**Моделирование и обработка космического эксперимента**

**Лабораторная работа № 1**

**«Простейший расчет потенциала электрического поля вокруг кубсата 1U (плоский вариант)»**

**4ПМ Вехорев Герман Алексеевич**

**Условие**

Имеется малый спутник, размером 10 х 10 см.

Электрический потенциал на его поверхности *U0*=10 Вольт (внутренне граничное условие).

Необходимо решить уравнение Лапласа

в квадрате 1 х 1 метр (внешнее граничное условие на границе этого квадрата). Значение потенциала на внешней границе определить как потенциал не кубика, а шара диаметром 10 см.

Для определения это потенциала на границе найти аналитическое выражение для потенциала шара (круга для плоскости), которое зависит от расстояния от центра шара до соответствующей точки на границе.

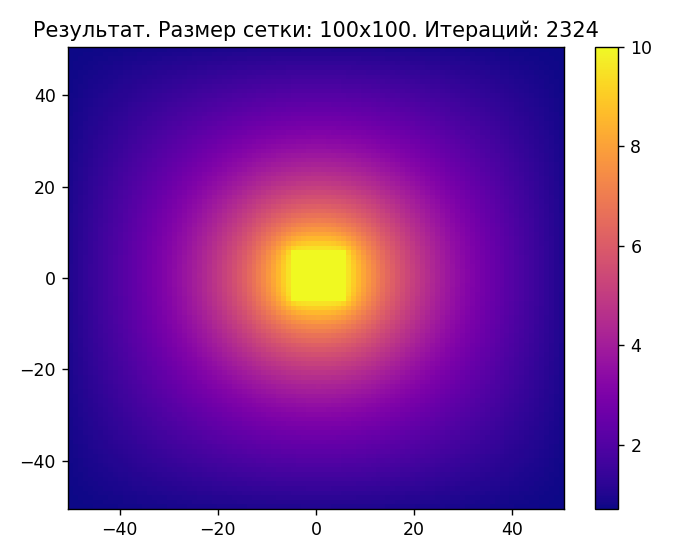
Потенциал шара на поверхности и вне её , на поверхности *R0*=5 cм

Следовательно, потенциал на внешней границе равен , или

Здесь r - расстояние от центра области до граничной точки. Пространственный шаг сетки – 1 см.

r

**Результаты**



**Код (Python)**

import matplotlib  
import matplotlib.pyplot as plt  
import numpy as np  
  
  
# Правая часть уравнения d2U/dx2 + d2U/dy2 = f  
def f(x\_: float, y\_: float) -> float:  
 return 0.  
  
  
# Внешнее граничное условие (на границе исследуемой области).  
def u\_outer(x\_: float, y\_: float) -> float:  
 r = np.sqrt((x\_ - sat\_pos[0]) \*\* 2 + (y\_ - sat\_pos[1]) \*\* 2)  
 return R\_0 \* U\_0 / r  
  
  
# Внутреннее граничное условие (на границе/поверхности спутника).  
def u\_inner(x\_: float, y\_: float) -> float:  
 return U\_0  
  
  
def step(U):  
 h = step\_size[0]  
 for i in range(1, grid\_size[0] - 1):  
 for j in range(1, grid\_size[1] - 1):  
 if (sat\_shape\_start\_x\_index <= i <= sat\_shape\_end\_x\_index  
 and sat\_shape\_start\_y\_index <= j <= sat\_shape\_end\_y\_index):  
 continue  
 U[i, j] = (U[i + 1, j] + U[i - 1, j] + U[i, j + 1] + U[i, j - 1]) / 4 + (h \*\* 2) \* f(x[i], y[j])  
 return U  
  
  
def solve() -> tuple[object, int]:  
 # Инициализируем всё нулями.  
 U = np.zeros(shape=(grid\_size[0], grid\_size[1]), dtype=float)  
  
 # Внешнее граничное условие (на границе исследуемой области).  
 for i in range(grid\_size[0]):  
 U[i, 0] = u\_outer(x[i], y[0])  
 U[i, -1] = u\_outer(x[i], y[-1])  
  
 for j in range(grid\_size[1]):  
 U[0, j] = u\_outer(x[0], y[j])  
 U[-1, j] = u\_outer(x[-1], y[j])  
  
 # Внутреннее граничное условие (на границе/поверхности спутника).  
 sat\_shape\_start\_x\_real = sat\_pos[0] - sat\_size[0] / 2.  
 sat\_shape\_end\_x\_real = sat\_pos[0] + sat\_size[0] / 2.  
 sat\_shape\_start\_y\_real = sat\_pos[1] - sat\_size[1] / 2.  
 sat\_shape\_end\_y\_real = sat\_pos[1] + sat\_size[1] / 2.  
  
 global sat\_shape\_start\_x\_index, sat\_shape\_end\_x\_index, sat\_shape\_start\_y\_index, sat\_shape\_end\_y\_index  
 sat\_shape\_start\_x\_index = None  
 sat\_shape\_end\_x\_index = None  
 sat\_shape\_start\_y\_index = None  
 sat\_shape\_end\_y\_index = None  
  
 for i, xi in enumerate(x):  
 if sat\_shape\_start\_x\_index is None and xi >= sat\_shape\_start\_x\_real:  
 sat\_shape\_start\_x\_index = i  
 if sat\_shape\_end\_x\_index is None and xi >= sat\_shape\_end\_x\_real:  
 sat\_shape\_end\_x\_index = i  
  
 for j, yj in enumerate(y):  
 if sat\_shape\_start\_y\_index is None and yj >= sat\_shape\_start\_y\_real:  
 sat\_shape\_start\_y\_index = j  
 if sat\_shape\_end\_y\_index is None and yj >= sat\_shape\_end\_y\_real:  
 sat\_shape\_end\_y\_index = j  
  
 for i in range(sat\_shape\_start\_x\_index, sat\_shape\_end\_x\_index + 1):  
 for j in range(sat\_shape\_start\_y\_index, sat\_shape\_end\_y\_index + 1):  
 U[i, j] = u\_inner(x[i], y[j])  
  
 U1 = step(U.copy())  
 k = 1  
  
 while np.max(np.abs(U - U1)) > eps:  
 U = U1  
 U1 = step(U.copy())  
 k += 1  
  
 return U1, k  
  
  
###################################################################################################  
  
  
sat\_pos = (0., 0.) # координаты центра спутника [см;см]  
sat\_size = (10., 10.) # размеры спутника [см;см]  
area\_size = (100., 100.) # размеры исследуемой области около спутника [см;см]  
  
R\_0 = 5. # радиус шара (круга), которым мы "приближаем" исследуемую квадратную область  
U\_0 = 10. # электрический потенциал на поверхности спутника  
  
precision = 4 # точность решения (число знаков после запятой)  
eps = 10.0 \*\* (-precision) # погрешность решения  
step\_size = (1., 1.) # величина шага (по X, по Y) [см]  
grid\_size: tuple[int, int] = (  
 int(area\_size[0] // step\_size[0]),  
 int(area\_size[1] // step\_size[1])  
) # размер сетки (по X, по Y) = длина стороны / величина шага  
  
x = np.linspace(sat\_pos[0] - area\_size[0] / 2., sat\_pos[0] + area\_size[1] / 2., grid\_size[0])  
y = np.linspace(sat\_pos[1] - area\_size[1] / 2., sat\_pos[1] + area\_size[1] / 2., grid\_size[1])  
  
  
def main():  
 plots\_color\_theme = "plasma"  
  
 # Для красивого вывода чисел в массивах numpy.  
 np.set\_printoptions(linewidth=100, precision=precision, suppress=True, floatmode="fixed")  
  
 # Для интерактивных графиков в matplotlib.  
 # Ещё в настройках PyCharm (Settings -> Tools -> Python Scientific) нужно отключить Show plots in tool window.  
 matplotlib.use("TkAgg")  
  
 print("...")  
  
 # Вывод ответа.  
 U, k = solve()  
 title = f"Результат. Размер сетки: {grid\_size[0]}x{grid\_size[1]}. Итераций: {k}"  
 print(title)  
 print(U)  
  
 X, Y = np.meshgrid(x, y)  
 plt.figure()  
 plt.pcolormesh(X, Y, U, cmap=plots\_color\_theme)  
 plt.colorbar()  
 plt.title(title)  
 plt.show()  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()