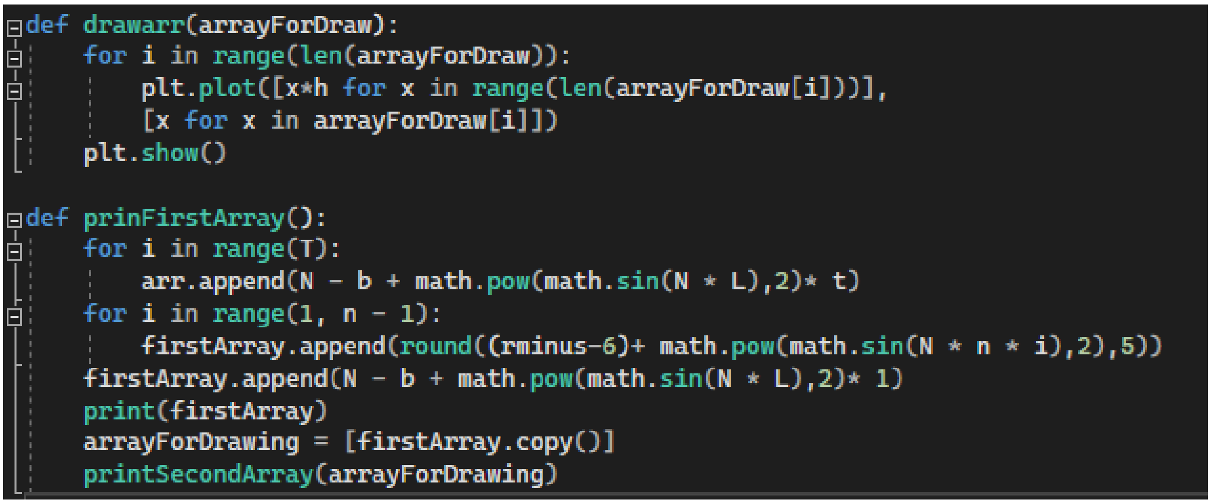
Посилання на репозиторій з кодом:

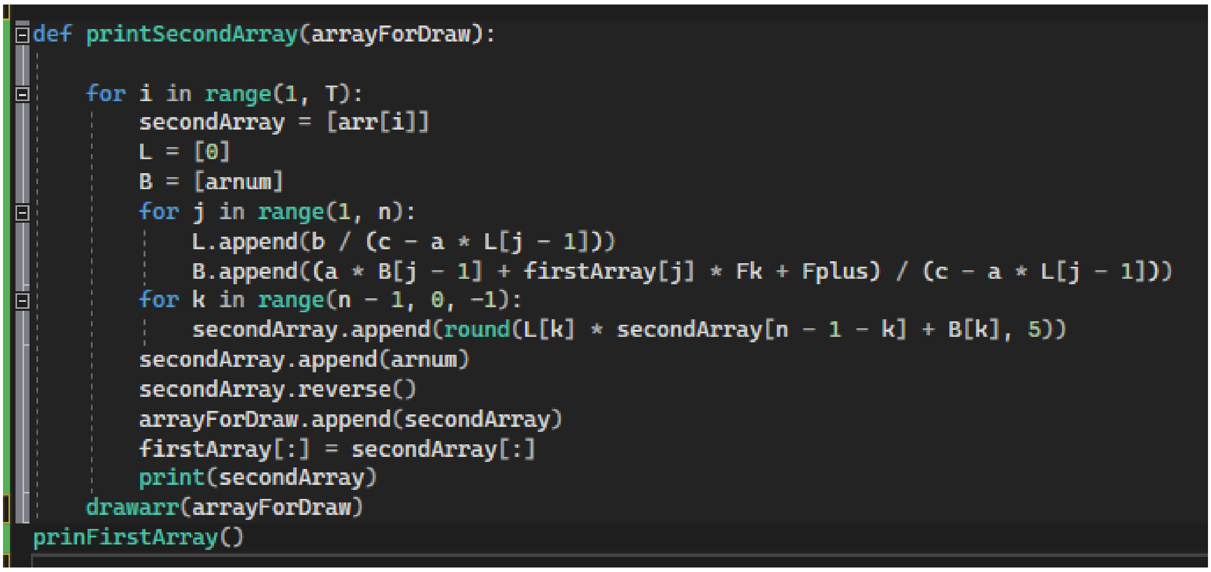
Лабораторна робота №1

N=9

Код:

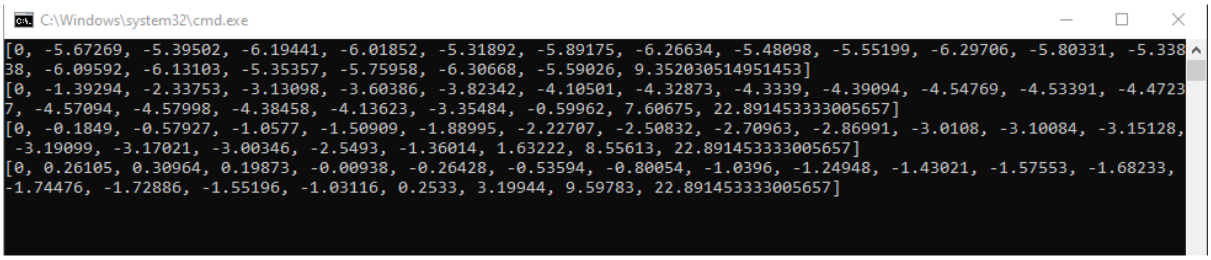


```python
def drawarr(arrayForDraw):
    for i in range(len(arrayForDraw)):
        plt.plot([x*h for x in range(len(arrayForDraw[i]))],
        [x for x in arrayForDraw[i]])
    plt.show()

def prinFirstArray():
    for i in range(T):
        arr.append(N - b + math.pow(math.sin(N * L),2)* t)
    for i in range(1, n - 1):
        firstArray.append(round((rminus-6)+ math.pow(math.sin(N * n * i),2),5))
    firstArray.append(N - b + math.pow(math.sin(N * L),2)* 1)
    print(firstArray)
    arrayForDrawing = [firstArray.copy()]
    printSecondArray(arrayForDrawing)
```



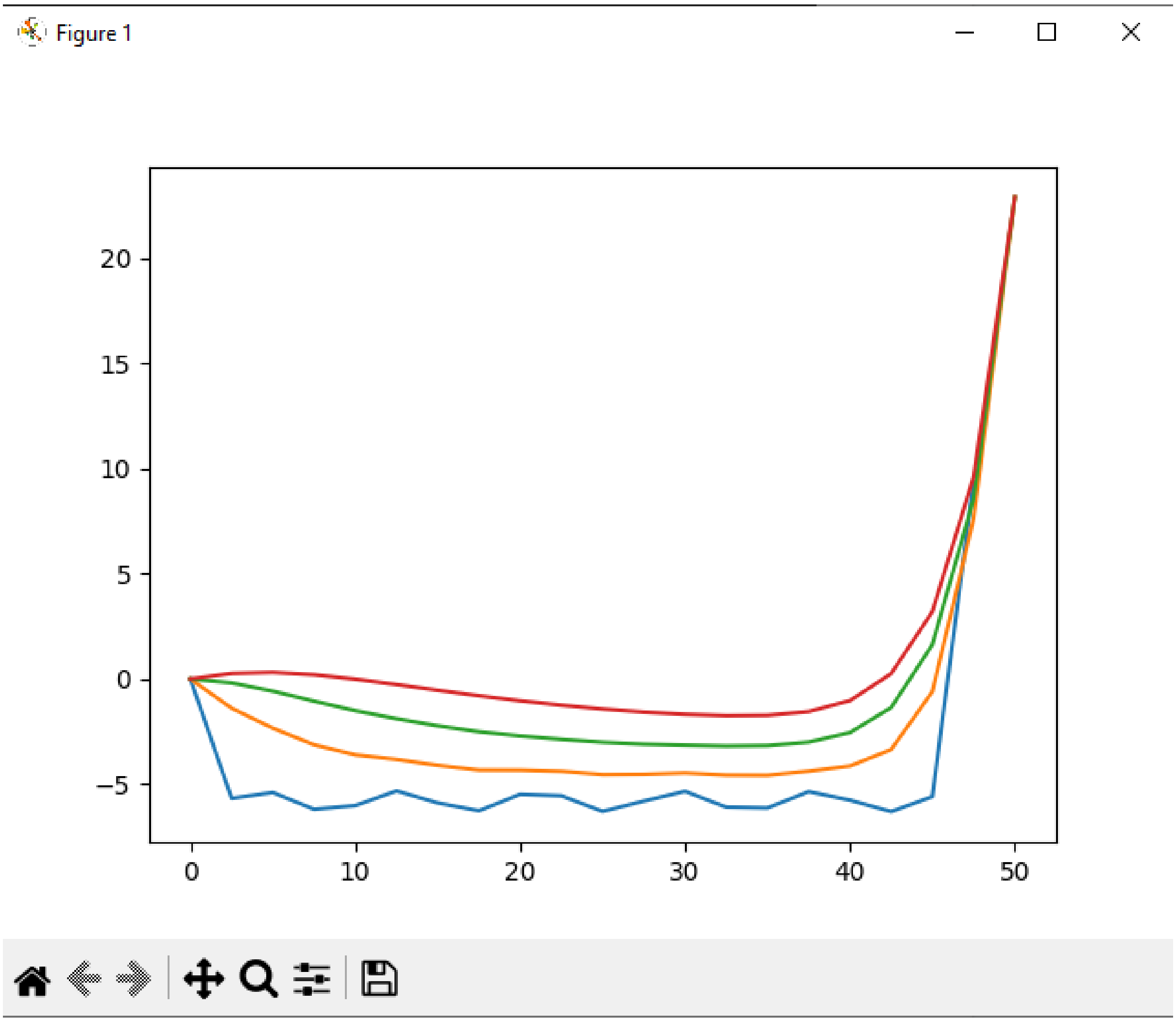
```python
def printSecondArray(arrayForDraw):

    for i in range(1, T):
        secondArray = [arr[i]]
        L = [0]
        B = [arnum]
        for j in range(1, n):
            L.append(b / (c - a * L[j - 1]))
            B.append((a * B[j - 1] + firstArray[j] * Fk + Fplus) / (c - a * L[j - 1]))
        for k in range(n - 1, 0, -1):
            secondArray.append(round(L[k] * secondArray[n - 1 - k] + B[k], 5))
        secondArray.append(arnum)
        secondArray.reverse()
        arrayForDraw.append(secondArray)
        firstArray[:] = secondArray[:]
        print(secondArray)
    drawarr(arrayForDraw)
prinFirstArray()
```

Результат:



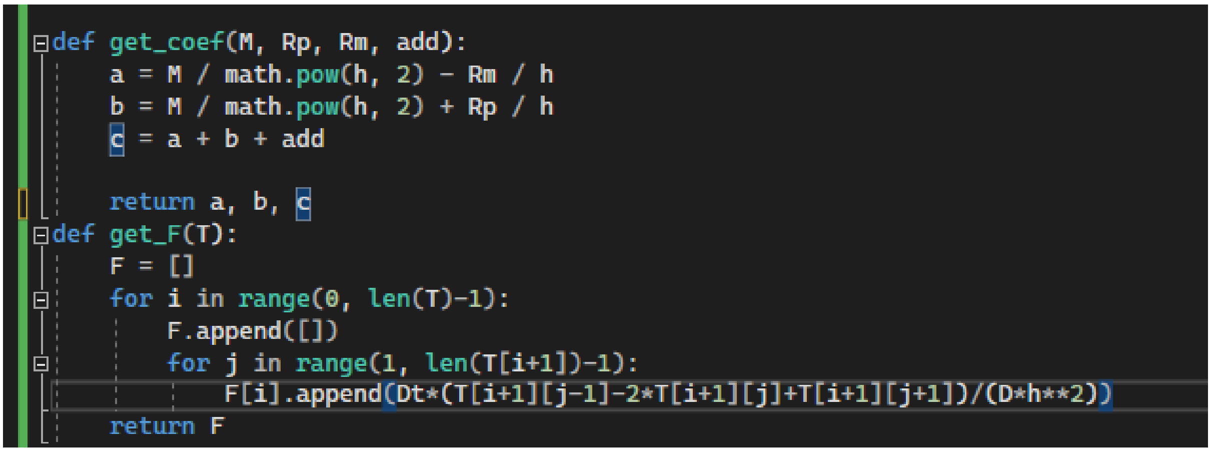
```
[0, -5.67269, -5.39502, -6.19441, -6.01852, -5.31892, -5.89175, -6.26634, -5.48098, -5.55199, -6.29706, -5.80331, -5.338
38, -6.09592, -6.13103, -5.35357, -5.75958, -6.30668, -5.59026, 9.352030514951453]
[0, -1.39294, -2.33753, -3.13098, -3.60386, -3.82342, -4.10501, -4.32873, -4.3339, -4.39094, -4.54769, -4.53391, -4.4723
7, -4.57094, -4.57998, -4.38458, -4.13623, -3.35484, -0.59962, 7.60675, 22.891453333005657]
[0, -0.1849, -0.57927, -1.0577, -1.50909, -1.88995, -2.22707, -2.50832, -2.70963, -2.86991, -3.0108, -3.10084, -3.15128,
 -3.19099, -3.17021, -3.00346, -2.5493, -1.36014, 1.63222, 8.55613, 22.891453333005657]
[0, 0.26105, 0.30964, 0.19873, -0.00938, -0.26428, -0.53594, -0.80054, -1.0396, -1.24948, -1.43021, -1.57553, -1.68233,
-1.74476, -1.72886, -1.55196, -1.03116, 0.2533, 3.19944, 9.59783, 22.891453333005657]
```

