:

$$P_k = P\,\frac{k}{n},$$

gdzie P jest polem kwadratu opisanego na tym kole (dla R=1 i P=4)



Funkcja przedsawiająca działanie metody Monte Carlo:

```python
def metoda_monte_carlo(xp, xk, dx):
    LP1 = 0
    LPP1 = 0
    # zbiór_punktow = generator(liczba_pkt_podzialowych)
    for i in range(100):
        zbior_punktow = []
        for j in range(liczba_pkt_podzialowych):
            zbior_punktow.append(generator(liczba_pkt_podzialowych, xp, xk)[j])
        if(zbior_punktow[i+1] <= funkcje(zbior_punktow[i], xk)):
            LPP1 += 1
        LP1 += 1

    return LPP1/LP1
```

Otrzymuje ona jako argument poczatek oraz koniec przedziału a także wielkość przedziału (delta x):
dx = xk – xp
Funkcja 100 krotnie wyznacza zbiór punktów podziałowych dla zadanej liczby_pkt_podziałowych, korzystając z powyższego generatora, oraz sprawdza czy wylosowany przez nas punkt jest mniejszy bądź równy wartości dla punktu w funckji, która przyjmuje go jako nasz argument. Jeżeli tak to dodajemy do licznika prób udanych (LPP). Po wykonaniu lososowania 100 razy otrzymaliśmy wynik

```
0.681
0.133
```

Oszacowana całka dla funkcji 1 wynosi 0.68, a dla funkcji drugiej 0.133

Zadanie 2

Zadanie drugie polegało na oszacowaniu liczby rozwiązań dopuszczalnych w wybranej instacji problemu plecakowego.

Problem plecakowy to jeden z najczęściej poruszanych problemów optymalizacyjnych. Zagadnienie polega na maksymalizacji problemu wyboru przedmiotów, tak by ich sumaryczna wartość była jak największa i jednoczesnie spełniała ograniczenie plecaka – mieściła się do niego. Przy podanym zbiorze elementów o podanej wadze i wartości, należy wybrać taki podzbiór by suma wartości była możliwie jak największa, a suma wag była nie większa od danej pojemności plecaka.

Wybrany przeze mnie problem plecakow polega na przyjęciu pojemności plecaka jako 10, ilości elementów jako 20 oraz losowane wagi z zakresu (0,2].

Program oszacujący liczbę rozwiązań dopuszczalnych w wybranym przeze mnie problemie składa się z następujących funkcji: generator, generator_normalny, losowanie_wagi_elementów, f_zero_jeden, zbior_0_1, problem plecakowy, main.

```python
def generator(numbers):
    a = 226954477
    c = 71917
    m = pow(2, 32)
    seed = 4359
    x = (seed * a + c) % m
    u = []
    u.append(abs((2*x/m)-1))
    for i in range((numbers-1)):
        x = (x * a + c) % m
        u.append(abs((2 * x / m) - 1))
    # print(u)
    return u
```

Generator pierwszy zwraca zbiór o rozkładzie równomiernym, wykorzystamy go do losowania macierzy 0 – 1, która będzie odpowiadać za to jakie przedmioty biezremy do plecaka: 1- bierzemy, 0 – nie bierzemy.

```python
def generator_normalny():
    a = 58363
    c = 71917
    m = 102259
    seed = 4359
    temp = []
    x = (seed*a+c) % m
    temp.append(x/m)
    for i in range(239):
        x = (x * a + c) % m
        temp.append(x/m)
    NN = []

    for i in range(0, 240, 12):
        NN.append(abs(((sum(temp[i:i+11]))-6) * 0.5 + 0.5))
    # print (( (sum(temp[i:j]) ) - 6 ) * 0.5 + 0.5)
    # print(NN)
    return NN
```

Generator_normalny zwraca zbiór o rozkładzie normalnym. Wykorzystamy go do stworzenia listy wag przedmiotów.

Losowanie wagi przedmiotów:

```python
def losowanie_wagi_elementow(ilosc_elementow):
    waga_elementow = []
    zbior = generator_normalny()
    for i in range(ilosc_elementow):
        waga_elementow.append(zbior[i]*2)

    return waga_elementow
```

Funkcja polega na wylosowaniu za pomocą geratora o rozkładzie normalnym zbioru 20 punktów. Ponieważ zakres liczb zwracanych ze zbioru jest dwa razy mniejszy niż określiłam zakres zbioru wag elementów, mnożę każdy element zbioru razy dwa otrzymując wtedy zakres (0,2].

