Дан тонкий однородный стержень длиной l, на концах которого поддерживается нулевая температура. Начальное распределение температуры стержня описывается функцией $f(x)$. Найти распределение температуры вдоль стержня при $t > 0$. Найти решение в случае, когда

$$f(x) = \begin{cases} x, & 0 < x \le \frac{l}{2}, \\ l - x, & \frac{l}{2} < x < l. \end{cases}$$

Был выбран метод конечных разностей. Взяв информацию из лекции (формулы) и реализовав всё на языке программирования Python, проанализируем результаты.

Для начала приведём код программы:

//ПРИМЕЧАНИЕ//

Обозначения в коде:

Слева – код, справа – из лекции

ui0 = u(i,0) – массив

uij = u(i,j) – массив

uij_next = u(i, j+1) – массив

uij[i+1] = u(i+1,j) – элемент из массива uij

uxt = u(x,t)

uij_previous = u(i, j-1) – массив

и т.д.

//ПРИМЕЧАНИЕ//

```
# Метод конечных разностей
import math

# для примера и сравнения Аналитического решения и Численного,
# возьмём любые значения, так чтобы r <= 1/2, где r = ((alpha**2)*k)/h**2
l = 1
n = 10
h = l/n
b = 0.5
m = 10
k = b/m
alpha = 0.05
r = ((alpha**2)*k)/h**2
print("r = ",r)

# Из условия задачи - начальное распределение температуры:
def f1(xi):
    return xi

def f2(xi):
    return (l - xi)

# заполняем температуру в n узлах
ui0 = [f1(i*h) if (i*h <= l/2) else f2(i*h) for i in range(n+1)] # начальная
температура (начальное условие)
```

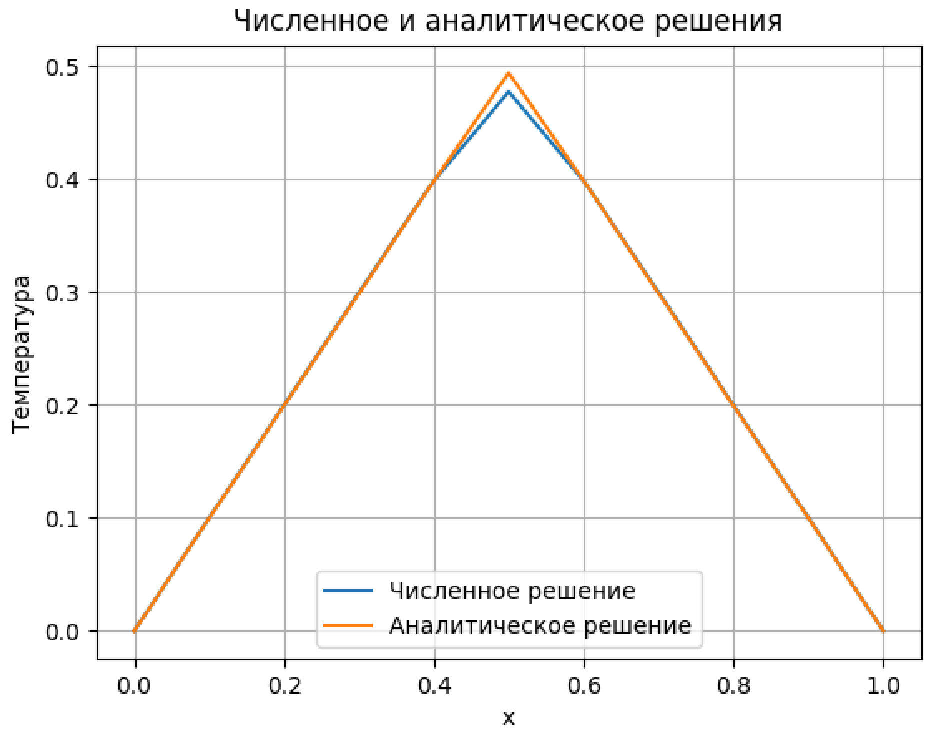
```
uij = ui0 # текущая температура
uij_next = [0] * (n+1) # следующая температура, пока заполняем 0, т.е. [0, 0, 0, ..., 0]

# Вычисления в узлах:
for j in range(1,m+1):
    for i in range(1,n):
        uij_next[i] = (1 - 2*r)*uij[i] + r*(uij[i+1] + uij[i-1])
    uij = uij_next

# Аналитическое решение из тетради, при тех же данных
def analitic(x, t):
    uxt = 0.0
    for m in range(1,1000,2):
        uxt += (((4*math.sin(math.pi*m/2))/(math.pi*m)**2) * math.sin(math.pi * m * x / l) * math.exp(-(alpha**2)*math.pi*m*t/l))
    return uxt

analitic_solution = []
for i in range(n+1):
    xi = i*h
    analitic_solution.append(analitic(xi, b))

for el1,el2 in zip(uij,analitic_solution):
    print(el1, '    ', el2)

# Графическая визуализация
import matplotlib.pyplot as plt
plt.plot([i*h for i in range(n+1)], uij, label='Численное решение')
plt.plot([i*h for i in range(n+1)], analitic_solution, label='Аналитическое решение')
plt.xlabel('x')
plt.ylabel('Температура')
plt.title('Численное и аналитическое решения')
plt.legend()
plt.grid(True)
plt.show()
```

При запуске на данных: l = 1 (длина стержня), b = 0.5 (время моделирования), n = m = 10 (размерность сетки). Вычислив шаги и запустив итерационный процесс для численного вычисления, затем вычислив аналитическое решение в числовом конечном виде (т.к. в тетради оно в общем виде), получаем следующие результаты:

```
r =  0.0125
0     0.0
0.09999998614081113     0.09974578388513343
0.19999908992980098     0.1994634304391886
0.2999588256354392     0.29910541292996057
0.3987653684421055     0.3985336221648433
0.47752473200232326     0.49424343148952143
0.39853966066019253     0.3985336221648433
0.2999303279287457     0.29910541292996046
0.19999729279336323     0.19946343043918863
0.09999990932621251     0.0997457838851335
0     4.306762092793323e-19
```

Левый столбец – численное решение, правый – аналитическое решение, r ≤ ½ - выполняется. Таким образом, результаты различаются несильно, на погрешность влияет – количество итераций в аналитическом решении и размерности сетки подобранные для численного решения.

