numpy

Хашин С.И.
http://math.ivanovo.ac.ru/dalgebra/Khashin/index.html

Ивановский университет

.

Случайные числа в Питоне

.

Иваново-2023

План

π

Псевдослучайные числа

Случайные величины

numpy

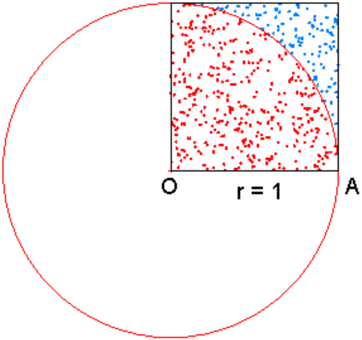
Полная вероятность

Список слайдов

Список слайдов, продолжение

π

Как приближенно найти число $\pi \approx 3.1415926535897$?



Площадь четверти круга равна $\pi/4$. Генерируем пару случайных чисел (x, y) в интервале (0,1) Если $x^2 + y^2 < 1$, увеличиваем счётчик.

π на Питоне

```
import random
def a_pi(n):
    cnt = 0
    for i in range(n):
        x = random.random()
        y = random.random()
        if x * x + y * y < 1: cnt += 1
    return 4*cnt/n

for i in range(10): print(a_pi(10**6))
```

π на Питоне

При 10^2 испытаний: $3.08, 2.80, 3.36, 2.96, 3.2, 3.08, 3.16$.
При 10^4 испытаний: $3.1416, 3.1512, 3.1216, 3.1380, 3.1220$.
При 10^6 испытаний: $3.1419, 3.1443, 3.1428, 3.1397, 3.1449$.
При 10^8 испытаний: $3.14132, 3.14153, 3.14149, 3.14174, 3.14194, 3.14169, 3.14142, 3.14144, 3.14153, 3.14143$.
$(\pi \approx 3.141593)$.

Псевдослучайные числа

Как они строятся?

```
def rand1(x):
  a, b = 137, 0.6177
  x = a*x + b
  return x - int(x)
```

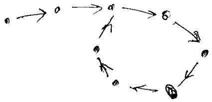
Период 5..30 млн.

```
def rand2(x):
  n, a, b = 2**32, 173, 2346
  return (a*x + b)%n
```

Период 2 млрд. (2^{31})

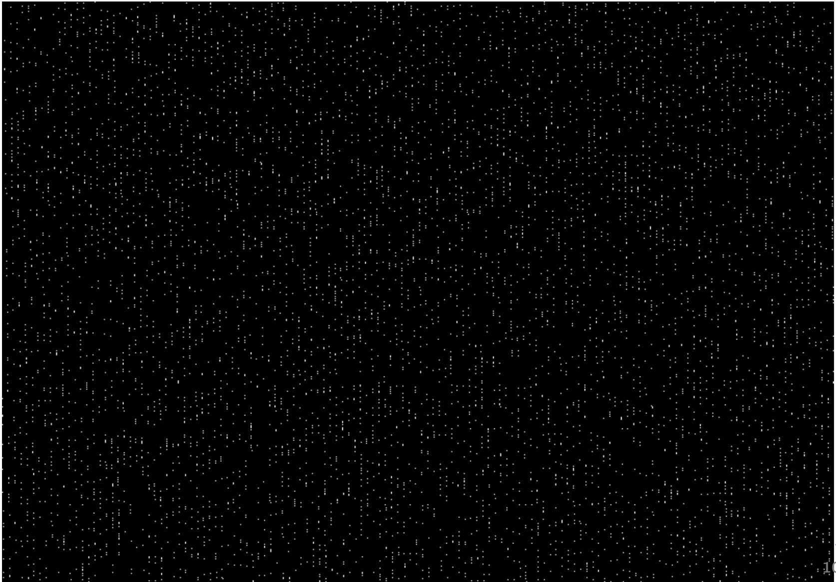
Как найти период рекуррентной последовательности?

Рассмотрим последовательность x_n, находящуюся по формуле: $x_{n+1} = f(x_n)$ для некоторой функции $f(x)$.



