In [16]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import math
import tqdm
import matplotlib.pyplot as plt
```

Домашнее задание №4

Выполнил: Мартиросян Арутюн Артушович, группа ВВО-19

Алгоритм Метрополиса

Задание

In [3]:
```
from IPython.display import Image
Image(filename='task4_method(pic1).JPG', width=700, height=400)
```

Out[3]:

1) Пусть N частиц вначале распределены произвольным образом внутри объема V (для примера на плоскости)
2) Возьмем некоторую частицу j (x_j^n, y_j^n), окружим ее квадратом $a \times a$, где точка j в центре квадрата, и переместим эту частицу случайным образом в любую точку квадрата

   $x_j^{n+1} = x_j^n + a(R_1 - 1/2)$

   $y_j^{n+1} = y_j^n + a(R_2 - 1/2)$

   где R_1, R_2 значения равномерно распределенной на $(0,1)$ случайной величины.

В результате на шаге $n+1$ образуется новая система

$(x_1^n, y_1^n, ..., x_j^{n+1}, y_j^{n+1}, ..., x_N^n, y_N^n)$

При перемещении частицы произошло изменение потенциальной энергии ΔU

$$\Delta U = \sum_{\substack{i=1 \\ i \neq j}}^{N} \{\Phi(|\vec{r}_i^{\,n+1} - \vec{r}_j^{\,n+1}|) - \Phi(|\vec{r}_i^{\,n} - \vec{r}_j^{\,n}|)\}$$

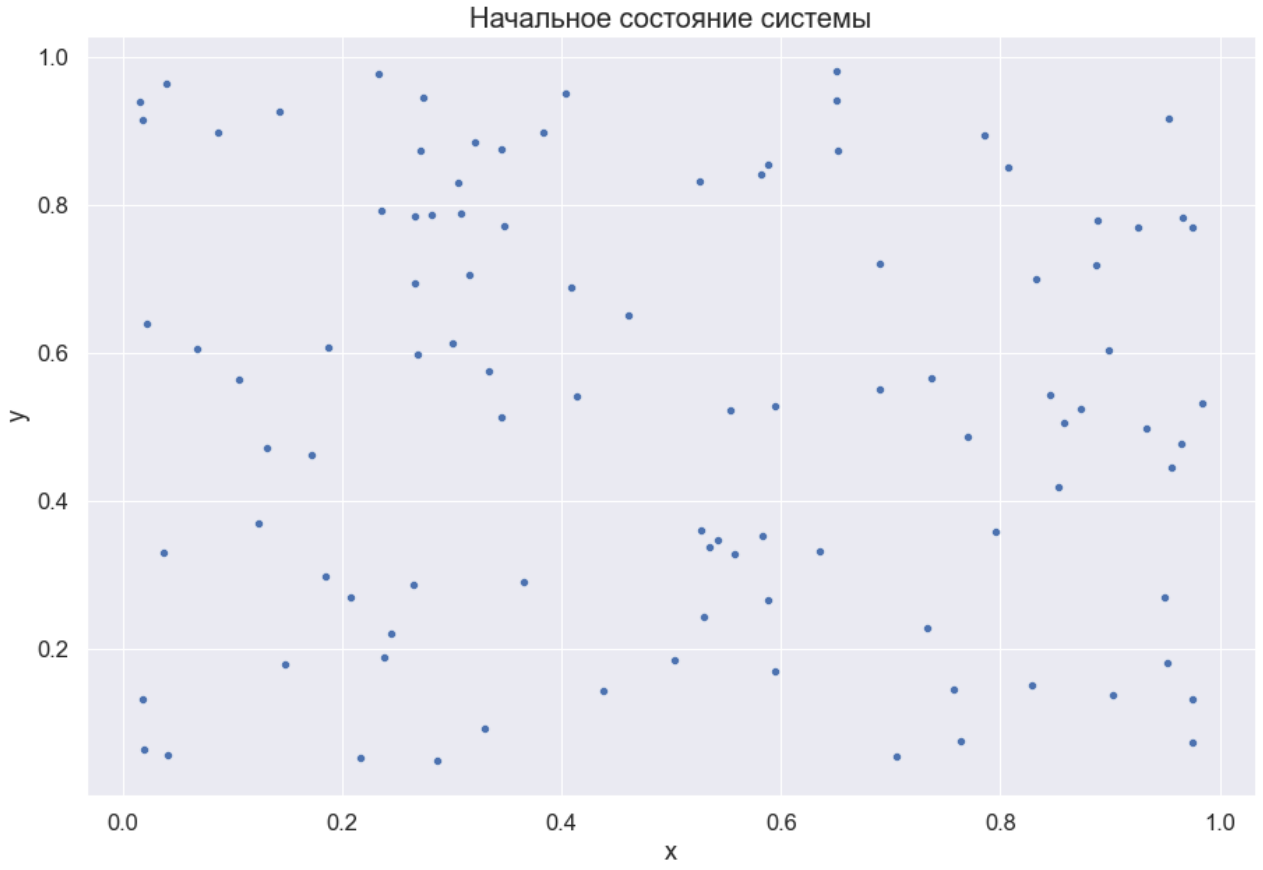
In [4]:
```
Image(filename='task4_method(pic2).JPG', width=700, height=200)
```

Out[4]:

Перед нами стоит задача построить канонический ансамбль так, чтобы частота появления систем, образующих ансамбль была пропорциональна $e^{-U/kT}$. Такое распределение можно построить, решая вопрос, является ли вновь построенная система допустимой или нет. Если $\Delta U < 0$, т.