16.24 **17.9** 18.25 **19.23** 20.26 **21.20 22.2** 23.2 24.6

**17.9** Координаты случайной точки А в пространстве (X, Y, Z) подчинены нормальному закону   
Определить вероятность того, что точка А окажется внутри эллипсоида с главными полудиаметрами .

Эллипсоид с главными полудиаметрами описывается неравенством:

Нам необходимо найти вероятность того, что точка A окажется внутри этого эллипсоида. Это эквивалентно интегрированию плотности вероятности по объёму этого эллипсоида.

Для упрощения расчётов перейдём к нормированным координатам:

В новых координатах плотность вероятности становится:

а эллипсоид:

Теперь интегрируем плотность вероятности f(u,v,w) по объёму нового эллипсоида:

Преобразуем интеграл в сферические координаты:

Якобиан перехода к сферическим координатам равен .

Тогда вероятность P выражается как:

Рассмотрим интеграл по r:

Сделаем замену переменной:

Пределы интегрирования меняются от 0 до ρk. Тогда интеграл становится:

Используем известный результат для интеграла:

Подставляя a=ρk, получаем:

Теперь интегрируя по углам θ, φ:

import numpy as np  
from scipy.special import erf  
  
def generate\_random\_point(rho, E1, E2, E3, mu1=0, mu2=0, mu3=0):  
 x = np.random.rand(2)  
  
 z0 = np.sqrt(-2 \* np.log(x[0])) \* np.cos(2 \* np.pi \* x[1])  
 z1 = np.sqrt(-2 \* np.log(x[0])) \* np.sin(2 \* np.pi \* x[1])  
 z2 = np.sqrt(-2 \* np.log(np.random.rand())) \* np.cos(2 \* np.pi \* np.random.rand())  
  
 x\_A = rho \* z0 \* E1 + mu1  
 y\_A = rho \* z1 \* E2 + mu2  
 z\_A = rho \* z2 \* E3 + mu3  
 return np.array([x\_A, y\_A, z\_A])  
  
def is\_inside\_ellipsoid(point, k, E1, E2, E3):  
 return (point[0]\*\*2 / (k\*E1)\*\*2 + point[1]\*\*2 / (k\*E2)\*\*2 + point[2]\*\*2 / (k\*E3)\*\*2) <= 1  
  
rho = 1.0  
E1 = 1.0  
E2 = 1.0  
E3 = 1.0  
  
  
k = 4.0  
  
  
num\_points = 100000  
  
  
count\_inside = 0  
  
  
for \_ in range(num\_points):  
 point = generate\_random\_point(rho, E1, E2, E3)  
 if is\_inside\_ellipsoid(point, k, E1, E2, E3):  
 count\_inside += 1  
  
  
probability = count\_inside / num\_points  
  
analytical\_result = erf(k) - (2 \* rho \* k / np.sqrt(np.pi)) \* np.exp(-(rho \* k)\*\*2)  
  
print("Экспериментальная вероятность:", probability)  
print("Аналитическая вероятность:", analytical\_result)

Результат:  
Экспериментальная вероятность: 0.99885

Аналитическая вероятность: 0.9999994766533552

**19.23** При взвешивании на чашку весов положено 10 разновесов, точность изготовления каждого из которых и точность процесса взвешивания характеризуется средним квадратическим отклонением соответственно равным 0,01 и 0,02 г. Найти среднее квадратическое отклонение ошибки определения веса взвешиваемого тела.

Имеем σ1 = 0,01, σ2 = 0,02 г

**21.20**  – независимые случайные величины, каждая из которых может принимать только два значения: единицу с вероятностью p и нуль с вероятностью q = 1 – p. Найти ряд распределения случайной величины .

Используем метод порождающих функций. Порождающая функция Xj задаётся как:

теперь, чтобы найти порождающую функцию для суммы Y, мы просто умножаем порождающие функции Xj​, так как они независимы:

Теперь мы можем найти ряд распределения YYY раскрывая эту порождающую функцию в ряд Маклорена:

Это означает, что вероятность того, что Y = k, равна коэффициенту tk в разложении GY(t), то есть . Таким образом, ряд распределения Y имеет вид:

Распределение соответствует биномиальному

Предлагаю рассмотреть новейшее распределение полученное путём естественного валяния дурака в процессе исполнения второй расчётной работы. За автомат 4 я готов раскрыть секрет этого чем-бы-оно-ни-было

Изображение выглядит как диаграмма, График, линия

Автоматически созданное описание

**24.6** При изготовлении отливок вероятность получения дефектной равна 0,2. Сколько необходимо запланировать отливок к изготовлению, чтобы с вероятность не менее 0,95 была обеспечена программа выпуска изделий, для выполнения которой необходимо 50 бездефектных отливок, если качество каждой из них не зависит от остальных?*70*