Potejncał elektryczny

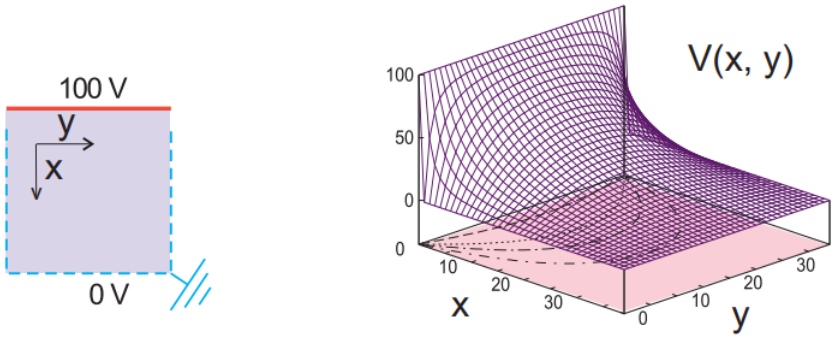
Matejko Marek, Mazur Krzysztof, Paszkot Dawid

15 stycznia 2022

Spis treści

1 Wprowadzenie

Naszym celem było znalezienie potencjału elektrycznego we wszystkich punktach wewnątrz kwadratu wolnego od ładunku.



Rysunek 1: Dwu i Trzy wymiarowa reprezentacja problemu

Dno i boki regionu składają się z drutów, które są „uziemione” (utrzymywane w 0 V). Górny przewód jest podłączony do akumulatora, który utrzymuje go na stałym poziomie 100 V.

2 Materiały i metody

Materiały dydaktyczne

W celu zapoznania się z tematem i postawionymi przed nami zadaniami, przeczytaliśmy dostarczoną literaturę 1ElektrostatykaICieplo.pdf). Posłużyliśmy się również materiałami dostępnymi w internecie: różne artykuły związane z tematem zadania na stronie *(en.wikipedia.org)* posłużyły za uzupełnienie naszej wiedzy oraz *(translate.google.com)* w celu sprawdzania tłumaczenia z wyżej podanych materiałów.

Metody

Programy pisaliśmy w języku Python w środowiskach Visual Studio Code oraz Jupyter notebook. Użyliśmy bibliotek *numpy*, *matplotlib*.

Algorytm skończonej różnicy z którego korzystaliśmy w tym programie wyglada nastepująco:

$$U_{i,j} = \frac{1}{4}\left[U_{i+1,j} + U_{i-1,j} + U_{i,j+1} + U_{i,j-1}\right] + \pi\rho(i\Delta, j\Delta)\Delta^2.$$

Istnieje wiele sposobów iteracji naszego algorytmu, w celu przekształcenia warunków brzegowych w rozwiązanie. Jego najbardziej podstawową formą jest metoda Jacobiego.

Raczej oczywiste ulepszenie metody Jacobiego wykorzystuje zaktualizowane domysły dotyczące potencjału w algorytmie, gdy tylko są one dostępne.

Technika ta, znana jako metoda Gaussa–Seidela (GS), zwykle prowadzi ona do przyspieszonej konwergencji, co z kolei prowadzi do mniejszego błędu zaokrąglania. Zużywa również mniej pamięci, ponieważ nie ma potrzeby przechowywania dwóch generacji domysłów.

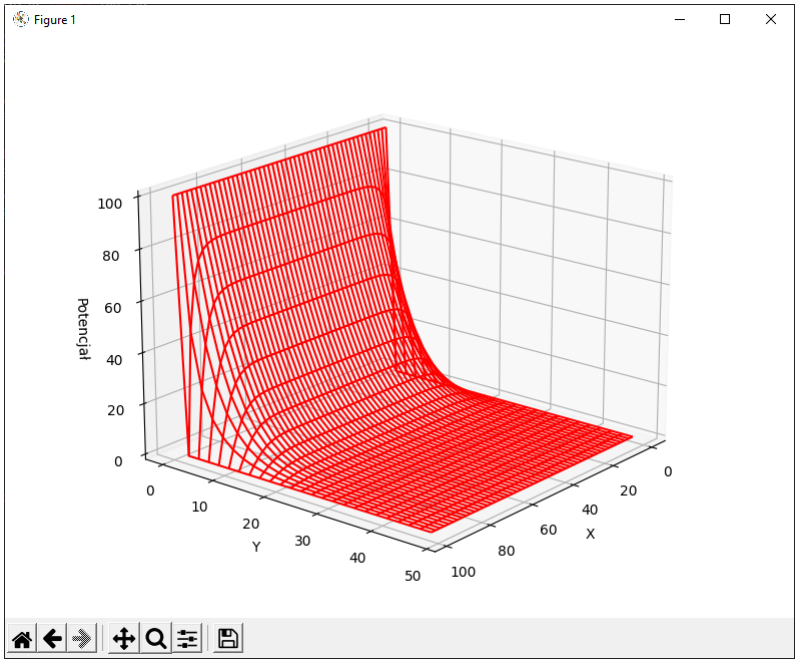
$$U_{i,j}^{(\mathrm{new})} = \frac{1}{4}\left[U_{i+1,j}^{(\mathrm{old})} + U_{i-1,j}^{(\mathrm{new})} + U_{i,j+1}^{(\mathrm{old})} + U_{i,j-1}^{(\mathrm{new})}\right]$$

3 Rezultaty

Za pomocą jezyka programowania Python pokazalismy zachowanie potecjału elektrycznego w kwadracie wolnego od ładunku

```python
# from numpy import *
import numpy as np
import matplotlib.pylab as p;
from mpl_toolkits.mplot3d import Axes3D

print("Initializing")
Nmax = 100; Niter = 70; V = np.zeros((Nmax, Nmax), float)                    # float maybe Float

print( "Working hard, wait for the figure while I count to 60")
for k in range(0, Nmax-1): V[k,0] = 100.0                                    # line at 100V

for iter in range(Niter):                                                    # iterations over algorithm
    if iter % 10 == 0: print( iter)
    for i in range(1, Nmax-2):
        for j in range(1,Nmax-2): V[i,j] = 0.25*(V[i+1,j]+V[i-1,j]+V[i,j+1]+V[i,j-1])

x = range(0, Nmax-1, 2); y = range(0, 50, 2)                                 # plot every other point
X, Y = p.meshgrid(x,y)

def functz(V):                                                               # Function returns V(x, y)
    z = V[X,Y]
    return z

Z = functz(V)
fig = p.figure()                          # Create figure
ax = Axes3D(fig)                          # plot axes
ax.plot_wireframe(X, Y, Z, color = 'r')   # red wireframe
ax.set_xlabel('X')                        # label axes
ax.set_ylabel('Y')
ax.set_zlabel('Potential')
p.show()                                  # display fig, close shell to quit
```

Rysunek 2: Kod programu w języku Python



Rysunek 3: Wykres potencjału

4 Dyskusja

Udało nam się otrzymać rezultaty zgodne z przewidywaniami teoretycznymi. Wykres wykazuje odpowiednie zachowania.