Графік концентрації забруднення в часових шарах

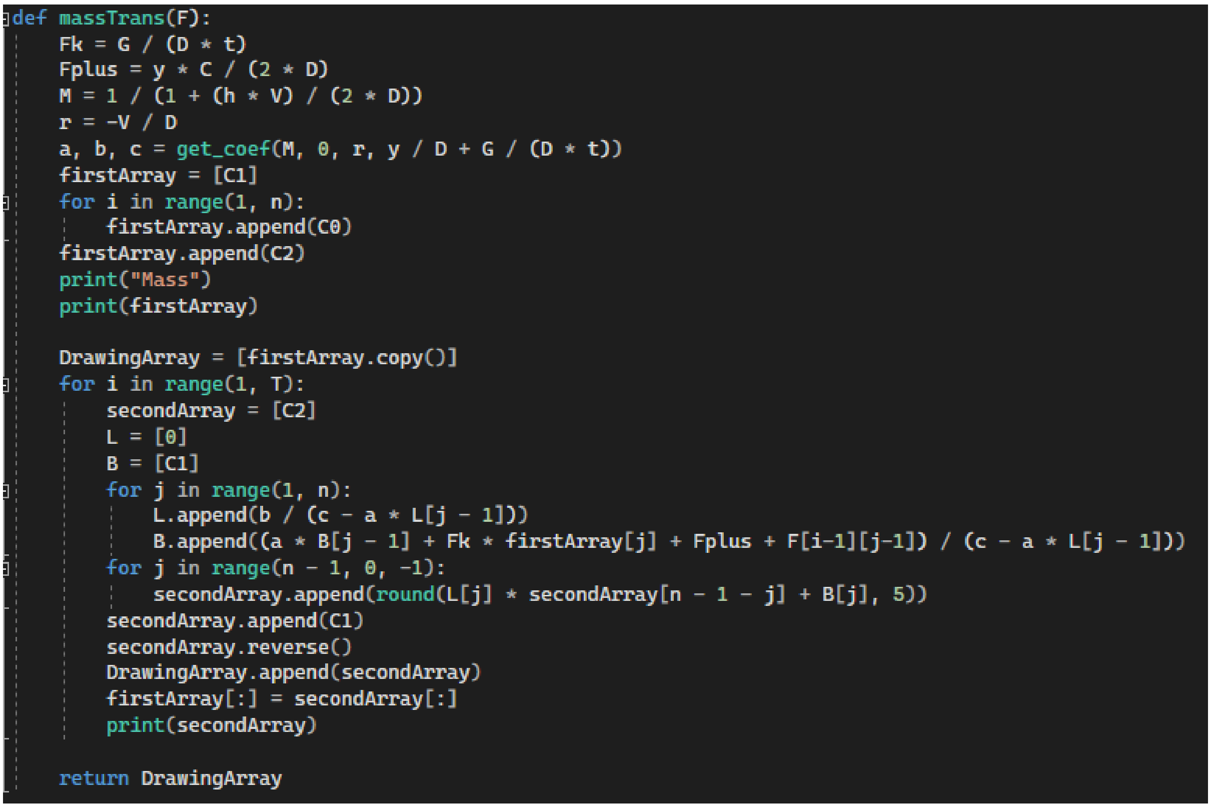


Лабораторна робота №2
N=9

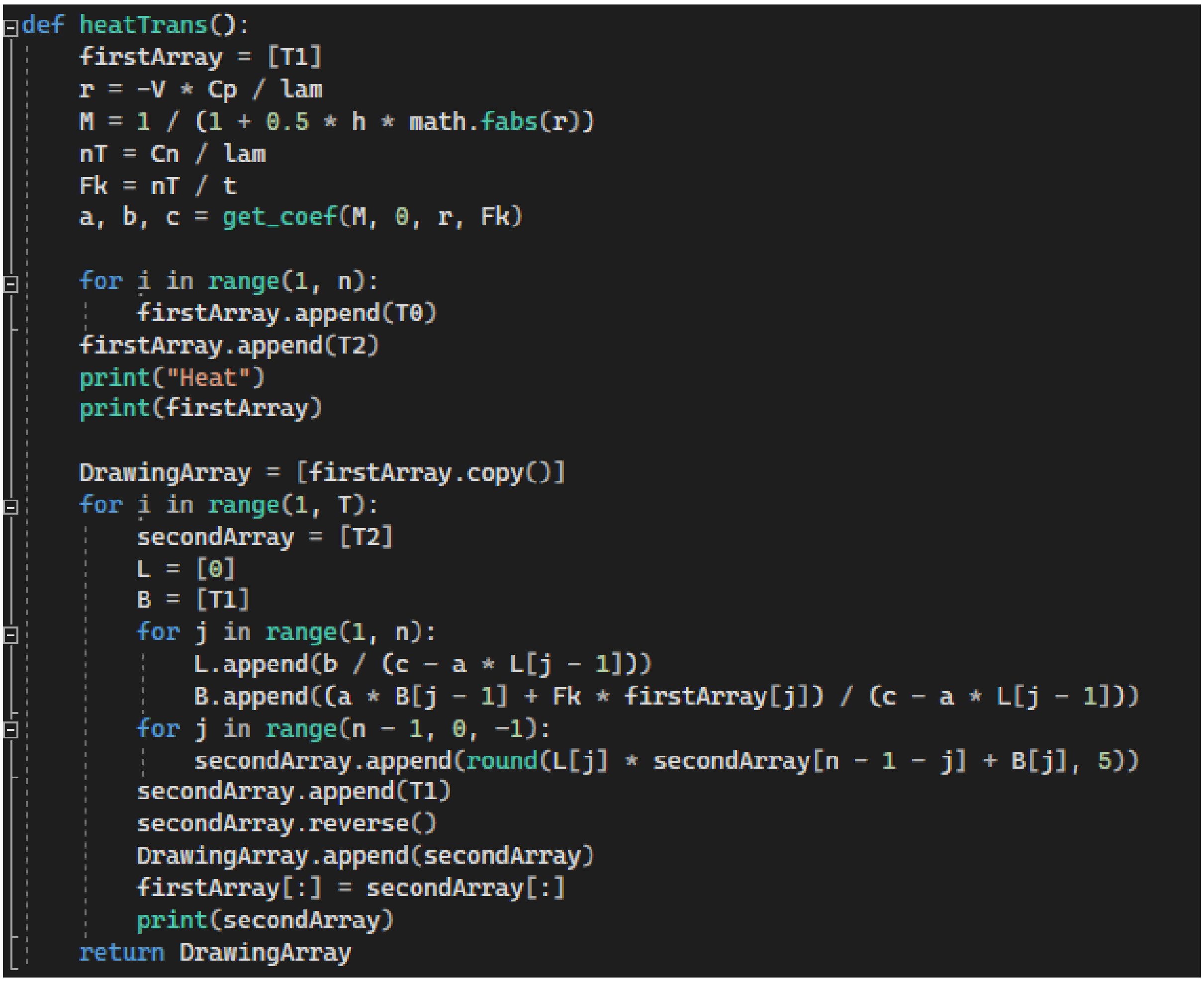
Код:



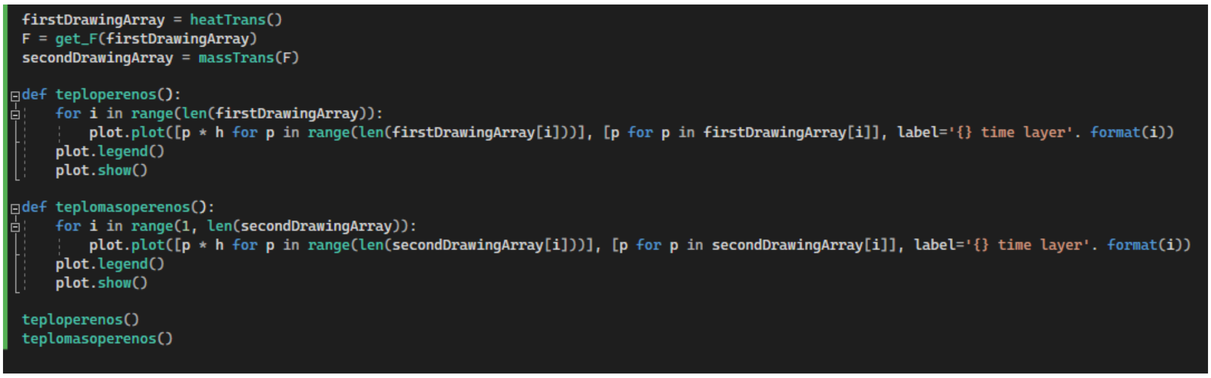
```python
def get_coef(M, Rp, Rm, add):
    a = M / math.pow(h, 2) - Rm / h
    b = M / math.pow(h, 2) + Rp / h
    c = a + b + add

    return a, b, c
def get_F(T):
    F = []
    for i in range(0, len(T)-1):
        F.append([])
        for j in range(1, len(T[i+1])-1):
            F[i].append(Dt*(T[i+1][j-1]-2*T[i+1][j]+T[i+1][j+1])/(D*h**2))
    return F
```



```python
def massTrans(F):
    Fk = G / (D * t)
    Fplus = y * C / (2 * D)
    M = 1 / (1 + (h * V) / (2 * D))
    r = -V / D
    a, b, c = get_coef(M, 0, r, y / D + G / (D * t))
    firstArray = [C1]
    for i in range(1, n):
        firstArray.append(C0)
    firstArray.append(C2)
    print("Mass")
    print(firstArray)

    DrawingArray = [firstArray.copy()]
    for i in range(1, T):
        secondArray = [C2]
        L = [0]
        B = [C1]
        for j in range(1, n):
            L.append(b / (c - a * L[j - 1]))
            B.append((a * B[j - 1] + Fk * firstArray[j] + Fplus + F[i-1][j-1]) / (c - a * L[j - 1]))
        for j in range(n - 1, 0, -1):
            secondArray.append(round(L[j] * secondArray[n - 1 - j] + B[j], 5))
        secondArray.append(C1)
        secondArray.reverse()
        DrawingArray.append(secondArray)
        firstArray[:] = secondArray[:]
        print(secondArray)

    return DrawingArray
```

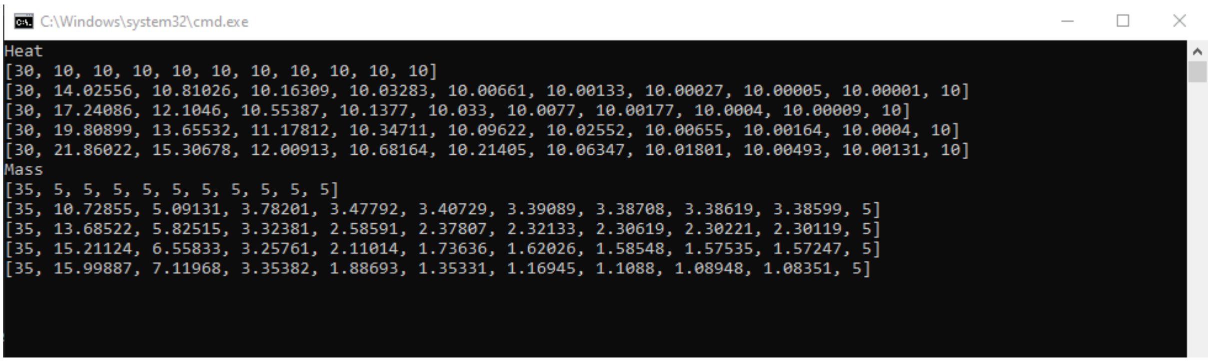


```python
def heatTrans():
    firstArray = [T1]
    r = -V * Cp / lam
    M = 1 / (1 + 0.5 * h * math.fabs(r))
    nT = Cn / lam
    Fk = nT / t
    a, b, c = get_coef(M, 0, r, Fk)

    for i in range(1, n):
        firstArray.append(T0)
    firstArray.append(T2)
    print("Heat")
    print(firstArray)

    DrawingArray = [firstArray.copy()]
    for i in range(1, T):
        secondArray = [T2]
        L = [0]
        B = [T1]
        for j in range(1, n):
            L.append(b / (c - a * L[j - 1]))
            B.append((a * B[j - 1] + Fk * firstArray[j]) / (c - a * L[j - 1]))
        for j in range(n - 1, 0, -1):
            secondArray.append(round(L[j] * secondArray[n - 1 - j] + B[j], 5))
        secondArray.append(T1)
        secondArray.reverse()
        DrawingArray.append(secondArray)
        firstArray[:] = secondArray[:]
        print(secondArray)
    return DrawingArray
```



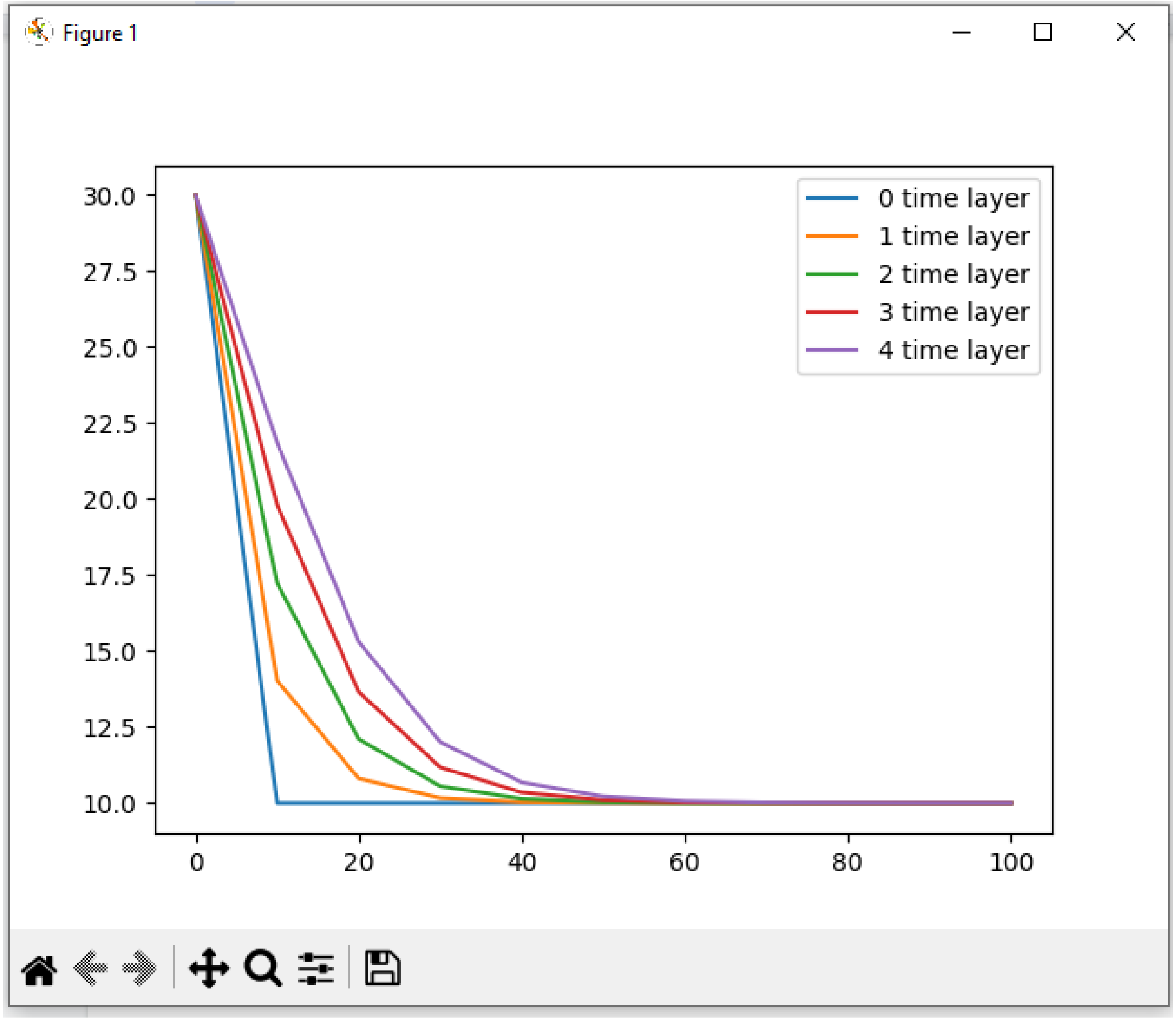
```python
firstDrawingArray = heatTrans()
F = get_F(firstDrawingArray)
secondDrawingArray = massTrans(F)

def teploperenos():
    for i in range(len(firstDrawingArray)):
        plot.plot([p * h for p in range(len(firstDrawingArray[i]))], [p for p in firstDrawingArray[i]], label='{} time layer'. format(i))
    plot.legend()
    plot.show()

def teplomasoperenos():
    for i in range(1, len(secondDrawingArray)):
        plot.plot([p * h for p in range(len(secondDrawingArray[i]))], [p for p in secondDrawingArray[i]], label='{} time layer'. format(i))
    plot.legend()
    plot.show()

teploperenos()
teplomasoperenos()
```

