Лабораторная работа №2

Методы оценки статических характеристик, связанных с распределением пользователей на плоскости

Задание 1. Сгенерировать выборку случайных чисел размером 100 и 1000 для двух распределений – экспоненциального и нормального. Для созданных выборок сделать следующее:

1. Посчитать выборочное среднее и дисперсию, сравнить с математическим ожиданием соответствующих распределений;
2. Посчитать 0.5 и 0.99 квантили, сравнить с соответствующими теоретическими значениями;
3. Построить гистограмму распределения;
4. Построить функцию распределения случайной величины на основе выборки (на одном графике показать функции распределения, полученные из выборок разного размера и теоретическую);
5. Построить плотность распределения случайной величины на основе выборки (на одном графике показать плотности распределения, полученные из выборок разного размера и теоретическую).

Подготовка функций и импорт библиотек

In [1]:

```
import numpy as np
import matplotlib.pyplot as plt
import math
from scipy import special
```

In [2]:

```
# Определение функции распределения
def cdf(rnd_list):
    cdf_x = []
    cdf_y = []
    l = sorted(rnd_list)
    N = len(l)
    for i in range(0, N):
        count = 0
        for j in range(0, N):
            if l[i] > l[j]:
                count += 1
        cdf_x.append(l[i])
        cdf_y.append(count/N)
    return {'x': cdf_x, 'y': cdf_y}
```

In [3]:

```
# Определение плотности распределения
def pdf(k, rnd_list):
    pdf_x = []
    pdf_y = []
    n = len(rnd_list)
    h = (max(rnd_list) - min(rnd_list)) / k
    a = min(rnd_list)
    for i in range(0, k):
        count = 0
        for j in rnd_list:
            if (a + i * h) < j < (a + (i * h) + h):
                count = count + 1
        pdf_x.append(a + i * h + h / 2)
        pdf_y.append(count / (n * h))
    d = {'x': pdf_x, 'y': pdf_y}
    return d
```

In [4]:

```
def cdf_exp(x, exp_lambda):
    return 1 - math.e**(-exp_lambda * x)
```

In [5]:

```
def cdf_norm(x, mean, std):
    return (1.0/2.0) * (1 + math.erf((x - mean)/(std * math.sqrt(2))))
```

In [6]:

```
def pdf_exp(x, exp_lambda):
    return exp_lambda * (math.e**(-exp_lambda*x))
```

In [7]:

```
def pdf_norm(x, mean, std):
    tmp1 = 1.0 / (std  * math.sqrt(2 * math.pi))
    tmp2 = ((x - mean) /std)**2
    tmp3 = -0.5 * tmp2
    return tmp1 * math.e**(tmp3)
```

Генерация данных

In [8]:

```
exp_lambda = 2.0
distrib_exp_100 = np.random.exponential(scale = 1.0/exp_lambda, size = 100)
distrib_exp_1000 = np.random.exponential(scale = 1.0/exp_lambda, size = 1000)
```

In [9]:

```
norm_mean = 5
norm_std = 3.0
distrib_norm_100 = np.random.normal(norm_mean, norm_std, size = 100)
distrib_norm_1000 = np.random.normal(norm_mean, norm_std, size = 1000)
```

Выборочное среднее, дисперсия.

In [10]:

```
m_e = np.mean(distrib_exp_100)
d_e = np.var(distrib_exp_100)
theor_m_e = 1.0/exp_lambda
theor_d_e = 1.0/(exp_lambda**2)
print("Экспоненциальное распределение (100 элементов): \n"
"Mξ_выборочное = {}, Dξ_выборочное = {}\n"
"Mξ_теоретическое = {}, Dξ_теоретическое = {}\n".format(m_e, d_e, theor_m_e, theor_
print("Разница между выборочным значением и теоретическим: \n |Mξ| = {}, |Dξ| = {}"
      .format(abs(theor_m_e - m_e), abs(theor_d_e - d_e)))
```

```
Экспоненциальное распределение (100 элементов):
Mξ_выборочное = 0.5108374975530433, Dξ_выборочное = 0.286427041272317
4
Mξ_теоретическое = 0.5, Dξ_теоретическое = 0.25

Разница между выборочным значением и теоретическим:
 |Mξ| = 0.010837497553043307, |Dξ| = 0.036427041272317384
```