Также в конце был сделан график, с использованием модуля matplotlib:



Дана струна, закрепленная на концах $x = 0, x = 10$. В начальный момент струна имеет форму параболы $u(x,0) = x^2 - 10x$. Определить смещение точек струны от положения равновесия, если их начальные скорости отсутствуют. Уравнение колебаний имеет вид:

$$u_{tt} = 25 \cdot u_{xx} + 7 \cdot \sin t \cdot \sin \frac{3\pi x}{10}.$$

Был также выбран метод конечных разностей. Взяв информацию из лекции (формулы) и реализовав всё на языке программирования Python, проанализируем результаты.

Для начала приведём код программы:

```
# Метод конечных разностей
import math

# для примера и сравнения Аналитического решения и Численного,
# возьмём любые значения, так чтобы r <= 1, где r = (alpha*k)/h
x_start = 0
x_end = 10
l = abs(x_end - x_start)
n = 100
h = l/n
b = 5
m = 10000
k = b/m
alpha = 5
r = (alpha*k)/h
print("r = ",r)

# Функция p(x, t), т.к. колебания у нас вынужденные
def p(x, t):
    return 7*math.sin(t)*math.sin(math.pi*3*x/10)

# заполняем смещение в n узлах
ui0 = [((h*i)**2-10*(h*i)) for i in range(n+1)] # начальное смещение - 
парабола (начальное условие)
ui1 = ui0.copy() # u(i,1) = u(i,0) + k*g(x), g(x) = ut в т.t=0, но из условия 
задачи скорость равна 0 изначально
uij = ui1.copy() # текущее смещение
uij_next = [0] * (n+1) # следующее смещение, пока заполняем 0, т.е. [0, 0, 0, 
..., 0]
uij_previous = ui0.copy() # предыдущее примем для начала перед итерациями 
равным u(i,0)

# Вычисления в узлах:
for j in range(2,m+1):
    for i in range(1,n):
        uij_next[i] = 2*(1 - r**2)*uij[i] + (r**2)*(uij[i+1] + uij[i-1]) - 
uij_previous[i] + (k**2)*p(i*h, j*k)
    uij_previous = uij.copy()
    uij = uij_next.copy()

# Аналитическое решение из тетради, при тех же данных
def analitic(x, t):
    uxt = 0.0
    for m in range(1,10000,2):
        uxt += 
((1/((2*m+1)**3))*(math.cos(math.pi*(2*m+1)*t/2))*(math.sin(math.pi*(2*m+1)*x
```

```
/10)))
    return ((-800/math.pi**3)*uxt + (28/(9*math.pi**2-
4))*(math.sin(3*math.pi*x/10))*(math.sin(t) -
(2/(3*math.pi))*math.sin(3*math.pi*t/2)))

analitic_solution = []
for i in range(n+1):
    xi = i*h
    analitic_solution.append(analitic(xi, b))

for el1,el2 in zip(uij,analitic_solution):
    print(el1, '    ', el2)

# Графическая визуализация
import matplotlib.pyplot as plt
plt.plot([i*h for i in range(n+1)], uij, label='Численное решение')
plt.plot([i*h for i in range(n+1)], analitic_solution, label='Аналитическое
решение')
plt.xlabel('x')
plt.ylabel('Смещение')
plt.title('Численное и аналитическое решения')
plt.legend()
plt.grid(True)
plt.show()
```

При запуске на данных: x_start = 0, x_end = 10 (концы закрепления), b = 5 (время моделирования), n = 100, m = 10000 (размерность сетки), alpha = 5 (известно). Вычислив шаги и запустив итерационный процесс для численного вычисления, затем вычислив аналитическое решение в числовом конечном виде (т.к. в тетради оно в общем виде), получаем следующие результаты:

```
r =  0.024999999999999998
0     -0.0
-0.022356179226563692     -0.023195899511714337
-0.04624727148723652     -0.04618591057723599
-0.06997251066443756     -0.06876597223079628
-0.0926734909863794     -0.09073566224662875
-0.11348234012970769     -0.11189997610083428
-0.1327837175324608     -0.1320710578453398
-0.15237403830155083     -0.1510698675305927
-0.17086762418037904     -0.1687277703768281
-0.1875691665223862     -0.18488803358831205
-0.2028123966748279     -0.19940721752473228
-0.21519357186344779     -0.21215644888160023
-0.22545226363126142     -0.22302256457882555
-0.23427207585195509     -0.2319091162042254
-0.24177934683810196     -0.23873722609645967
-0.24863053188470233     -0.24344628746873673
```

Левый столбец – численное решение, правый – аналитическое решение, r ≤ 1 - выполняется. Таким образом, результаты различаются несильно, на погрешность влияет –

количество итераций в аналитическом решении и размерности сетки подобранные для численного решения.

Также в конце был сделан график, с использованием модуля matplotlib: