

ОТЧЁТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ № 6
«ЛИНЕЙНОЕ УРАВНЕНИЕ ПЕРЕНОСА»

(Вариант 9)

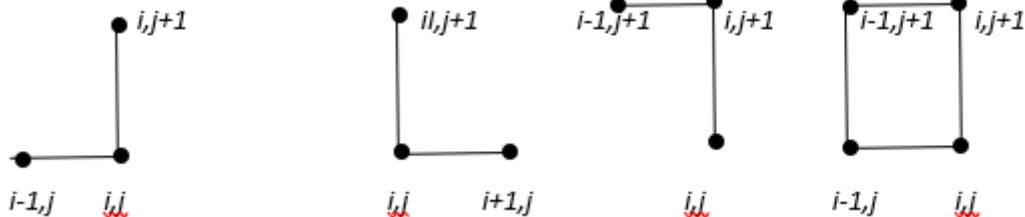
*Выполнил студент 3 курса ПМИИ
Кондратьев Виталий*

Численно решить уравнение переноса

$$\frac{\partial U}{\partial t} + a \frac{\partial U}{\partial x} = f(x, t)$$

- 1) Для полуплоскости $-\infty < x < \infty; t \geq 0$
- 2) В прямоугольнике $0 \leq x \leq 1; 0 \leq t \leq 10$

Во всех случаях $a = \text{const}$. Применить следующие шаблоны для явных и неявных схем.



Схемы выбирать в зависимости от знака a . Причем параметр a принимает два значения: $a=2$ и $a=-2$

Для полуплоскости и для прямоугольной области решить задачу от 0 до 1 с шагом 0.1 по x и от 0 до 10 по времени с шагом, отвечающим условиям устойчивости. Применить все возможные схемы.

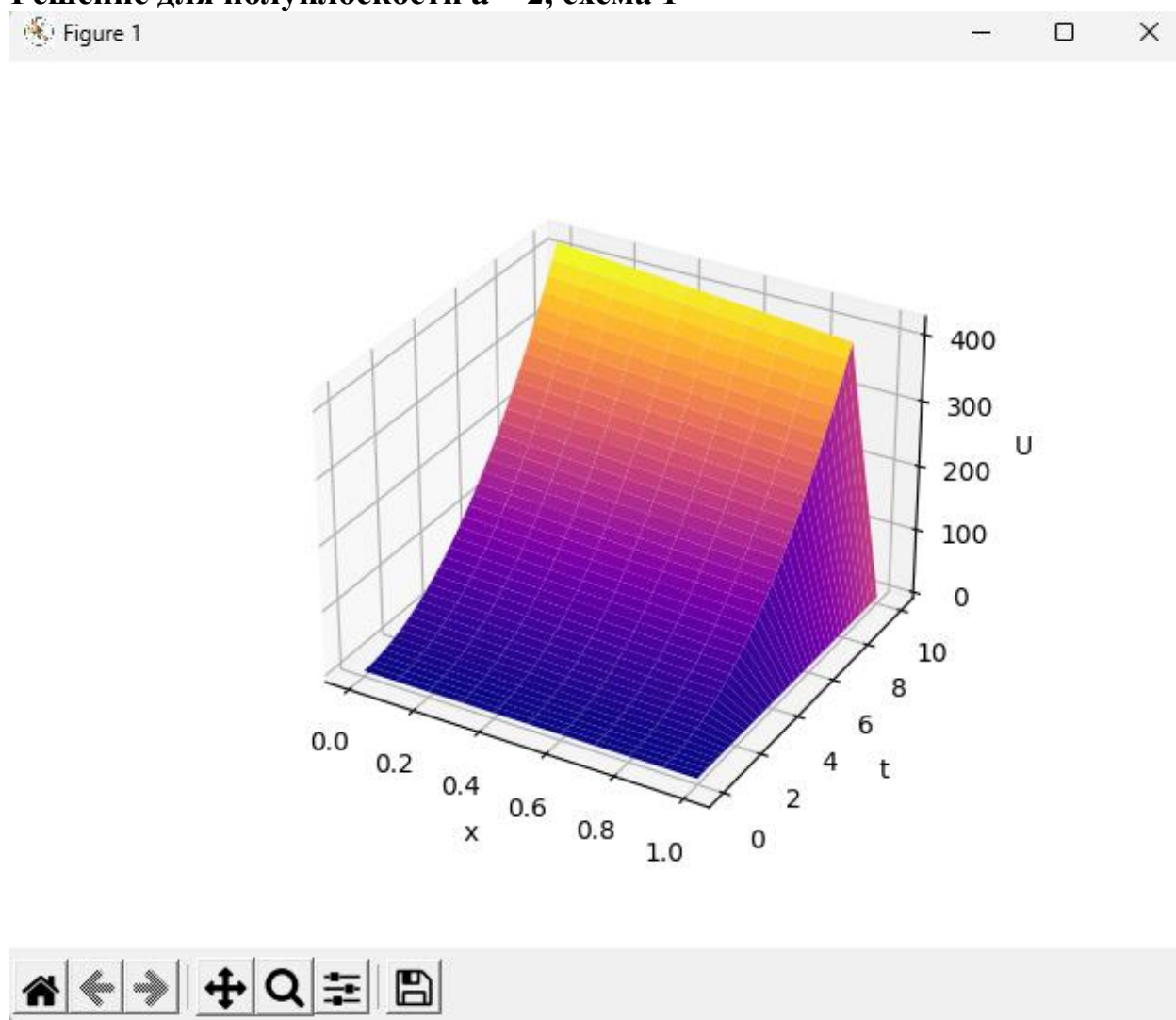
Рассмотреть и решить задачу для всех возможных схем для двух случаев: однородного уравнения с нулевой правой частью и правой частью, указанной в варианте задания.

