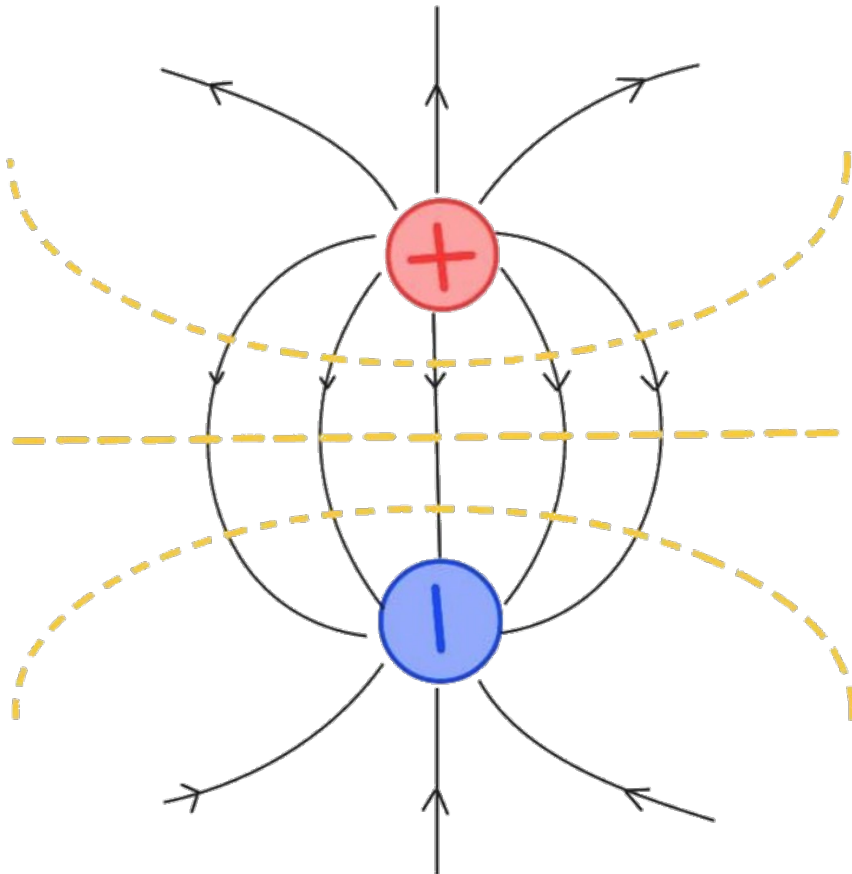


Распределение потенциалов в жидкостях

Поместите два электрода в емкость с водой и подайте на них безопасное напряжение. **Определите распределение.** Исследуйте, насколько найденные эквипотенциальные поверхности соответствуют предположениям **для различных условий эксперимента.**