Результат:

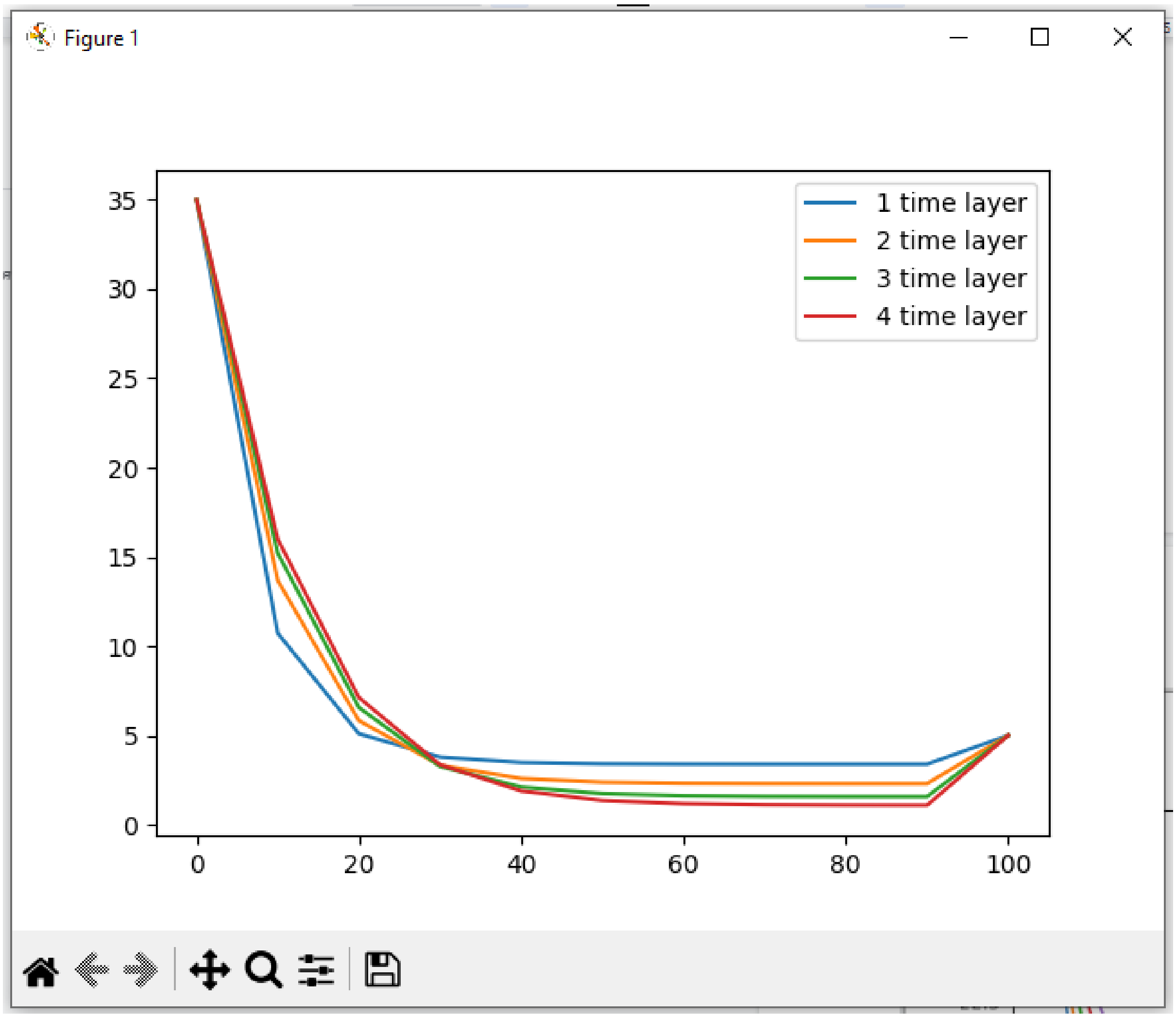


```
Heat
[30, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10]
[30, 14.02556, 10.81026, 10.16309, 10.03283, 10.00661, 10.00133, 10.00027, 10.00005, 10.00001, 10]
[30, 17.24086, 12.1046, 10.55387, 10.1377, 10.033, 10.0077, 10.00177, 10.0004, 10.00009, 10]
[30, 19.80899, 13.65532, 11.17812, 10.34711, 10.09622, 10.02552, 10.00655, 10.00164, 10.0004, 10]
[30, 21.86022, 15.30678, 12.00913, 10.68164, 10.21405, 10.06347, 10.01801, 10.00493, 10.00131, 10]
Mass
[35, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5]
[35, 10.72855, 5.09131, 3.78201, 3.47792, 3.40729, 3.39089, 3.38708, 3.38619, 3.38599, 5]
[35, 13.68522, 5.82515, 3.32381, 2.58591, 2.37807, 2.32133, 2.30619, 2.30221, 2.30119, 5]
[35, 15.21124, 6.55833, 3.25761, 2.11014, 1.73636, 1.62026, 1.58548, 1.57535, 1.57247, 5]
[35, 15.99887, 7.11968, 3.35382, 1.88693, 1.35331, 1.16945, 1.1088, 1.08948, 1.08351, 5]
```

Теплоперенос

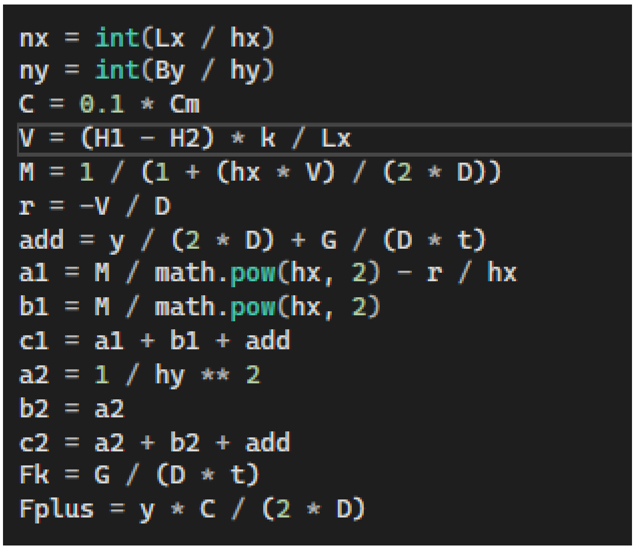


Тепломасоперенос

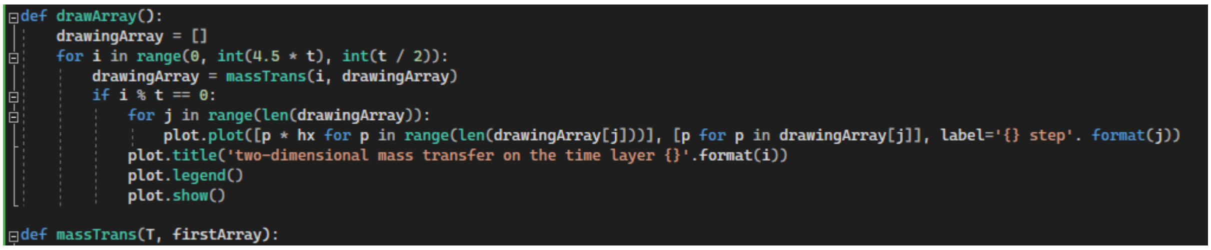


Лабораторна робота №3
N=9

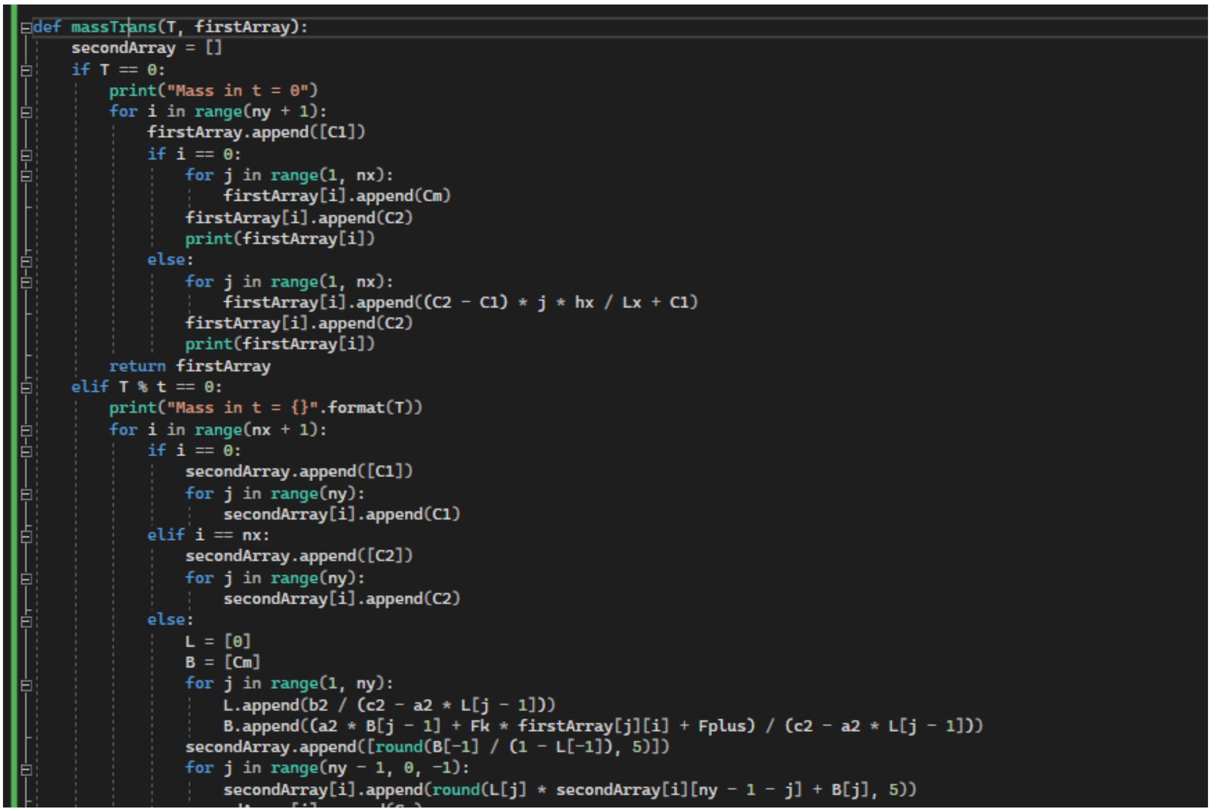
Код:



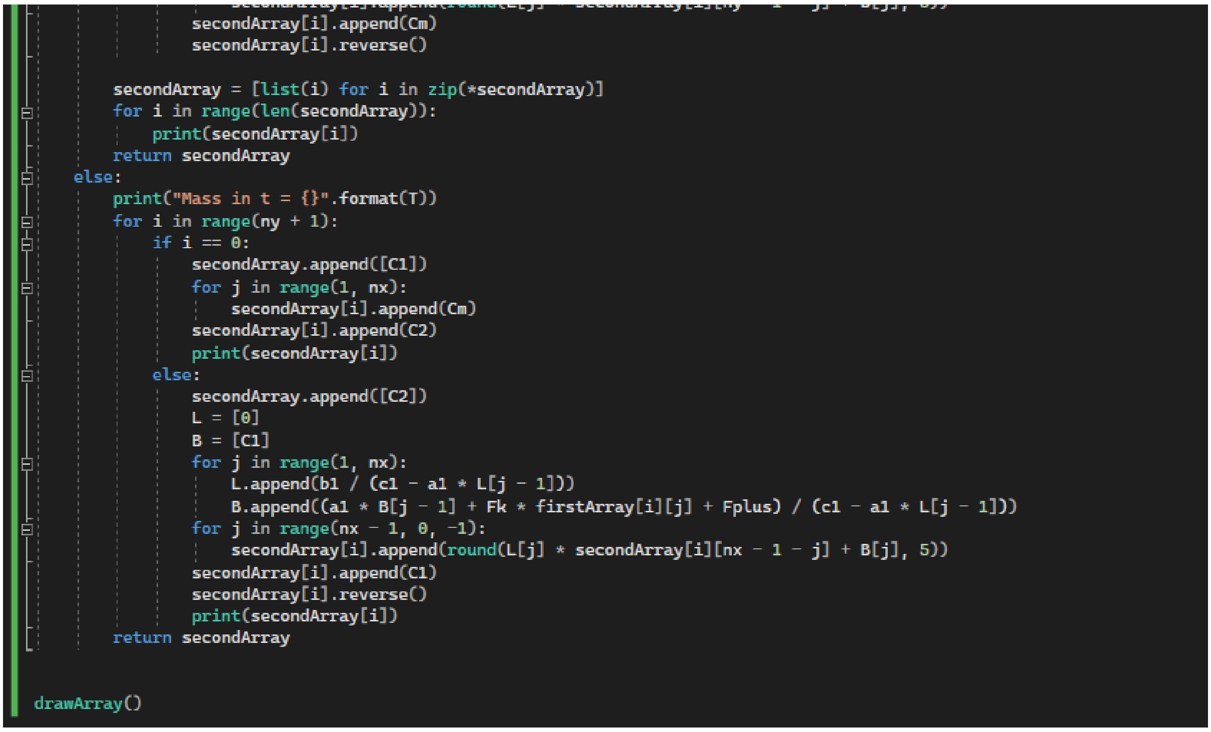
```
nx = int(Lx / hx)
ny = int(By / hy)
C = 0.1 * Cm
V = (H1 - H2) * k / Lx
M = 1 / (1 + (hx * V) / (2 * D))
r = -V / D
add = y / (2 * D) + G / (D * t)
a1 = M / math.pow(hx, 2) - r / hx
b1 = M / math.pow(hx, 2)
c1 = a1 + b1 + add
a2 = 1 / hy ** 2
b2 = a2
c2 = a2 + b2 + add
Fk = G / (D * t)
Fplus = y * C / (2 * D)
```



```
def drawArray():
    drawingArray = []
    for i in range(0, int(4.5 * t), int(t / 2)):
        drawingArray = massTrans(i, drawingArray)
        if i % t == 0:
            for j in range(len(drawingArray)):
                plot.plot([p * hx for p in range(len(drawingArray[j]))], [p for p in drawingArray[j]], label='{} step'. format(j))
            plot.title('two-dimensional mass transfer on the time layer {}'.format(i))
            plot.legend()
            plot.show()

def massTrans(T, firstArray):
```

