Правительство Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования "Национальный исследовательский университет "Высшая школа экономики"

Московский институт электроники и математики Национального исследовательского университета "Высшая школа экономики"
Департамент прикладной математики

ОТЧЕТ

по Лабораторной работе №7
По курсу «Численные методы»

ЧИСЛЕННОЕ РЕШЕНИЕ СТАЦИОНАРНЫХ И НЕСТАЦИОНАРНЫХ ЗАДАЧ ТЕПЛОПРОВОДНОСТИ

ФИО студента	Номер группы	Вариант 15	Дата
Пугач Виктория Павловна	БПМ-211	10.1.15, 10.2.8, 10.4.15, 10.5.15, 10.6.15	09.10.2024

Москва – 2024 г.

Оглавление

Задача 10.1.15

Задача 10.1. Промоделировать стационарные процессы теплопроводности стержня в зависимости от входных данных задачи:

$$\begin{cases} -\dfrac{d}{dx}\left(K(x)\dfrac{du}{dx}\right) = f, \\ u(a) = UA, \quad u(b) = UB. \end{cases}$$

ПОРЯДОК РЕШЕНИЯ ЗАДАЧИ:

1. Представить коэффициент теплопроводности *K(x)* в виде функции двух переменных *x* и *c*: *K(x)=K(x,c),* где *c* - параметр.
2. При заданных в индивидуальном варианте функциях *k(x)* (что соответ-ствует *K(x,*1)), *f(x)* и значениях *UA, UB* найти аналитическое решение задачи символьно (см. *ПРИЛОЖЕНИЯ 10.B* и *10.C*).
3. Изменяя значения параметра *c* в коэффициенте теплопроводности, найти решения задачи для наборов параметров 1-3 (см. таблицу ниже).
4. На одном чертеже построить графики найденных решений. Сравнить полученные результаты.
5. Аналогично п.2, найти аналитическое решение для набора параметров 4. На одном чертеже построить графики решений для наборов 1 и 4. Сравнить полученные результаты.
6. Изменяя граничные условия *UA, UB*, построить решения для наборов параметров 5-7.

Таблица наборов параметров

Параметры	1 набор	2 набор	3 набор	4 набор	5 набор	6 набор	7 набор
c	1	10	0.1	1	1	1	1
K(x)	*k(x)*	*ck(x)*	*ck(x)*	*1/k(x)*	*k(x)*	*k(x)*	*k(x)*
UA	*ua*	*ua*	*ua*	*ua*	*-ua*	*ua*	*-ua*
UB	*ub*	*ub*	*ub*	*ub*	*ub*	*-ub*	*-ub*

Пункт 1-4

№	*k(x)*	*f(x)*	*a*	*UA*	*b*	*UB*
10.1.15	$x^{-1/3}$	$x+\sqrt{x}$	1.5	3	2.5	-3

In[1]:= $f[x\_] := x + \sqrt{x}$

In[2]:= $k[x\_] := x^{-\frac{1}{3}}$

№10.1.15

$$\begin{cases} -\frac{d}{dx}\left(K(x)\cdot\frac{du}{dx}\right)=f \\ u|_a = UA \qquad a=1{,}5 \quad UA=3 \\ u|_b = UB \qquad b=2{,}5 \quad UB=-3 \end{cases}$$

$f(x)=x+\sqrt{x}$

$k(x)=x^{-\frac{1}{3}}$

$K(x)\cdot u'_x = -\int f\,dx = -\frac{2x^{\frac{3}{2}}}{3} - \frac{x^2}{2} + C_1$

In[47]:= $-\int\left(x+\sqrt{x}\right)dx$

Out[47]= $-\frac{2x^{3/2}}{3} - \frac{x^2}{2}$

$u = \int \frac{1}{K(x)}\cdot\left[-\frac{2x^{\frac{3}{2}}}{3} - \frac{x^2}{2} + C_1\right]dx$

Решения для наборов 1-3

1) $K(x) = k(x)$

In[48]:= $$\int\left(\frac{-\frac{2x^{3/2}}{3}-\frac{x^2}{2}+c1}{k[x]}\right)dx$$