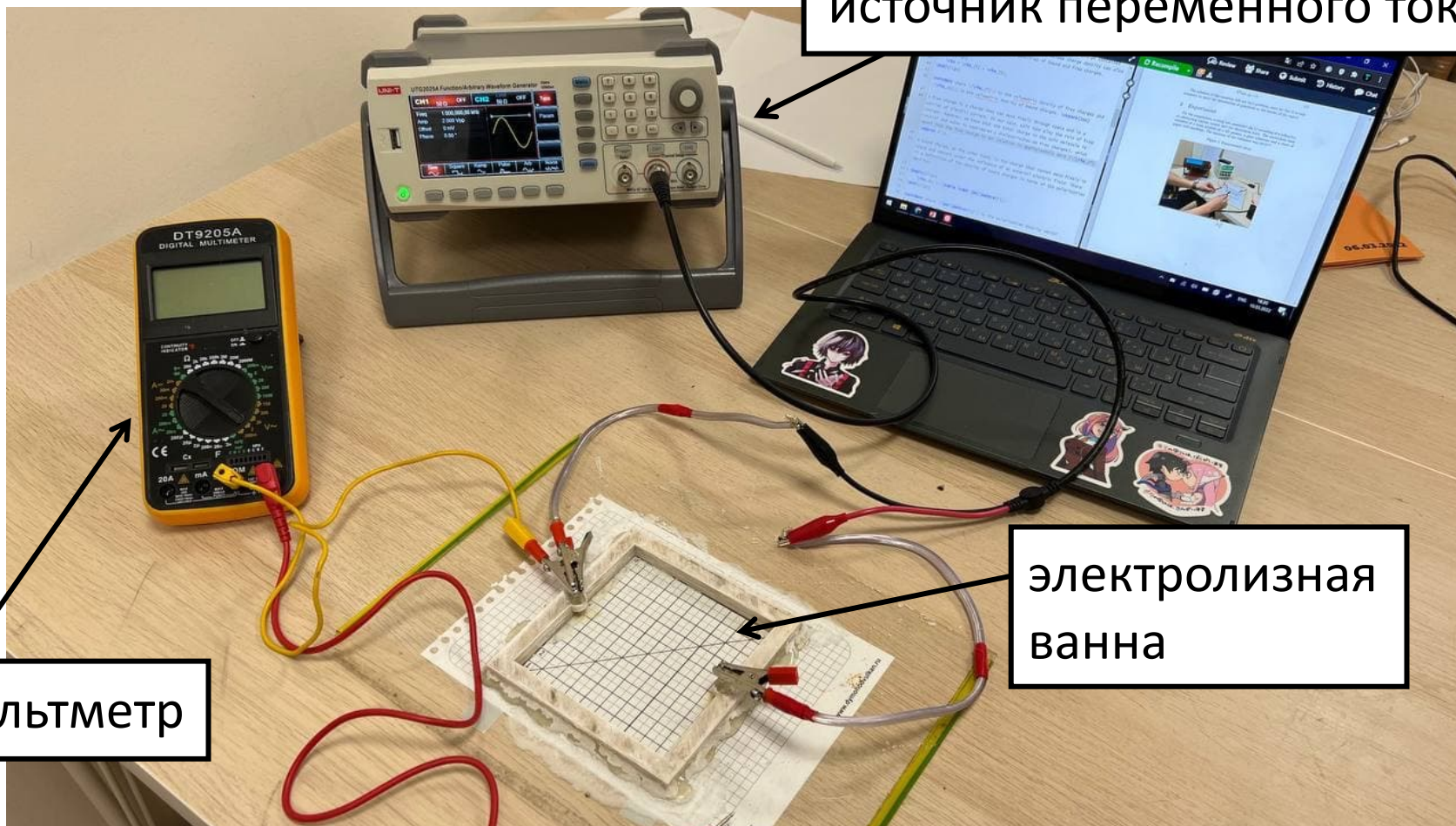
Github

Экспериментальная установка

источник переменного тока

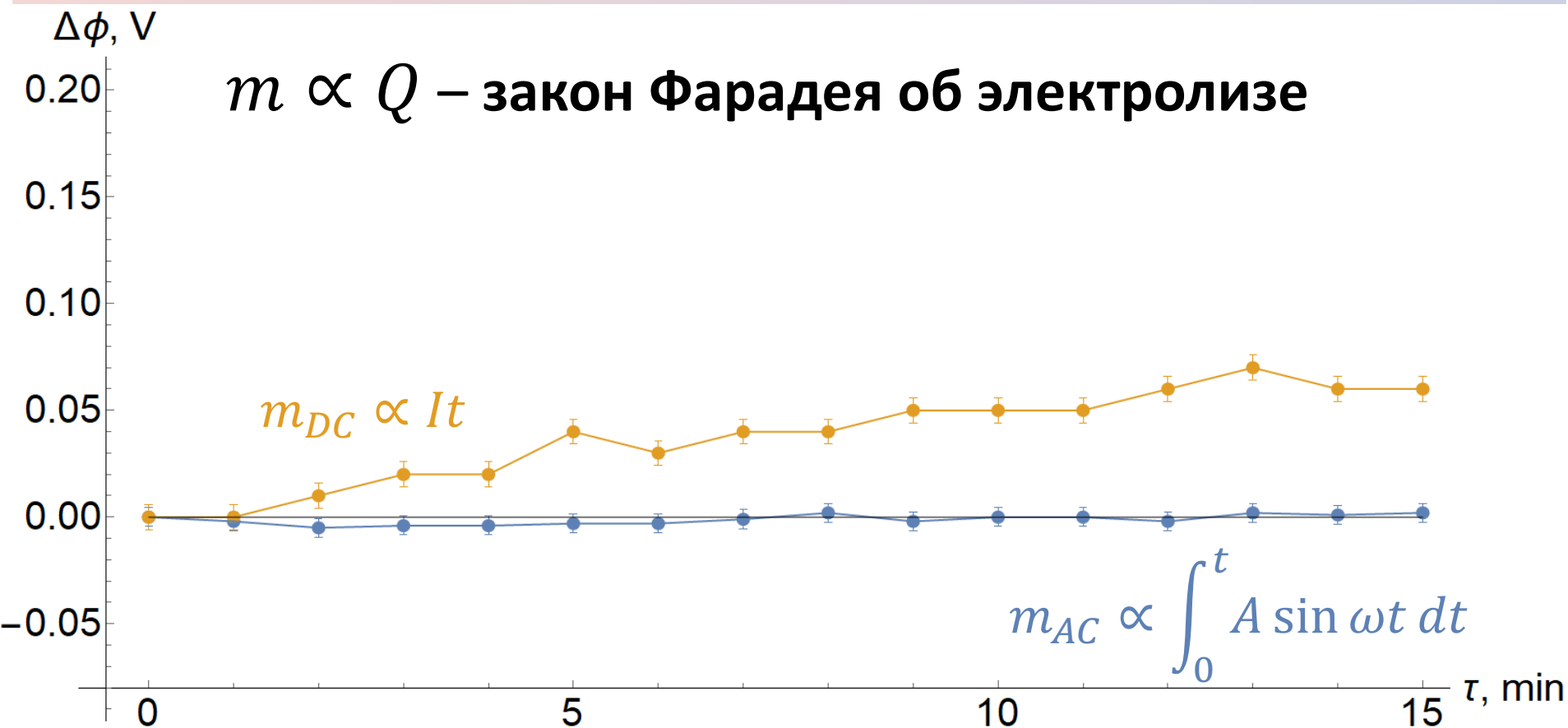
электролизная
ванна

вольтметр



Введение

Переменный и постоянный ток



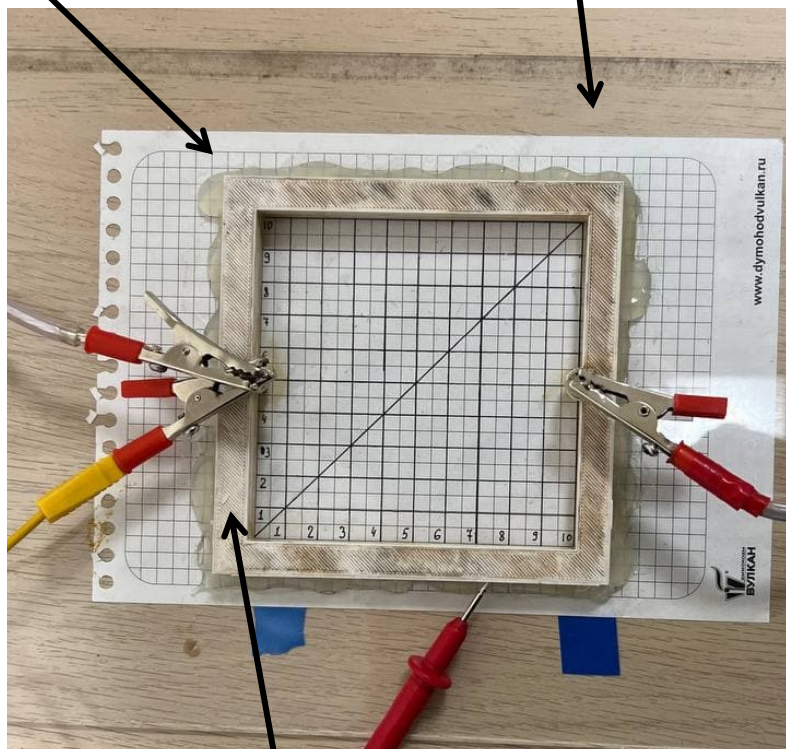
В дальнейших экспериментах пользуемся
высокочастотным переменным током

Введение

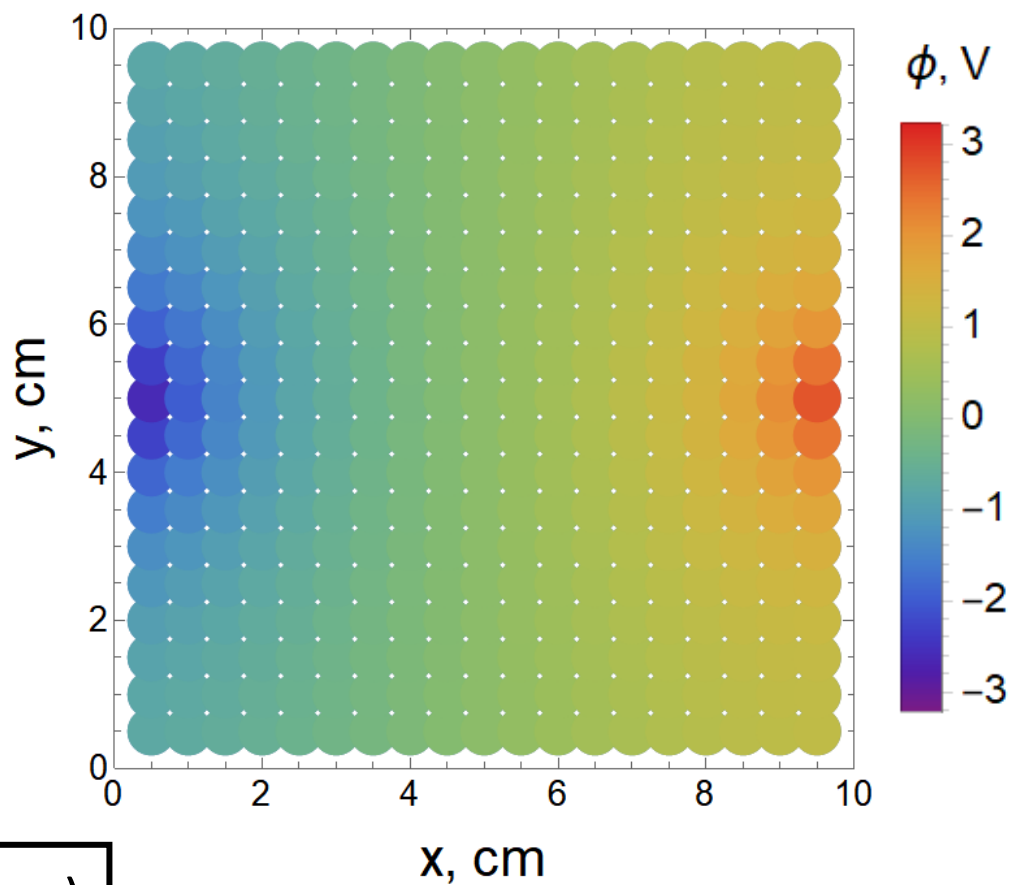
Дискретность потенциала

сетка

стеклянная база



пластиковый корпус (3D-печать)