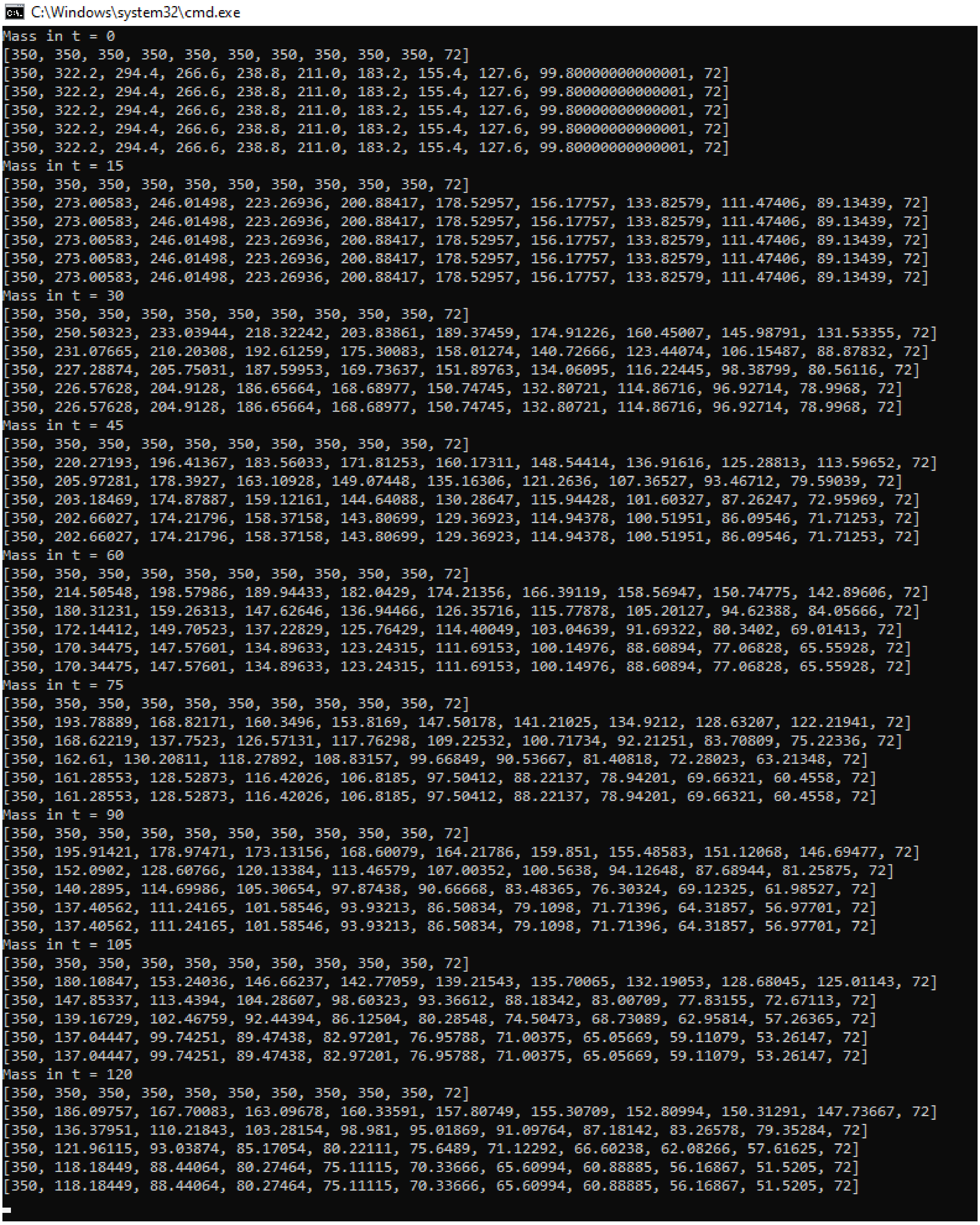


```
def massTrans(T, firstArray):
    secondArray = []
    if T == 0:
        print("Mass in t = 0")
        for i in range(ny + 1):
            firstArray.append([C1])
            if i == 0:
                for j in range(1, nx):
                    firstArray[i].append(Cm)
                firstArray[i].append(C2)
                print(firstArray[i])
            else:
                for j in range(1, nx):
                    firstArray[i].append((C2 - C1) * j * hx / Lx + C1)
                firstArray[i].append(C2)
                print(firstArray[i])
        return firstArray
    elif T % t == 0:
        print("Mass in t = {}".format(T))
        for i in range(nx + 1):
            if i == 0:
                secondArray.append([C1])
                for j in range(ny):
                    secondArray[i].append(C1)
            elif i == nx:
                secondArray.append([C2])
                for j in range(ny):
                    secondArray[i].append(C2)
            else:
                L = [0]
                B = [Cm]
                for j in range(1, ny):
                    L.append(b2 / (c2 - a2 * L[j - 1]))
                    B.append((a2 * B[j - 1] + Fk * firstArray[j][i] + Fplus) / (c2 - a2 * L[j - 1]))
                secondArray.append([round(B[-1] / (1 - L[-1]), 5)])
                for j in range(ny - 1, 0, -1):
                    secondArray[i].append(round(L[j] * secondArray[i][ny - 1 - j] + B[j], 5))
```

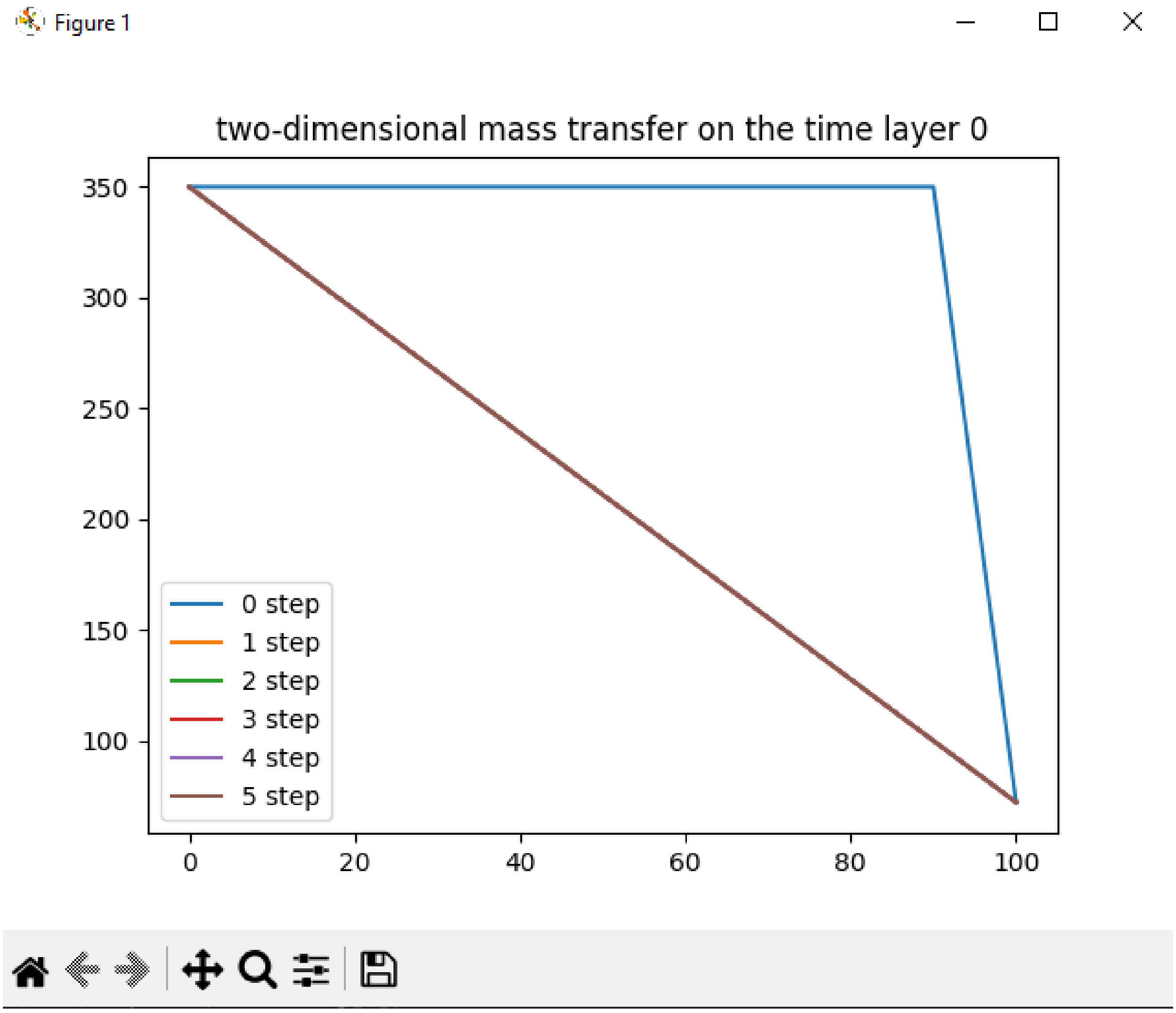


```
                    secondArray[i].append(round(L[j] * secondArray[i][ny - 1 - j] + B[j], 5))
                secondArray[i].append(Cm)
                secondArray[i].reverse()

        secondArray = [list(i) for i in zip(*secondArray)]
        for i in range(len(secondArray)):
            print(secondArray[i])
        return secondArray
    else:
        print("Mass in t = {}".format(T))
        for i in range(ny + 1):
            if i == 0:
                secondArray.append([C1])
                for j in range(1, nx):
                    secondArray[i].append(Cm)
                secondArray[i].append(C2)
                print(secondArray[i])
            else:
                secondArray.append([C2])
                L = [0]
                B = [C1]
                for j in range(1, nx):
                    L.append(b1 / (c1 - a1 * L[j - 1]))
                    B.append((a1 * B[j - 1] + Fk * firstArray[i][j] + Fplus) / (c1 - a1 * L[j - 1]))
                for j in range(nx - 1, 0, -1):
                    secondArray[i].append(round(L[j] * secondArray[i][nx - 1 - j] + B[j], 5))
                secondArray[i].append(C1)
                secondArray[i].reverse()
                print(secondArray[i])
        return secondArray


drawArray()
```