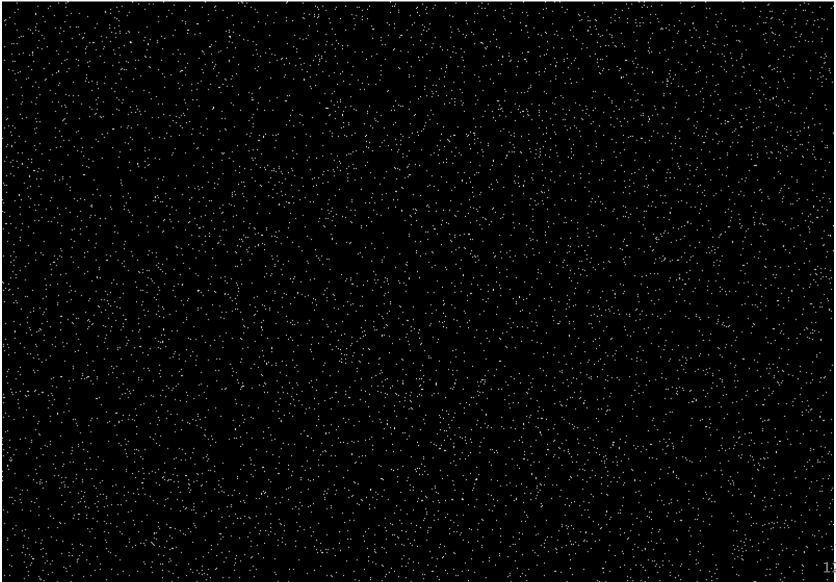
```
def cycle_len(f, x0):
    y = x0
    for i in range(1, 30):
        if i>20: print(i)
        x = y
        for j in range(2**i):
            y = f(y)
            if y==x: return j+1
```

10 процентов случайных белых точек, rand1



10 процентов случайных белых точек, rand2

10 процентов случайных белых точек, Python



Как построить псевдослучайную последовательность x_n?

Будем строить её в виде $x_n = f(n)$, где $f(n)$ — комбинация функций:

- $C_1 + x$.
- $C_2 * x$, C_2 — нечётное.
- $C_3\ XOR\ x$.
- циклический сдвиг на $11, 19, 25$ бит
- $j(C_4, x)$, где

$$j(C_4, x) = \begin{cases} 2x, & \text{если старший бит } = 0, \\ 2x + C_4, & \text{иначе} \end{cases}$$

.

При этом функция seed(k) должна устанавливать $n = C_5 * k$.

Случайные величины

Случайная величина - числовая величина, значение которой меняется в зависимости от случая. Примеры.

- Число мальчиков среди ста новорожденных - случайная величина, которая может принимать значения 0,. . . ,100.
- Количество выпавших «орлов» при 100 бросаниях монеты.
- Сумма баллов при бросании 2-х кубиков.
- Время ожидания автомобиля на заправке, - непрерывная СВ.

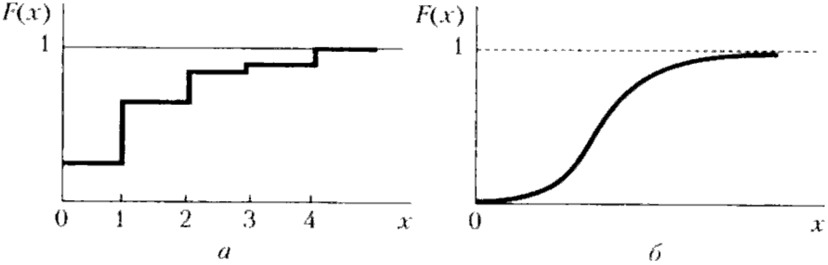
Функция распределения СВ

Функцией распределения случайной величины ξ называется функция $F(x)$, выражающая вероятность того, что ξ примет значение, меньшее чем x: $F(x) = P(\xi < x)$. Свойства.