Вариант	$U(x,0)$	$U(0,t)$ для $a>0$ и прямоугольной области	$U(1,t)$ для $a<0$ и прямоугольной области	$f(x,t)$
9	x^2-x-1	t^2-t-1	t^2-t-1	x

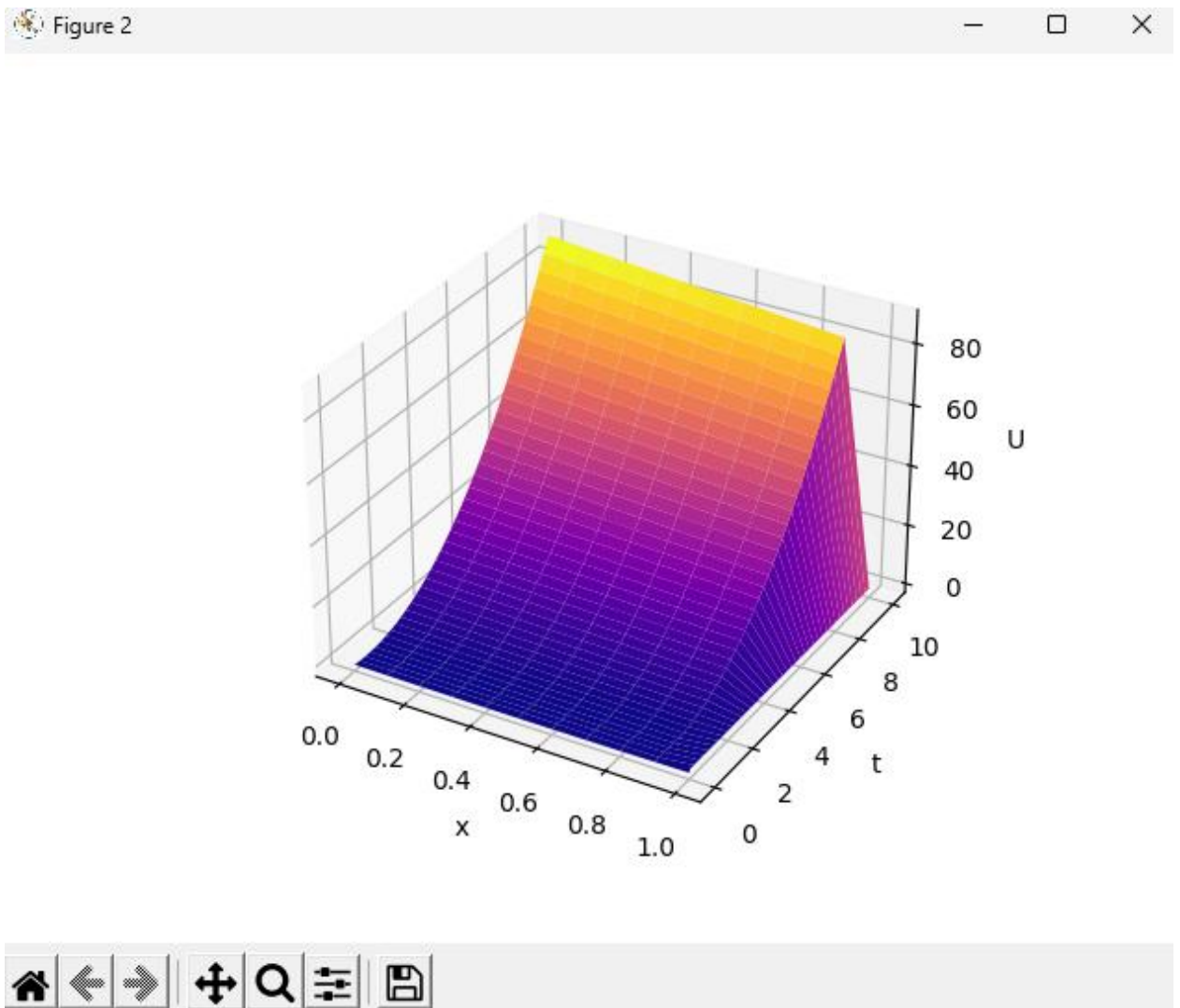
Результаты каждого расчета вывести в виде трехмерных графиков. $U(x,t)$ и двумерных таблиц.

Решения с нулевой правой частью:

