**ОТЧЁТ**

**ПО ЛАБОРАТОРНОЙ РАБОТЕ № 6**

**ЛИНЕЙНОЕ УРАВНЕНИЕ ПЕРЕНОСА**

**(Вариант 10)**

*Выполнил студент 3 курса МОиАИС*

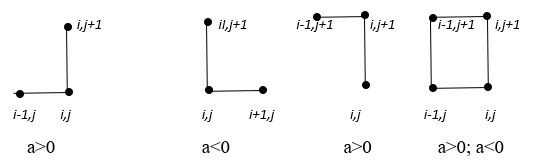
*Соколов Арсений*

**Задание:**

Численно решить уравнение переноса

1. Для полуплоскости
2. В прямоугольнике

Во всех случаях *a* – const. Применить следующие шаблоны для явных и неявных схем.



Схемы выбирать в зависимости от знака a (см. на схеме). Параметр a принимает два значения: a=2 и a= -2

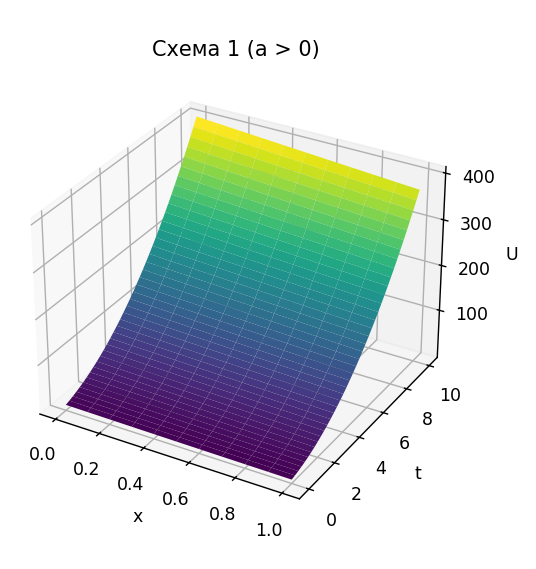
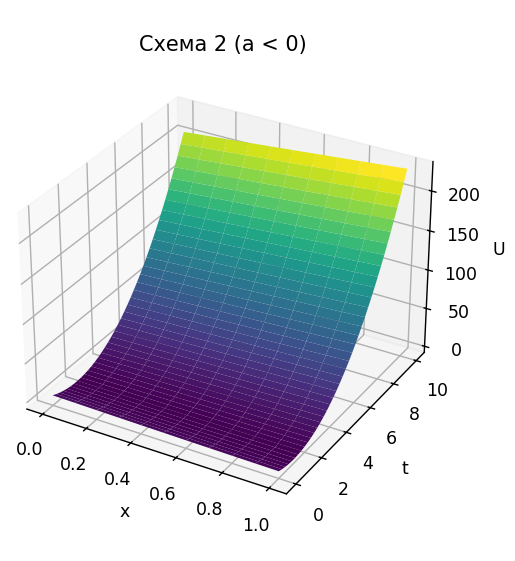
Для полуплоскости и для прямоугольной области решить задачу от 0 до 1 с шагом 0.1 по х и от 0 до 10 по времени с шагом, отвечающим условиям устойчивости. Для полуплоскости применить схему 1 и 2 , для прямоугольной области применить все возможные схемы.

Результаты вывести в виде трехмерных рисунков.