Введение

Получение эквипотенциальных линий

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
file = '/Users/sofiakanukova/Downloads/killme.xlsx'
xl = pd.ExcelFile(file)
data = xl.parse('voltage5', skiprows=1, usecols='B:D')
n = len(data)
x = []
y = []
v = []
```

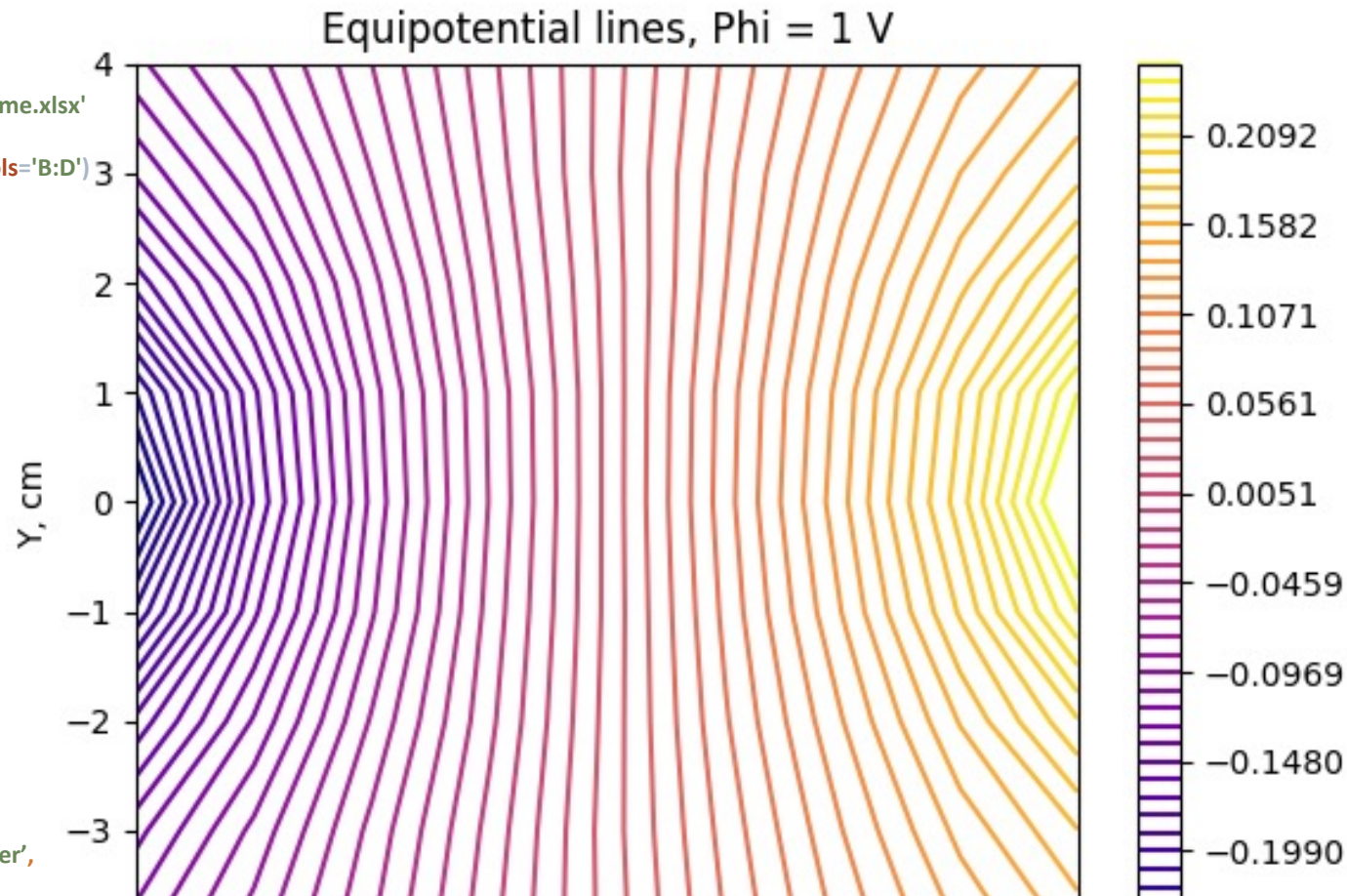
```
for i in range(n):
    a = int(data['x, cm'].iloc[i])
    x.append(int(data['x, cm'].iloc[i]))
    a = int(data['y, cm'].iloc[i])
    y.append(int(data['y, cm'].iloc[i]))
    a = int(data['phi, V'].iloc[i])
    v.append(float(data['phi, V'].iloc[i]))
```

```
V = np.zeros((9, 9))
```

```
for i in range(9):
    for j in range(9):
        V[i, j] = v[9*i+j] - 0.5
```