Решение для полуплоскости $a = 2$, схема 1

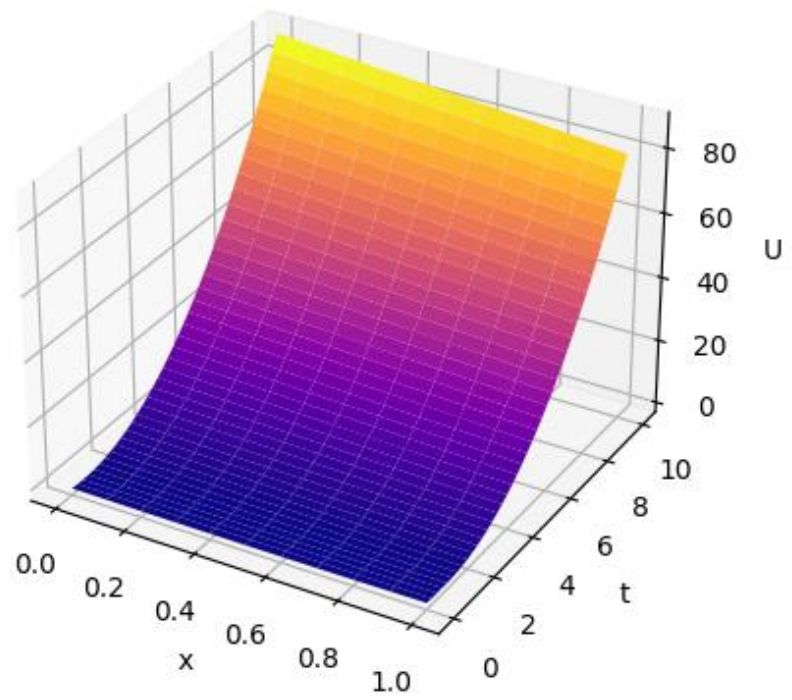


Решение для полуплоскости $a = -2$, схема 2

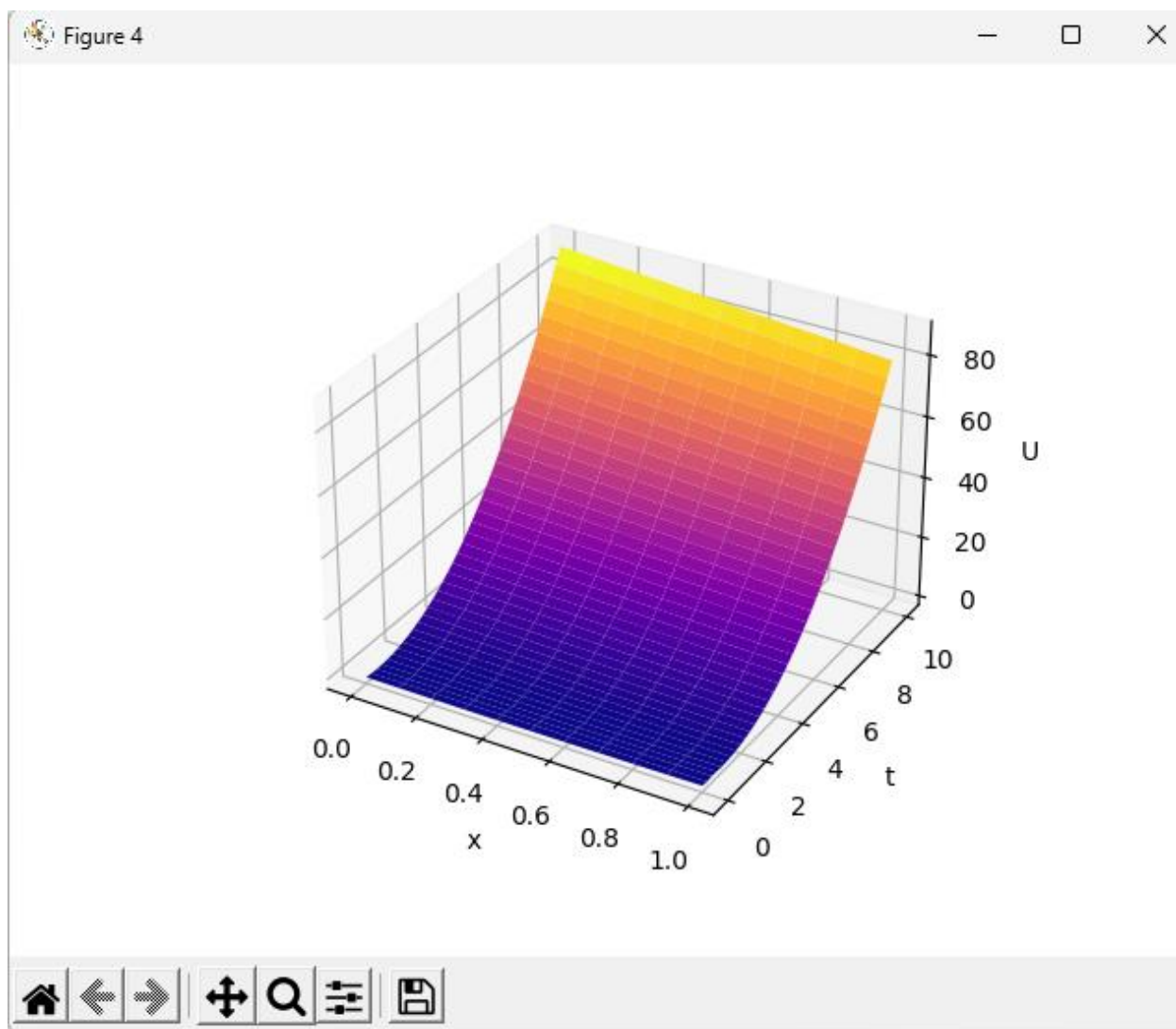


Решение в прямоугольнике $a = 2$, схема 1

Figure 3

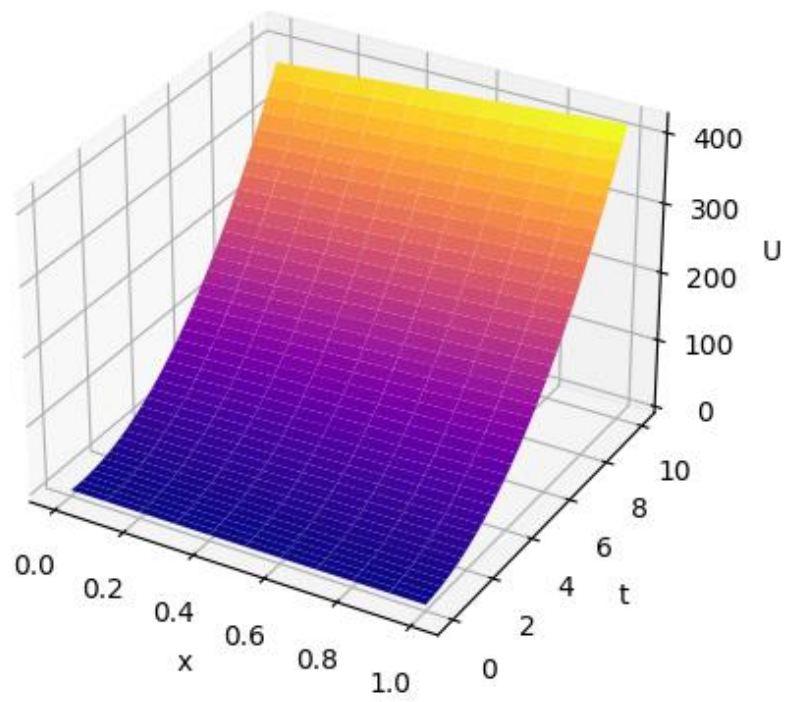