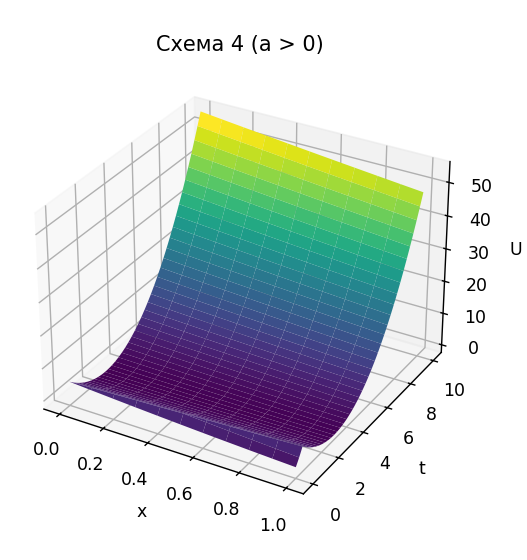
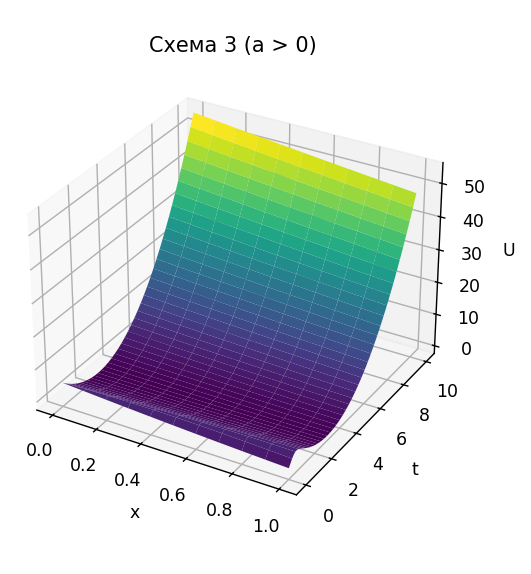
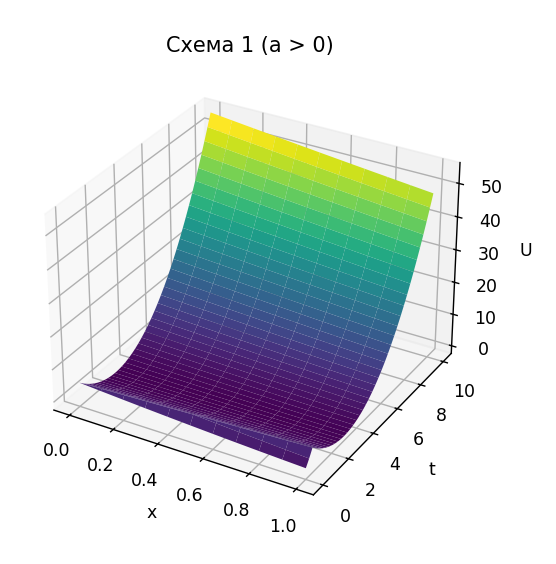
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Вариант | U(x,0) | U(0,t) для a>0 и прямоугольной области | U(1,t) для a<0 и прямоугольной области | *f(x,t)* |
| 10 | x2-5x+5 | t2-5t+5 | t2-5t+1 | x |

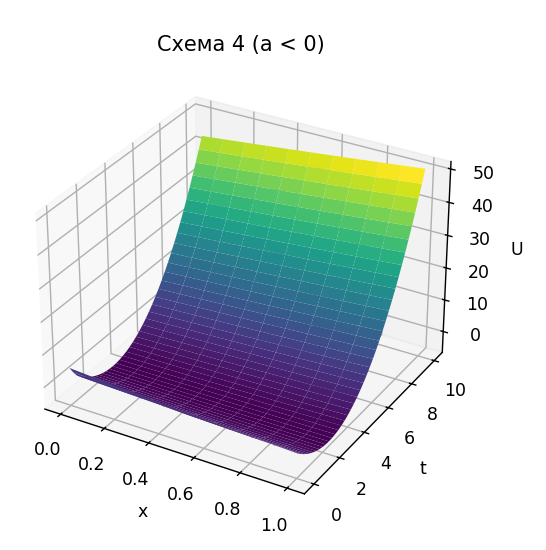
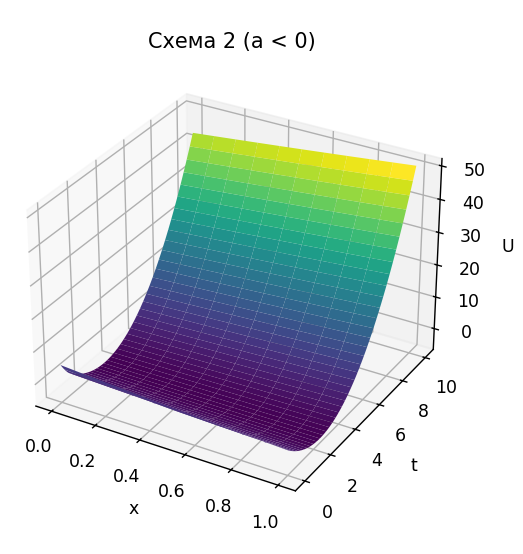
**Решение**