Результат:



```
Mass in t = 0
[350, 350, 350, 350, 350, 350, 350, 350, 350, 350, 72]
[350, 322.2, 294.4, 266.6, 238.8, 211.0, 183.2, 155.4, 127.6, 99.80000000000001, 72]
[350, 322.2, 294.4, 266.6, 238.8, 211.0, 183.2, 155.4, 127.6, 99.80000000000001, 72]
[350, 322.2, 294.4, 266.6, 238.8, 211.0, 183.2, 155.4, 127.6, 99.80000000000001, 72]
[350, 322.2, 294.4, 266.6, 238.8, 211.0, 183.2, 155.4, 127.6, 99.80000000000001, 72]
[350, 322.2, 294.4, 266.6, 238.8, 211.0, 183.2, 155.4, 127.6, 99.80000000000001, 72]
Mass in t = 15
[350, 350, 350, 350, 350, 350, 350, 350, 350, 350, 72]
[350, 273.00583, 246.01498, 223.26936, 200.88417, 178.52957, 156.17757, 133.82579, 111.47406, 89.13439, 72]
[350, 273.00583, 246.01498, 223.26936, 200.88417, 178.52957, 156.17757, 133.82579, 111.47406, 89.13439, 72]
[350, 273.00583, 246.01498, 223.26936, 200.88417, 178.52957, 156.17757, 133.82579, 111.47406, 89.13439, 72]
[350, 273.00583, 246.01498, 223.26936, 200.88417, 178.52957, 156.17757, 133.82579, 111.47406, 89.13439, 72]
[350, 273.00583, 246.01498, 223.26936, 200.88417, 178.52957, 156.17757, 133.82579, 111.47406, 89.13439, 72]
Mass in t = 30
[350, 350, 350, 350, 350, 350, 350, 350, 350, 350, 72]
[350, 250.50323, 233.03944, 218.32242, 203.83861, 189.37459, 174.91226, 160.45007, 145.98791, 131.53355, 72]
[350, 231.07665, 210.20308, 192.61259, 175.30083, 158.01274, 140.72666, 123.44074, 106.15487, 88.87832, 72]
[350, 227.28874, 205.75031, 187.59953, 169.73637, 151.89763, 134.06095, 116.22445, 98.38799, 80.56116, 72]
[350, 226.57628, 204.9128, 186.65664, 168.68977, 150.74745, 132.80721, 114.86716, 96.92714, 78.9968, 72]
[350, 226.57628, 204.9128, 186.65664, 168.68977, 150.74745, 132.80721, 114.86716, 96.92714, 78.9968, 72]
Mass in t = 45
[350, 350, 350, 350, 350, 350, 350, 350, 350, 350, 72]
[350, 220.27193, 196.41367, 183.56033, 171.81253, 160.17311, 148.54414, 136.91616, 125.28813, 113.59652, 72]
[350, 205.97281, 178.3927, 163.10928, 149.07448, 135.16306, 121.2636, 107.36527, 93.46712, 79.59039, 72]
[350, 203.18469, 174.87887, 159.12161, 144.64088, 130.28647, 115.94428, 101.60327, 87.26247, 72.95969, 72]
[350, 202.66027, 174.21796, 158.37158, 143.80699, 129.36923, 114.94378, 100.51951, 86.09546, 71.71253, 72]
[350, 202.66027, 174.21796, 158.37158, 143.80699, 129.36923, 114.94378, 100.51951, 86.09546, 71.71253, 72]
Mass in t = 60
[350, 350, 350, 350, 350, 350, 350, 350, 350, 350, 72]
[350, 214.50548, 198.57986, 189.94433, 182.0429, 174.21356, 166.39119, 158.56947, 150.74775, 142.89606, 72]
[350, 180.31231, 159.26313, 147.62646, 136.94466, 126.35716, 115.77878, 105.20127, 94.62388, 84.05666, 72]
[350, 172.14412, 149.70523, 137.22829, 125.76429, 114.40049, 103.04639, 91.69322, 80.3402, 69.01413, 72]
[350, 170.34475, 147.57601, 134.89633, 123.24315, 111.69153, 100.14976, 88.60894, 77.06828, 65.55928, 72]
[350, 170.34475, 147.57601, 134.89633, 123.24315, 111.69153, 100.14976, 88.60894, 77.06828, 65.55928, 72]
Mass in t = 75
[350, 350, 350, 350, 350, 350, 350, 350, 350, 350, 72]
[350, 193.78889, 168.82171, 160.3496, 153.8169, 147.50178, 141.21025, 134.9212, 128.63207, 122.21941, 72]
[350, 168.62219, 137.7523, 126.57131, 117.76298, 109.22532, 100.71734, 92.21251, 83.70809, 75.22336, 72]
[350, 162.61, 130.20811, 118.27892, 108.83157, 99.66849, 90.53667, 81.40818, 72.28023, 63.21348, 72]
[350, 161.28553, 128.52873, 116.42026, 106.8185, 97.50412, 88.22137, 78.94201, 69.66321, 60.4558, 72]
[350, 161.28553, 128.52873, 116.42026, 106.8185, 97.50412, 88.22137, 78.94201, 69.66321, 60.4558, 72]
Mass in t = 90
[350, 350, 350, 350, 350, 350, 350, 350, 350, 350, 72]
[350, 195.91421, 178.97471, 173.13156, 168.60079, 164.21786, 159.851, 155.48583, 151.12068, 146.69477, 72]
[350, 152.0902, 128.60766, 120.13384, 113.46579, 107.00352, 100.5638, 94.12648, 87.68944, 81.25875, 72]
[350, 140.2895, 114.69986, 105.30654, 97.87438, 90.66668, 83.48365, 76.30324, 69.12325, 61.98527, 72]
[350, 137.40562, 111.24165, 101.58546, 93.93213, 86.50834, 79.1098, 71.71396, 64.31857, 56.97701, 72]
[350, 137.40562, 111.24165, 101.58546, 93.93213, 86.50834, 79.1098, 71.71396, 64.31857, 56.97701, 72]
Mass in t = 105
[350, 350, 350, 350, 350, 350, 350, 350, 350, 350, 72]
[350, 180.10847, 153.24036, 146.66237, 142.77059, 139.21543, 135.70065, 132.19053, 128.68045, 125.01143, 72]
[350, 147.85337, 113.4394, 104.28607, 98.60323, 93.36612, 88.18342, 83.00709, 77.83155, 72.67113, 72]
[350, 139.16729, 102.46759, 92.44394, 86.12504, 80.28548, 74.50473, 68.73089, 62.95814, 57.26365, 72]
[350, 137.04447, 99.74251, 89.47438, 82.97201, 76.95788, 71.00375, 65.05669, 59.11079, 53.26147, 72]
[350, 137.04447, 99.74251, 89.47438, 82.97201, 76.95788, 71.00375, 65.05669, 59.11079, 53.26147, 72]
Mass in t = 120
[350, 350, 350, 350, 350, 350, 350, 350, 350, 350, 72]
[350, 186.09757, 167.70083, 163.09678, 160.33591, 157.80749, 155.30709, 152.80994, 150.31291, 147.73667, 72]
[350, 136.37951, 110.21843, 103.28154, 98.981, 95.01869, 91.09764, 87.18142, 83.26578, 79.35284, 72]
[350, 121.96115, 93.03874, 85.17054, 80.22111, 75.6489, 71.12292, 66.60238, 62.08266, 57.61625, 72]
[350, 118.18449, 88.44064, 80.27464, 75.11115, 70.33666, 65.60994, 60.88885, 56.16867, 51.5205, 72]
[350, 118.18449, 88.44064, 80.27464, 75.11115, 70.33666, 65.60994, 60.88885, 56.16867, 51.5205, 72]
```

Figure 1
two-dimensional mass transfer on the time layer 0
350
300
250
200
150
100
0
20
40
60
80
100
0 step
1 step
2 step
3 step
4 step
5 step

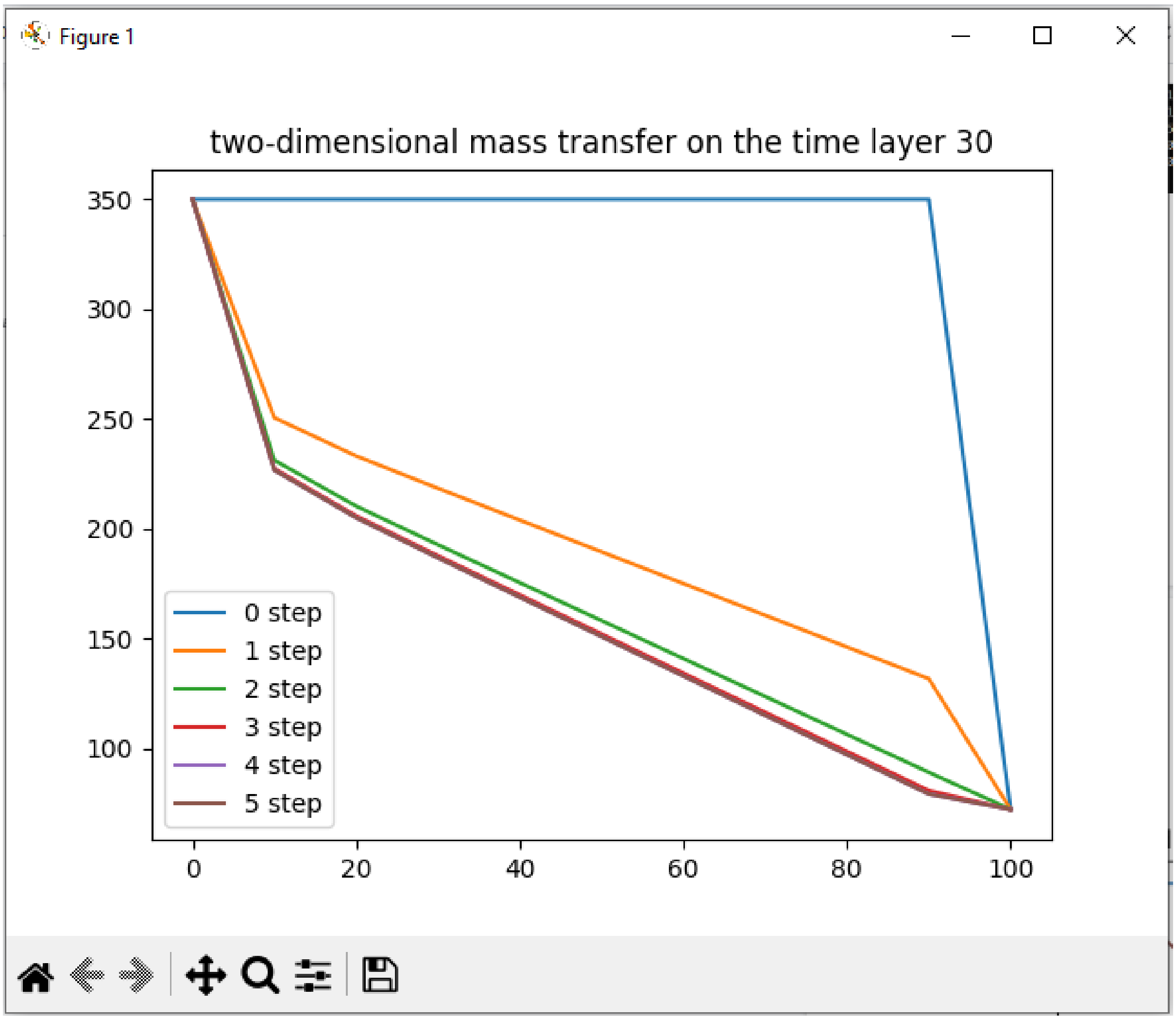


Figure 1
two-dimensional mass transfer on the time layer 30
350
300
250
200
150
100
0
20
40
60
80
100
0 step
1 step
2 step
3 step
4 step
5 step

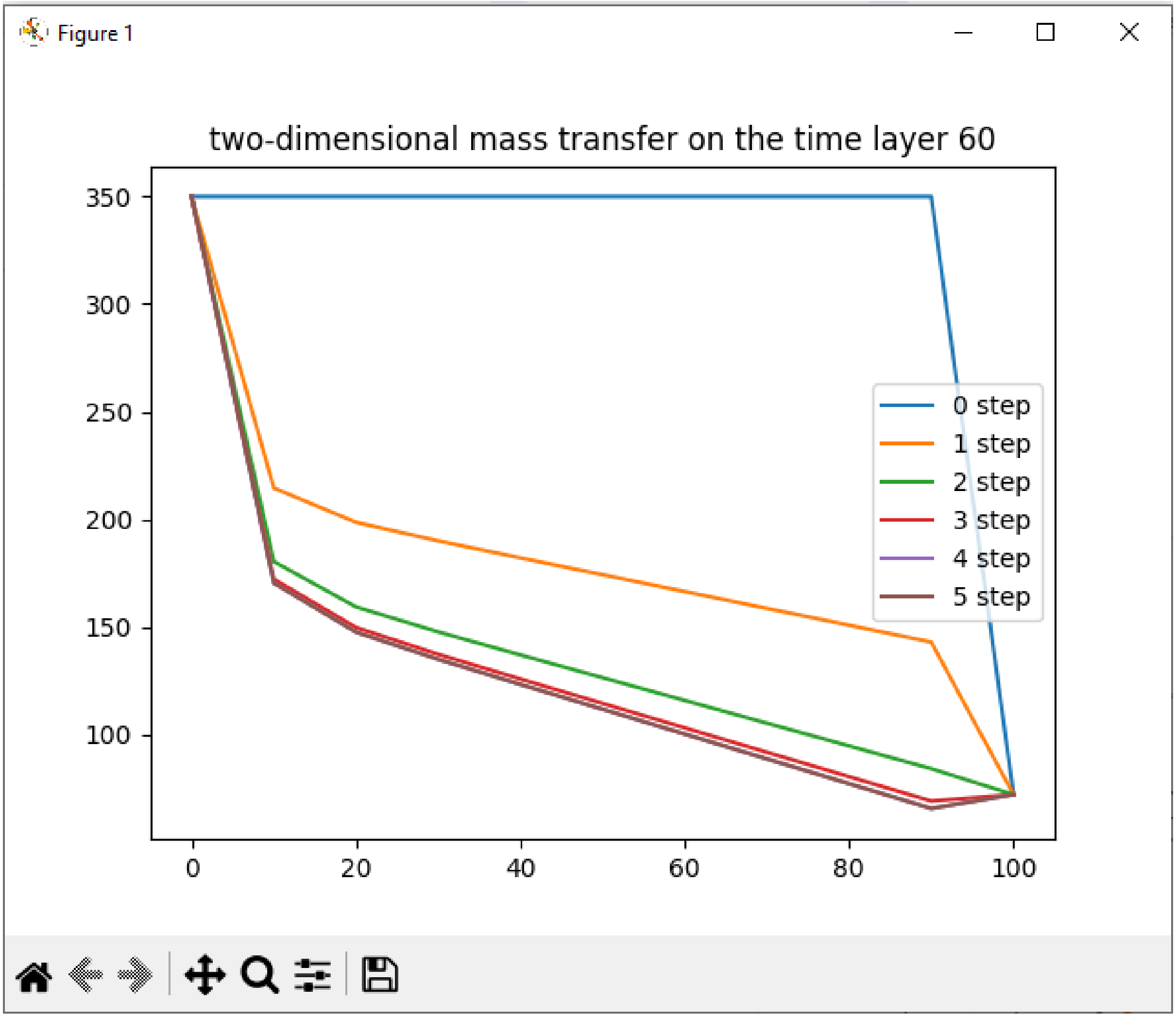


Figure 1
two-dimensional mass transfer on the time layer 60
350
300
250
200
150
100
0
20
40
60
80
100
0 step
1 step
2 step
3 step
4 step
5 step

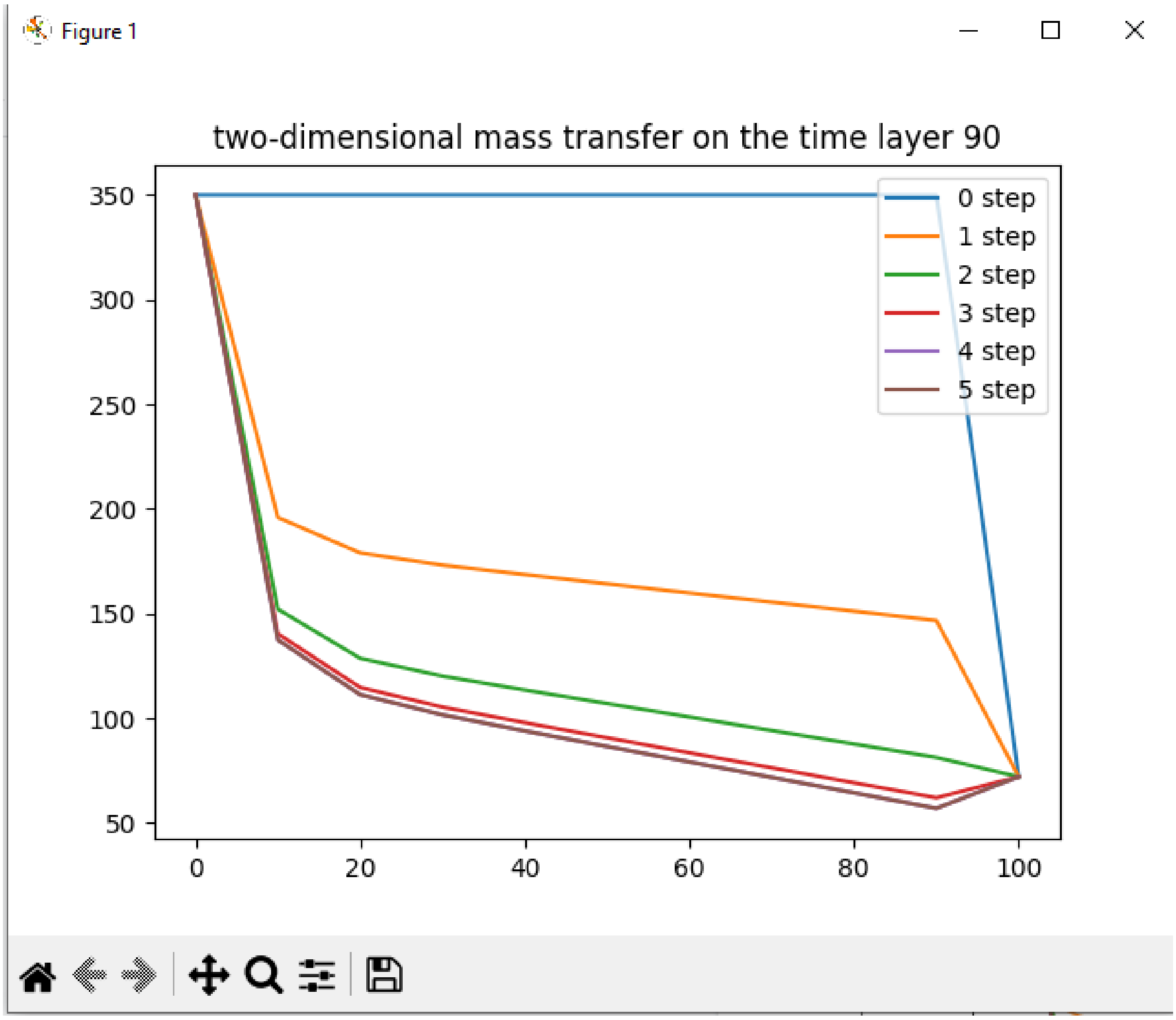


Figure 1
two-dimensional mass transfer on the time layer 90
350
300
250
200
150
100
50
0
20
40
60
80
100
0 step
1 step
2 step
3 step
4 step
5 step