Решение в прямоугольнике $a = 2$, схема 2

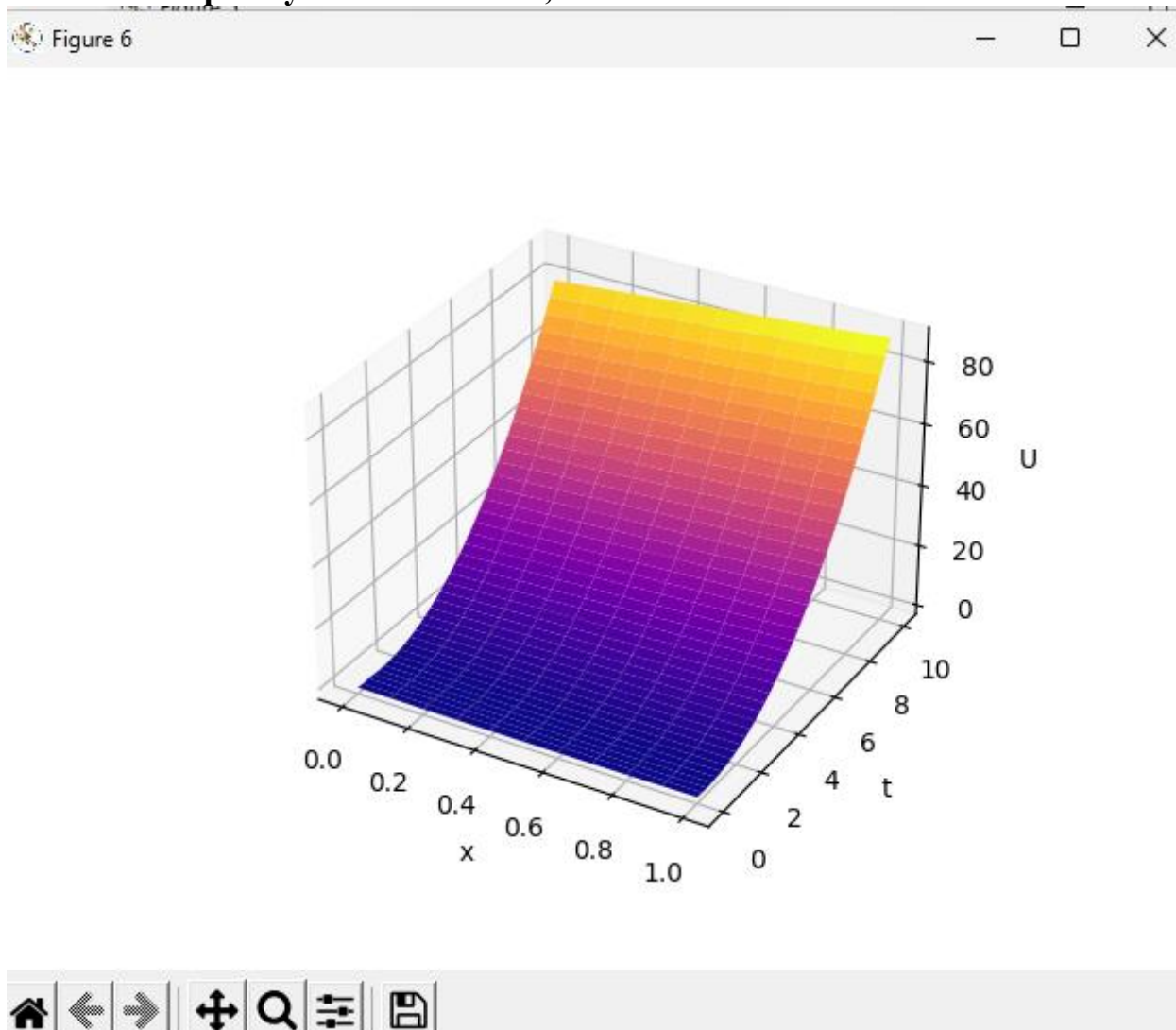


Решение в прямоугольнике $a = 2$, схема 3

Figure 5

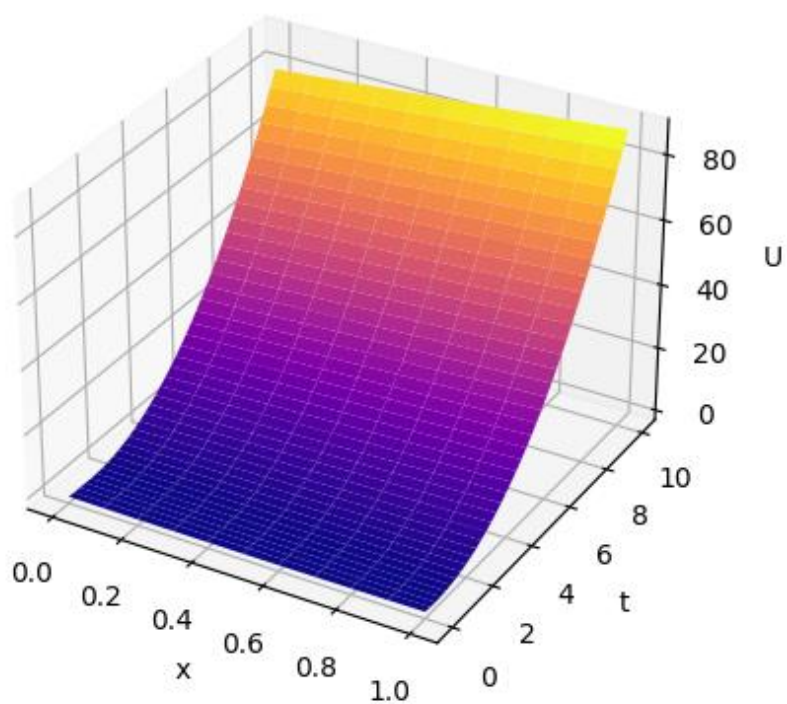


Решение в прямоугольнике $a = -2$, схема 1



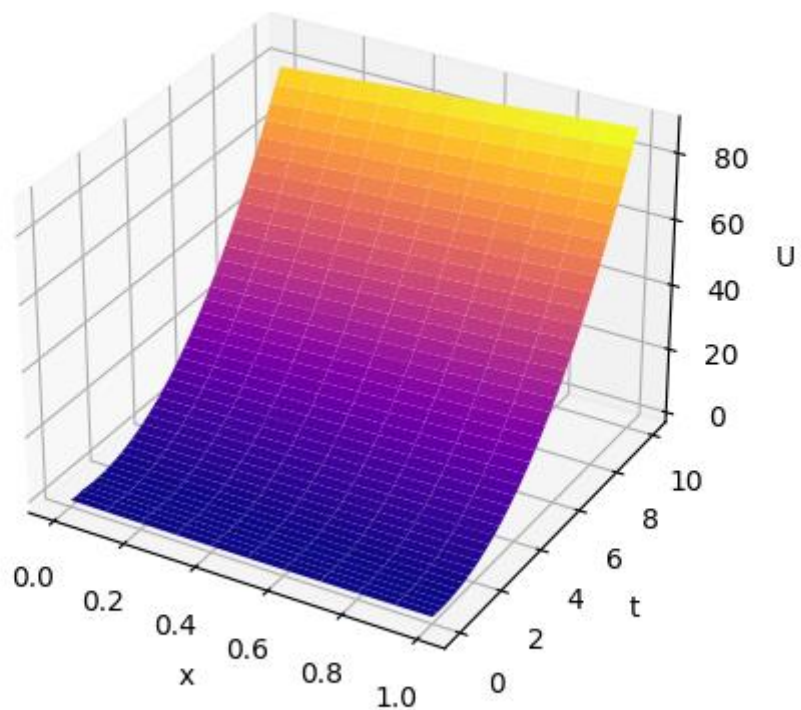
Решение в прямоугольнике $a = -2$, схема 2