```
lvl = np.linspace(-0.25, 0.25, 50)
```

```
plt.contour(V.T, cmap='plasma', origin='lower',
levels=lvl, extent=[-4.5, 4.5, -4.5, 4.5])
plt.colorbar()
plt.xlabel('X, cm')
plt.ylabel('Y, cm')
plt.title('Equipotential lines, Phi = 1 V')
plt.show()
```



!

Эквипотенциальные линии получены

Введение

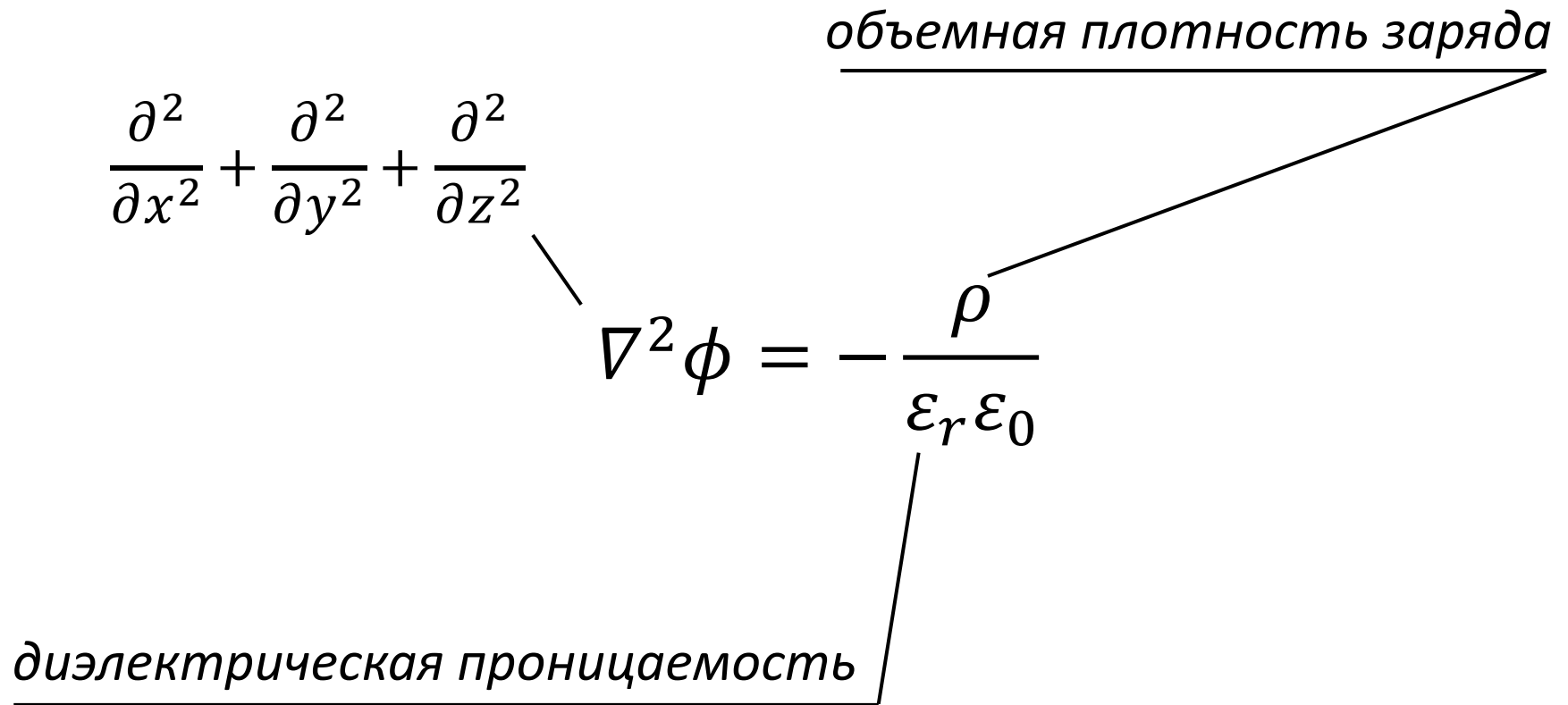
Теоретическая модель

Уравнение Пуассона

$$\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2} \quad \nabla^2 \phi = - \frac{\rho}{\epsilon_r \epsilon_0}$$

объемная плотность заряда

диэлектрическая проницаемость



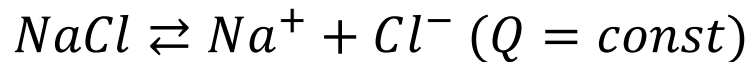
Теоретическая модель

Объемная плотность заряда

свободный & связанный заряд

$$\rho = V^{-1}(Q_f + Q_b)$$

Свободный заряд:



свободный заряд сохраняется в
процессе реакции

$$Q_f = 0$$

Связанный заряд:

$$Q_b = -\frac{\chi}{1 + \chi} Q_f$$

$$Q_b = 0$$

! Объемная плотность заряда **равна 0**



Теоретическая модель

Уравнение Лапласа

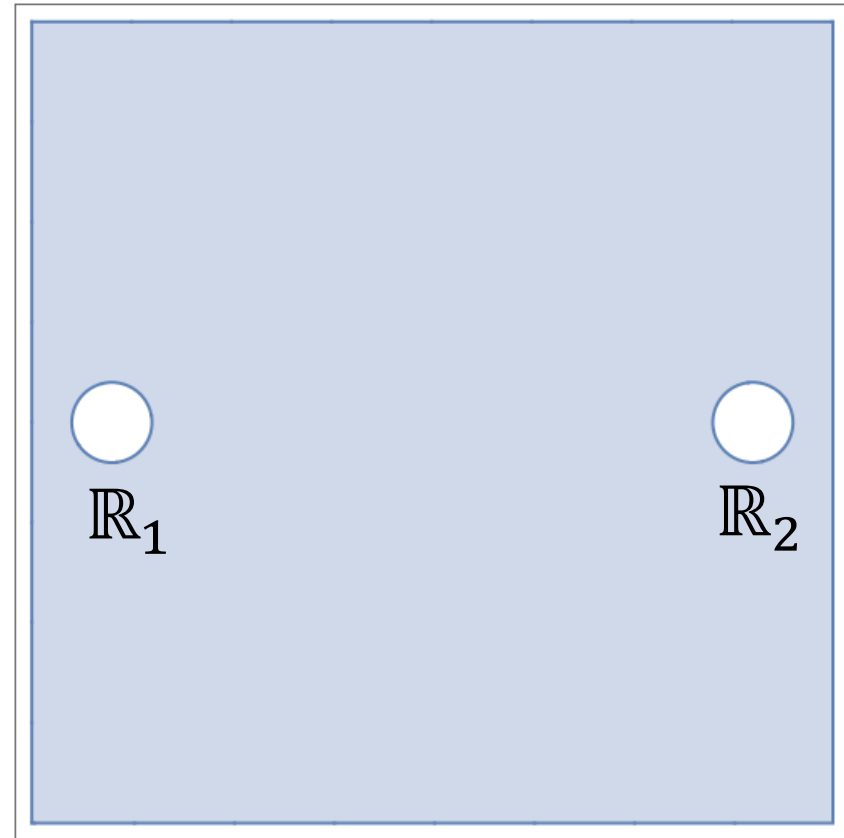
$$\nabla^2 \phi = -\frac{\rho}{\epsilon_r \epsilon_0} \rightarrow \nabla^2 \phi = 0$$

! Можно решить численно

Граничные условия:

подаваемая разность потенциалов

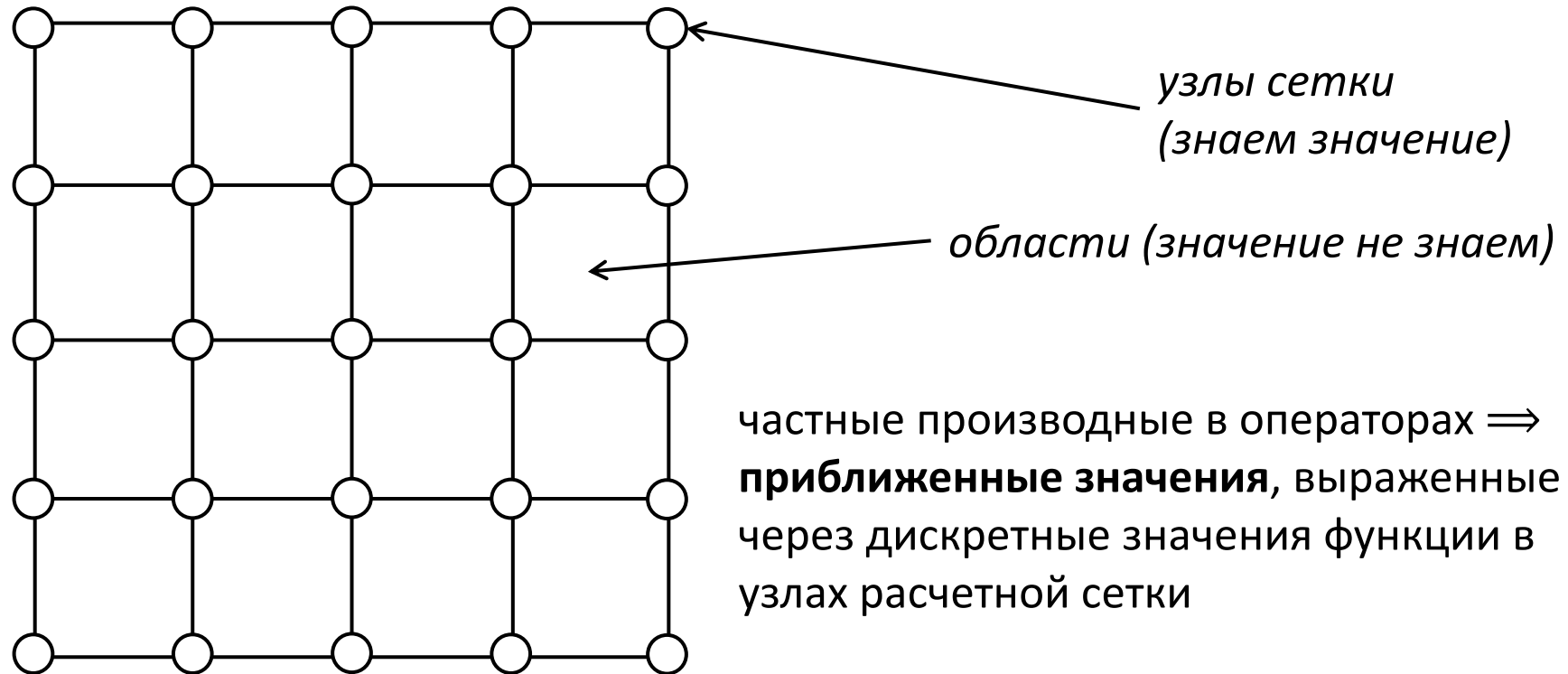
$$\begin{cases} \phi(x, y, z) = -\frac{U}{2}, \{x, y, z\} \in \mathbb{R}_1 \\ \phi(x, y, z) = \frac{U}{2}, \{x, y, z\} \in \mathbb{R}_2 \end{cases}$$



Теоретическая модель

Модель Python

Метод конечных разностей



Формула конечных разностей для уравнения Лапласа:

$$V(i, j) = \frac{V(i - 1, j) + V(i + 1, j) + V(i, j - 1) + V(i, j + 1)}{4}$$

Модель Python

Алгоритм

- Задать **границы** области и **число узлов** сетки в каждом направлении;
- Определить **шаг сетки** в каждом направлении, создать ее;
- Задать **начальные условия** для функции в узлах сетки;
- Задать **граничные условия** на границе области и круговых электродах;
- На каждой итерации вычислить значения функции в узлах сетки, используя **формулу конечных разностей**;
- Повторять шаг 6 до тех пор, пока значения функции не сойдутся;

Используемые библиотеки