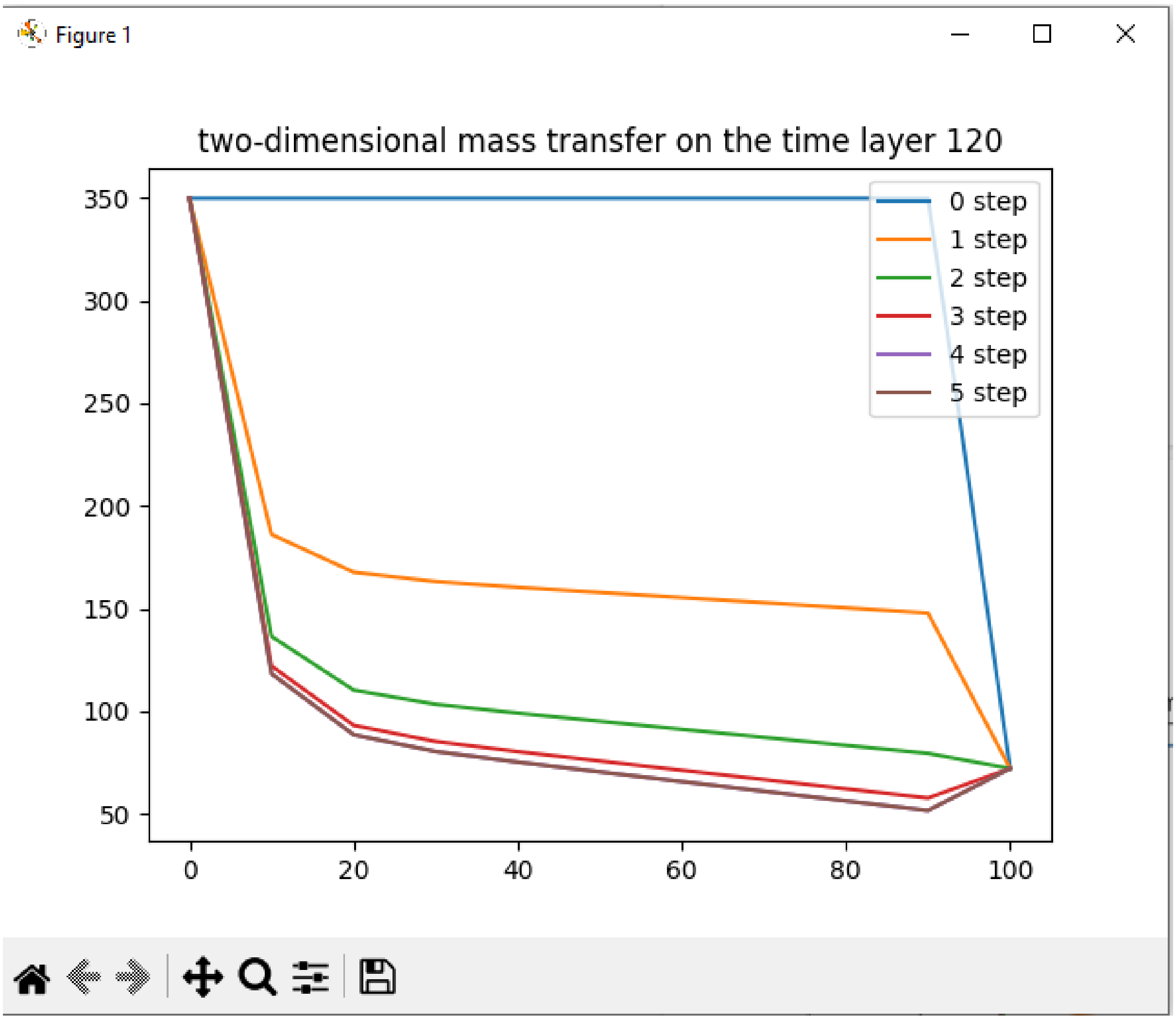
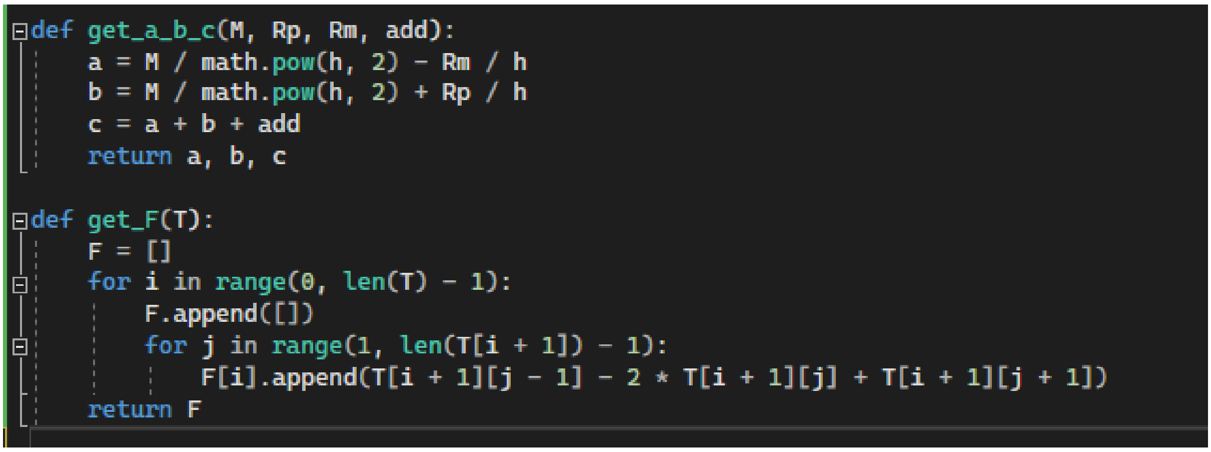


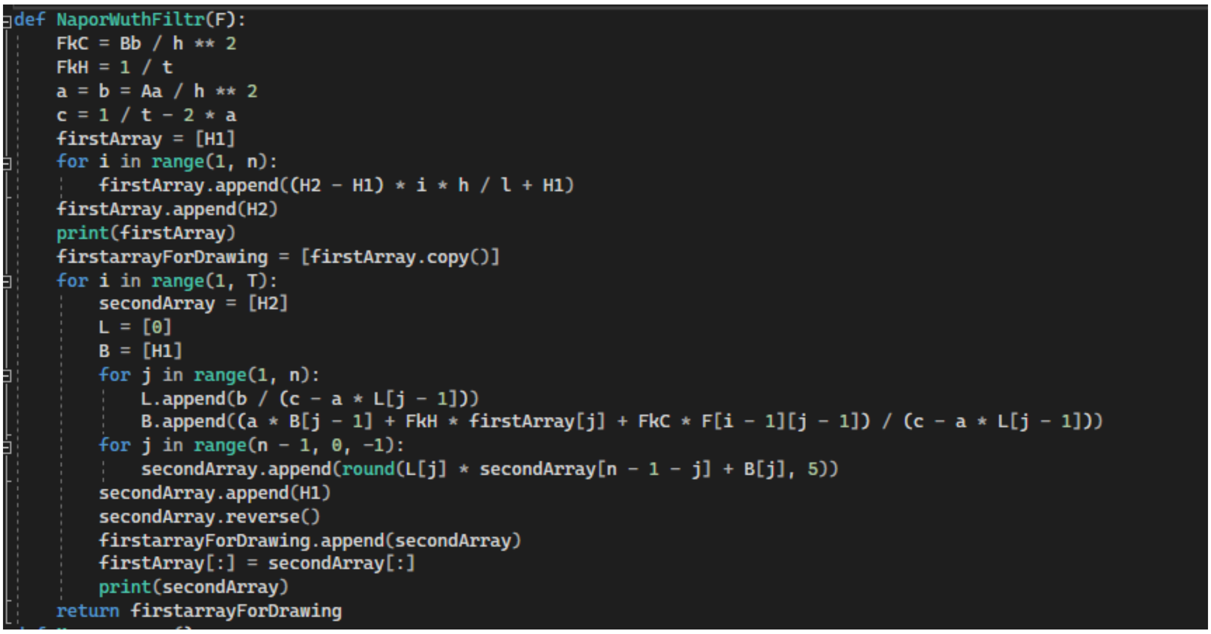
Figure 1
two-dimensional mass transfer on the time layer 120
350
300
250
200
150
100
50
0
20
40
60
80
100
0 step
1 step
2 step
3 step
4 step
5 step

Лабораторна робота №5
N=9

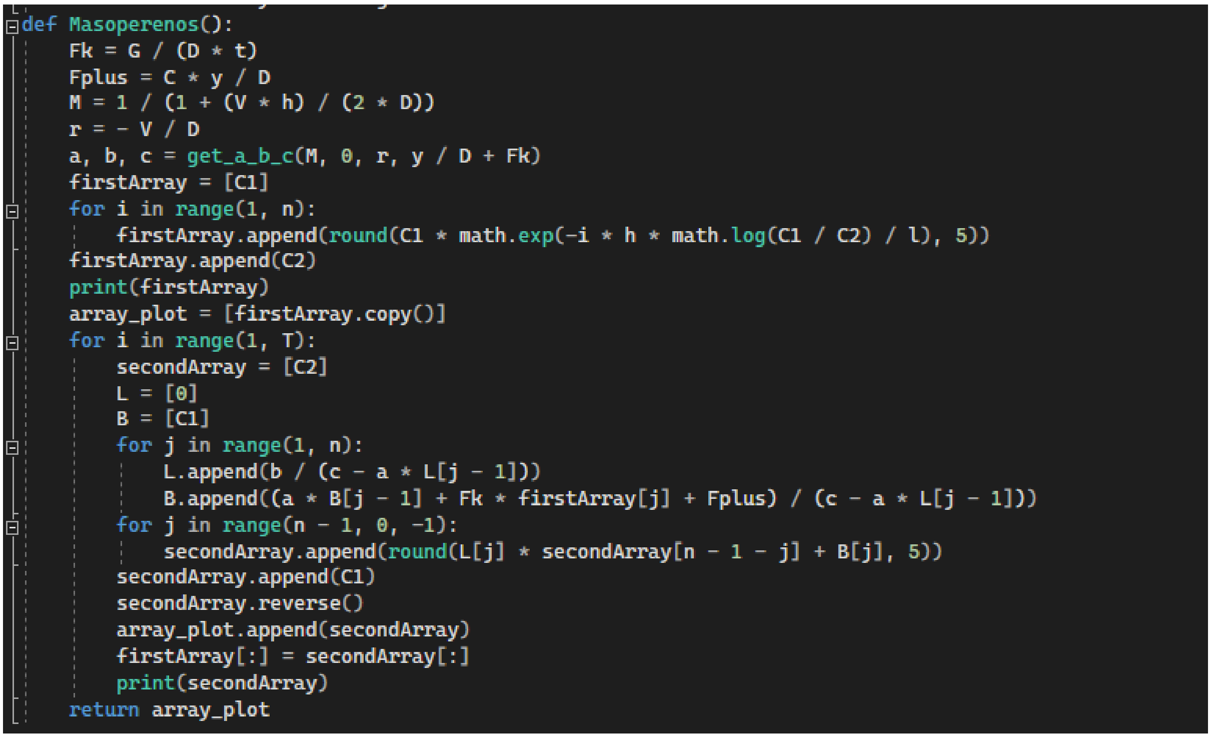
Код:



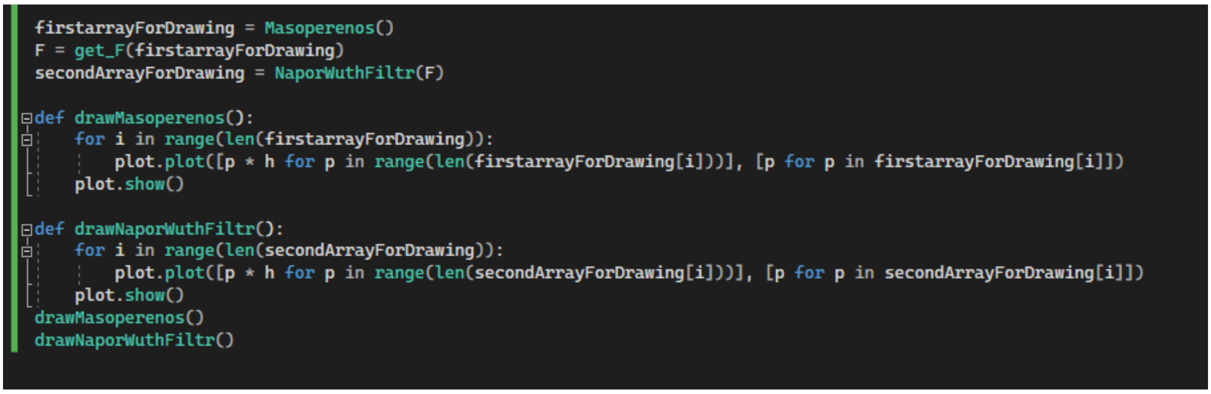
```python
def get_a_b_c(M, Rp, Rm, add):
    a = M / math.pow(h, 2) - Rm / h
    b = M / math.pow(h, 2) + Rp / h
    c = a + b + add
    return a, b, c

def get_F(T):
    F = []
    for i in range(0, len(T) - 1):
        F.append([])
        for j in range(1, len(T[i + 1]) - 1):
            F[i].append(T[i + 1][j - 1] - 2 * T[i + 1][j] + T[i + 1][j + 1])
    return F
```



```python
def NaporWuthFiltr(F):
    FkC = Bb / h ** 2
    FkH = 1 / t
    a = b = Aa / h ** 2
    c = 1 / t - 2 * a
    firstArray = [H1]
    for i in range(1, n):
        firstArray.append((H2 - H1) * i * h / l + H1)
    firstArray.append(H2)
    print(firstArray)
    firstarrayForDrawing = [firstArray.copy()]
    for i in range(1, T):
        secondArray = [H2]
        L = [0]
        B = [H1]
        for j in range(1, n):
            L.append(b / (c - a * L[j - 1]))
            B.append((a * B[j - 1] + FkH * firstArray[j] + FkC * F[i - 1][j - 1]) / (c - a * L[j - 1]))
        for j in range(n - 1, 0, -1):
            secondArray.append(round(L[j] * secondArray[n - 1 - j] + B[j], 5))
        secondArray.append(H1)
        secondArray.reverse()
        firstarrayForDrawing.append(secondArray)
        firstArray[:] = secondArray[:]
        print(secondArray)
    return firstarrayForDrawing
```