Funkcja f_zero_jeden odpowiada za wypełnienie macierzy jedynkami i zerami w zależności od wartości i-tego elementu zbioru wygenerowanego rozkładem równomiernym. Dla x < 0,5 funkcja przyjmuje 1 w innym przypadku przyjmuje ona wartość 0.

```python
def f_zero_jeden(x):
    if (x < 0,5):
        return 1
    else:
        return 0
```

```python
def zbior_0_1(zbior_wykorzystania):
    macierz = []
    print(zbior_wykorzystania)
    for i in range(len(zbior_wykorzystania)):
        macierz.append(f_zero_jeden(zbior_wykorzystania[i]))
        print(macierz[i])
    return macierz
```

Zadaniem funkcji zbior_0_1 jesy wypełnienie macierzy przedmiotów zerami i jedynkami wykorzystując funkcje f_zero_jeden.

Funkcja Problem Plecakowy:

```python
def problem_plecakowy(pojemnosc, ilosc_elementow, waga_elementow):
    proby_udane = 0
    proby_nieudane = 0
    wszystkie_proby = 0
    for i in range(2000):
        suma = 0
        zbior = generator(ilosc_elementow)
        macierz_wykorzystania = zbior_0_1(zbior)
        # print(macierz_wykorzystania)
        for j in range(ilosc_elementow-1):
            x = 0.0
            # print(macierz_wykorzystania[i][j])
            x = macierz_wykorzystania[j] * waga_elementow[j]
            suma += x
        if (suma <= pojemnosc):
            proby_udane += 1
        else:
            proby_nieudane += 1
        wszystkie_proby += 1
    # print(proby_udane/wszystkie_proby)
    wynik = proby_udane/wszystkie_proby
    print(f'{wynik* 100}%  oraz {proby_udane}')
    return wynik
```

Zadaniem funkcji problem_plecakowy jest wyliczenie ile prób zapakowania plecaka jest pozytywnych przy wybranych przeze mnie z założeniach. Dla 2000 prób sprawdzamy czy wylosowane przez nas zapakowanie plecaka spełnia narzucone ograniczenia.

Wyszło mi, że w wybranym przeze mnie przypadku 1522 na 2000 przypadkach udało nam się zapakować plecak. Co daje 76,1% skuteczności.

```
76.1%  oraz 1522
```

Zadanie 3

W zadaniu 3 należało oszacować zysk firmy produkującej bezpieczniki. Zysk ten zależał od: kosztu produkcji produktu, wachającego się między 2-3 zł; ceny_sprzedaży – cena : 5-10 zł, oraz miesięcznej możliości produkcyjnej między 1 tys a 2 tys. Przyjeliśmy też jako koszt utrzymania firmy przez rok 50000 zł. Aby wyliczyć prognozowany zysk firmy należało:

- wylosować koszt produkcji z przedziału 2 a 3 zł,
- wylosować cene sprzedaży 5 – 10 zł,
- wylosować możliwości produkcyjne 1-2 tysiące sztuk.

```
def losowanie_ceny_sprzedazy():
    cena = generator_z_podanego_zakresu(5.0, 10.0)
    return cena

def losowanie_koszt_produkcji():
    cena = generator_z_podanego_zakresu(2.0, 3.0)
    return cena

    sowanie_produkcji():
    dukcja = generator_z_podanego_zakresu(1000, 2000)
    urn produkcja
```

Wylosowałam je używając napisanego przeze mnie generatora, który zwraca liczbę z podanego zakresu.

```
def generator_z_podanego_zakresu(gora, dol):
    ArrayOfData = []
    mGet = 134456
    aGet = 24091
    cGet = 90821
    xGet = time.time()
    ArrayOfData.append(xGet)
    x = ((aGet * ArrayOfData[len(ArrayOfData)-1]) + cGet) % mGet
    finish = (dol + (x) % (gora - dol))
    return finish
```

Aby zasymulować i oszacować prognozowany zysk firmy napisałam funkcje symulacja_zysku()

```
def symulacja_zysku():
    zysk = 0
    for i in range(999):
        zysk_ze_sprzedazy = 0
        koszt_produkcji = losowanie_koszt_produkcji()
        cena_sprzedazy = losowanie_ceny_sprzedazy()
        for j in range(11):
            sztuki = losowanie_produkcji()
            zysk_ze_sprzedazy += (cena_sprzedazy - koszt_produkcji) * sztuki
        zysk += (zysk_ze_sprzedazy - budzet)
    print(f'planowany zysk ze sprzedaży {zysk/1000 }')
    return zysk
```

Przyjmuje ona na początku, że suma wszystkich prób zysku wynosi 0.

Następnie w pętli for 1000 krotnie losuje koszt i cene produktu a następnie 12 razy losuje miesięczną produkcję bezpieczników, na której podstawie oblicza roczny zysk.

Następnie dla każedo z 1000 prognozowanych roków produkcyjnych zlicza prognozowany zysk po odjęciu od niego kosztów utrzymania firmy. Następnie całą uzyskaną kwotę dzieli na ilość wykonanych prób, w tym przypadku 1000.

W tym przypadku otrzymaliśmy wartość: 34350.21 zł

```
planowany zysk ze sprzedaży 34350.207028198245
```