- Функция распределения есть неубывающая функция.
- $\lim\limits_{x \to -\infty} F(x) = 0.$
- $\lim\limits_{x \to +\infty} F(x) = 1.$

Функция распределения СВ

а) Дискретная СВ
б) Непрерывная СВ



Математическое ожидание

Математическое ожидание СВ — её среднее значение за много испытаний.
Пусть дискретная СВ ξ принимает значения $\{x_1, \dots, x_r\}$ с вероятностями $\{p_1, \dots, p_r\}$. Тогда за N испытаний мы получим Np_1 раз x_1, Np_2 раз x_2, и т.д.. Среднее значение:

$$M(\xi) = p_1 x_1 + \dots + p_r x_r.$$

Для непрерывной величины с плотностью $f(x)$:

$$M(\xi) = \int f(x) x \, dx,$$

причём интеграл по всем вещественным числам, от $-\infty$ до $+\infty$.

Среднеквадратичное отклонение и дисперсия

Мат.ожидание показывает среднее значение СВ. А насколько она отклоняется от среднего? Это показывается другая характеристика — дисперсия.
Дисперсией СВ ξ называется

$$D(\xi) = M((\xi - M(\xi))^2),$$

то есть среднее значение квадрата отклонения величины ξ от её среднего значения.
Квадратный корень из дисперсии и есть среднеквадратичное отклонение:

$$\sigma(\xi) = \sqrt{D(\xi)}.$$

Свойства мат.ожидания и дисперсии

- Мат.ожидание суммы случайных величин равно сумме мат.ожиданий.
- Дисперсия суммы НЕЗАВИСИМЫХ величин равно сумме дисперсий.
- Если умножить СВ на n, то мат.ожидание увеличатся в n раз, дисперсия — в n^2 раз, а среднеквадратичное отклонение — в n раз.

Список случайных величин

- Бернуллиевская СВ (p)
- Биномиальная СВ (n,p)
- Целочисленная СВ равномерно распределённая на отрезке.
- Вещественная СВ равномерно распределённая на отрезке.
- Геометрическая СВ(p) – до первого успеха
- Пуассоновская СВ (λ)
- Экспоненциальная СВ (λ)
- Нормальная СВ (a, σ)
- Распределение хи-квадрат(k)

Бернуллиевская СВ(p)

Принимает лишь два значения: 0 и 1,

- 0 с вероятностью q=1-p,
- 1 с вероятностью p.

Мат.ожидание = p.
Дисперсия = pq.

ξ

x	P
0	q
1	p

ξ-M(ξ)

x	P
-p	q
1-p	p

$(\xi\text{-}M(\xi))^2$

x	P
p^2	q
q^2	p

$M = 0 \cdot q + 1 \cdot p = p$

$M = p^2 \cdot q + q^2 \cdot p = pq(p+q) = pq$

На Питоне:

```
def Bernully(p):    # Бернуллиевская СВ (p)
    return 1 if random.random()<p else 0
```

Биномиальная СВ(n, p)

Сумма n независимых бернуллиевских СВ(p). Принимает значения: $0, 1, \ldots, n$.
$P(\xi = k) = C_n^k p^k q^{n-k}$, где C_n^k - биномиальный коэффициент

$$C_n^k = \frac{n!}{(n-k)!k!}.$$

Мат.ожидание $= np$.
Дисперсия $= npq$.
На Питоне:

```
def Binom(n, p):    # Биномиальная СВ(n,p)
  return sum(Bernully(p) for i in range(n))
```

Задание. Построить график бином.СВ при $(n, p) = (10, 1/2), (100, 1/2), (60, 1/6), (600, 1/6)$.