1. Для полуплоскости :

1. В прямоугольнике :





**ПРИЛОЖЕНИЕ 1**

***Программа вычисления уравнения для полуплоскости***

import numpy as np  
import matplotlib.pyplot as plt  
from mpl\_toolkits.mplot3d import Axes3D  
  
# Параметры задачи  
x\_start, x\_end = 0, 1  
t\_start, t\_end = 0, 10  
dx = 0.1  
a1 = 2  
a2 = -2  
  
# Условие устойчивости  
dt = dx / max([abs(a1), abs(a2)])  
x = np.arange(x\_start, x\_end + dx, dx)  
t = np.arange(t\_start, t\_end + dt, dt)  
  
# Размерности сетки  
nx = len(x)  
nt = len(t)  
  
# Начальное и граничные условия  
def U\_x(x):  
 return x\*\*2 - 5 \* x + 5  
  
def U\_t1(t):  
 return t\*\*2 - 5 \* t + 5  
  
def U\_t2(t):  
 return t\*\*2 - 5 \* t + 1  
  
def f(x, t):  
 return x  
  
# Схема 1 для a > 0  
def scheme\_1(a):  
 U = np.zeros((nt, nx))  
 U[0, :] = U\_x(x) # Начальное условие  
 U[:, 0] = U\_t1(t) # Граничные условия  
  
 for j in range(0, nt - 1):  
 for i in range(1, nx):  
 U[j + 1, i] = (U[j, i] - (a \* dt / dx) \* (U[j, i] - U[j, i - 1]) +  
 dt \* f(x[i], t[j]))  
 return U  
  
# Схема 2 для a < 0  
def scheme\_2(a):  
 U = np.zeros((nt, nx))  
 U[0, :] = U\_x(x) # Начальное условие  
 U[:, -1] = U\_t2(t) # Граничные условия  
  
 for j in range(0, nt - 1):  
 for i in range(nx - 2, -1, -1):  
 U[j + 1, i] = (U[j, i] - (a \* dt / dx) \* (U[j, i + 1] - U[j, i]) +  
 dt \* f(x[i], t[j]))  
 return U  
  
# Построение графика  
def plot\_3d(U, title):  
 X, T = np.meshgrid(x, t)  
 fig = plt.figure(figsize=(10, 6))  
 ax = fig.add\_subplot(111, projection='3d')  
 ax.plot\_surface(X, T, U, cmap='viridis')  
 ax.set\_title(title)  
 ax.set\_xlabel('x')  
 ax.set\_ylabel('t')  
 ax.set\_zlabel('U(x, t)')  
 plt.show()  
  
# Решение для a > 0  
plot\_3d(scheme\_1(a1), "Схема 1 (a > 0)")  
  
# Решение для a < 0  
plot\_3d(scheme\_2(a2), "Схема 2 (a < 0)")

**ПРИЛОЖЕНИЕ 2**

***Программа вычисления уравнения для прямоугольника***

import numpy as np  
import matplotlib.pyplot as plt  
  
  
def f(x): # f(x, t)  
 return x  
  
  
def U\_x(x):  
 return x \*\* 2 - 5 \* x + 5  
  
  
def U\_t1(t):  
 return t \*\* 2 - 5 \* t + 5  
  
  
def U\_t2(t):  
 return t \*\* 2 - 5 \* t + 1  
  
  
def scheme\_1(I, J, dt, U, f, x):  
 for i in range(1, I + 1):  
 for j in range(0, J):  
 U[i][j + 1] = U[i - 1][j] + dt \* f(x[i])  
 return U  
  
  
def scheme\_2(I, J, dt, U, f, x):  
 for i in range(I - 1, -1, -1):  
 for j in range(0, J):  
 U[i][j + 1] = U[i + 1][j] - dt \* f(x[i])  
 return U  
  