Out[48]= $$\frac{1}{340}\left(255\,c1\,x^{4/3}-80\,x^{17/6}-51\,x^{10/3}\right)+C_2$$

$$\Rightarrow \quad U|_{x=1,5}=3 \qquad U|_{x=2,5}=-3$$

In[51]:= $$\frac{1}{340}\left(255\,c1\,x^{4/3}-80\,x^{17/6}-51\,x^{10/3}\right)+c2\ /.\ x\to 1.5$$

Out[51]= $$c2+\frac{1}{340}(-449.391+437.853\,c1)$$

In[52]:= $$\frac{1}{340}\left(255\,c1\,x^{4/3}-80\,x^{17/6}-51\,x^{10/3}\right)+c2\ /.\ x\to 2.5$$

Out[52]= $$c2+\frac{1}{340}(-2154.49+865.221\,c1)$$

In[57]:= Solve[
решить уравнения

$$\left\{c2+\frac{1}{340}(-449.39086059919043`+437.85319777664944`\,c1)==3,\ c2+\frac{1}{340}(-2154.4935427777295`+865.2206152896264`\,c1)==-3\right\},$$
{c1, c2}]

Out[57]= {{c1 → −0.783629, c2 → 5.3309}}

In[59]:= $$\frac{1}{340}\left(255\,c1\,x^{4/3}-80\,x^{17/6}-51\,x^{10/3}\right)+c2\ /.$$
{c1 → −0.783628568996586`, c2 → 5.330897457069074`}

Out[59]= $$5.3309+\frac{1}{340}\left(-199.825\,x^{4/3}-80\,x^{17/6}-51\,x^{10/3}\right)$$

2) $K(x) = c\cdot k(x) = 10\cdot k(x)$

In[49]:= $$\int\left(\frac{-\frac{2x^{3/2}}{3}-\frac{x^2}{2}+c1}{10\,k[x]}\right)dx$$

Out[49]= $$\frac{255\,c1\,x^{4/3}-80\,x^{17/6}-51\,x^{10/3}}{3400}+C_2$$

$$\Rightarrow$$

In[60]:= $$\frac{255\,c1\,x^{4/3}-80\,x^{17/6}-51\,x^{10/3}}{3400}+c2\ /.\ x\to 1.5$$

Out[60]= $$\frac{-449.391+437.853\,c1}{3400}+c2$$

In[61]:= $$\frac{255\,c1\,x^{4/3}-80\,x^{17/6}-51\,x^{10/3}}{3400}+c2\ /.\ x\to 2.5$$

Out[61]= $$\frac{-2154.49+865.221\,c1}{3400}+c2$$

In[62]:= Solve[
решить уравнения
$$\left\{\frac{-449.39086059919043`+437.85319777664944`\,c1}{3400}+c2==3,\ \frac{-2154.4935427777295`+865.2206152896264`\,c1}{3400}+c2==-3\right\},$$
{c1, c2}]

Out[62]= {{c1 → −43.7443, c2 → 8.76558}}

In[63]:= $$\frac{255\,c1\,x^{4/3}-80\,x^{17/6}-51\,x^{10/3}}{3400}+c2\ /.$$
{{c1 → −43.74432058160774`, c2 → 8.76558279759501`}}

Out[63]= $$\left\{8.76558+\frac{-11154.8\,x^{4/3}-80\,x^{17/6}-51\,x^{10/3}}{3400}\right\}$$

3) $K(x) = c\cdot k(x) = 0,1\cdot k(x)$

In[50]:= $$\int\left(\frac{-\frac{2x^{3/2}}{3}-\frac{x^2}{2}+c1}{0.1\,k[x]}\right)dx$$

Out[50]= $$-5.\,x^{4/3}\left(-1.5\,c1+0.470588\,x^{3/2}+0.3\,x^2\right)+C_2$$

$$\Rightarrow$$

$$-5.`\,x^{4/3}\left(-1.5`\,c1+0.47058823529411764`\,x^{3/2}+0.3`\,x^2\right)+c2\ /.\ x\to 1.5$$

Out[64]= $$-8.58536\,(1.53953-1.5\,c1)+c2$$

In[65]:= $$-5.`\,x^{4/3}\left(-1.5`\,c1+0.47058823529411764`\,x^{3/2}+0.3`\,x^2\right)+c2\ /.\ x\to 2.5$$