График биномиальной СВ

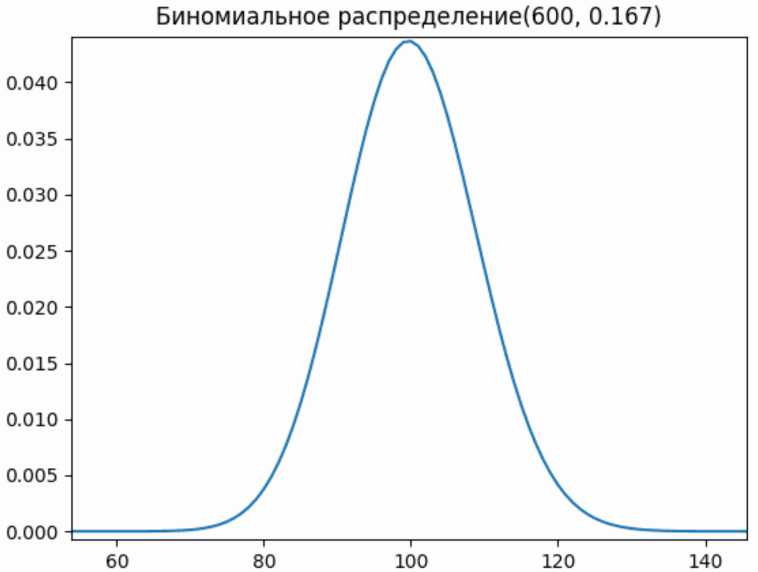
```
from math import factorial
import matplotlib.pyplot as plt

def Binom_co(n,k): # C_n^k
    return factorial(n)/(factorial(k)*factorial(n-k))

def Binom(n,p,k): # P(биномиальная CB(n,p))= k
    return Binom_co(n,k) * p**k * (1-p)**(n-k)

 n, p = 600, 1/6
 pp = [Binom(n,p,k) for k in range(n+1)]
 plt.plot(np.arange(n+1), pp)
 plt.title(f'Биномиальное распределение({n}, {p:.3f})')
 plt.show()
```

График биномиальной СВ



Целочисленная равномерно распределённая на отрезке

Целочисленная СВ равномерно распределённая на отрезке [a,b]
Принимает значения $[a, a+1, \ldots b]$, причем все с одинаковой вероятностью $1/(b-a+1)$.
Мат.ожидание $= (a+b)/2$.
Дисперсия: $(n^2-1)/12$, где $n = b-a+1$.
На Питоне:

```
random.randint(a, b)
```

ВНИМАНИЕ! Здесь ВКЛЮЧАЯ b! То есть `random.randint(0,9)` — все цифры от 0 до 9.

Вещественная равномерно распределённая на отрезке

Вещественная СВ равномерно распределённая на отрезке [a,b]
Принимает все значения их отрезка $[a, b]$.
Мат.ожидание $= (a + b)/2$.
Дисперсия: $n^2/12$, где $n = b - a$.
На Питоне:

```
random.random() # на отрезке [0,1]
random.uniform(a,b) # на отрезке [a,b]
```

Но равномерное распределение на отрезке, например $[-1, 1]$ можно получить и так:

```
2*random.random()-1 # на отрезке [-1,1]
```

МО и дисперсия равномерного распределения $[-1, 1]$

Задание. Найти МО и дисперсию через интегралы.
Математическое ожидание:

$$M(\xi) = \int_{-\infty}^{\infty} x \cdot f(x) dx = \int_{-1}^{1} \frac{x \cdot dx}{2} = 0$$

Дисперсия:

$$D(\xi) = \int_{-\infty}^{\infty} x^2 \cdot f(x) dx = \int_{-1}^{1} \frac{x^2 dx}{2} = \frac{x^3}{6} \Big|_{-1}^{1} = \frac{1}{3}$$

Геометрическая СВ(p)

Выполняем бернуллиевскую СВ до первого успеха, берем количество неудачных попыток.
Принимает значения: $0, 1, 2, \ldots$.
$P(\xi = 0) = p$,
$P(\xi = 1) = pq$, где $q = 1 - p$,,
$P(\xi = 2) = pq^2$,
$P(\xi = 3) = pq^3$,
$\ldots$
МО $= q/p$, Дисперсия $= q/p^2$.
На Питоне:

```
def GeomSV(p):  # Геометрическая CB(p)
  k = 0
  while Bernully(p)==0: k += 1
  return k
```