```
import numpy as np  
import matplotlib.pyplot as plt
```

Модель Python

Задание параметров

```
r = 0.5 #радиус круговых электродов
L = 3.5 #размер области
N = 100 #количество точек на оси x и y
h = L / (N-1) #шаг сетки
r1 = 0.1*L #расстояния до электродов
r2 = 0.9*L
phi = 40 #разность потенциалов
x = np.linspace(0, L, N)
y = np.linspace(0, L, N)
```


Создание сетки и начальных условий

```
X, Y = np.meshgrid(x, y)  
V = np.zeros((N, N))
```

Граничные условия

```
for i in range(N):
    for j in range(N):
        if i == 0 or i == N-1 or j == 0 or j == N-1:
            V[i,j] = 0.0
        elif np.sqrt((X[i, j]-r1)**2 + (Y[i, j]-L/2)**2) < r:
            V[i,j] = phi/2
        elif np.sqrt((X[i, j]-r2)**2 + (Y[i, j]-L/2)**2) < r:
            V[i, j] = -phi/2
V_old = V.copy() #текущее решение
V_new = V.copy() #новое решение
```

Модель Python

Реализация МКР

`tol = 1e-2` #порог точности

`delta = 1` #изменение решения на текущей итерации

`while delta > tol:`

`for i in range(1, N - 1):`

`for j in range(1, N - 1):`

`if np.sqrt((X[i, j]-r1)**2 + (Y[i, j]-L/2)**2) >= r and np.sqrt((X[i, j]-r2)**2 + (Y[i, j]-L/2)**2) >= r:`

`V_new[i, j] = 0.25 * (V_old[i - 1, j] + V_old[i + 1, j] + V_old[i, j - 1] + V_old[i, j + 1])`

`delta = np.max(np.abs(V_new - V_old))` #оценка изменения решения на итерации

`V_old = V_new.copy()` #обновление текущего решения

Модель Python

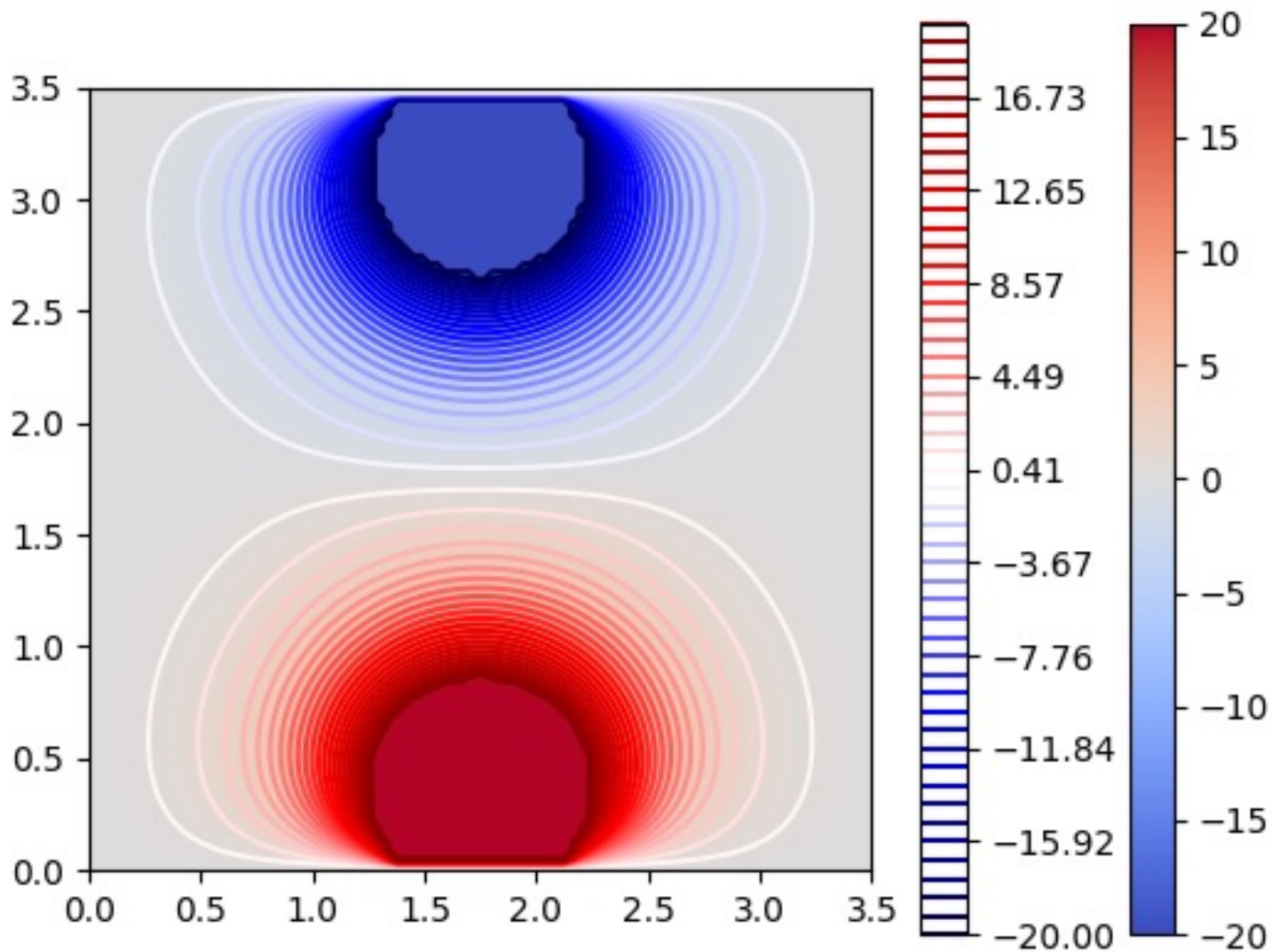
Визуализация