Out[65]= $$-16.9651\,(3.73516-1.5\,c1)+c2$$

In[66]:= Solve[
решить уравнения
{−8.585356819149988` (1.5395257915705334` − 1.5` c1) + c2 == 3,
−16.965110103718164` (3.7351633295108115` − 1.5` c1) + c2 ==
−3}, {c1, c2}]

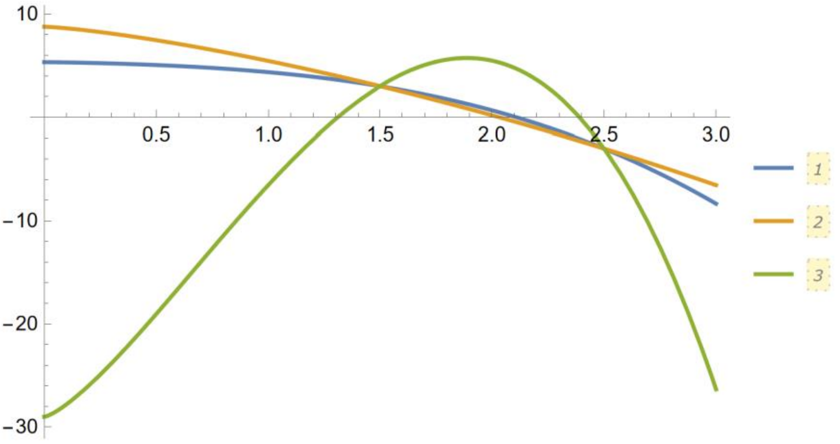
Out[66]= {{c1 → 3.51244, c2 → −29.016}}

In[67]:= $$-5.`\,x^{4/3}\left(-1.5`\,c1+0.47058823529411764`\,x^{3/2}+0.3`\,x^2\right)+c2\ /.$$
{c1 → 3.5124406322645263`, c2 → −29.015955948190253`}

Out[67]= $$-29.016-5.\,x^{4/3}\left(-5.26866+0.470588\,x^{3/2}+0.3\,x^2\right)$$

```
Plot[{f1[x], f2[x], f3[x]}, {x, 0, 3},
 PlotLegends → Automatic]
```
график функции
легенды графика автоматический

Out[80]=



Пункт 5

$$\int \left(\left(-\frac{2\,x^{3/2}}{3} - \frac{x^2}{2} + c1\right) k[x]\right) dx$$

Out[70]= $\frac{1}{208}\left(312\,c1\,x^{2/3} - 64\,x^{13/6} - 39\,x^{8/3}\right)$

In[71]:= $\frac{1}{208}\left(312\,c1\,x^{2/3} - 64\,x^{13/6} - 39\,x^{8/3}\right) + c2\ /.\ x \to 1.5$

Out[71]= $\frac{1}{208}\left(-269.053 + 408.836\,c1\right) + c2$

In[72]:= $\frac{1}{208}\left(312\,c1\,x^{2/3} - 64\,x^{13/6} - 39\,x^{8/3}\right) + c2\ /.\ x \to 2.5$

Out[72]= $\frac{1}{208}\left(-914.989 + 574.709\,c1\right) + c2$

In[73]:= $\text{Solve}\left[\left\{\frac{1}{208}\left(-269.05252859736277\text{`} + 408.8356574965878\text{`}\ c1\right) + c2 == 3,\right.\right.$
решить уравнения

$\left.\left.\frac{1}{208}\left(-914.9885591970822\text{`} + 574.7089137879003\text{`}\ c1\right) + c2 == -3\right\},\ \{c1, c2\}\right]$