In [11]:

```
m_e = np.mean(distrib_exp_1000)
d_e = np.var(distrib_exp_1000)
print("Экспоненциальное распределение (1000 элементов): \n"
"Mξ_выборочное = {}, Dξ_выборочное = {}\n"
"Mξ_теоретическое = {}, Dξ_теоретическое = {}\n".format(m_e, d_e, theor_m_e, theor_
print("Разница между выборочным значением и теоретическим: \n |Mξ| = {}, |Dξ| = {}"
      .format(abs(theor_m_e - m_e), abs(theor_d_e - d_e)))
```

```
Экспоненциальное распределение (1000 элементов):
Mξ_выборочное = 0.4847536024970952, Dξ_выборочное = 0.281179120075406
8
Mξ_теоретическое = 0.5, Dξ_теоретическое = 0.25

Разница между выборочным значением и теоретическим:
 |Mξ| = 0.015246397502904796, |Dξ| = 0.031179120075406797
```

In [12]:

```
m_e = np.mean(distrib_norm_100)
d_e = np.var(distrib_norm_100)
theor_m_e = norm_mean
theor_d_e = norm_std ** 2
print("Нормальное распределение (100 элементов): \n"
"Mξ_выборочное = {}, Dξ_выборочное = {}\n"
"Mξ_теоретическое = {}, Dξ_теоретическое = {}\n".format(m_e, d_e, theor_m_e, theor_
print("Разница между выборочным значением и теоретическим: \n |Mξ| = {}, |Dξ| = {}"
      .format(abs(theor_m_e - m_e), abs(theor_d_e - d_e)))
```

```
Нормальное распределение (100 элементов):
Mξ_выборочное = 4.596211363057753, Dξ_выборочное = 9.058084653635873
Mξ_теоретическое = 5, Dξ_теоретическое = 9.0

Разница между выборочным значением и теоретическим:
 |Mξ| = 0.403788636942247, |Dξ| = 0.05808465363587345
```

In [13]:

```
m_e = np.mean(distrib_norm_1000)
d_e = np.var(distrib_norm_1000)
print("Нормальное распределение (1000 элементов): \n"
"Mξ_выборочное = {}, Dξ_выборочное = {}\n"
"Mξ_теоретическое = {}, Dξ_теоретическое = {}\n".format(m_e, d_e, theor_m_e, theor_
print("Разница между выборочным значением и теоретическим: \n |Mξ| = {}, |Dξ| = {}"
      .format(abs(theor_m_e - m_e), abs(theor_d_e - d_e)))
```

```
Нормальное распределение (1000 элементов):
Mξ_выборочное = 5.012565787988866, Dξ_выборочное = 8.341878382624051
Mξ_теоретическое = 5, Dξ_теоретическое = 9.0

Разница между выборочным значением и теоретическим:
 |Mξ| = 0.012565787988865651, |Dξ| = 0.658121617375949
```

Квантили (0.5, 0.99)

In [14]:

```
qu_05 = np.quantile(distrib_exp_100, .5)
qu_099 = np.quantile(distrib_exp_100, .99)
theor_qu_05 = - (math.log(1 - .5)/ exp_lambda)
theor_qu_099 = - (math.log(1 - .99)/ exp_lambda)
print("Квантили для экспоненциального распределения (100 элементов) ")
print("\tВыборочный квантиль 0.5 = {} \n\tТеоретический квантиль 0.5 = {} \n\tРазни
      .format(qu_05, theor_qu_05, abs(qu_05 - theor_qu_05)))
print("\n\tВыборочный квантиль 0.99 = {}\n\tтеоретический квантиль 0.99 = {}\n\traз
      .format(qu_099, theor_qu_099, abs(qu_099 - theor_qu_099)))
```

```
Квантили для экспоненциального распределения (100 элементов)
        Выборочный квантиль 0.5 = 0.32059941337334796
        Теоретический квантиль 0.5 = 0.34657359027997264
        Разница = 0.02597417690662468

        Выборочный квантиль 0.99 = 2.4141029580618047
        теоретический квантиль 0.99 = 2.3025850929940455
        разница = 0.11151786506775929
```