Figure 7



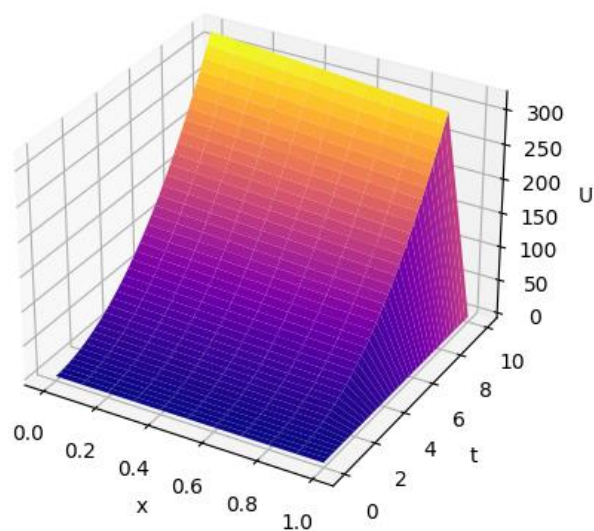
Решение в прямоугольнике $a = -2$, схема 3

Figure 8

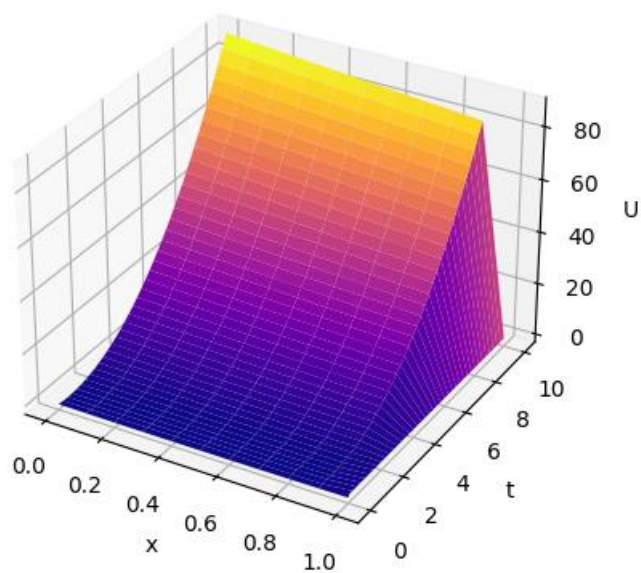


Решения для правой части из варианта:

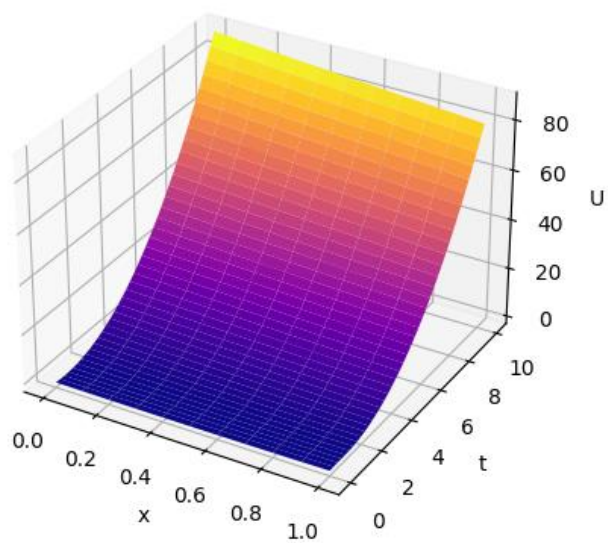
Решение для полуплоскости $a = 2$, схема 1