Out[73]= $\{\{c1 \to -3.62966,\ c2 \to 11.4278\}\}$

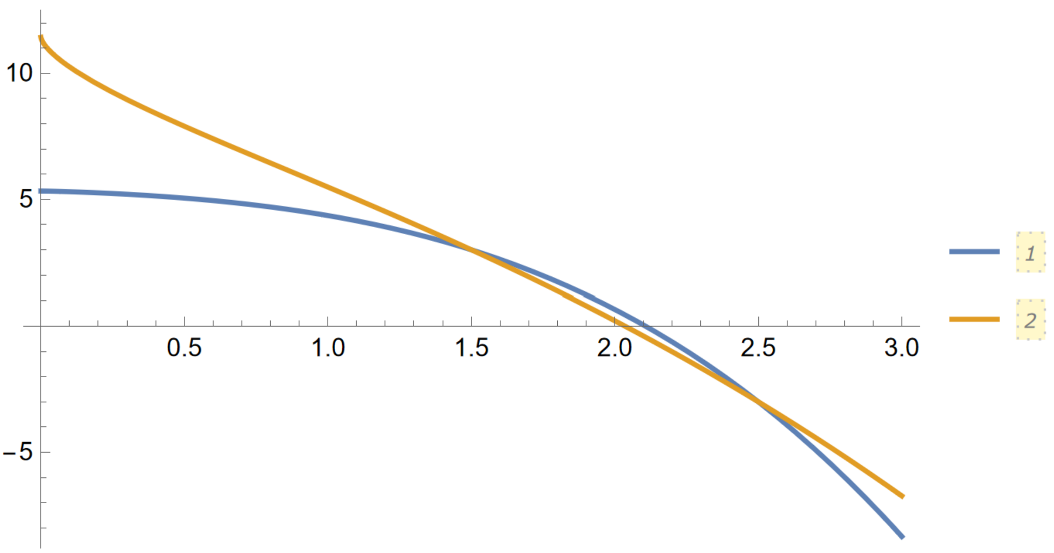
In[74]:= $\frac{1}{208}\left(312\,c1\,x^{2/3} - 64\,x^{13/6} - 39\,x^{8/3}\right) + c2\ /.\ \{c1 \to -3.6296626886187995\text{`},\ c2 \to 11.427827213411838\text{`}\}$

Out[74]= $11.4278 + \frac{1}{208}\left(-1132.45\,x^{2/3} - 64\,x^{13/6} - 39\,x^{8/3}\right)$

In[81]:=

```
Plot[{f1[x], f4[x]}, {x, 0, 3}, PlotLegends → Automatic]
```
график функции легенды графика автоматически

Out[81]=



Пункт 6

У нас уже есть решение для первого пункта. Найдем константы.

In[48]:= $\int \left(\frac{\left(-\frac{2 x^{3/2}}{3} - \frac{x^2}{2} + c1\right)}{k[x]} \right) dx$

Out[48]= $\frac{1}{340} \left(255\, c1\, x^{4/3} - 80\, x^{17/6} - 51\, x^{10/3}\right)$

Для -ua, ub

In[82]:= $\frac{1}{340} \left(255\, c1\, x^{4/3} - 80\, x^{17/6} - 51\, x^{10/3}\right) + c2$ /. x → 1.5

Out[82]= $\frac{1}{340} (-449.391 + 437.853\, c1) + c2$

In[83]:= $\frac{1}{340} \left(255\, c1\, x^{4/3} - 80\, x^{17/6} - 51\, x^{10/3}\right) + c2$ /. x → 2.5

Out[83]= $\frac{1}{340} (-2154.49 + 865.221\, c1) + c2$

In[84]:= Solve[{$\frac{1}{340}$ (-449.39086059919043\` + 437.85319777664944\` c1) + c2 == -3,
решить уравнения
$\frac{1}{340}$ (-2154.4935427777295\` + 865.2206152896264\` c1) + c2 == -3}, {c1, c2}]

Out[84]= {{c1 → 3.98978, c2 → -6.81632}}

In[86]:= $\frac{1}{340} \left(255\, c1\, x^{4/3} - 80\, x^{17/6} - 51\, x^{10/3}\right) + c2$ /. {c1 → 3.9897816546268743\`, c2 → -6.816317045028817\`}

Out[86]= $-6.81632 + \frac{1}{340} \left(1017.39\, x^{4/3} - 80\, x^{17/6} - 51\, x^{10/3}\right)$

In[87]:= f5[x_] := $-6.816317045028817` + \frac{1}{340} \left(1017.3943219298529`\, x^{4/3} - 80\, x^{17/6} - 51\, x^{10/3}\right)$

Для ua, -ub

In[88]:= Solve[{$\frac{1}{340}$ (-449.39086059919043\` + 437.85319777664944\` c1) + c2 == 3,
решить уравнения
$\frac{1}{340}$ (-2154.4935427777295\` + 865.2206152896264\` c1) + c2 == 3}, {c1, c2}]

Out[88]= {{c1 → 3.98978, c2 → -0.816317}}

In[89]:= $\frac{1}{340} \left(255\, c1\, x^{4/3} - 80\, x^{17/6} - 51\, x^{10/3}\right) + c2$ /. {c1 → 3.9897816546268743\`, c2 → -0.8163170450288177\`}

Out[89]= $-0.816317 + \frac{1}{340} \left(1017.39\, x^{4/3} - 80\, x^{17/6} - 51\, x^{10/3}\right)$

In[90]:= f6[x_] := $-0.8163170450288177` + \frac{1}{340} \left(1017.3943219298529`\, x^{4/3} - 80\, x^{17/6} - 51\, x^{10/3}\right)$

Для -ua, -ub

In[91]:= Solve[{$\frac{1}{340}$ (-449.39086059919043\` + 437.85319777664944\` c1) + c2 == -3,
решить уравнения
$\frac{1}{340}$ (-2154.4935427777295\` + 865.2206152896264\` c1) + c2 == 3}, {c1, c2}]

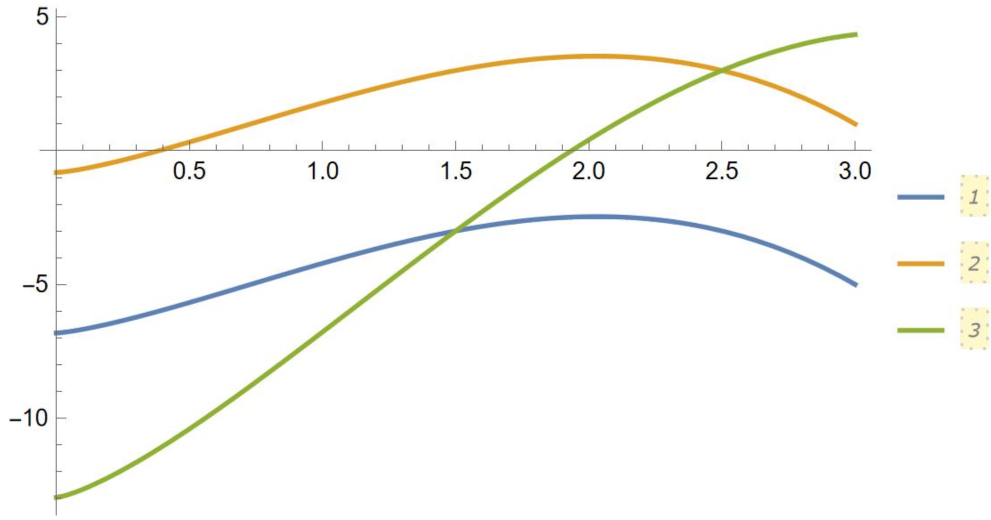
Out[91]= {{c1 → 8.76319, c2 → -12.9635}}

In[92]:= $\frac{1}{340} \left(255\, c1\, x^{4/3} - 80\, x^{17/6} - 51\, x^{10/3}\right) + c2$ /. {c1 → 8.763191878250336\`, c2 → -12.963531547126708\`}

Out[92]= $-12.9635 + \frac{1}{340} \left(2234.61\, x^{4/3} - 80\, x^{17/6} - 51\, x^{10/3}\right)$

In[93]:= f7[x_] := $-12.963531547126708` + \frac{1}{340} \left(2234.6139289538355`\, x^{4/3} - 80\, x^{17/6} - 51\, x^{10/3}\right)$

In[94]:= `Plot[{f5[x], f6[x], f7[x]}, {x, 0, 3}, PlotLegends → Automatic]`
график функции　　　　легенды графика　автоматически

Out[94]=


Задача 10.2.8

Задача 10.2. Найти приближенное решение краевой задачи методом конечных разностей:

$$\begin{cases} u'' + p(x)u' + q(x)u = f(x), \quad x \in (a,b), \\ u(a) = UA, \quad u(b) = UB. \end{cases}$$

с заданной точностью ε и построить его график.