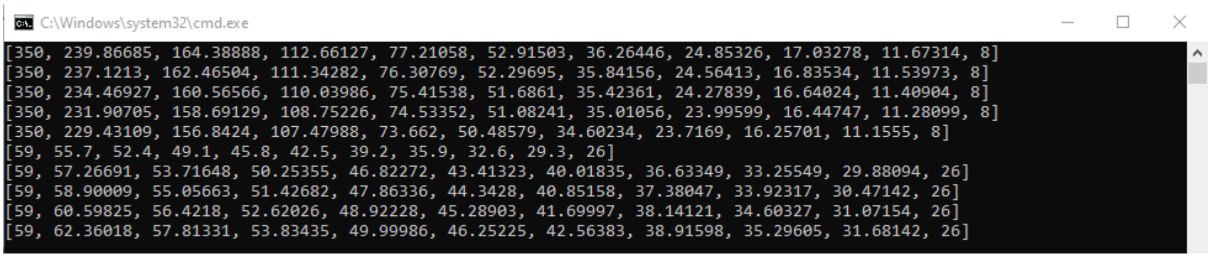


```
def Masoperenos():
    Fk = G / (D * t)
    Fplus = C * y / D
    M = 1 / (1 + (V * h) / (2 * D))
    r = - V / D
    a, b, c = get_a_b_c(M, 0, r, y / D + Fk)
    firstArray = [C1]
    for i in range(1, n):
        firstArray.append(round(C1 * math.exp(-i * h * math.log(C1 / C2) / l), 5))
    firstArray.append(C2)
    print(firstArray)
    array_plot = [firstArray.copy()]
    for i in range(1, T):
        secondArray = [C2]
        L = [0]
        B = [C1]
        for j in range(1, n):
            L.append(b / (c - a * L[j - 1]))
            B.append((a * B[j - 1] + Fk * firstArray[j] + Fplus) / (c - a * L[j - 1]))
        for j in range(n - 1, 0, -1):
            secondArray.append(round(L[j] * secondArray[n - 1 - j] + B[j], 5))
        secondArray.append(C1)
        secondArray.reverse()
        array_plot.append(secondArray)
        firstArray[:] = secondArray[:]
        print(secondArray)
    return array_plot
```



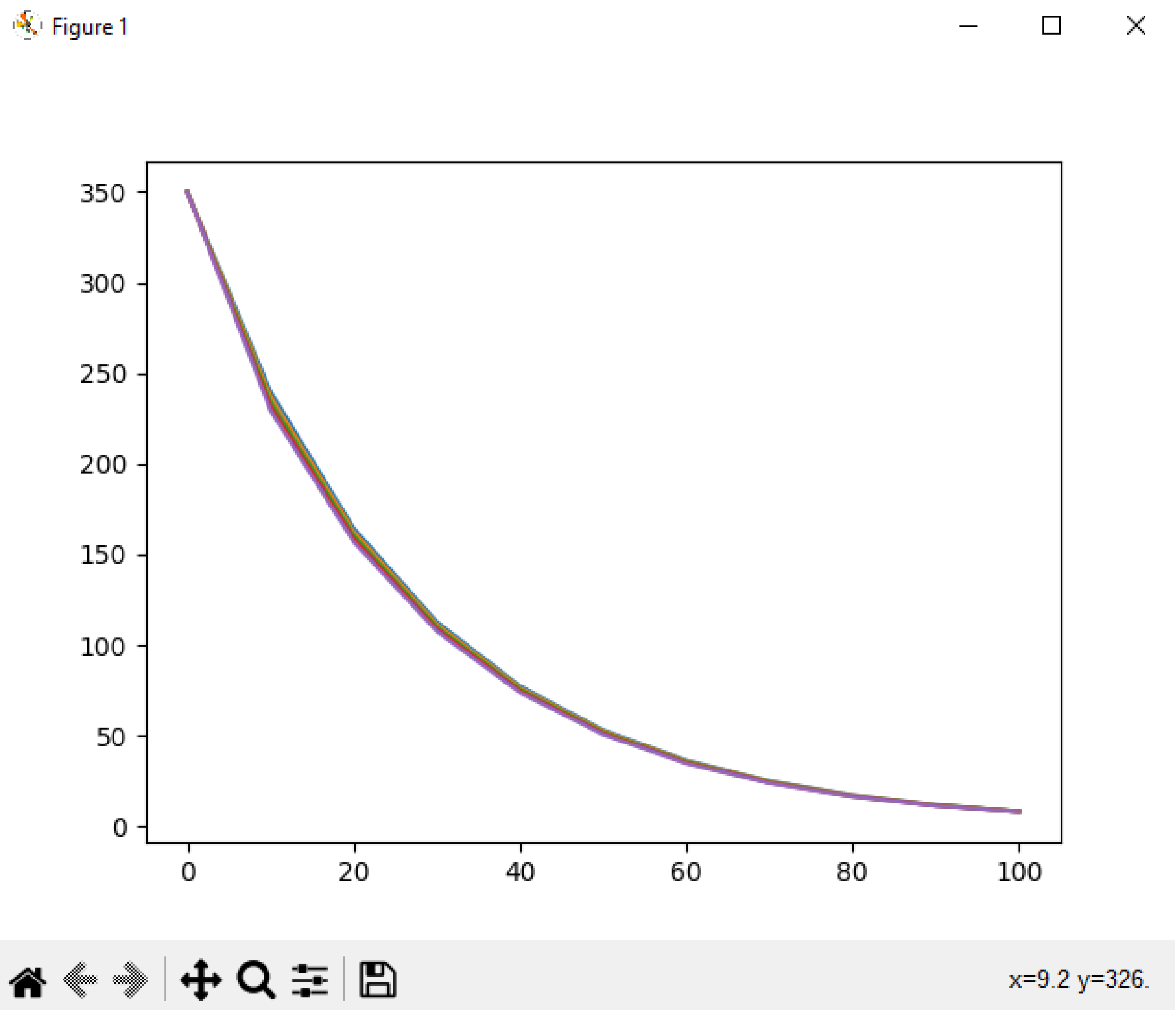
```
firstarrayForDrawing = Masoperenos()
F = get_F(firstarrayForDrawing)
secondArrayForDrawing = NaporWuthFiltr(F)

def drawMasoperenos():
    for i in range(len(firstarrayForDrawing)):
        plot.plot([p * h for p in range(len(firstarrayForDrawing[i]))], [p for p in firstarrayForDrawing[i]])
    plot.show()

def drawNaporWuthFiltr():
    for i in range(len(secondArrayForDrawing)):
        plot.plot([p * h for p in range(len(secondArrayForDrawing[i]))], [p for p in secondArrayForDrawing[i]])
    plot.show()
drawMasoperenos()
drawNaporWuthFiltr()
```

Результат:



```
[350, 239.86685, 164.38888, 112.66127, 77.21058, 52.91503, 36.26446, 24.85326, 17.03278, 11.67314, 8]
[350, 237.1213, 162.46504, 111.34282, 76.30769, 52.29695, 35.84156, 24.56413, 16.83534, 11.53973, 8]
[350, 234.46927, 160.56566, 110.03986, 75.41538, 51.6861, 35.42361, 24.27839, 16.64024, 11.40904, 8]
[350, 231.90705, 158.69129, 108.75226, 74.53352, 51.08241, 35.01056, 23.99599, 16.44747, 11.28099, 8]
[350, 229.43109, 156.8424, 107.47988, 73.662, 50.48579, 34.60234, 23.7169, 16.25701, 11.1555, 8]
[59, 55.7, 52.4, 49.1, 45.8, 42.5, 39.2, 35.9, 32.6, 29.3, 26]
[59, 57.26691, 53.71648, 50.25355, 46.82272, 43.41323, 40.01835, 36.63349, 33.25549, 29.88094, 26]
[59, 58.90009, 55.05663, 51.42682, 47.86336, 44.3428, 40.85158, 37.38047, 33.92317, 30.47142, 26]
[59, 60.59825, 56.4218, 52.62026, 48.92228, 45.28903, 41.69997, 38.14121, 34.60327, 31.07154, 26]
[59, 62.36018, 57.81331, 53.83435, 49.99986, 46.25225, 42.56383, 38.91598, 35.29605, 31.68142, 26]
```

Масоперенос



Напор з урахуванням фільтрації

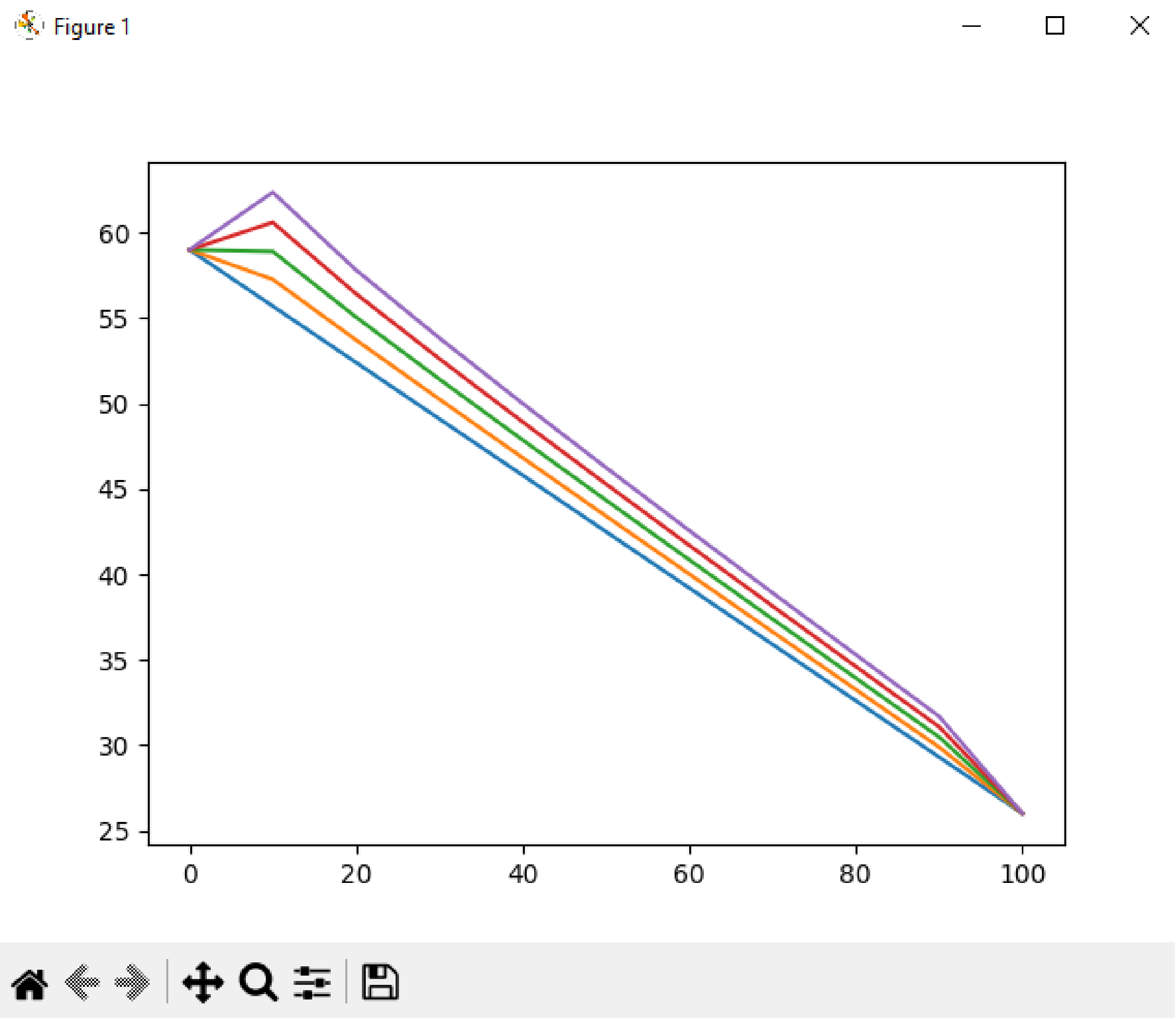
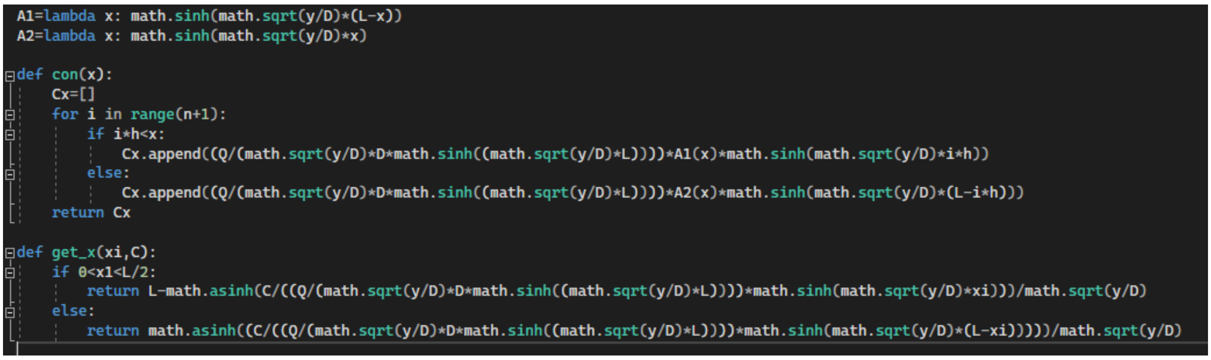


