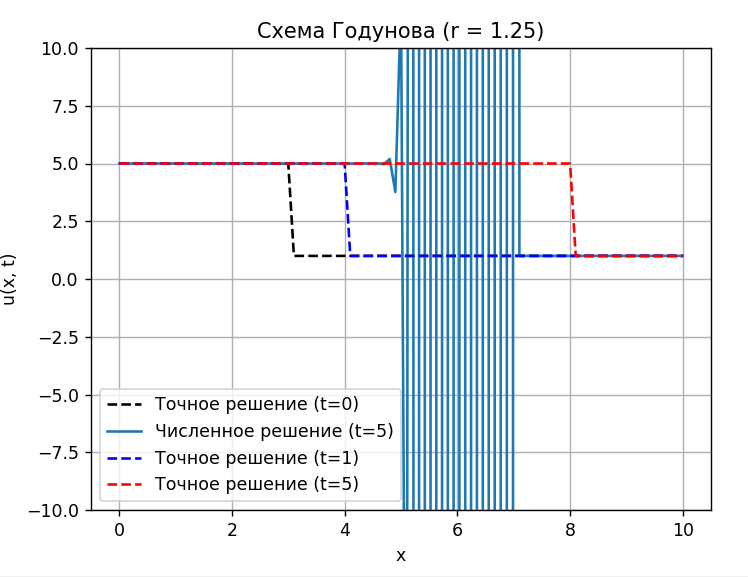
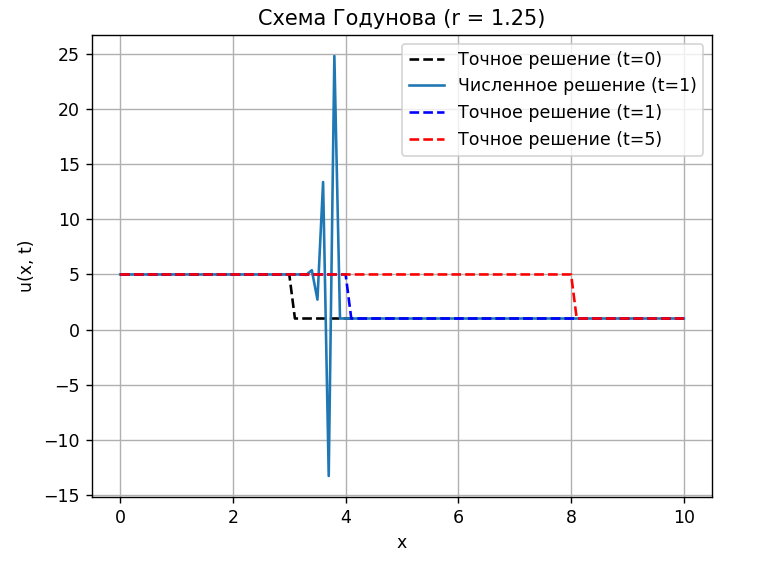
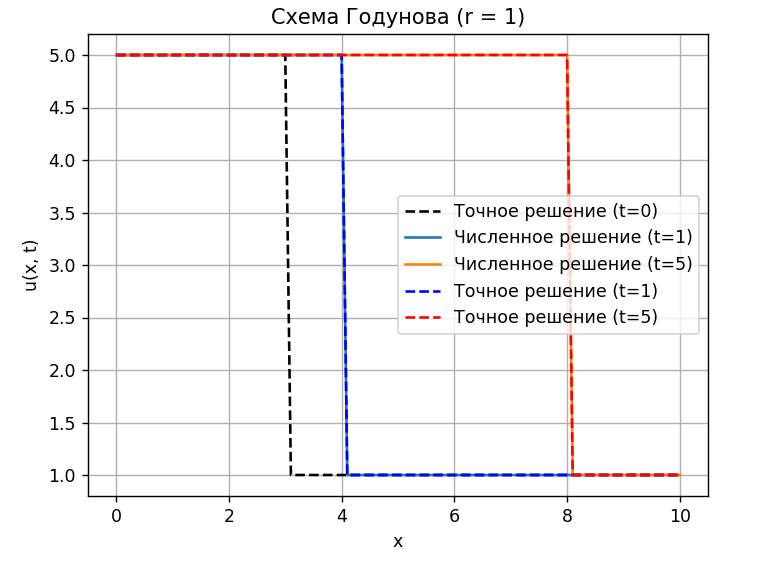