ПОРЯДОК РЕШЕНИЯ ЗАДАЧИ:

1. Составить разностную схему второго порядка точности и выписать коэффициенты матрицы системы уравнений и коэффициенты правой части.
2. Подготовить тестовый пример и провести расчет для него. Построить на одном чертеже графики приближенного и точного решений для тестового примера. После проверки правильности работы программы перейти к решению основной задачи.
3. Для вычисления решения задачи с заданной точностью произвести расчет с начальным шагом h, затем уменьшить шаг вдвое. Вывести на экран два соседних приближенных решения и сравнить результаты. Если заданная точность не достигнута, то продолжить уменьшение шага.
4. Построить график найденного решения и указать шаг, при котором заданная точность достигается.

№	p(x)	q(x)	f(x)	a	b	UA	UB	ε
10.2.8	$1+\cos^2(x)$	x^2+1	$(x^2+1)\cos(x)$	1	3	–1	4	0.05

Пункт 1

```
# Метод конечных разностей
def finite_difference_method(a, b, UA, UB, h):
    N = int((b - a) / h)  # Число точек
    x = np.linspace(a, b, N+1)  # Сетка по x
    A = np.zeros((N+1, N+1))  # Матрица коэффициентов
    B = np.zeros(N+1)  # Вектор правой части

    # Граничные условия
    A[0, 0] = 1
    B[0] = UA
    A[N, N] = 1
    B[N] = UB

    # Заполнение системы уравнений
    for i in range(1, N):
        xi = x[i]
        A[i, i-1] = 1/h**2 - pt(xi) / (2*h)
        A[i, i] = -2/h**2 + qt(xi)
        A[i, i+1] = 1/h**2 + pt(xi) / (2*h)
        B[i] = ft(xi)

    # Решение системы
    U = np.linalg.solve(A, B)

    return x, U
```

Пункт 2

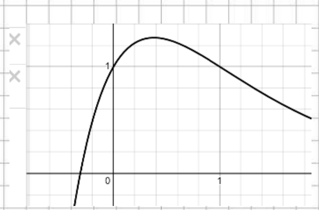
Тестовая задача и её аналитическое решение

№10.2.8

тестовый пример:

$$\begin{cases} u'' + p(x)u' + q(x)u = f(x), \; x \in (a,b) \\ u|_{x=a} = UA \\ u|_{x=b} = UB \end{cases}$$

$$\left.\begin{aligned} p(x) &= 3 \\ q(x) &= 2 \\ f(x) &= 0 \\ a &= 0 \\ b &= 1 \\ UA, UB &= 1 \end{aligned}\right\} \Rightarrow \begin{aligned} & u'' + 3u' + 2u = 0 \\ & \lambda^2 + 3\lambda + 2 = 0 \\ & \lambda_1 = -1 \\ & \lambda_2 = -2 \\ & u = C_1 \cdot e^{-x} + C_2 \cdot e^{-2x} \end{aligned}$$



$$\begin{aligned} u(0)&: C_1 + C_2 = 1 \\ u(1)&: \frac{C_1}{e} + \frac{C_2}{e^2} = 1 \end{aligned} \Bigg\} \begin{aligned} & C_1 = 1 - C_2 \\ & \frac{1 - C_2}{e} + \frac{C_2}{e^2} = 1 \end{aligned} \Rightarrow e - C_2 \cdot e + C_2 = e^2$$

$$C_2(1 - e) = e^2 - e$$

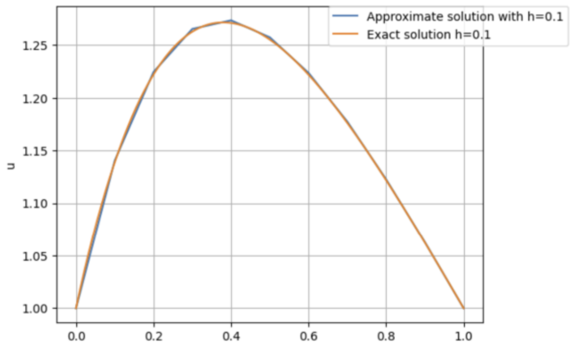
$$C_2 = \frac{e^2 - e}{1 - e} = \frac{e(e-1)}{1-e} = -e$$

$$C_1 = 1 - C_2 = 1 + e$$

Ответ: $u(x) = (1+e)e^{-x} - e \cdot e^{-2x}$

```python
# Параметры ТЕСТОВОЙ задачи
at, bt = 0, 1
UAt, UBt = 1, 1
epsilont = 0.05
ht = 0.1 # Начальный шаг

def pt(x):
    return 3

def qt(x):
    return 2

def ft(x):
    return 0
```