```
lvl = np.linspace(-20, 20, 50) #задание контуров

plt.imshow(V_new.T, cmap='coolwarm', origin='lower',
extent=[0, L, 0, L], aspect=1)
plt.colorbar()
plt.contour(V_new.T, cmap='seismic', origin='lower',
levels=lvl, extent=[0, L, 0, L])
plt.colorbar()
plt.figure(figsize=(cm_to_inch(20), cm_to_inch(20)))

plt.show()
```

Результат



Модель Python

Эксперимент

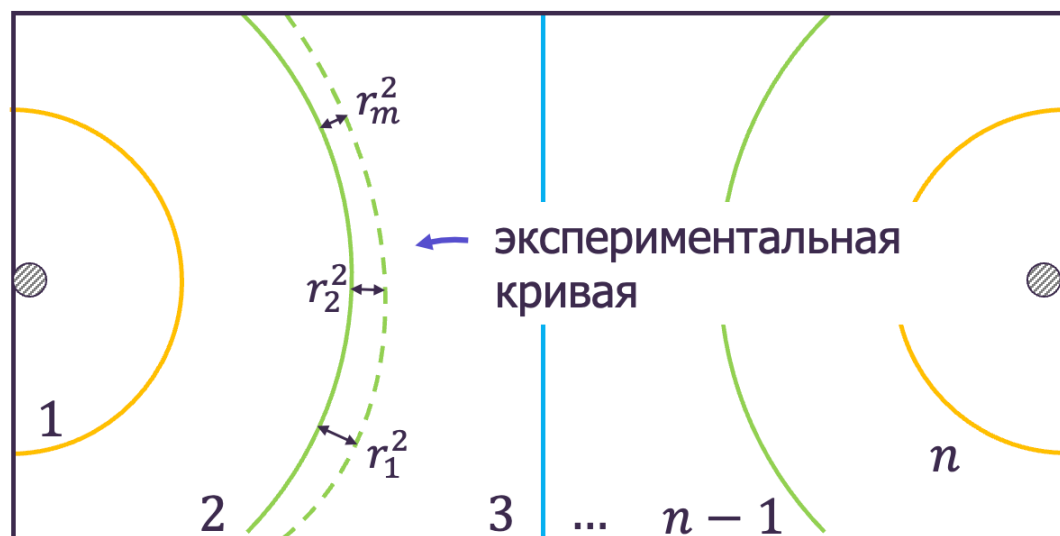
η метрика

Показывает относительное расстояние между экспериментальными и теоретическими линиями

$$\eta = \frac{\sum_i^n w_i \cdot \Delta r_i}{l \cdot \sum_j^n w_j}$$

w_i – длина i -й экспериментальной кривой

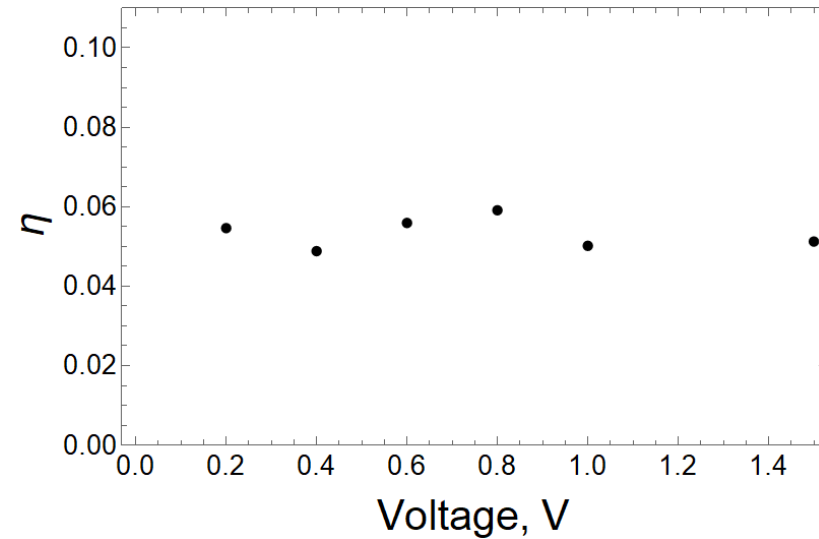
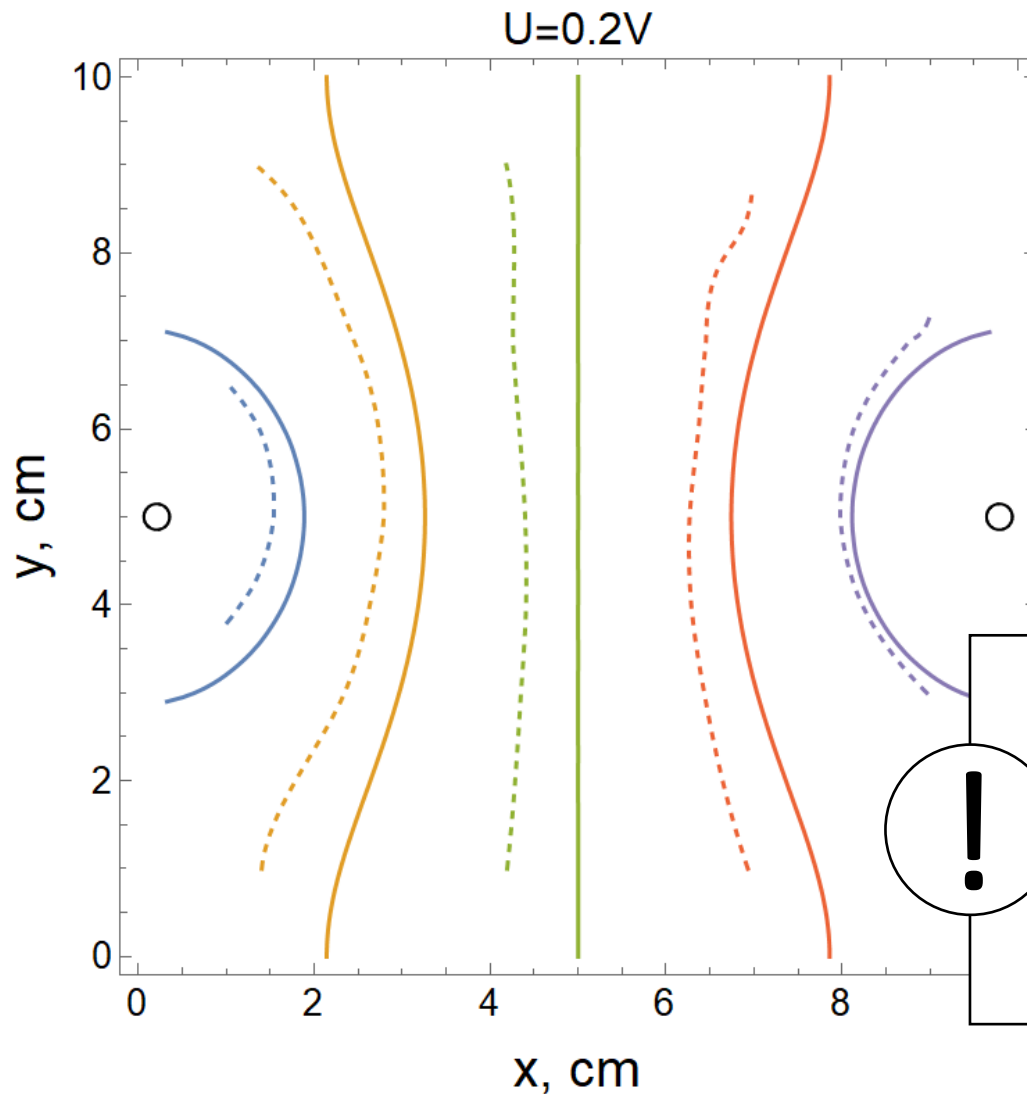
l – характерный размер электролизной ванны



$$\Delta r_2 = \frac{r_1^2 + r_2^2 + \dots + r_m^2}{m}$$

Эксперимент

Зависимость от разности потенциалов

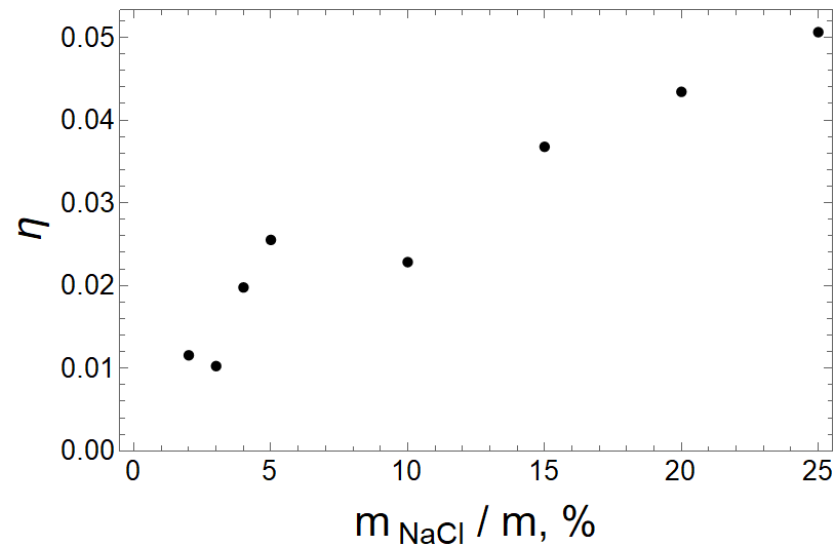
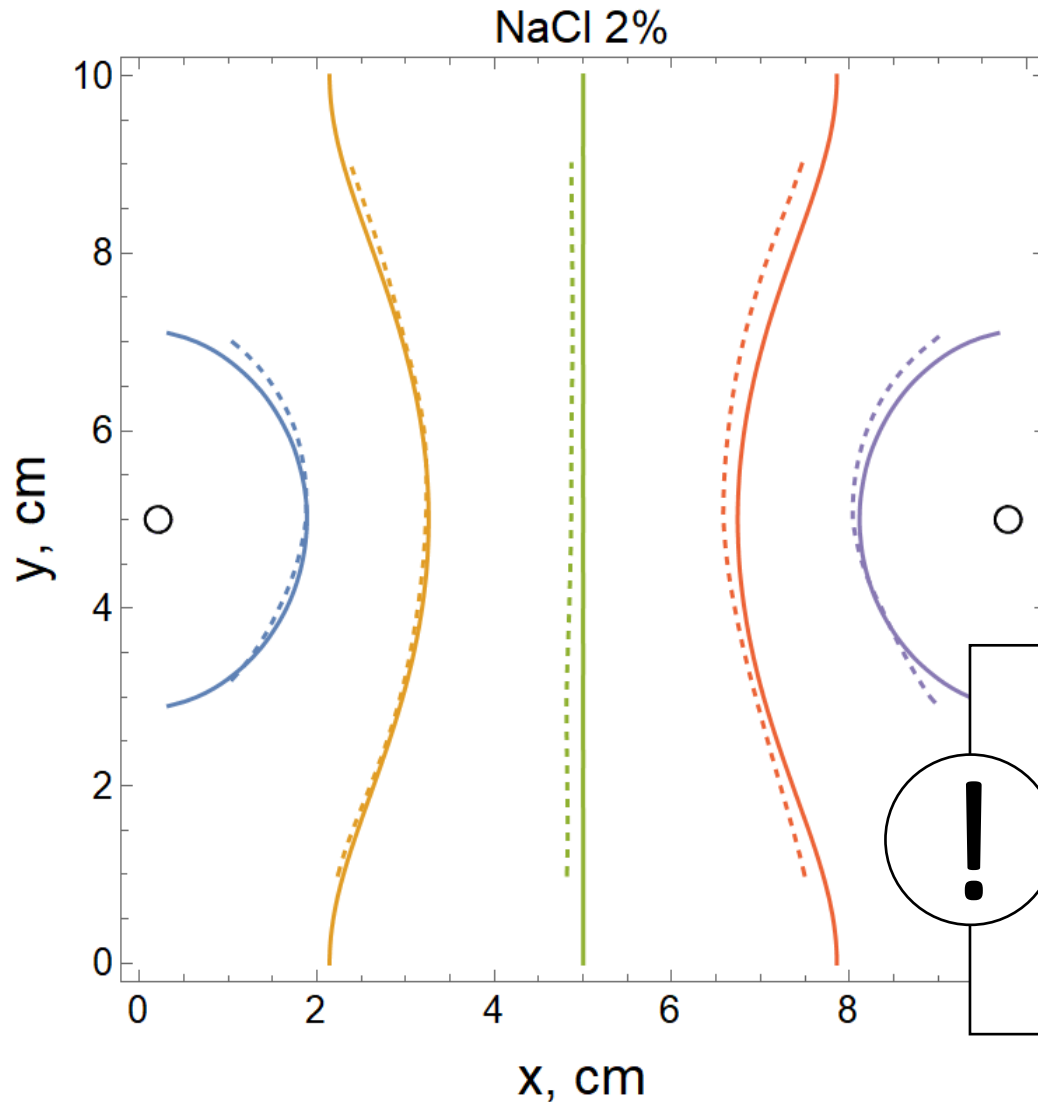


!

Как и ожидалось,