е. перемещение приводит к системе с меньшей энергией, то новая система включается в ансамбль. С другой стороны, если образовалась система с большей энергией, то система включается в ансамбль лишь с вероятностью $e^{-\Delta U/kT}$. Для этого выбирается третье равномерно распределенное на $(0,1)$ случайное число R_3, и если $R_3 < e^{-\Delta U/kT}$, то новая система присоединяется к ансамблю, если $R_3 > e^{-\Delta U/kT}$ то новая система отбрасывается, и в ансамбль еще раз добавляется старая система. Выбор a расстояние между частицами $a \sim L/N$, где L линейный размер области.

In [7]:
```
matrix = np.random.rand(100,2)
fractions = pd.DataFrame(matrix, columns=['x','y'])
```

In [11]:
```
plt.style.use('seaborn')
sns.set(font_scale=1.5)
plt.figure(figsize = (15,10))
plt.title('Начальное состояние системы', fontsize=20)
_ = sns.scatterplot(data=fractions, x='x', y='y')
```



In [12]:
```
a = 0.01
def point_change(matrix, i):
    result = matrix.copy()
    while True:
        new_coord_x = result[i,0] + a*(np.random.rand(1)[0] - 0.5)
        if new_coord_x < 1 and new_coord_x > 0:
            result[i,0] = new_coord_x
            break
    while True:
        new_coord_y = result[i,1] + a*(np.random.rand(1)[0] - 0.5)
        if new_coord_y < 1 and new_coord_y > 0:
            result[i,1] = new_coord_y
            break
    return result
```

In [13]:
```
indices = []
for i,j in np.array([(i,j) for i in range(len(matrix)) for j in range(len(matrix)) if i != j]):
    if [j,i] not in indices:
        indices.append([i,j])

def potential_energy(matrix):
    res = 0
    for i,j in indices:
        r = np.sqrt((matrix[i,0] - matrix[j,0])**2
                    + (matrix[i,1] - matrix[j,1])**2)
        F = 1/r**2 - 1/r
        res += 2*F
    return res
```

In [14]:
```
k = 1.38*10**(-23)
T = 25
```

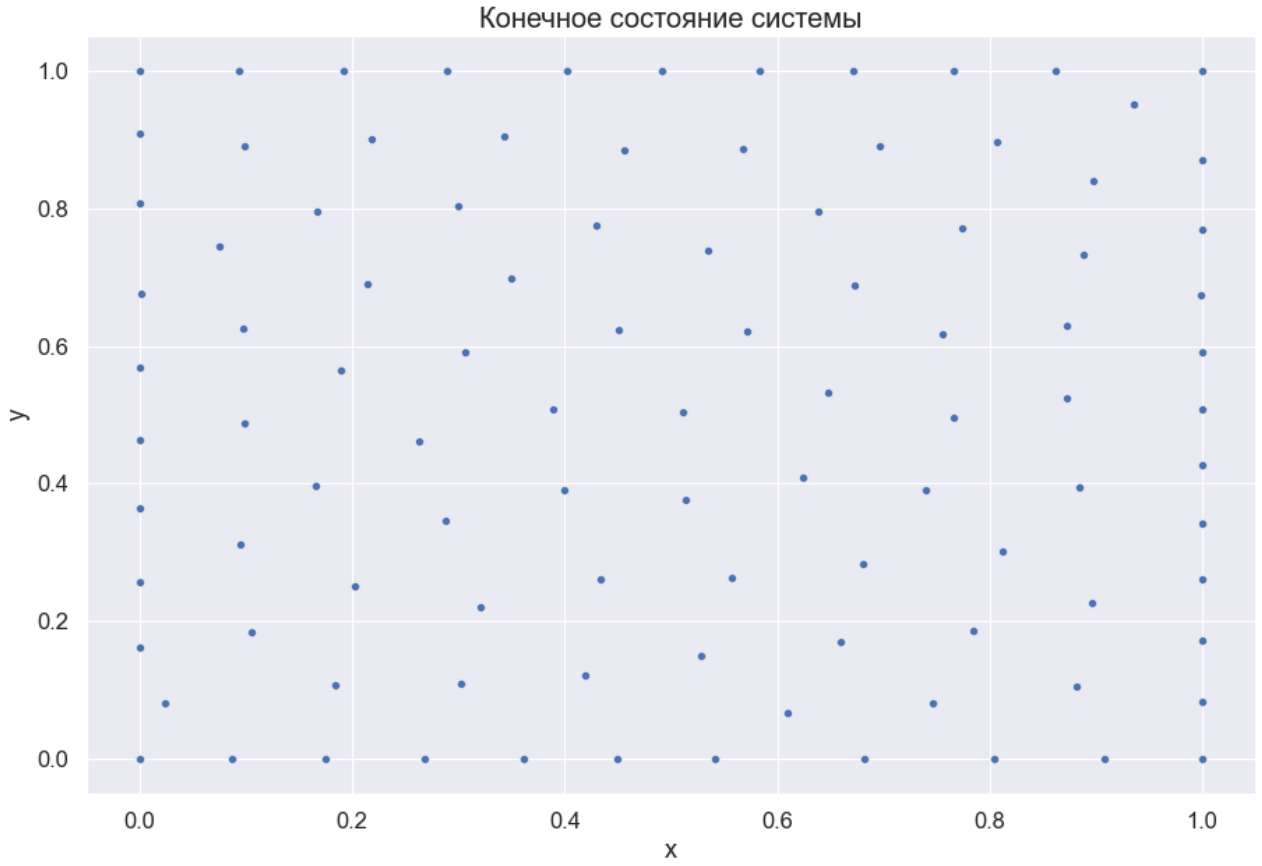
```
def calculate(matrix):
    m = matrix.copy()
    energy = []
    for iteration in tqdm.tqdm(range(20000)):
        i = np.random.randint(99)
        new_matrix = point_change(m, i)
        deltaU = potential_energy(new_matrix) - potential_energy(m)
        if deltaU <= 0:
            m = new_matrix.copy()
        else:
            probability = np.exp(-deltaU/(k*T))
            r = np.random.rand(1)[0]
            if r < probability:
                m = new_matrix.copy()
        energy.append(potential_energy(m))
    return m, energy
```

In [17]:
```
matrix_new, energy = calculate(matrix)
```

```
100%|██████████████████████████████████| 20000/20000 [19:44<00:00, 16.89it/s]
```

In [18]:
```
fractions_new = pd.DataFrame(matrix_new, columns=['x','y'])
sns.set(font_scale=1.5)
plt.figure(figsize = (15,10))
plt.title('Конечное состояние системы', fontsize=20)
sns.scatterplot(data=fractions_new, x='x', y='y')
```

Out[18]: <AxesSubplot:title={'center':'Конечное состояние системы'}, xlabel='x', ylabel='y'>



In [21]:
```
plt.figure(figsize = (15,10))
plt.title('График изменения потенциальной энергии системы', fontsize=20)
_ = plt.plot(energy)
```


График изменения потенциальной энергии системы
180000
160000
140000
120000
100000
80000
60000
0
2500
5000
7500
10000
12500
15000
17500
20000