In [15]:

```
qu_05 = np.quantile(distrib_exp_1000, 0.5)
qu_099 = np.quantile(distrib_exp_1000, 0.99)
print("Квантили для экспоненциального распределения (1000 элементов) ")
print("\tВыборочный квантиль 0.5 = {} \n\tТеоретический квантиль 0.5 = {} \n\tРазни
      .format(qu_05, theor_qu_05, abs(qu_05 - theor_qu_05)))
print("\n\tВыборочный квантиль 0.99 = {}\n\tтеоретический квантиль 0.99 = {}\n\traз
      .format(qu_099, theor_qu_099, abs(qu_099 - theor_qu_099)))
```

```
Квантили для экспоненциального распределения (1000 элементов)
        Выборочный квантиль 0.5 = 0.3090972479545059
        Теоретический квантиль 0.5 = 0.34657359027997264
        Разница = 0.03747634232546676

        Выборочный квантиль 0.99 = 2.402981011575928
        теоретический квантиль 0.99 = 2.3025850929940455
        разница = 0.10039591858188235
```

In [16]:

```
qu_05 = np.quantile(distrib_norm_100, .5)
qu_099 = np.quantile(distrib_norm_100, .99)
theor_qu_05 = math.sqrt(2) * norm_std * special.erfinv(2 * 0.5 - 1) + norm_mean
theor_qu_099 = math.sqrt(2) * norm_std * special.erfinv(2 * 0.99 - 1) + norm_mean
print("Квантили для нормального распределения (100 элементов) ")
print("\tВыборочный квантиль 0.5 = {} \n\tТеоретический квантиль 0.5 = {} \n\tРазни
      .format(qu_05, theor_qu_05, abs(qu_05 - theor_qu_05)))
print("\n\tВыборочный квантиль 0.99 = {}\n\tтеоретический квантиль 0.99 = {}\n\traз
      .format(qu_099, theor_qu_099, abs(qu_099 - theor_qu_099)))
```

```
Квантили для нормального распределения (100 элементов)
        Выборочный квантиль 0.5 = 4.148386433286639
        Теоретический квантиль 0.5 = 5.0
        Разница = 0.8516135667133611

        Выборочный квантиль 0.99 = 12.029126798506468
        теоретический квантиль 0.99 = 11.979043622122525
        разница = 0.050083176383942885
```

In [17]:

```
qu_05 = np.quantile(distrib_norm_1000, .5)
qu_099 = np.quantile(distrib_norm_1000, .99)
print("Квантили для нормального распределения (1000 элементов) ")
print("\tВыборочный квантиль 0.5 = {} \n\tТеоретический квантиль 0.5 = {} \n\tРазни
      .format(qu_05, theor_qu_05, abs(qu_05 - theor_qu_05)))
print("\n\tВыборочный квантиль 0.99 = {}\n\tтеоретический квантиль 0.99 = {}\n\traз
      .format(qu_099, theor_qu_099, abs(qu_099 - theor_qu_099)))
```

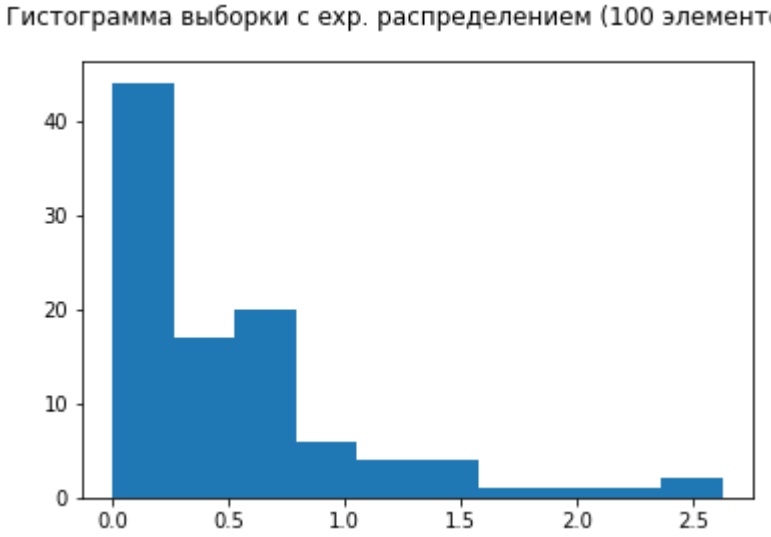
```
Квантили для нормального распределения (1000 элементов)
        Выборочный квантиль 0.5 = 4.95741636966741
        Теоретический квантиль 0.5 = 5.0
        Разница = 0.04258363033259016

        Выборочный квантиль 0.99 = 11.79260387747181
        теоретический квантиль 0.99 = 11.979043622122525
        разница = 0.18643974465071445
```

Гистограммы распределений

In [18]:

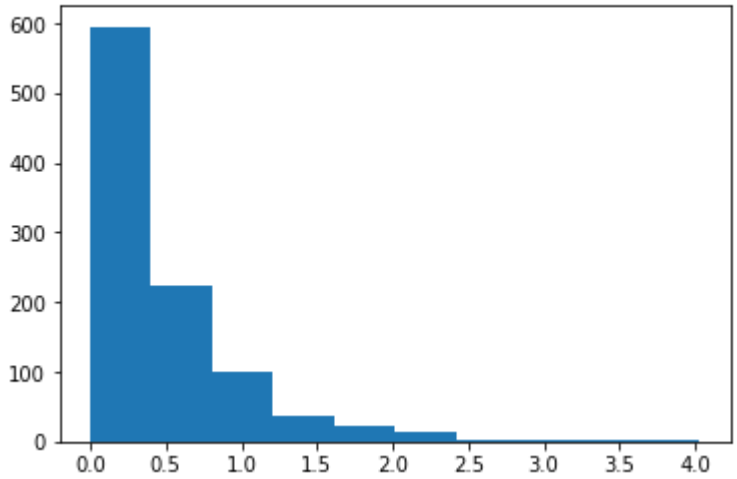
```
_ = plt.hist(distrib_exp_100)
_ = plt.suptitle('Гистограмма выборки с exp. распределением (100 элементов)')
```



In [19]:

```
_ = plt.hist(distrib_exp_1000)
_ = plt.suptitle('Гистограмма выборки с exp. распределением (1000 элементов)')
```

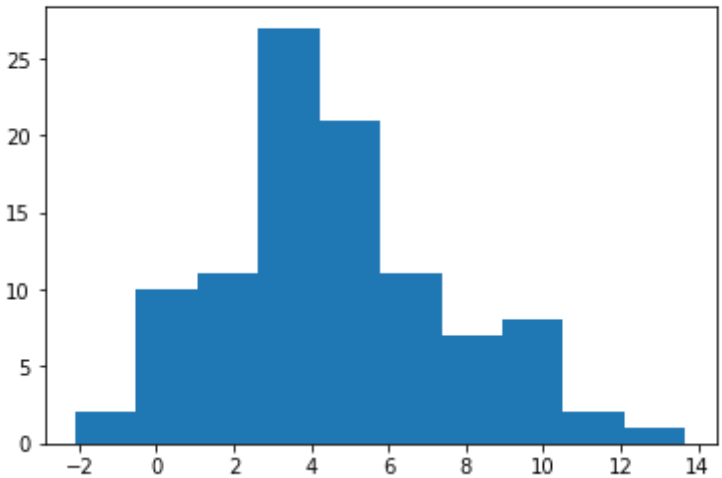
Гистограмма выборки с exp. распределением (1000 элементов)



In [20]:

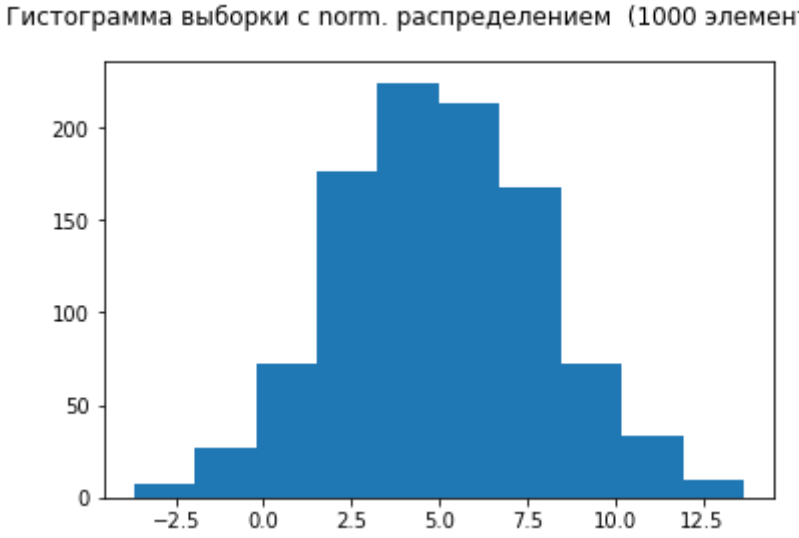
```
_ = plt.hist(distrib_norm_100)
_ = plt.suptitle('Гистограмма выборки с norm. распределением  (100 элементов)')
```

Гистограмма выборки с norm. распределением (100 элементов)