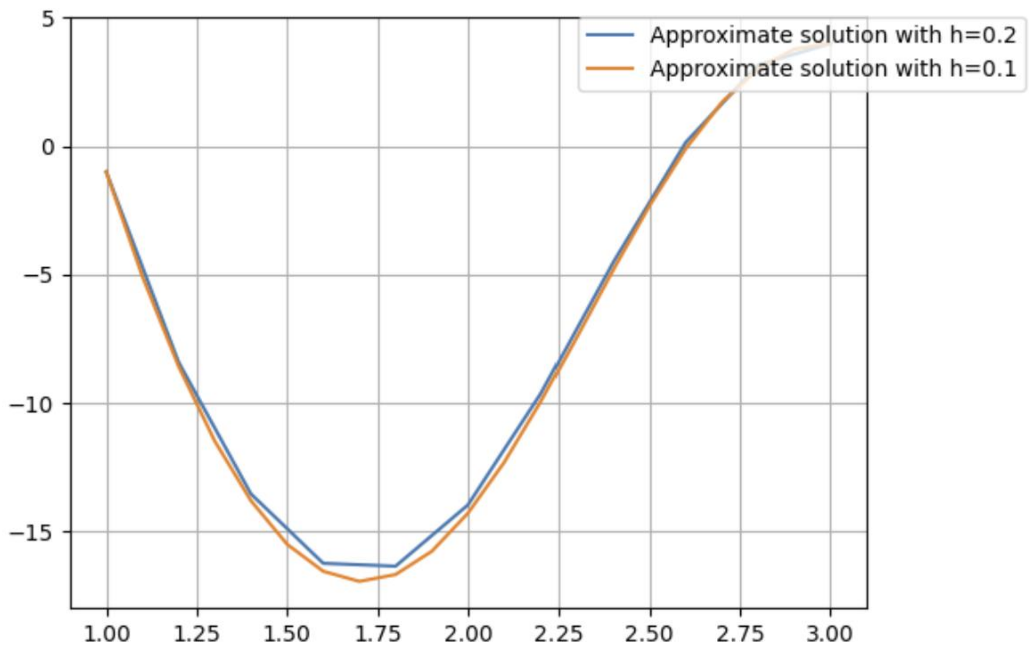
```python
# Решение ТЕСТОВОЙ задачи
xt, Ut = finite_difference_method(at, bt, UAt, UBt, ht)
x_exact_test = np.linspace(at, bt,  1000 )  # Сетка по x
u_exact_test = (1 + e)*np.exp(-x_exact_test) - e*np.exp(-2*x_exact_test)

# Построение графика
plt.plot(xt, Ut, label=f'Approximate solution with h={ht}')
plt.plot(x_exact_test, u_exact_test, label=f'Exact solution h={ht}'
)

plt.xlabel('')
plt.ylabel('u')
plt.legend(bbox_to_anchor=(1.2, 1), loc='upper right', borderaxespad=0)
plt.grid(True)
plt.show()
```



Как мы видим, полученные графики почти совпадают, поэтому перейдем к решению поставленной задачи.

Пункт 3

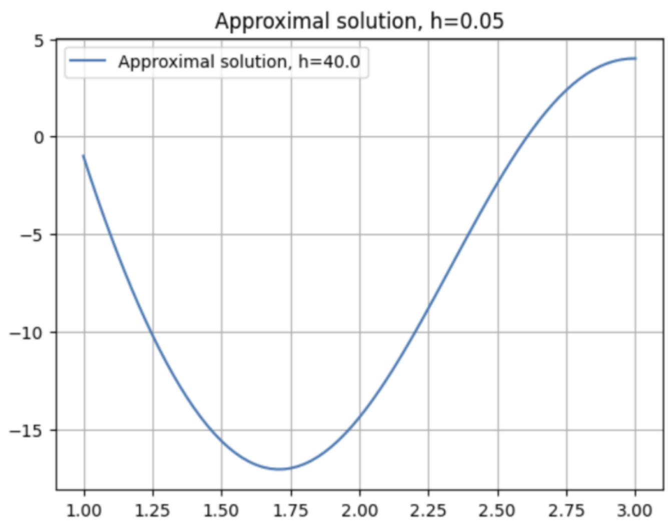
```
# Начальный шаг
h_start = 0.2

# Решение поставленной задачи
x1, U1 = finite_difference_method(a, b, UA, UB, h_start)
x2, U2 = finite_difference_method(a, b, UA, UB, h_start/2)

# Построение графика
plt.plot(x1, U1, label=f'Approximate solution with h={h_start}')
plt.plot(x2, U2, label=f'Approximate solution with h={h_start/2}')

plt.legend(bbox_to_anchor=(1.2, 1), loc='upper right', borderaxespad=0)
plt.grid(True)
plt.show()
```