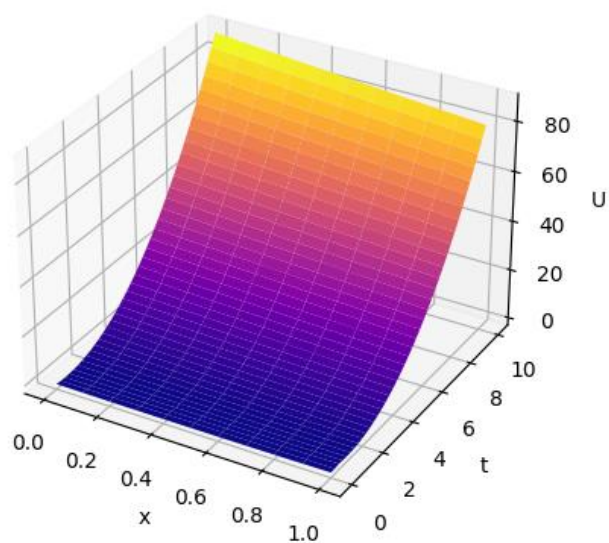
Решение для полуплоскости $a = -2$, схема 2



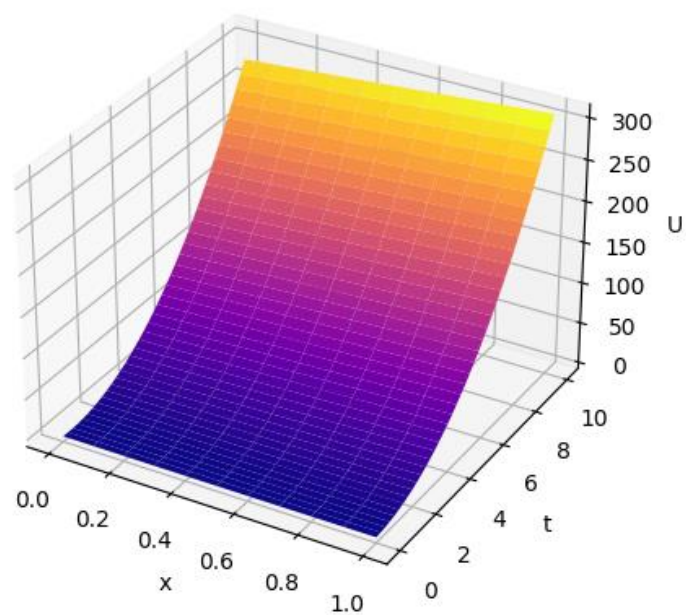
Решение в прямоугольнике $a = 2$, схема 1



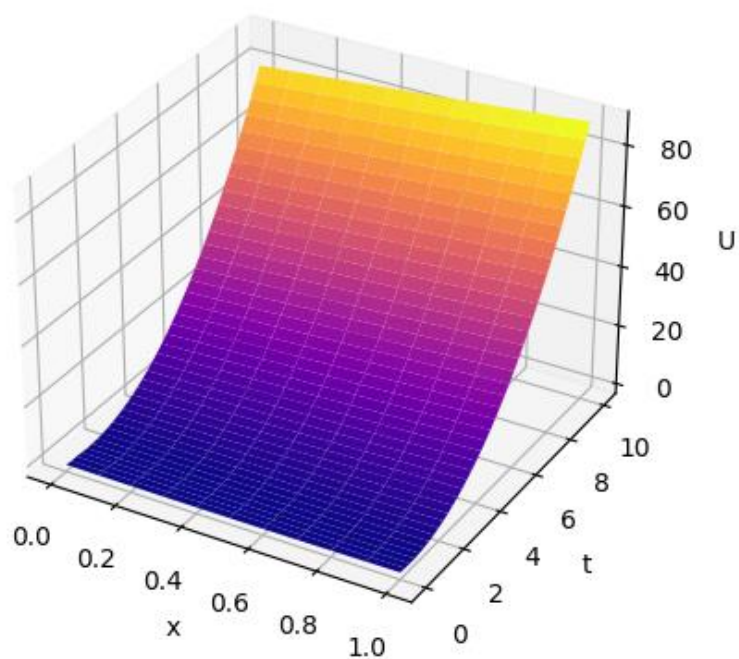
Решение в прямоугольнике $a = 2$, схема 2



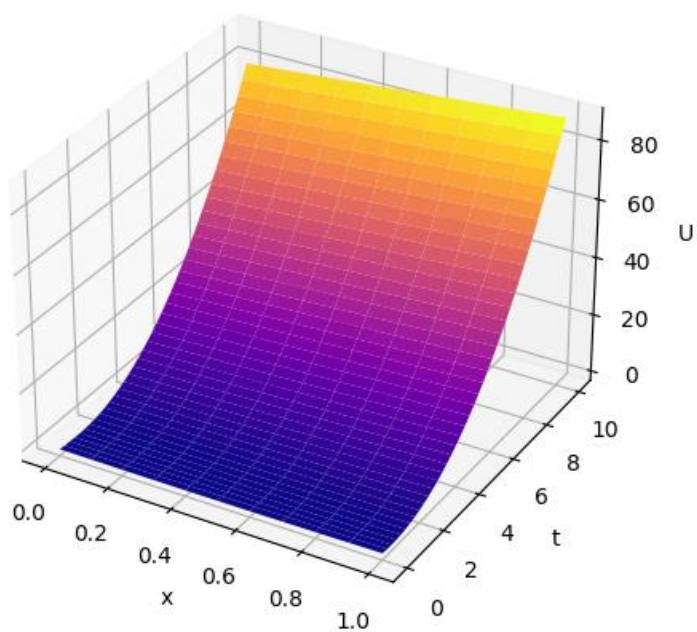
Решение в прямоугольнике $a = 2$, схема 3



