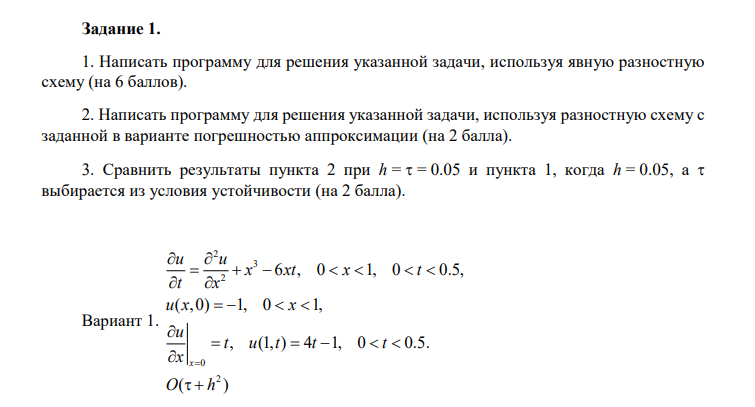
**Постановка задачи:**

****

**Программная реализация:**

import numpy as np  
import matplotlib.pyplot as plt  
from mpl\_toolkits.mplot3d import Axes3D  
  
# Заданные параметры  
L = 1 # Длина промежутка по x  
T = 0.5 # Промежуток времени  
nx = 20 # Количество узлов по x  
nt = 10 # Количество узлов по времени  
hx = L / nx # Шаг по x  
ht = T / nt # Шаг по времени  
  
# Заданные начальные и граничные условия  
u0 = -1  
ut = lambda t: 4 \* t - 1  
ux0 = lambda t: t  
  
# Функция для решения дифференциального уравнения с использованием явной разностной схемы  
def solve\_explicit\_scheme(L, T, nx, nt, u0, ut, ux0):  
 hx = L / nx  
 ht = T / nt  
  
 # Инициализация сетки  
 x\_values = np.linspace(0, L, nx + 1)  
 t\_values = np.linspace(0, T, nt + 1)  
 u = np.zeros((nt + 1, nx + 1))  
  
 # Начальное условие  
 u[0, :] = u0  
  
 # Граничные условия  
 u[:, -1] = ut(t\_values)  
  
 # Разностная схема  
 r = ht / hx \*\* 2  
 for n in range(nt):  
 for i in range(1, nx):  
 u[n + 1, i] = u[n, i] + r \* (u[n, i + 1] - 2 \* u[n, i] + u[n, i - 1]) + ht \* (x\_values[i] \*\* 3 - 6 \* x\_values[i] \* t\_values[n])  
  
 # Граничное условие на левом краю (производная)  
 u[n + 1, 0] = u[n + 1, 1] - hx \* ux0(t\_values[n + 1])  
  
 return x\_values, t\_values, u  
  
# Решение с использованием явной разностной схемы  
x\_values\_explicit, t\_values\_explicit, u\_explicit = solve\_explicit\_scheme(L, T, nx, nt, u0, ut, ux0)  
  
# Визуализация результатов  
X, T = np.meshgrid(x\_values\_explicit, t\_values\_explicit)  
fig = plt.figure()  
ax = fig.add\_subplot(111, projection='3d')  
ax.plot\_surface(X, T, u\_explicit, cmap='viridis')  
ax.set\_xlabel('x')  
ax.set\_ylabel('t')  
ax.set\_zlabel('u')  
ax.set\_title('Solution using Explicit Scheme')  
plt.show()  
  
# Вывод решения в виде набора двумерных графиков-кривых  
fix\_times\_row1 = [0, 0.1, 0.2] # Фиксированные моменты времени для первого ряда  
fix\_times\_row2 = [0.3, 0.4, 0.5] # Фиксированные моменты времени для второго ряда  
  
fig, axs = plt.subplots(2, len(fix\_times\_row1), figsize=(15, 10))  
  
# Первый ряд графиков  
for i, t\_fix in enumerate(fix\_times\_row1):  
 # Находим ближайший временной шаг к заданному моменту времени  
 idx\_time = np.abs(t\_values\_explicit - t\_fix).argmin()  
 t\_plot = t\_values\_explicit[idx\_time]  
 u\_plot = u\_explicit[idx\_time]  
  
 # Построение графика значения функции u(x) в заданный момент времени  
 axs[0, i].plot(x\_values\_explicit, u\_plot, label=f't = {t\_plot}')  
 axs[0, i].set\_xlabel('x')  
 axs[0, i].set\_ylabel('u(x)')  
 axs[0, i].set\_title(f'Solution at t = {t\_plot}')  
 axs[0, i].grid(True)  
 axs[0, i].legend()  
  
# Второй ряд графиков  
for i, t\_fix in enumerate(fix\_times\_row2):  
 # Находим ближайший временной шаг к заданному моменту времени  
 idx\_time = np.abs(t\_values\_explicit - t\_fix).argmin()  
 t\_plot = t\_values\_explicit[idx\_time]  
 u\_plot = u\_explicit[idx\_time]  
  
 # Построение графика значения функции u(x) в заданный момент времени  
 axs[1, i].plot(x\_values\_explicit, u\_plot, label=f't = {t\_plot}')  
 axs[1, i].set\_xlabel('x')  
 axs[1, i].set\_ylabel('u(x)')  
 axs[1, i].set\_title(f'Solution at t = {t\_plot}')  
 axs[1, i].grid(True)  
 axs[1, i].legend()  
  
plt.tight\_layout()  
plt.show()

**Полученные результаты:**

