Этап 3. Комплексы программ. Описание программной реализации проекта.

Математическое моделирование

Королёв Иван Андреевич Оганнисян Давит Багратович Шуплецов Александр Кудряшов Артем Мугари Абдурахим

Содержание

Список иллюстраций

Список таблиц

# 1 Назначение этапа

Цель данного этапа - реализация численного алгоритма моделирования электрического пробоя в виде рабочего программного комплекса. Программа разрабатывается на языке Python с использованием научных библиотек и включает в себя все ключевые модули: от инициализации до визуализации

# 2 Язык и инструменты

* Язык программирования Python
* Основные библиотеки:
* NumPy - для работы с массивами и векторными вычислениями
* Matplotlib - для визуализации распределения потенциала и полей

# 3 Модуль инициализации

* Заданы размеры расчетной области: 1.0 м х 1.0 м.
* Сетка дискретизации: 100×100 узлов, шаги dx и dy.
* Определены параметры среды:
* диэлектрическая проницаемость ε (однородная);
* плотность заряда ρ (нулевая на старте);
* электрическая проводимость σ (низкое значение).
* Установлены граничные условия:
  1. нижняя граница потенциала = 0 В;
  2. верхняя граница потенциала = 1000 В.

Программный код:

# Параметры среды и расчётной области  
Lx, Ly = 1.0, 1.0  
Nx, Ny = 100, 100  
dx, dy = Lx / Nx, Ly / Ny  
x = np.linspace(0, Lx, Nx)  
y = np.linspace(0, Ly, Ny)  
  
# Диэлектрическая проницаемость, плотность заряда, проводимость  
epsilon = np.ones((Nx, Ny)) \* 8.85e-12  
rho = np.zeros((Nx, Ny))  
sigma = np.ones((Nx, Ny)) \* 1e-10  
  
# Начальные условия  
phi = np.zeros((Nx, Ny))  
E\_x = np.zeros((Nx, Ny))  
E\_y = np.zeros((Nx, Ny))  
tol = 1e-4  
  
# Граничные условия  
phi[:, 0] = 0 # нижняя граница  
phi[:, -1] = 1000 # верхняя граница

# 4 Модуль решения уравнения Пуассона

**1. Метод конечных разностей для дискретизации**

Метод конечных разностей — это способ замены производных в дифференциальных уравнениях на разностные выражения. Он позволяет перейти от непрерывного уравнения Пуассона:

к дискретной форме, пригодной для программной реализации. На равномерной сетке шагами и , лапласиан аппроксимируется следующим образом (в двумерном случае):

Отсюда выражается новая итерационная формула для :

**2. Метод Якоби (простых итераций)**

Метод Якоби — это итерационная схема, в которой новое значение потенциала рассчитывается исключительно на основе значений предыдущего шага . Все новые значения рассчитываются одновременно, независимо друг от друга, используя значения с предыдущей итерации.

Преимущества:

* Простота реализации.
* Хорошо подходит для параллелизации.

Недостатки:

* Медленная сходимость.
* Требует большого числа итераций по сравнению с другими методами (например, Гаусса–Зейделя).

**3. Критерий завершения итераций**

Чтобы остановить итерационный процесс, проверяется условие сходимости: если изменения потенциала между двумя последовательными шагами малы во всей расчетной области, то решение считается достигнутым.

Конкретно, в коде используется норма максимального изменения потенциала:

if np.max(np.abs(phi - phi\_old)) < tol:  
 break

где:

* phi\_old — массив потенциала с предыдущей итерации;
* phi — массив текущих значений;
* tol = 1e-4 — заданный порог точности.

Это означает, что итерации будут выполняться до тех пор, пока максимум абсолютных изменений между итерациями не станет меньше 0.0001 В.

Программный код:

def solve\_poisson(phi, rho, epsilon):  
 for \_ in range(5000):  
 phi\_old = phi.copy()  
 phi[1:-1, 1:-1] = 0.25 \* (  
 phi[:-2, 1:-1] + phi[2:, 1:-1] +  
 phi[1:-1, :-2] + phi[1:-1, 2:] +  
 dx\*\*2 \* rho[1:-1, 1:-1] / epsilon[1:-1, 1:-1]  
 )  
 if np.max(np.abs(phi - phi\_old)) < tol:  
 break  
 return phi

# 5 Модуль расчета электрического поля

* Компоненты электрического поля E\_x и E\_y рассчитываются как градиент потенциала:
* с использованием центральной разности (через функцию np.roll).
* Обновление поля происходит после каждого перерасчета потенциала.