Геометрическая СВ(p), МО

Найдем МО:

$$M = 0 \cdot p + 1 \cdot pq + 2 \cdot pq^2 + 3 \cdot pq^3 + \dots =$$

$$= pq(1 + 2q + 3q^2 + 4q^3 + \dots).$$

$$f(q) = 1 + q + q^2 + q^3 + \dots = 1/(1-q).$$

$$f'(q) = 1 + 2q + 3q^2 + \dots = 1/(1-q)^2 = 1/p^2.$$

$$M = pq(1 + 2q + 3q^2 + 4q^3 + \dots) = pq/p^2 = q/p.$$

Аналогично, но более громоздко: дисперсия $\sigma^2 = q/p^2$.

Экспоненциальная СВ (λ)

Пусть есть поток независимых случайных событий, причем в среднем за минуту их происходит λ штук. СВ ξ = время ожидания очередного события. Плотность распределения

$$f(x) = \lambda e^{-\lambda x} \text{ при } x > 0.$$

Мат.ожидание $= 1/\lambda$. Дисперсия $= 1/\lambda^2$.

```
def ExpSV(lam): # Экспоненциальная СВ (lam)
    return -np.log(random.random())/lam
```

random.seed()

random.seed(777) — фиксируем начальное число

```
import random
random.seed(77)
for i in range(5):
  print(random.randint(0,100), end=' ')
print()
random.seed(77)
for i in range(5):
  print(random.randint(0,100), end=' ')
# без random.seed
 32 41 25 30 24
 14 37 60 71 78
# c random.seed
 32 41 25 30 24
 32 41 25 30 24
```

choise, sample

```
import random
L = [i**2 for i in range(10)]
print(L)  # [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
print(random.choice(L))  # 16
print(random.sample(L,8))
> [25, 9, 64, 1, 0, 4, 49, 16]
print(random.choices(L,k=8))
> [81, 4, 25, 36, 0, 4, 64, 16]
print(random.sample(L,11))     # Ошибка!
print(random.choices(L,k=11))  # Ok
```

numpy.random

```
np.random.seed(777) - инициализация датчика
np.random.random(9) - массив из 9 чисел [0,1]
np.random.random((3,4)) - матрица 3*4 чисел [0,1]
np.random.randint(0, 3, 7) - массив из 7 чисел [0,1,2]
np.random.randint(0, 3, size=(3,4)) матрица [0,1,2]
np.random.shuffle(a) - случайное перемешивание a
np.random.choice(a, size=None, replace=True, p=None)
   replace : если True, то одно значение может выбираться
             более одного раза.
   p : вероятности. Это означает, что элементы можно
       выбирать с неравными вероятностями.
```

numpy.random.random

```
print(np.random.random(9))
> [0.8911 0.9423 0.4454 0.6836 0.1352
   0.8339 0.6374 0.6146 0.0695]
print(np.random.random((3,4)))
[[0.7509 0.1127 0.744  0.697 ]
 [0.9518 0.8398 0.1435 0.5806]
 [0.3583 0.0495 0.0678 0.3907]]

print(np.random.uniform(0,100, size=6))
> [19.4575  8.8079 47.1339 95.3348 36.4089 33.3351]

print(np.random.randint(0, 3, 7))
  [0 1 1 2 0 2 0]
```

Пуассоновская СВ

Распределение Пуассона описывает вероятность наступления k независимых событий за данное время при средней интенсивности событий λ.
Примеры: количество автомашин за минуту, количество покупателей за минуту, количество упавших звёзд за час.

$$P(\xi = k) = \frac{\lambda^k}{k!} e^{-\lambda}.$$

МО$(\xi) = \lambda$,
Дисперсия $= \lambda$.

```
np.random.poisson(lam=1.0, size=None)
# в среднем за минуту проходит 5 а/м.
# сколько за каждую из 7 минут:
print(np.random.poisson(5, size=7))
> [ 6  3  5  2  2 11  8]
```

Нормальная СВ (a, σ)

Плотность:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-a)^2}{2\sigma^2}}$$

Мат.ожидание $= a$.
Дисперсия $= \sigma^2$.
Стандартная СВ – это нормальная с парамерами $(0, 1)$.