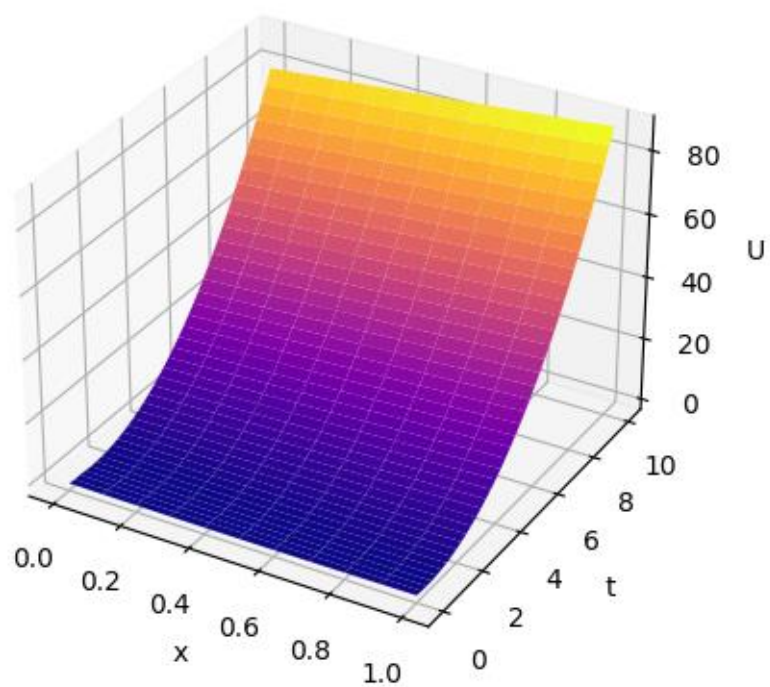
Решение в прямоугольнике $a = -2$, схема 1



Решение в прямоугольнике $a = -2$, схема 2



Решение в прямоугольнике $a = -2$, схема 3



ПРИЛОЖЕНИЕ

```
import numpy as np
import math
import matplotlib.pyplot as plt

def f(x, t):
    return x

def f0(x, t):
    return 0

def Ux(x):
    return x ** 2 - x - 1

def Ut_0(t):
    return t ** 2 - t - 1

def Ut_1(t):
    return t ** 2 - t - 1

def Scheme_One(I, J, tau, U, f, x, t):
    for i in range(1, I):
        for j in range(0, J):
            U[i][j + 1] = U[i - 1][j] + tau * f(x[i], t[j])
    return U

def Scheme_Two(I, J, tau, U, f, x, t):
    for i in range(I - 1, -1, -1):
        for j in range(0, J):
            U[i][j + 1] = U[i + 1][j] - tau * f(x[i], t[j])
    return U

def Scheme_Three(I, J, tau, U, f, x, t, a):
    if a > 0:
        for i in range(1, I + 1):
            for j in range(0, J):
                U[i][j + 1] = (U[i][j] + U[i - 1][j + 1] + tau * f(x[i],
t[j])) / 2
    else:
        for i in range(I - 1, -1, -1):
            for j in range(0, J):
                U[i][j + 1] = (U[i][j] + U[i + 1][j + 1] - tau * f(x[i],
t[j])) / 2
    return U

def Scheme_Four(I, J, tau, U, f, x, t, a, h):
    if a > 0:
        for i in range(1, I + 1):
            for j in range(0, J):
                U[i][j + 1] = U[i - 1][j] + tau * f(x[i] + h / 2, t[j] + tau
/ 2)
    else:
        for i in range(I - 1, -1, -1):
            for j in range(0, J):
                U[i][j + 1] = U[i + 1][j] - tau * f(x[i] + h / 2, t[j] + tau
/ 2)
```



```

    return U

# End of block

def init(I, J, h, tau, a, fx, ft, rectangle):
    if rectangle:
        I1 = I
    else:
        I1 = I + J
    U = np.zeros((I1 + 1, J + 1))
    x = np.zeros(I1 + 1)
    t = np.zeros(J + 1)
    if rectangle:
        for i in range(I + 1):
            x[i] = h * i
            U[i][0] = fx(x[i])
        if (a > 0):
            for j in range(J + 1):
                t[j] = tau * j
                if (j != 0): U[0][j] = ft(t[j])
        else:
            for j in range(J + 1):
                t[j] = tau * j
                if (j != 0): U[I][j] = ft(t[j])
    else:
        if (a > 0):
            for i in range(J + I + 1):
                x[i] = h * (i - J)
                U[i][0] = fx(x[i])
        else:
            for i in range(I + J + 1):
                x[i] = i * h
                U[i][0] = fx(x[i])
            for i in range(J + 1):
                t[i] = i * tau
    return x, t, U

def draw_plot(graph):
    for gr in graph:
        gr[0], gr[1] = np.meshgrid(gr[0], gr[1])
        gr[0], gr[1] = gr[0].T, gr[1].T
        fig = plt.figure()
        ax = fig.add_subplot(projection='3d')
        ax.set_xlabel("x")
        ax.set_ylabel("t")
        ax.set_zlabel("U")
        ax.set_rasterization_zorder(1)
        ax.plot_surface(gr[0], gr[1], gr[2], cmap='plasma')
    plt.show()

rectangle = False
x_start = 0
x_end = 1
time_start = 0
time_end = 10
h = 0.1
tau = 0.05
a = -2

I = int((x_end - x_start) / h)