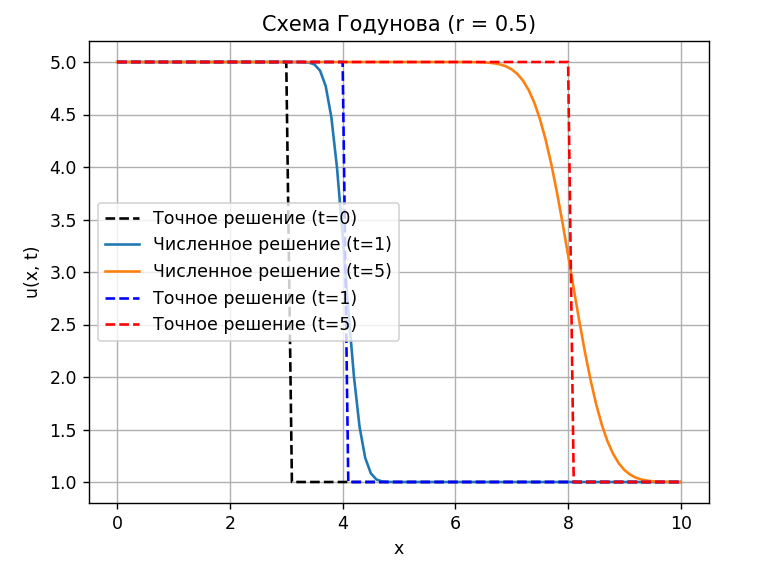
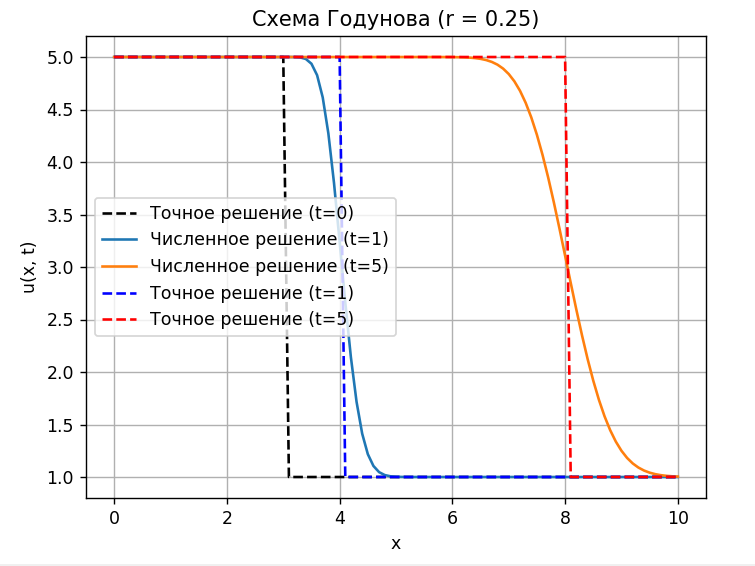
**3 задание (разностные схемы)**