Нормальная СВ (a, σ)

Центральная предельная теорема (грубо)

Сумма большого количества любых независимым СВ является нормальной СВ.
Напомню, что при сложении независимых СВ мат.ожидания и дисперсии складываются.
Правило 3σ. С вероятностью 99.5% нормальная СВ отклоняется от своего среднего значения не более, чем на 3σ.
$\sigma =$ корень из дисперсии.

Стандартная СВ

Пусть ξ — нормальная СВ с параметрами (a, σ). Тогда

$$\xi_1 = \frac{\xi - a}{\sigma}$$

— нормальная СВ с параметрами $(0, 1)$. Такая СВ называется стандартной. То есть из любой нормальной СВ можно получить стандартную.

Задание 1. Пусть y — среднее значение N независимых стандартных СВ. В каких пределах должно находится y по «правилу 3σ»?

Задание 2. Проверить на Питоне.

numpy.random.normal

Нормальная СВ с МО=loc, ср.кв.отклонением = scale. Можно получать не только одиночное значение, но и массив произвольного размер, если задать параметр size.

```
np.random.normal(loc=0.0, scale=1.0, size=None)

print(np.random.normal(10, 2, size=(3,4)))
>[[ 7.8667 10.1878  7.6642  7.1231]
> [11.6345  4.4653 10.9779  9.0358]
> [10.1661  7.8844 11.4476 11.6764]]

print(np.random.normal(size=(3,4))) # loc=0, scale=1
>[[ 0.0766  0.8322  1.2473  1.0879]
> [ 0.1301 -0.1036  0.1974 -0.0792]
> [-0.5861 -0.5602  1.1886  0.713 ]]
```

Задание

За 10 тыс. испытаний, какая доля отклонилась от среднего больше, чем на 2σ?
Больше, чем на 3σ?
Больше, чем на 4σ?

numpy.random.binomial

```
np.random.binomial(n, p, size) # биномиальное распределение

# 7 раз бросаем по 1000 монет:
print(np.random.binomial(1000, 1/2, size=7))
> [524 478 483 504 490 492 484]

# 7 раз бросаем по 100 кубиков,
# находим количество "6":
print(np.random.binomial(1000, 1/6, size=7))
> [178 167 170 166 167 182 162]
```

Формула Стирлинга

Формула Стирлинга приближенного вычисления факториалов больших чисел:

$$n! \approx \left(\frac{n}{e}\right)^n \sqrt{2\pi n}.$$

Приближенная формула для биномиальных коэффициентов. Пусть z близко в $1/2$:

$$z = \frac{2}{n}\left|\frac{n}{2} - k\right| << \sqrt[3]{1/n}$$

Тогда:

$$C_n^k \approx \frac{2^{n+1}}{\sqrt{2\pi n}} e^{-\frac{n-1}{2}z^2}$$

Формула Стирлинга, проверка

1. Найти относительную погрешность формулы Стирлинга при $n = 10, 30, 100, 300, 1000$.
2. Найти относительную погрешность формулы биномиальных коэффициентов при $n = 10, 30, 100, 300, 1000$ и k в пределах от $n/2 - 4\sigma$ до $n/2 + 4\sigma$.

Формула полной вероятности

Ковбой попадает в муху на стене с вероятностью 0.9, если стреляет из пристрелянного револьвера и 0.2 — если из непристрелянного. На столе лежит 10 револьверов, из них только 4 пристрелянные. Ковбой видит на стене муху, хватает первый попавшийся револьвер и стреляет в неё. Найдите вероятность того, что ковбой промахнётся.