  
def scheme\_3(I, J, dt, U, f, x, 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] + dt \* f(x[i])) / 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] - dt \* f(x[i])) / 2  
 return U  
  
  
def scheme\_4(I, J, dt, U, f, x, a, dx):  
 if a > 0:  
 for i in range(1, I + 1):  
 for j in range(0, J):  
 U[i][j + 1] = U[i - 1][j] + dt \* f(x[i] + dx / 2)  
 else:  
 for i in range(I - 1, -1, -1):  
 for j in range(0, J):  
 U[i][j + 1] = U[i + 1][j] - dt \* f(x[i] + dx / 2)  
 return U  
  
  
# Задаем сетку  
def init\_grid(I, J, dx, dt, a, U\_x, U\_t):  
 I1 = I  
 U = np.zeros((I1 + 1, J + 1))  
 x = np.zeros(I1 + 1)  
 t = np.zeros(J + 1)  
 for i in range(I + 1):  
 x[i] = dx \* i  
 U[i][0] = U\_x(x[i])  
 if a > 0:  
 for j in range(J + 1):  
 t[j] = dt \* j  
 if j != 0:  
 U[0][j] = U\_t(t[j])  
 elif a < 0:  
 for j in range(J + 1):  
 t[j] = dt \* j  
 if j != 0:  
 U[I][j] = U\_t(t[j])  
  
 return x, t, U  
  
  
# Функция для построения 3D графика  
def plot\_3d(graph, title):  
 graph[0], graph[1] = np.meshgrid(graph[0], graph[1])  
 graph[0], graph[1] = graph[0].T, graph[1].T  
 fig = plt.figure()  
 ax = fig.add\_subplot(projection='3d')  
 ax.set\_title(title)  
 ax.set\_xlabel("x")  
 ax.set\_ylabel("t")  
 ax.set\_zlabel("U")  
 ax.set\_rasterization\_zorder(1)  
 ax.plot\_surface(graph[0], graph[1], graph[2], cmap='viridis')  
 plt.show()  
  
  
# Параметры задачи  
x\_start, x\_end = 0, 1  
t\_start, t\_end = 0, 10  
dx = 0.1  
a1 = 2  
a2 = -2  
  
# Условие устойчивости  
dt = dx / max([abs(a1), abs(a2)])  
  
# Размерности сетки  
I = int((x\_end - x\_start) / dx)  
J = int((t\_end - t\_start) / dt)  
  
# Построение графиков  
x, t, U = init\_grid(I, J, dx, dt, a1, U\_x, U\_t1)  
U = scheme\_1(I, J, dt, U, f, x)  
plot\_3d([np.linspace(x\_start, x\_end, I + 1), np.linspace(t\_start, t\_end, J + 1), U], "Схема 1 (a > 0)")  
  
x, t, U = init\_grid(I, J, dx, dt, a1, U\_x, U\_t1)  
U = scheme\_3(I, J, dt, U, f, x, a1)  
plot\_3d([np.linspace(x\_start, x\_end, I + 1), np.linspace(t\_start, t\_end, J + 1), U], "Схема 3 (a > 0)")  
  
x, t, U = init\_grid(I, J, dx, dt, a1, U\_x, U\_t1)  
U = scheme\_4(I, J, dt, U, f, x, a1, dx)  
plot\_3d([np.linspace(x\_start, x\_end, I + 1), np.linspace(t\_start, t\_end, J + 1), U], "Схема 4 (a > 0)")  
  
x, t, U = init\_grid(I, J, dx, dt, a2, U\_x, U\_t2)  
U = scheme\_2(I, J, dt, U, f, x)  
plot\_3d([np.linspace(x\_start, x\_end, I + 1), np.linspace(t\_start, t\_end, J + 1), U], "Схема 2 (a < 0)")  
  
x, t, U = init\_grid(I, J, dx, dt, a2, U\_x, U\_t2)  
U = scheme\_4(I, J, dt, U, f, x, a2, dx)  
plot\_3d([np.linspace(x\_start, x\_end, I + 1), np.linspace(t\_start, t\_end, J + 1), U], "Схема 4 (a < 0)")