расположение линий **не**
зависит от подаваемого
напряжения

Эксперимент

Зависимость от среды



!

Текущая теория не
показывает зависимость
от среды, **отклонение**
растет с концентрацией

Эксперимент

Продвинутая теория

Potential distribution

Длина Дебая

$$\nabla^2 \phi = \lambda_d^{-2} \phi$$

Potential

$$\lambda_d^{-1} = \sqrt{\frac{\epsilon_r \epsilon_0 kT}{2e^2 I}}$$

ионная сила раствора

Coordinate



Осталось только суметь посчитать...

Эксперимент

Нововведения в коде

or k in range(1000):

#Используем метод конечных разностей для нахождения лапласиана

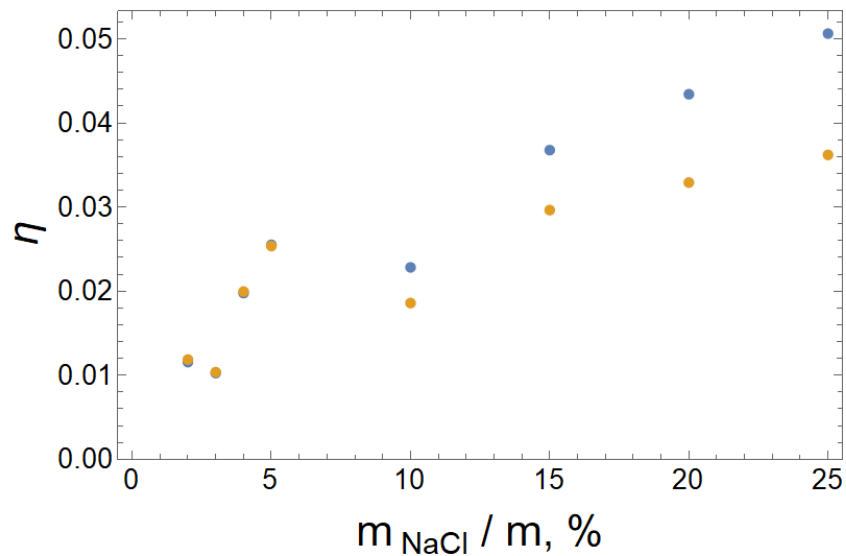
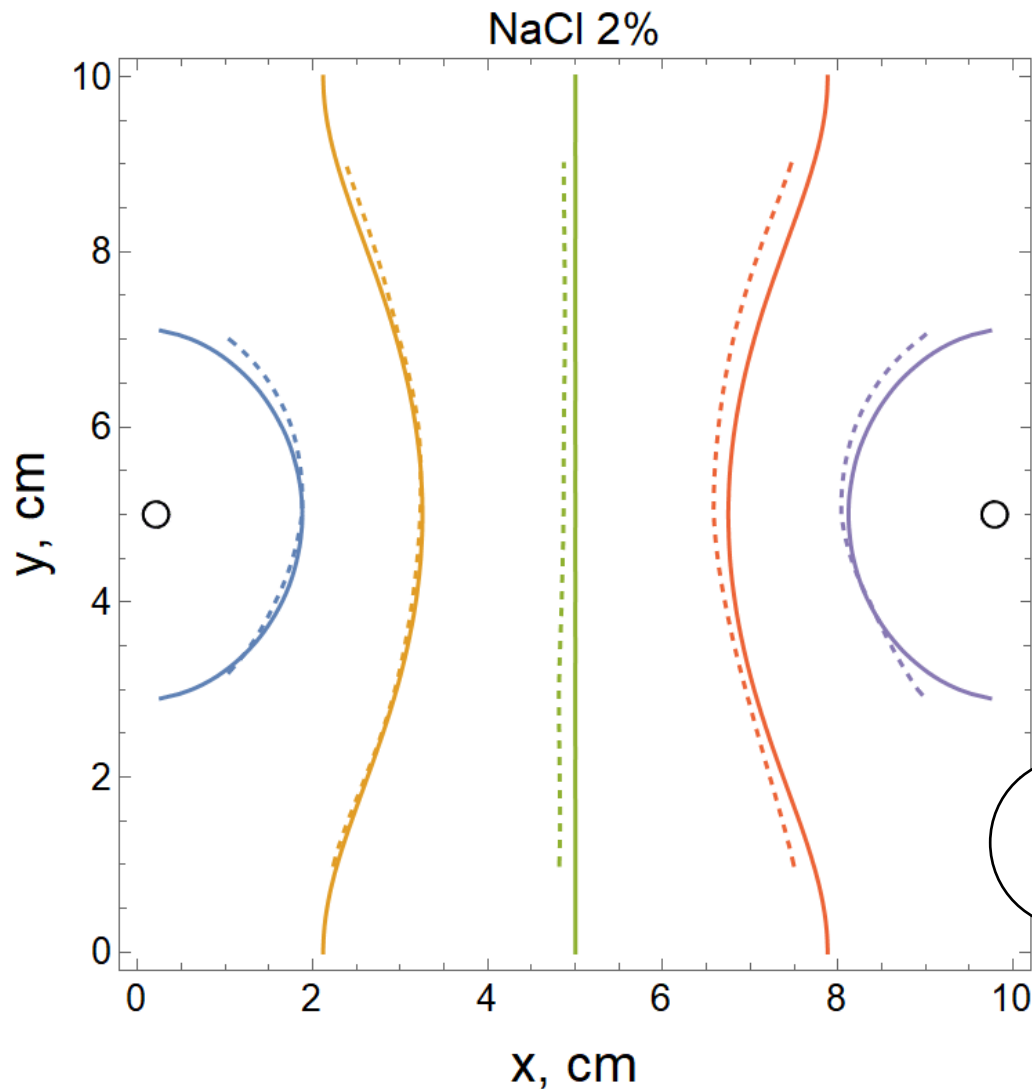
```
laplacian = (np.roll(V_old, 1, axis=0) + np.roll(V_old, -1, axis=0) + np.roll(V_old, 1, axis=1)+  
np.roll(V_old, -1, axis=1) - 2*V_old) / h**2 + (np.roll(V_old, 1, axis=0) + np.roll(V_old, -1,  
axis=0) + np.roll(V_old, 1, axis=1) + np.roll(V_old, -1, axis=1)-2*V_old) / h**2
```

#Обновляем значение функции phi