Выполнила: Жуйко Снежана, 22207

1. Численное решение линейного уравнения переноса u\_t+a\*u\_x=0, при a=1 с помощью схемы Годунова.

На семинаре были получены значения числа Курента ( r ), при которых схема Годунова устойчива. При r <= 1 схема может быть устойчива, при r > 1 – неустойчива и её не рекомендуется использовать.

Таким образом, на графиках видно, что чем меньше r ( => меньше tau => больше итераций при вычислении), тем ближе численное решение к точному. Но только для r <= 1. Когда r = 1.25(т.е. схема неустойчива), то график функции разносится по координатной оси, что и ожидалось.

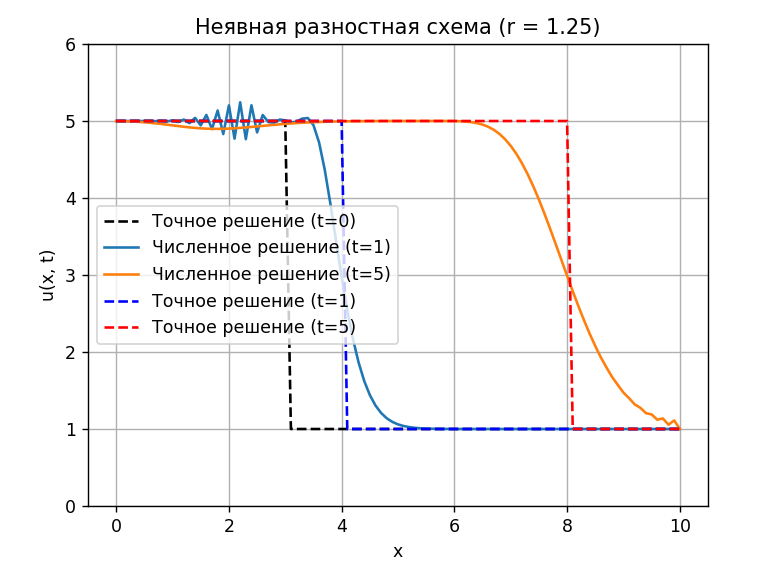
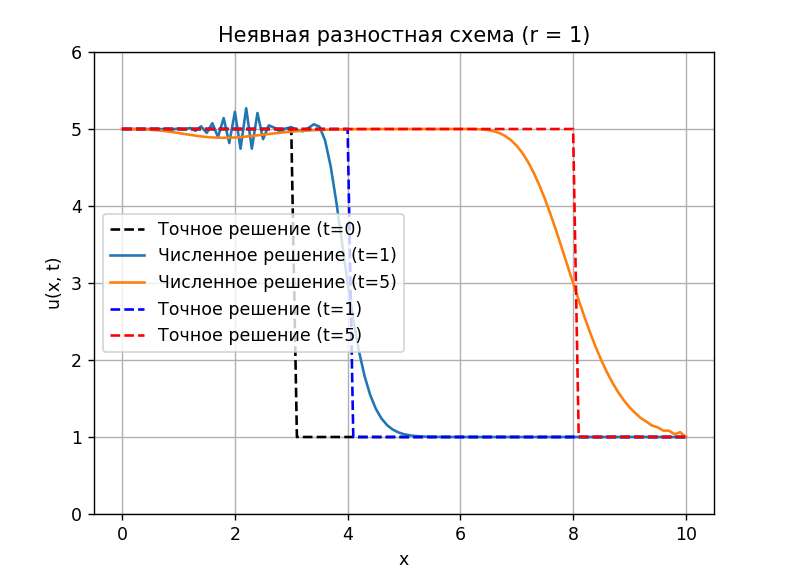
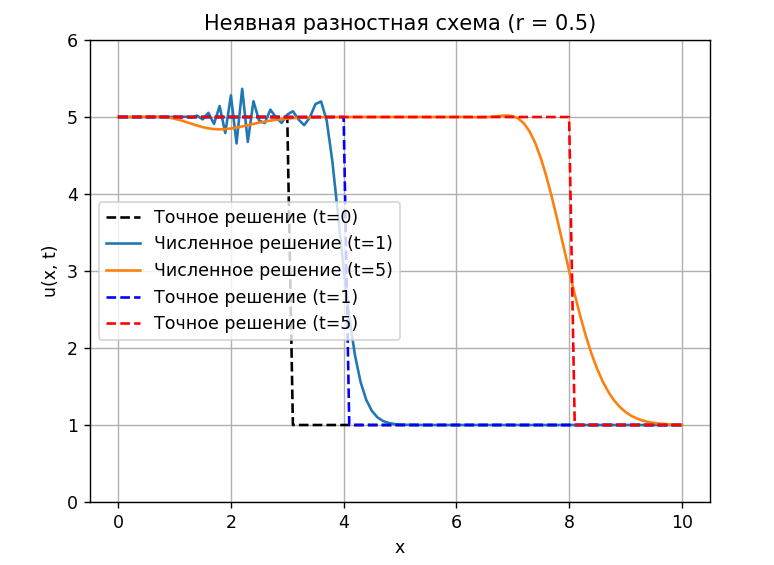
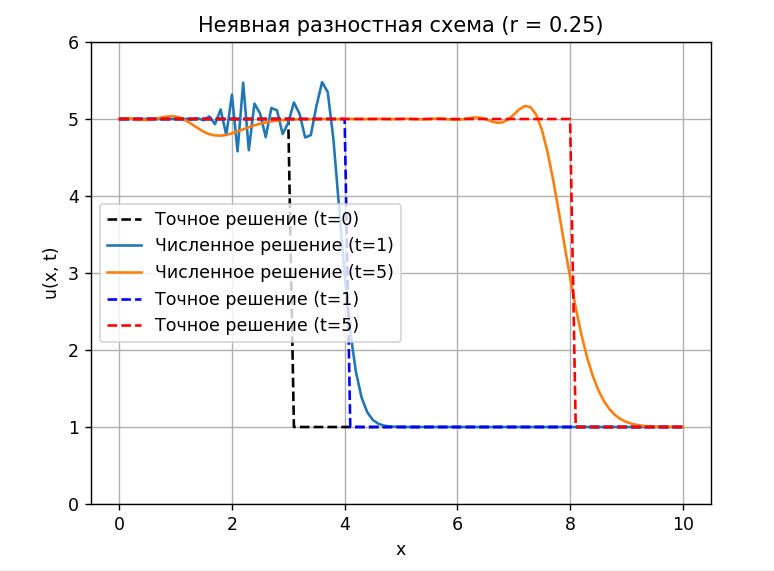
Графики для схемы Годунова: 

