РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Проект: Электрический пробой

Этап 3

дисциплина: *Математическое Моделирование*

Выполняли:

Дугаева Светлана (НФИбд-01-18),

Ли Тимофей (НФИбд-01-18),

Васильева Юлия (НФИбд-03-18),

Кученов Ирзилей (НФИбд-03-18),

Соколова Анастасия (НФИбд-03-18),

Назарьева Алена (НФИбд-03-18)

МОСКВА

2021г.

В прошлом этапе мы вывели алгоритм для вычисления электрического потенциала итерационным методом, сегодня мы бы хотели представить реализацию нашей задачи на python.
Мы задали граничные условия:
потенциал верхнего электрода равен нулю, потенциал нижнего мы задали равным 100, левая и правая граница-линейное падение потенциала от 0 до 100. Для этого мы задали двумерный массив размером 10x10, первая строка-нули, последняя-100, первый и последний столбец-номер строки умноженное на 10, остальные ячейки- рандомные целые числа от 0 до 100. Также мы задаем двумерный массив размером 10x10, где граничные значения-единицы, а остальные-нули.

```
import numpy as np
import math
m=10
n=10
U=100
M=np.zeros((m,n))
for i in range(len(M)):
    for j in range(len(M[i])):
        M[i][j]+=np.random.randint(0, U)
for i in range (len(M)):
        for j in range (len(M[i])):
            if i == 0 :
                M[i][j]=0
            elif i == (len(M)-1) :
                M[i][j]=U
            elif j == 0 or j==(len(M[i])-1) :
                M[i][j]=(U/(len(M)-1))*i
N = np.zeros((m,n))
for x in range(0, len(N)):
    for y in range(0, len(N[x])):
        if x==0:
            N[x][y]=1
        elif y==0:
            N[x][y]=1
        elif x==(len(N)-1):
            N[x][y]=1
        elif y==(len(N[x])-1):
            N[x][y]=1
N1=N
print(M)
print(N)
```

```
[[  0.          0.          0.          0.          0.
    0.          0.          0.          0.          0.        ]
 [ 11.11111111  32.         36.         81.         53.
    1.          4.         43.         94.         11.11111111]
 [ 22.22222222  24.          0.         36.         13.
    5.         16.         92.         28.         22.22222222]
 [ 33.33333333  94.          0.         43.         98.
   76.         36.         88.         75.         33.33333333]
 [ 44.44444444  52.          4.         62.         96.
   29.         33.         40.         72.         44.44444444]
 [ 55.55555556  79.         46.         29.          5.
   11.         28.         55.         63.         55.55555556]
 [ 66.66666667  53.         20.         12.         20.
   91.         52.         70.         50.         66.66666667]
 [ 77.77777778  91.         55.         18.         67.
   69.         32.         13.         45.         77.77777778]
 [ 88.88888889   1.         49.         67.         57.
   47.          0.         99.         31.         88.88888889]
 [100.        100.        100.        100.        100.
  100.        100.        100.        100.        100.        ]]
[[1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]]
```

Далее мы задаем цикл, где вычисляем значение потенциала в заданном узле. Повторяем эту процедуру до тех пор, пока новые значения будут отличаться от старых не более, чем на погрешность вычисления потенциала, которую мы задали равной 0.05. Если условие выполняется, мы записываем 1 вместо 0 в нашу матрицу N, и когда все значения в матрице N будут равны 1, мы выходим из цикла и считаем, что рассчитали потенциала в каждом возможном узле. Первый проход цикла:

```
while (np.sum(N)!=n*m):
    for i in range (1,len(M)-1):
        for j in range (1,len(M[i])-1):
            g=M[i][j]
            M[i][j] = (1/4)*(M[i-1][j]+M[i][j-1]+M[i+1][j]+M[i][j+1])
            if (M[i][j]-g<=0.05):
                N[i][j]=1
    print (M)
    print(N)
    N=N1
```

```
[[  0.          0.          0.          0.          0.
    0.          0.          0.          0.          0.        ]
 [ 11.11111111  17.77777778  24.69444444  28.42361111  10.60590278
    4.90147569  15.97536892  50.49384223  22.40123834  11.11111111]
 [ 22.22222222  33.5         23.54861111  26.99305556  35.14973958
   33.01280382  44.24704319  52.68522135  43.07717048  22.22222222]
 [ 33.33333333  29.70833333  25.06423611  53.01432292  65.04101562
   40.76345486  51.50262451  54.79696147  50.80186632  33.33333333]
 [ 44.44444444  39.28819444  43.08810764  55.27560764  38.57915582
   30.83565267  37.5845693   54.84538269  53.27292336  44.44444444]
 [ 55.55555556  48.4609375   35.13726128  26.85321723  24.10809326
   43.48593648  47.01762644  58.71575228  54.3860578   55.55555556]
 [ 66.66666667  56.53190104  39.66729058  26.13012695  52.05955505
   54.13637288  50.78849983  43.12606303  52.29469687  66.66666667]
 [ 77.77777778  47.5774197   38.56117757  49.67282613  56.9330953
   47.51736705  27.82646672  53.73813244  53.70265177  77.77777778]
 [ 88.88888889  71.36657715  69.23193868  68.9761912   68.22732162
   53.93617217  70.19065972  63.73219804  76.58093468  88.88888889]
 [100.         100.         100.         100.         100.
  100.         100.         100.         100.         100.        ]]
```

```
   100.         100.         100.
 [[1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
  [1. 1. 1. 1. 1. 0. 0. 0. 1. 1.]
  [1. 0. 0. 1. 0. 0. 0. 1. 0. 1.]
  [1. 1. 0. 0. 1. 1. 0. 1. 1. 1.]
  [1. 1. 0. 1. 1. 0. 0. 0. 1. 1.]
  [1. 1. 1. 1. 0. 0. 0. 0. 1. 1.]
  [1. 0. 0. 0. 0. 1. 1. 1. 0. 1.]
  [1. 1. 1. 0. 1. 1. 1. 0. 0. 1.]
  [1. 0. 0. 0. 0. 0. 0. 1. 0. 1.]
  [1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]]
```

Последний проход:

```
[[  0.          0.          0.          0.          0.
    0.          0.          0.          0.          0.        ]
 [ 11.11111111  11.04857845  11.00047486  10.97081685  10.9609811
   10.96985064  10.99423827  11.02950278  11.07026241  11.11111111]
 [ 22.22222222  22.11116363  22.02583139  21.97329904  21.95595732
   21.97178199  22.01508998  22.07763463  22.14987069  22.22222222]
 [ 33.33333333  33.19181022  33.08321832  33.01648365  32.99457239
   33.01484481  33.07002547  33.14960009  33.24142441  33.33333333]
 [ 44.44444444  44.29223798  44.17560569  44.10405416  44.08068809
   44.10260434  44.16194273  44.24738958  44.34590458  44.44444444]
 [ 55.55555556  55.4117212   55.30163316  55.23419858  55.21228121
   55.23308504  55.28915307  55.36978932  55.46268778  55.55555556]
 [ 66.66666667  66.54729751  66.45601574  66.40016521  66.38207845
   66.39940209  66.44592927  66.51278042  66.58975335  66.66666667]
 [ 77.77777778  77.69429719  77.6304947   77.59148507  77.57888048
   77.59102093  77.62355787  77.6702801   77.72405725  77.77777778]
 [ 88.88888889  88.84710406  88.8151755   88.79565916  88.78935841
   88.7954398   88.81172532  88.83510579  88.86201298  88.88888889]
 [100.         100.         100.         100.         100.
  100.         100.         100.         100.         100.        ]]
```

```
[[1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]]
```

всего 32

В итоге, мы выводим весь двумерный массив с округленными значениями потенциалами в каждом возможном узле и считаем задачу выполненной.

```
M=np.round(M,5)
for i in range(1,len(M)-1):
    print(*M[i][1:-1])
```

```
11.04858 11.00047 10.97082 10.96098 10.96985 10.99424 11.0295 11.07026
22.11116 22.02583 21.9733 21.95596 21.97178 22.01509 22.07763 22.14987
33.19181 33.08322 33.01648 32.99457 33.01484 33.07003 33.1496 33.24142
44.29224 44.17561 44.10405 44.08069 44.1026 44.16194 44.24739 44.3459
55.41172 55.30163 55.2342 55.21228 55.23309 55.28915 55.36979 55.46269
66.5473 66.45602 66.40017 66.38208 66.3994 66.44593 66.51278 66.58975
77.6943 77.63049 77.59149 77.57888 77.59102 77.62356 77.67028 77.72406
88.8471 88.81518 88.79566 88.78936 88.79544 88.81173 88.83511 88.86201
```

Список литературы

1) Д. А. Медведев, А. Л. Куперштох, Э. Р. Прууэл, Н. П. Сатонкина, Д. И. Карпов - “МОДЕЛИРОВАНИЕ ФИЗИЧЕСКИХ ПРОЦЕССОВ И ЯВЛЕНИЙ НА ПК”, Учебное пособие 2010