In [21]:

```
_ = plt.hist(distrib_norm_1000)
_ = plt.suptitle('Гистограмма выборки с norm. распределением  (1000 элементов)')
```



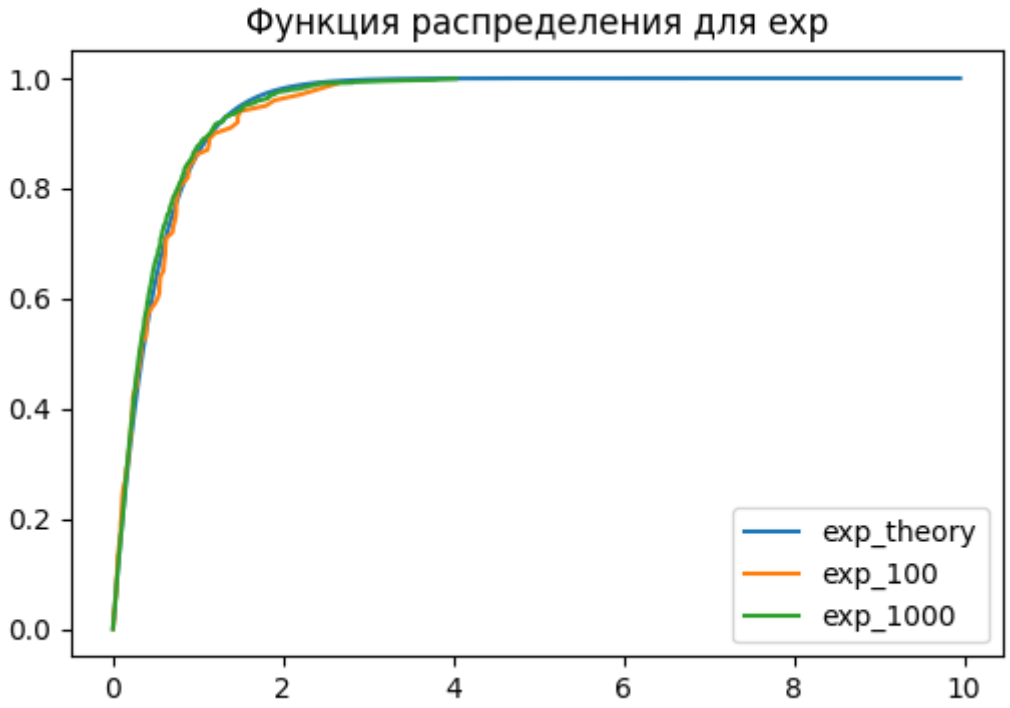
Функции распределения

In [22]:

```
cdf_exp_100 = cdf(distrib_exp_100)
cdf_exp_1000 = cdf(distrib_exp_1000)
cdf_norm_100 = cdf(distrib_norm_100)
cdf_norm_1000 = cdf(distrib_norm_1000)
x_exp = np.arange(0.0, 10.0, 0.05)
cdf_theory_exp = cdf_exp(x_exp, exp_lambda)
x_norm = np.arange(-10.0, 50.0, 0.05)
cdf_theory_norm = [cdf_norm(x, norm_mean, norm_std) for x in x_norm]
```