```
V_old += 0.1 * (c*V_old - laplacian)
```

Эксперимент

Зависимость от среды

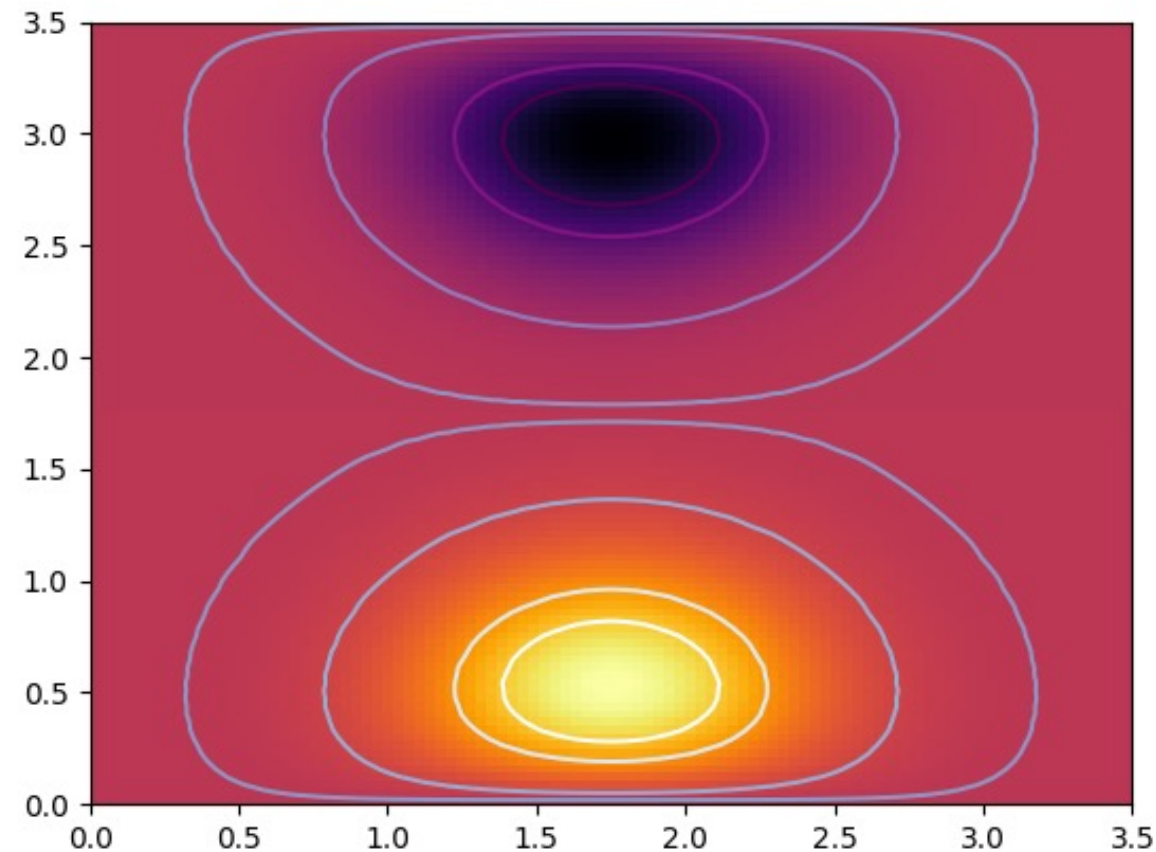


!

Продвинутая теория
работает лучше

Эксперимент

Спасибо за внимание!



Github