Figure 1
60
55
50
45
40
35
30
25
0
20
40
60
80
100

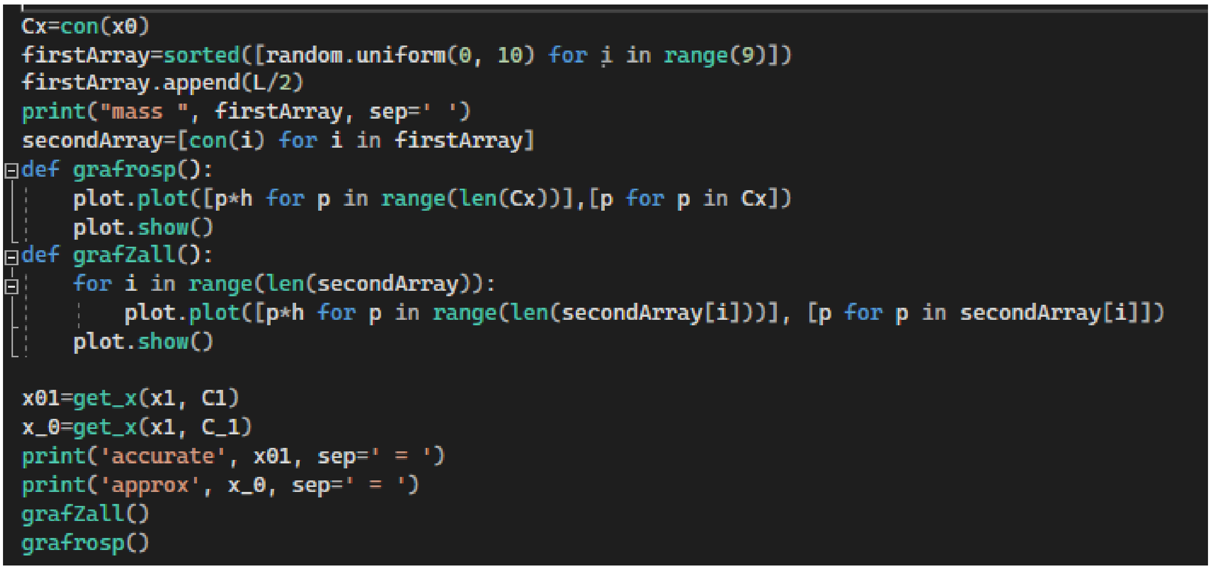
Лабораторна робота №9
Варіант 9

9.	$D = 1{,}7 \quad Q = 1{,}5, \quad \gamma = 0{,}05, \quad l = 20, \quad x_0 = 9$

Код:

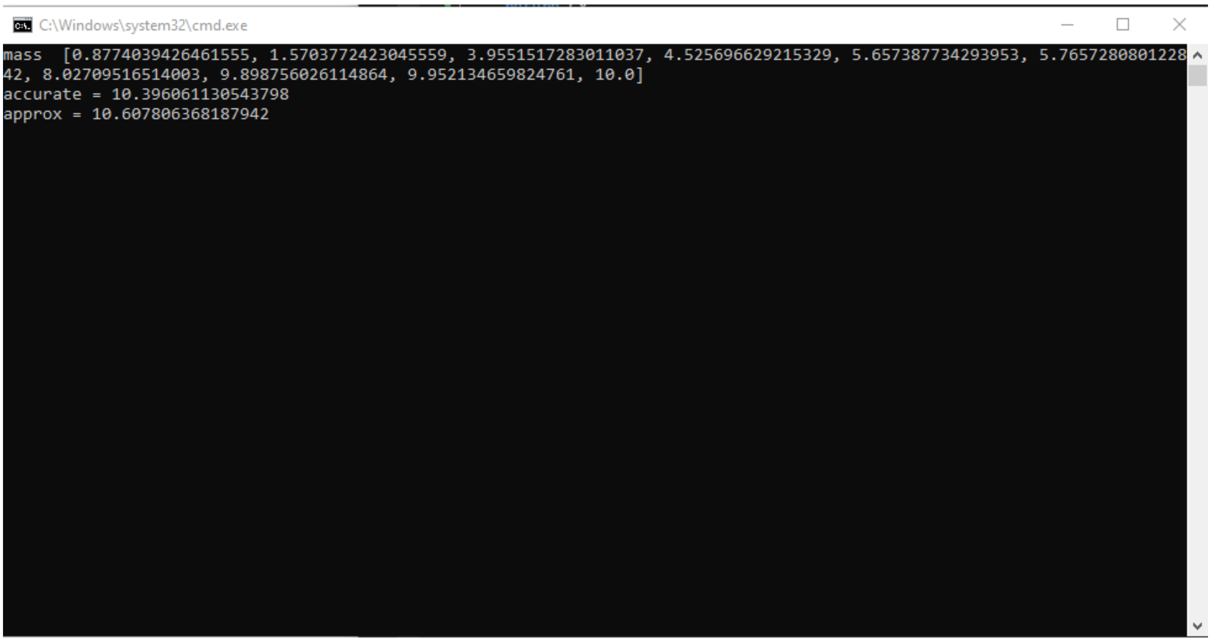


```
A1=lambda x: math.sinh(math.sqrt(y/D)*(L-x))
A2=lambda x: math.sinh(math.sqrt(y/D)*x)

def con(x):
    Cx=[]
    for i in range(n+1):
        if i*h<x:
            Cx.append((Q/(math.sqrt(y/D)*D*math.sinh((math.sqrt(y/D)*L))))*A1(x)*math.sinh(math.sqrt(y/D)*i*h))
        else:
            Cx.append((Q/(math.sqrt(y/D)*D*math.sinh((math.sqrt(y/D)*L))))*A2(x)*math.sinh(math.sqrt(y/D)*(L-i*h)))
    return Cx

def get_x(xi,C):
    if 0<x1<L/2:
        return L-math.asinh(C/((Q/(math.sqrt(y/D)*D*math.sinh((math.sqrt(y/D)*L))))*math.sinh(math.sqrt(y/D)*xi)))/math.sqrt(y/D)
    else:
        return math.asinh((C/((Q/(math.sqrt(y/D)*D*math.sinh((math.sqrt(y/D)*L))))*math.sinh(math.sqrt(y/D)*(L-xi)))))/math.sqrt(y/D)
```



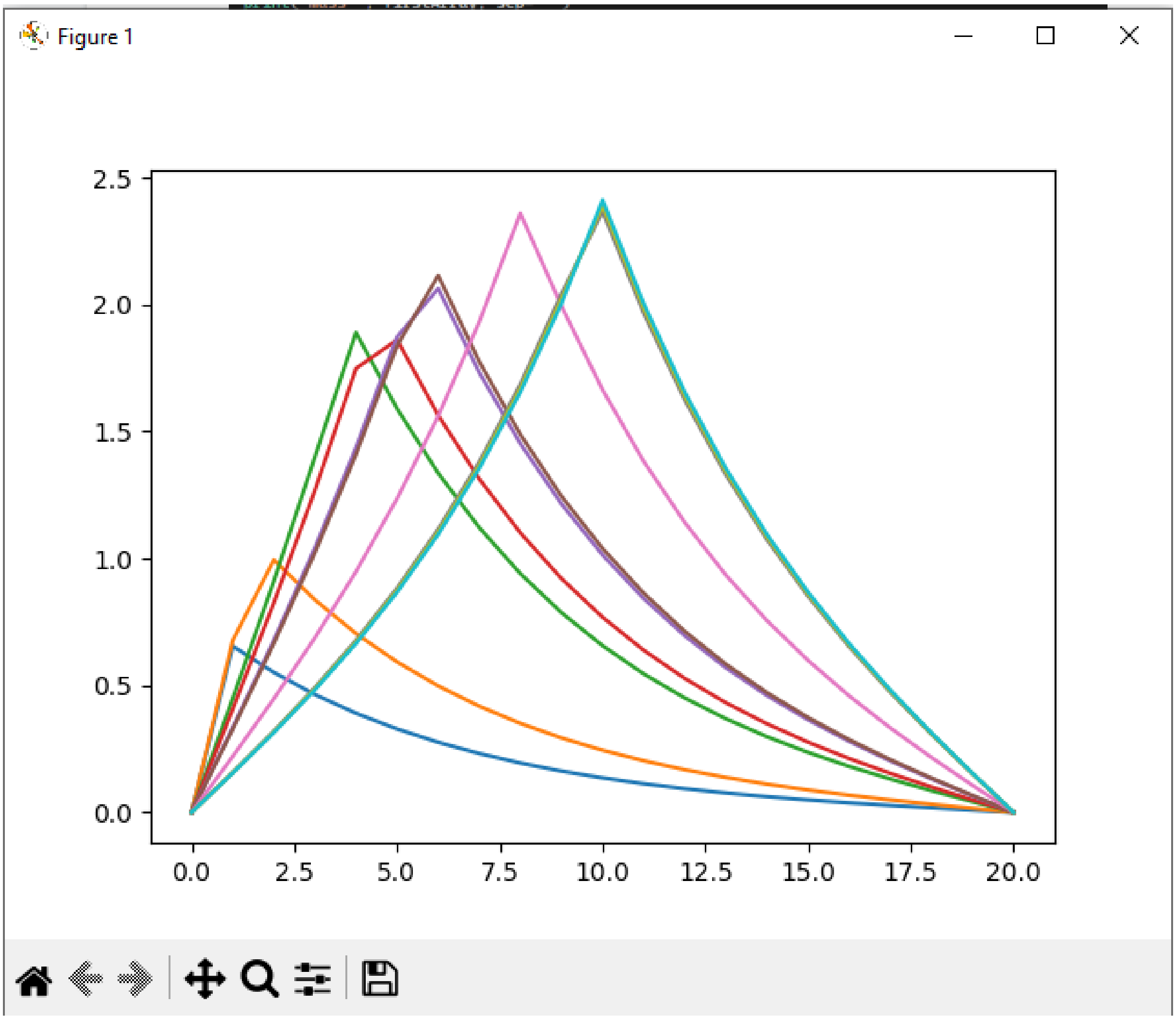
```
Cx=con(x0)
firstArray=sorted([random.uniform(0, 10) for i in range(9)])
firstArray.append(L/2)
print("mass ", firstArray, sep=' ')
secondArray=[con(i) for i in firstArray]
def grafrosp():
    plot.plot([p*h for p in range(len(Cx))],[p for p in Cx])
    plot.show()
def grafZall():
    for i in range(len(secondArray)):
        plot.plot([p*h for p in range(len(secondArray[i]))], [p for p in secondArray[i]])
    plot.show()

x01=get_x(x1, C1)
x_0=get_x(x1, C_1)
print('accurate', x01, sep=' = ')
print('approx', x_0, sep=' = ')
grafZall()
grafrosp()
```

Результат:



```
mass  [0.8774039426461555, 1.5703772423045559, 3.9551517283011037, 4.525696629215329, 5.657387734293953, 5.7657280801228
42, 8.02709516514003, 9.898756026114864, 9.952134659824761, 10.0]
accurate = 10.396061130543798
approx = 10.607806368187942
```

Графік залежностей



Графік розподілу концентрації

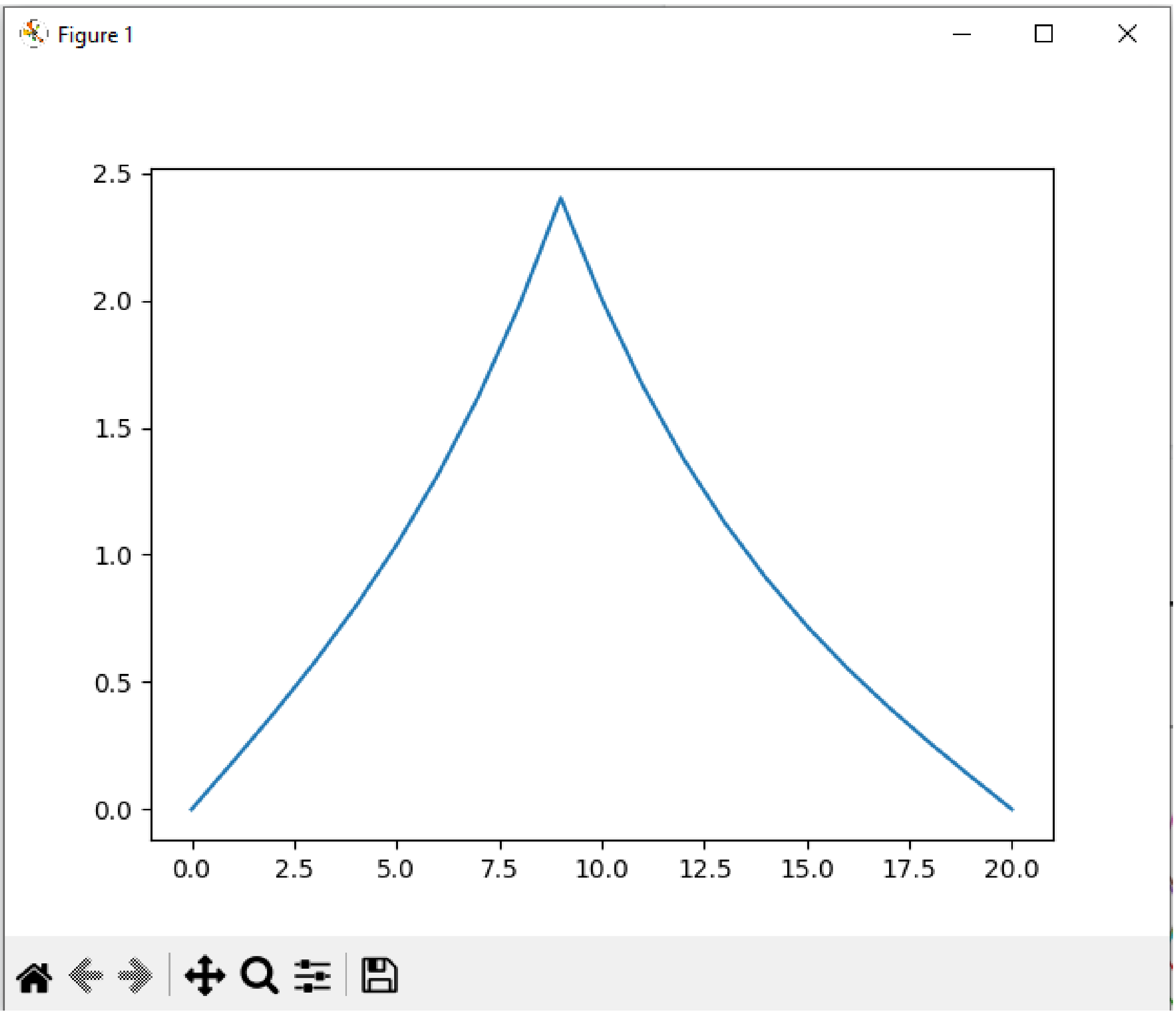


Figure 1
2.5
2.0
1.5
1.0
0.5
0.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0