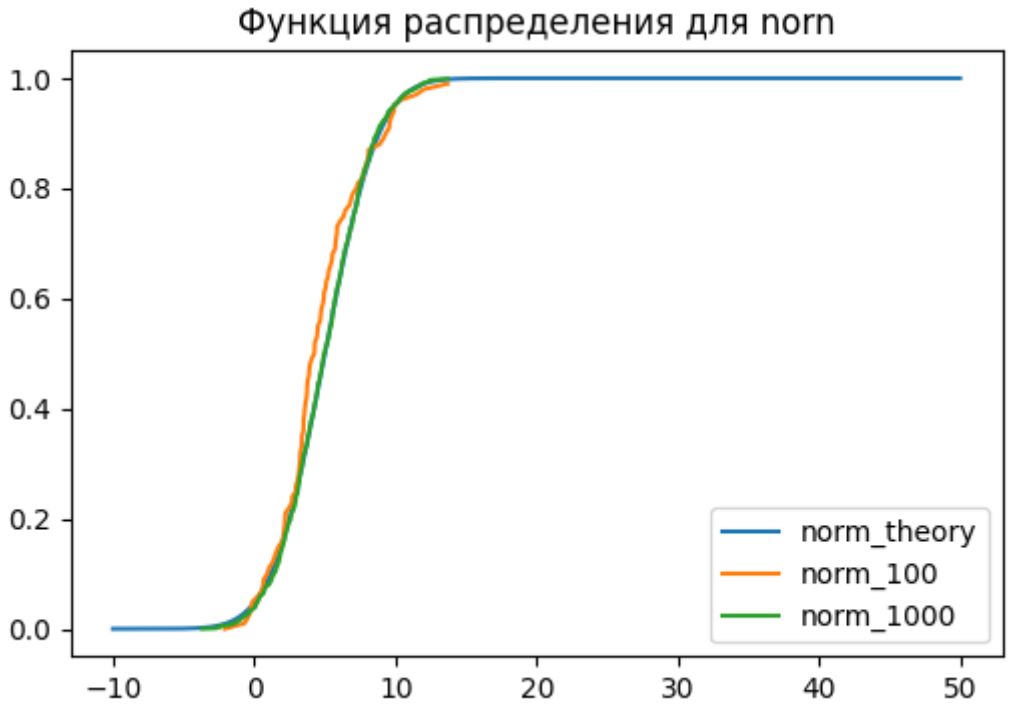
In [23]:

```
plt.figure(dpi=100)
plt.title("Функция распределения для exp")
plt.plot(x_exp, cdf_theory_exp, label='exp_theory')
plt.plot(cdf_exp_100['x'], cdf_exp_100['y'], label='exp_100')
plt.plot(cdf_exp_1000['x'], cdf_exp_1000['y'], label='exp_1000')
plt.legend()
plt.show()
```



In [24]:

```
plt.figure(dpi=100)
plt.title("Функция распределения для norn")
plt.plot(x_norm, cdf_theory_norm, label='norm_theory')
plt.plot(cdf_norm_100['x'], cdf_norm_100['y'], label='norm_100')
plt.plot(cdf_norm_1000['x'], cdf_norm_1000['y'], label='norm_1000')
plt.legend()
plt.show()
```



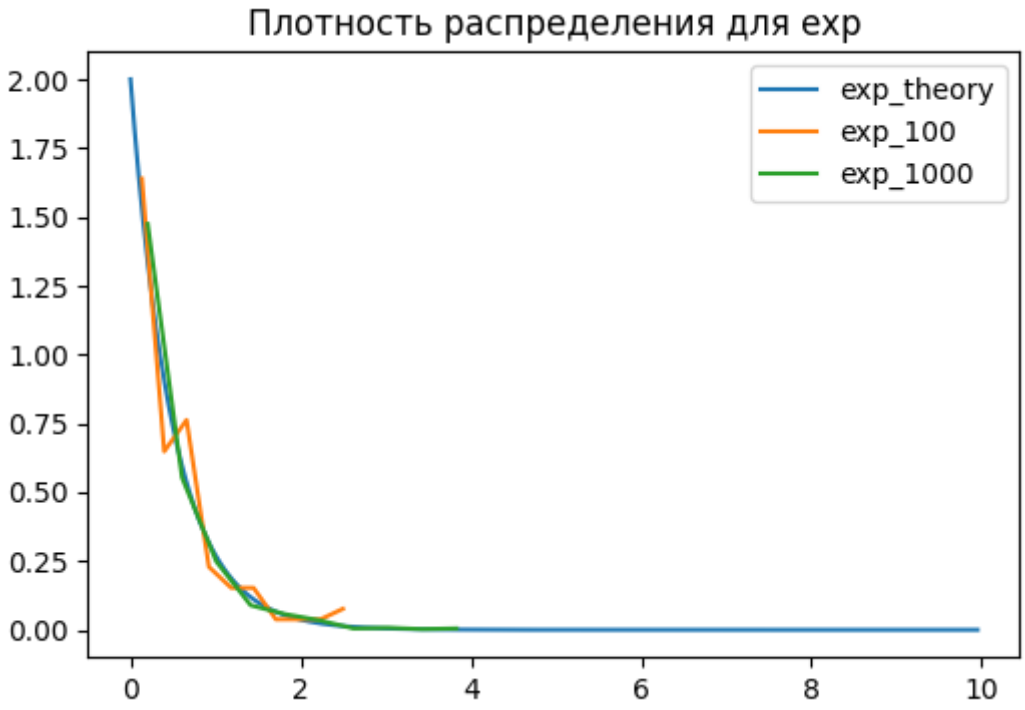
Плотность распределения

In [25]:

```
pdf_exp_100 = pdf(10, distrib_exp_100)
pdf_exp_1000 = pdf(10, distrib_exp_1000)
pdf_norm_100 = pdf(10, distrib_norm_100)
pdf_norm_1000 = pdf(10, distrib_norm_1000)
pdf_theory_exp = pdf_exp(x_exp, exp_lambda)
pdf_theory_norm = [pdf_norm(x, norm_mean, norm_std) for x in x_norm]
```