1. Численное решение линейного уравнения переноса u\_t+a\*u\_x=0, при a=1 с помощью неявной симметричной схемы.

На семинарах было выяснено, что данная схема является безусловно устойчивой.

Так, на графиках видно, что чем меньше число Курента ( => меньше tau => больше итераций при вычислении), тем ближе численное решение к точному, как и ожидалось.

Графики для неявной схемы:



**Код программы:**

import numpy as np

import matplotlib.pyplot as plt

L = 10

h = 0.1 # Шаг по пространству

x = np.arange(0, L + h, h)

N = len(x) # Количество узлов по пространству

def init(x):

u = np.zeros\_like(x)

for i in range(0, len(x)):

if x[i] <= 3.0:

u[i] = 5.0

else:

u[i] = 1.0

return u

def find\_solution(x, t):

u = np.zeros\_like(x)

for i in range(0, len(x)):

if x[i] - t <= 3.0:

u[i] = 5.0

else:

u[i] = 1.0

return u

def godunov\_scheme(u, r):

next = np.zeros\_like(u)

next[0] = u[0]

for i in range(1, N):

next[i] = u[i] \* (1 - r) + u[i - 1] \* r

return next

# Метод прогонки

def tridiagonal\_solve(a, b, c, d):

# a - нижняя диагональ, b -главная диагональ, c - верхняя диагональ.

# d - правая часть

n = len(d)

alpha = np.zeros(n)

beta = np.zeros(n)

x = np.zeros(n)

# Прямой ход

beta[0] = d[0] / b[0]

alpha[0] = -c[0] / b[0]

for i in range(1, n - 1):

tmp = b[i] + a[i] \* alpha[i - 1]

alpha[i] = -c[i] / tmp

beta[i] = (d[i] - a[i] \* beta[i - 1]) / tmp

tmp = b[-1] + a[-1] \* alpha[-2]

beta[-1] = (d[-1] - a[-1] \* beta[-2]) / tmp

# Обратный ход

x[-1] = beta[-1]

for i in range(n - 2, -1, -1):

x[i] = alpha[i] \* x[i + 1] + beta[i]

return x

def scheme(u, r):

n = len(u)

a = np.full(n, -r/2)

b = np.full(n, 1.0)

c = np.full(n, r/2)

a[0] = 0

c[-1] = 0

c[0] = 0

a[-1] = 0

next = tridiagonal\_solve(a, b, c, u)

return next

r\_array = [0.25, 0.5, 1, 1.25]

tay\_array = [r \* h for r in r\_array]

time\_points\_array = [1, 5]

for r, tay in zip(r\_array, tay\_array):

plt.figure()

plt.plot(x, init(x), 'k--', label='Точное решение (t=0)')

for time\_point in time\_points\_array:

u = init(x)

iters = int(time\_point / tay)

for i in range(iters):

# u = godunov\_scheme(u, r)

u = scheme(u, r)

plt.plot(x, u, label=f'Численное решение (t={time\_point})')

plt.plot(x, find\_solution(x, 1), 'b--', label=f'Точное решение (t=1)')

plt.plot(x, find\_solution(x, 5), 'r--', label=f'Точное решение (t=5)')

# plt.title(f'Схема Годунова (r = {r})')

plt.title(f'Неявная разностная схема (r = {r})')

plt.xlabel('x')

plt.ylabel('u(x, t)')

plt.ylim(0, 6)

plt.legend()

plt.grid()

plt.show()

Объяснение!!!