Событие A_1 — револьвер пристрелянный и A_2 — нет.
$P(A_1) = 0.4, P(A_2) = 0.6$.
Вероятность (A_1 и промах) $= 0.4 \cdot 0.1 = 0.04$.
Вероятность (A_2 и промах) $= 0.6 \cdot 0.8 = 0.48$.
Вероятность промаха $= 0.04 + 0.048 = 0.52$.

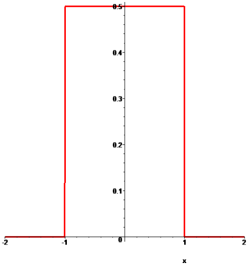
Формула полной вероятности

Автоматическая линия изготавливает батарейки. Вероятность того, что готовая батарейка неисправна, равна 0.02. Перед упаковкой каждая батарейка проходит систему контроля. Вероятность того, что система забракует неисправную батарейку, равна 0.99. Вероятность того, что система по ошибке забракует исправную батарейку, равна 0.01. Найдите вероятность того, что случайно выбранная батарейка будет забракована системой контроля.
Ответ: $0.02 \cdot 0.99 + 0.98 \cdot 0.01 = 0.0198 + 0.0098 = 0.0296$.

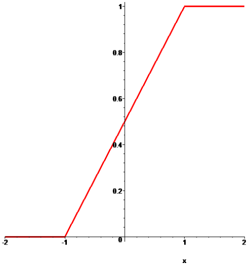
Сумма двух СВ

Пусть ξ_1, ξ_2 — две независимых СВ равномерно распределённых на отрезке $[-1, 1]$.

Плотность

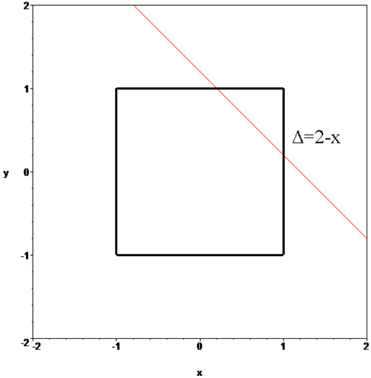


Функция



Как найти функцию распределения и плотность СВ $\xi_1 + \xi_2$?

Сумма двух СВ

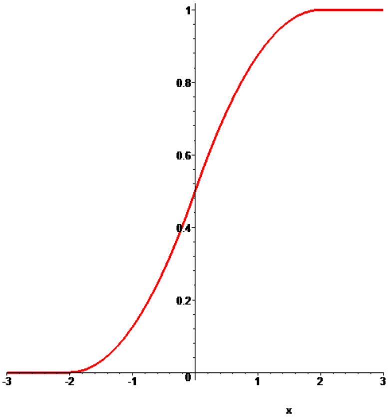


Площадь треугольника $= (2-x)^2/2$, площадь остальной части квадрата $= 4-(2-x)^2/2$. Поэтому при $x > 0$:

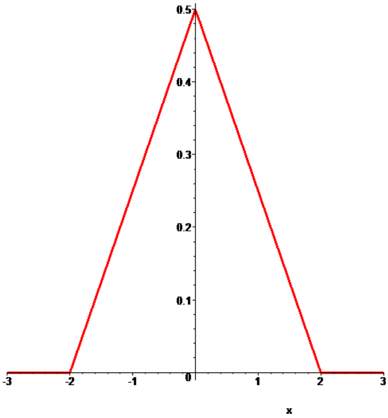
$$P(\xi_1 + \xi_2 < x) = (4-(2-x)^2/2)/4 = 1-(2-x)^2/8$$

Сумма двух СВ

Функция



Плотность



Остроугольный треугольник

Будет ли треугольник остроугольным:

```
def sc_prod(x1,y1, x2, y2):
    return x1*x2 + y1*y2

def ostroug(x1,y1, x2,y2, x3,y3):
    return sc_prod(x2-x1, y2-y1, x3-x1, y3-y1)>0 \
       and sc_prod(x1-x2, y1-y2, x3-x2, y3-y2)>0 \
       and sc_prod(x1-x3, y1-y3, x2-x3, y2-y3)>0

print(ostroug(0,0, 2,0, 1,0.2))
```