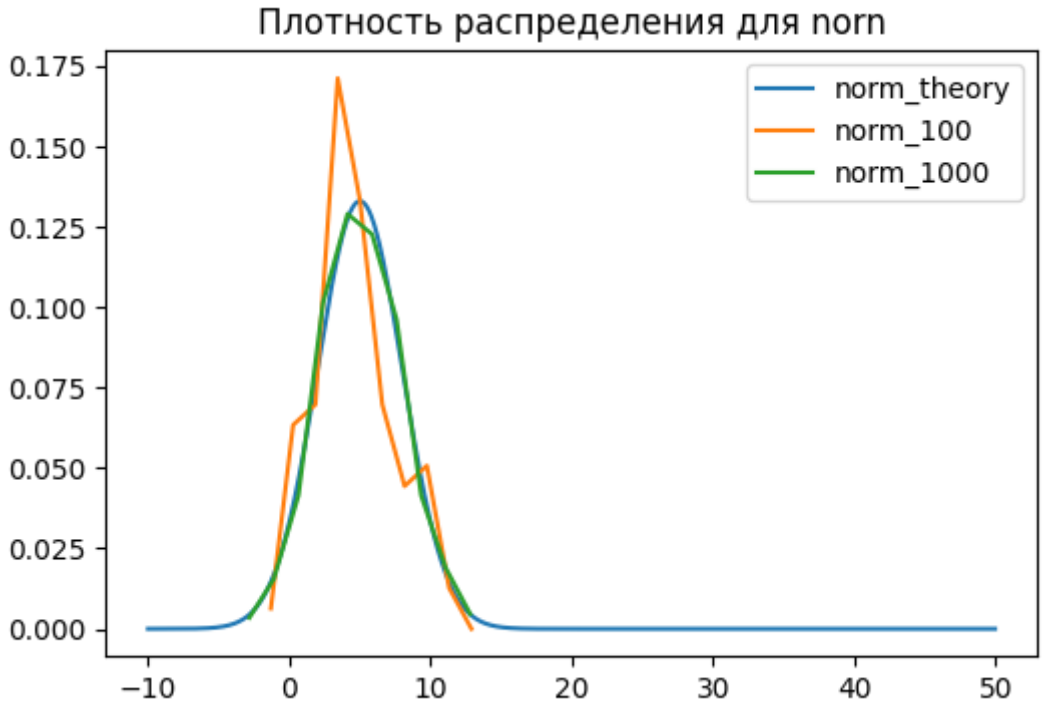
In [26]:

```
plt.figure(dpi=100)
plt.title("Плотность распределения для exp")
plt.plot(x_exp, pdf_theory_exp, label='exp_theory')
plt.plot(pdf_exp_100['x'], pdf_exp_100['y'], label='exp_100')
plt.plot(pdf_exp_1000['x'], pdf_exp_1000['y'], label='exp_1000')
plt.legend()
plt.show()
```



In [27]:

```
plt.figure(dpi=100)
plt.title("Плотность распределения для norn")
plt.plot(x_norm, pdf_theory_norm, label='norm_theory')
plt.plot(pdf_norm_100['x'], pdf_norm_100['y'], label='norm_100')
plt.plot(pdf_norm_1000['x'], pdf_norm_1000['y'], label='norm_1000')
plt.legend()
plt.show()
```



Задание 2. Сгенерировать три выборки размера 100, 1000 и 10000 для случайных расстояний между двумя точками, равномерно распределенные в прямоугольнике со сторонами 10 и 30. Получить среднее значение расстояния между точками, построить функцию распределения вероятностей и плотности вероятностей случайных расстояний. Показать разницу между соответствующими функциями на одном графике.