Программный код:

def compute\_electric\_field(phi):  
 E\_x = -(np.roll(phi, -1, axis=0) - np.roll(phi, 1, axis=0)) / (2 \* dx)  
 E\_y = -(np.roll(phi, -1, axis=1) - np.roll(phi, 1, axis=1)) / (2 \* dy)  
 return E\_x, E\_y

# 6 Модуль уравнения непрерывности

* Ток рассчитывается по закону Ома: j = σE.
* Дивергенция тока используется для обновления плотности заряда по уравнению непрерывности:
* ∂ρ/∂t = -∇·j.
* Плотность заряда обновляется на каждом временном шаге. Программный код:

def update\_rho(rho, E\_x, E\_y, sigma):  
 j\_x = sigma \* E\_x  
 j\_y = sigma \* E\_y  
 div\_j = (np.roll(j\_x, -1, axis=0) - np.roll(j\_x, 1, axis=0)) / (2 \* dx) + \  
 (np.roll(j\_y, -1, axis=1) - np.roll(j\_y, 1, axis=1)) / (2 \* dy)  
 rho += -div\_j \* 1e-9 # dt (в секундах)  
 return rho

# 7 Модуль проверки пробоя

* Проверяется величина электрического поля в каждой точке.
* Если модуль поля превышает заданный порог (3 МВ/м), пробой считается зафиксированным.
* Координаты точки пробоя выводятся в консоль. Программный код:

def check\_breakdown(E\_x, E\_y, threshold=3e6):  
 E\_mag = np.sqrt(E\_x\*\*2 + E\_y\*\*2)  
 mask = E\_mag >= threshold  
 if np.any(mask):  
 indices = np.argwhere(mask)  
 print("Пробой зафиксирован в точках:", indices)  
 return True  
 return False  
   
# Основной цикл моделирования  
for step in range(100):  
 phi = solve\_poisson(phi, rho, epsilon)  
 E\_x, E\_y = compute\_electric\_field(phi)  
 rho = update\_rho(rho, E\_x, E\_y, sigma)  
 if check\_breakdown(E\_x, E\_y):  
 break

# 8 Модуль визуализации

* Используется библиотека Matplotlib.
* Строится тепловая карта распределения электрического потенциала φ.
* Возможность сохранения и расширения для отображения E, ρ, стримеров и других данных. Программный код:

# Визуализация  
plt.figure(figsize=(6, 5))  
plt.title("Электрический потенциал φ")  
plt.imshow(phi.T, origin='lower', extent=[0, Lx, 0, Ly], cmap='plasma')  
plt.colorbar(label='φ (В)')  
plt.xlabel('x')  
plt.ylabel('y')  
plt.show()

# 9 Результат

Этот график представляет распределение электрического потенциала в двумерной расчетной области. Изображение наглядно показывает:

* **Градиент потенциала**: потенциал возрастает снизу вверх — от (нижняя граница) до (верхняя граница).
* **Плавное изменение цвета** указывает на равномерное распределение электрического поля в области.
* **Цветовая шкала справа** обозначает значение потенциала в вольтах.

Для наглядности в учебной модели проще всего снизить порог, чтобы увидеть срабатывание функции на проверку пробоя. Сделал 1000 В/м (рис. 1).

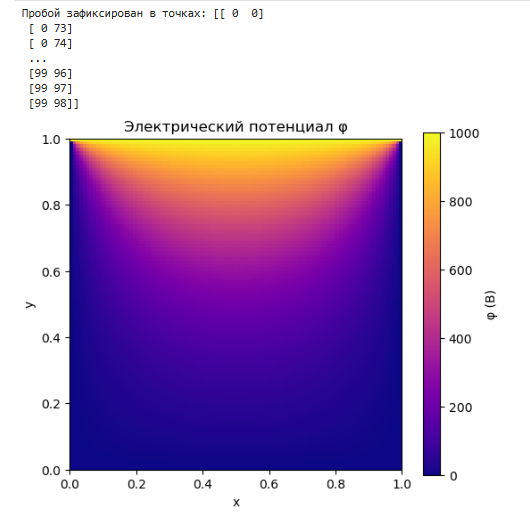


Рис. 1: Электрический пробой

# 10 Выводы

Созданный программный комплекс реализует ключевые этапы моделирования электрического пробоя. Он обеспечивает гибкость, визуализацию и масштабируемость и может быть использован как учебный, исследовательский и инженерный инструмент.

# Список литературы