```

```

J = int((time_end - time_start) / tau)
graph = []

for i in range(0, 2):
    a = -a
    rectangle = False
    if a > 0:
        x, t, U = init(I, J, h, tau, a, Ux, Ut_0, rectangle)
        U = Scheme_One(I + J, J, tau, U, f0, x, t)
        x, t = np.linspace(x_start, x_end, I + 1), np.linspace(time_start,
time_end, J + 1)
        U = U[J:J + I + 1, :]

        graph.append([x, t, U])

    else:
        x, t, U = init(I, J, h, tau, a, Ux, Ut_1, rectangle)
        U = Scheme_Two(I + J, J, tau, U, f0, x, t)
        x, t = np.linspace(x_start, x_end, I + 1), np.linspace(time_start,
time_end, J + 1)
        U = U[0:I + 1, :]

        graph.append([x, t, U])
    rectangle = True
    if a > 0:
        x, t, U = init(I, J, h, tau, a, Ux, Ut_0, rectangle)
        U = Scheme_One(I, J, tau, U, f0, x, t)
        x, t = np.linspace(x_start, x_end, I + 1), np.linspace(time_start,
time_end, J + 1)
        graph.append([x, t, U])

        x, t, U = init(I, J, h, tau, a, Ux, Ut_0, rectangle)
        U = Scheme_Three(I, J, tau, U, f0, x, t, a)
        x, t = np.linspace(x_start, x_end, I + 1), np.linspace(time_start,
time_end, J + 1)
        graph.append([x, t, U])

        x, t, U = init(I, J, h, tau, a, Ux, Ut_0, rectangle)
        U = Scheme_Four(I, J, tau, U, f0, x, t, a, h)
        x, t = np.linspace(x_start, x_end, I + 1), np.linspace(time_start,
time_end, J + 1)
        graph.append([x, t, U])
    else:
        x, t, U = init(I, J, h, tau, a, Ux, Ut_1, rectangle)
        U = Scheme_Two(I, J, tau, U, f0, x, t)
        x, t = np.linspace(x_start, x_end, I + 1), np.linspace(time_start,
time_end, J + 1)
        graph.append([x, t, U])

        x, t, U = init(I, J, h, tau, a, Ux, Ut_1, rectangle)
        U = Scheme_Three(I, J, tau, U, f0, x, t, a)
        x, t = np.linspace(x_start, x_end, I + 1), np.linspace(time_start,
time_end, J + 1)
        graph.append([x, t, U])

        x, t, U = init(I, J, h, tau, a, Ux, Ut_1, rectangle)
        U = Scheme_Four(I, J, tau, U, f0, x, t, a, h)
        x, t = np.linspace(x_start, x_end, I + 1), np.linspace(time_start,
time_end, J + 1)
        graph.append([x, t, U])

# Для правой части из варианта
'''
for i in range(0, 2):

```

```

a = -a
rectangle = False
if a > 0:
    x, t, U = init(I, J, h, tau, a, Ux, Ut_0, rectangle)
    U = Scheme_One(I + J, J, tau, U, f, x, t)
    x, t = np.linspace(x_start, x_end, I + 1), np.linspace(time_start,
time_end, J + 1)
    U = U[J:J + I + 1, :]

    graph.append([x, t, U])
else:
    x, t, U = init(I, J, h, tau, a, Ux, Ut_1, rectangle)
    U = Scheme_Two(I + J, J, tau, U, f, x, t)
    x, t = np.linspace(x_start, x_end, I + 1), np.linspace(time_start,
time_end, J + 1)
    U = U[0:I + 1, :]

    graph.append([x, t, U])
rectangle = True
if a > 0:
    x, t, U = init(I, J, h, tau, a, Ux, Ut_0, rectangle)
    U = Scheme_One(I, J, tau, U, f0, x, t)
    x, t = np.linspace(x_start, x_end, I + 1), np.linspace(time_start,
time_end, J + 1)
    graph.append([x, t, U])

    x, t, U = init(I, J, h, tau, a, Ux, Ut_0, rectangle)
    U = Scheme_Three(I, J, tau, U, f0, x, t, a)
    x, t = np.linspace(x_start, x_end, I + 1), np.linspace(time_start,
time_end, J + 1)
    graph.append([x, t, U])

    x, t, U = init(I, J, h, tau, a, Ux, Ut_0, rectangle)
    U = Scheme_Four(I, J, tau, U, f0, x, t, a, h)
    x, t = np.linspace(x_start, x_end, I + 1), np.linspace(time_start,
time_end, J + 1)
    graph.append([x, t, U])
else:
    x, t, U = init(I, J, h, tau, a, Ux, Ut_1, rectangle)
    U = Scheme_Two(I, J, tau, U, f0, x, t)
    x, t = np.linspace(x_start, x_end, I + 1), np.linspace(time_start,
time_end, J + 1)
    graph.append([x, t, U])

    x, t, U = init(I, J, h, tau, a, Ux, Ut_1, rectangle)
    U = Scheme_Three(I, J, tau, U, f0, x, t, a)
    x, t = np.linspace(x_start, x_end, I + 1), np.linspace(time_start,
time_end, J + 1)
    graph.append([x, t, U])

    x, t, U = init(I, J, h, tau, a, Ux, Ut_1, rectangle)
    U = Scheme_Four(I, J, tau, U, f0, x, t, a, h)
    x, t = np.linspace(x_start, x_end, I + 1), np.linspace(time_start,
time_end, J + 1)
    graph.append([x, t, U])
'''
draw_plot(graph)

```