In [28]:

```
def distance(x1, x2):
    return math.sqrt((x1[0] - x2[0])**2 + (x1[1] - x2[1])**2)
```

In [29]:

```
def distance_generator(N):
    a = 10
    b = 30
    distances = []
    for i in range(N):
        x1 = np.random.uniform(0,a)
        y1 = np.random.uniform(0,b)
        x2 = np.random.uniform(0,a)
        y2 = np.random.uniform(0,b)
        distances.append(distance([x1,y1], [x2,y2]))
    distances.sort()
    return distances
```

In [30]:

```
distances_100 = distance_generator(100)
distances_1000 = distance_generator(1000)
distances_10000 = distance_generator(10000)
```

In [31]:

```
print("Среднее значение между точками (100): {}".format(np.average(distances_100)))
print("Среднее значение между точками (1000): {}".format(np.average(distances_1000)))
print("Среднее значение между точками (10000): {}".format(np.average(distances_1000
```

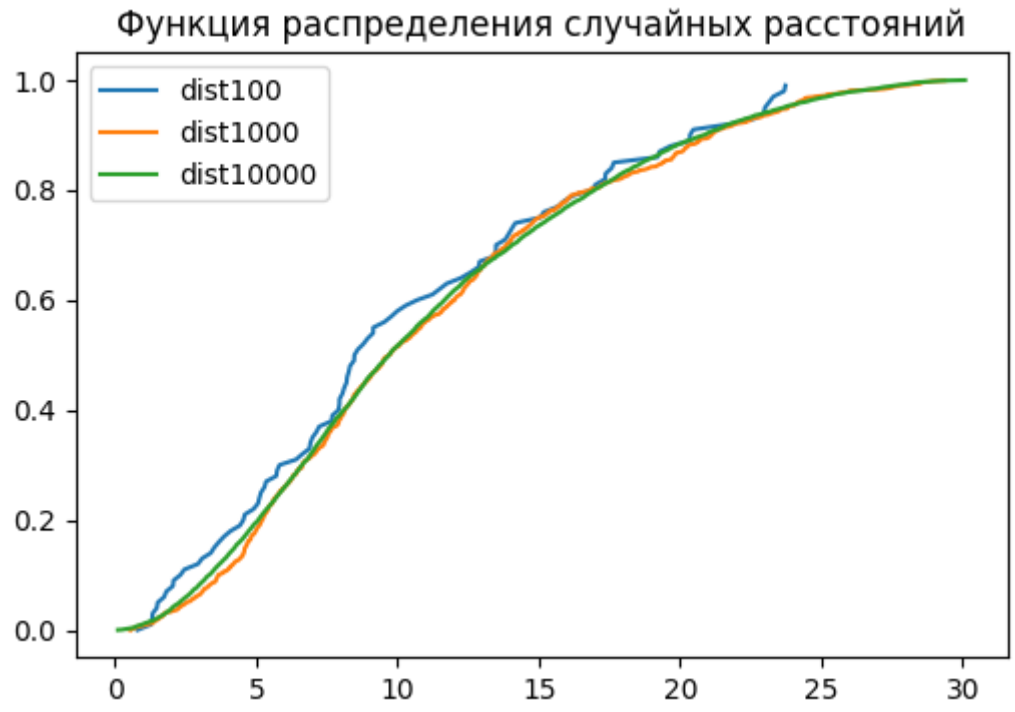
```
Среднее значение между точками (100): 10.211726037084679
Среднее значение между точками (1000): 11.110502048078281
Среднее значение между точками (10000): 11.00962327803038
```

In [32]:

```
cdf_dist_100 = cdf(distances_100)
cdf_dist_1000 = cdf(distances_1000)
cdf_dist_10000 = cdf(distances_10000)
```

In [33]:

```
plt.figure(dpi=100)
plt.title("Функция распределения случайных расстояний")
plt.plot(cdf_dist_100['x'], cdf_dist_100['y'], label='dist100')
plt.plot(cdf_dist_1000['x'], cdf_dist_1000['y'], label='dist1000')
plt.plot(cdf_dist_10000['x'], cdf_dist_10000['y'], label='dist10000')
plt.legend()
plt.show()
```



In [34]:

```
pdf_dist_100 = pdf(10, distances_100)
pdf_dist_1000 = pdf(10, distances_1000)
pdf_dist_10000 = pdf(10, distances_10000)
```

In [35]:

```
plt.figure(dpi=100)
plt.title("Плотность распределения случайных расстояний")
plt.plot(pdf_dist_100['x'], pdf_dist_100['y'], label='dist100')
plt.plot(pdf_dist_1000['x'], pdf_dist_1000['y'], label='dist1000')
plt.plot(pdf_dist_10000['x'], pdf_dist_10000['y'], label='dist10000')
plt.legend()
plt.show()
```