Пункт 4

```
# Функция для уменьшения шага
def adaptive_step(h_start, epsilon):
    h = h_start
    while True:
        x1, U1 = finite_difference_method(a, b, UA, UB, h)
        x2, U2 = finite_difference_method(a, b, UA, UB, h/2)

        # Интерполяция решения для сравнения
        U1_interp = np.interp(x2, x1, U1)

        # Проверка точности
        diff = np.max(np.abs(U1_interp - U2))
        if diff < epsilon:
            return x2, U2, h

        h /= 2  # уменьшаем шаг
```

```
x, U, h_final = adaptive_step(h_start, epsilon)
plt.plot(x, U, label=f'Approximal solution, h={(b-a)/h_final}')
plt.title(f'Approximal solution, h={h_final}')
plt.legend()
plt.grid(True)
plt.show()
```



Задача 10.4.15

Задача 10.4. Промоделировать стационарные процессы теплопроводности стержня в зависимости от входных данных задачи – переменного коэффициента теплопроводности *k(x)* и плотности источников тепла *f(x)*:

$$\begin{cases} -\dfrac{d}{dx}\left(k(x)\dfrac{du}{dx}\right) = f, \\ u(a) = UA, \quad u(b) = UB. \end{cases}$$

ПОРЯДОК РЕШЕНИЯ ЗАДАЧИ:

1. Составить разностную схему второго порядка точности для решения указанной задачи.
2. Взять исходные данные из 1-го набора параметров для задачи 10.1.Шаг сетки положить равным $h = (b-a)/150$.

3. Промоделировать процесс теплопроводности в зависимости от коэффициента *k(x):*

3.1. Пусть стержень состоит из 2-х материалов с различными свойствами:

$$k(x) = \begin{cases} k1, a \le x \le 0.5\cdot(b+a) \\ k2, 0.5(b+a) < x \le b \end{cases}, \quad \text{а) } k1 << k2, \qquad \text{б) } k1 >> k2.$$

3.2. Пусть стержень состоит из 3-х материалов с различными свойствами:

$$k(x) = \begin{cases} k1,\ a \le x \le a + (b-a)/3 \\ k2,\ a + (b-a)/3 \le x \le a + 2(b-a)/3 \\ k2,\ a + 2(b-a)/3 < x \le b \end{cases}$$

а) *k1<k2 <k1*, б) *k1>k2 >k3*,
в) *k1=k, k2=10k, k3=k*, г) *k1=100k, k2=k, k3=100k.*

4. Промоделировать процесс теплопроводности в зависимости от правой части – функции *f(x)*, предполагая, что *f(x)* - точечный источник тепла. Задать точечный источник тепла можно следующим образом: $f(x) = c\cdot\delta(x - x0)$, где c - некоторая константа (мощность источника), $\delta(x)$- дельта-функция, $x0$ - точка из отрезка $[a,b]$, в которую ставится источник. Рассмотреть следующие варианты расположения источника:
а) точечный источник поставлен в середину отрезка $[a,b]$;
б) два одинаковых по мощности источника поставлены в разные точки отрезка, симметричные относительно середины отрезка;
в) два различных по мощности источника поставлены симметрично;
г) предложить свой вариант расположения источников.

№	*k(x)*	*f(x)*	*a*	*UA*	*b*	*UB*
10.1.15	$x^{-1/3}$	$x+\sqrt{x}$	1.5	3	2.5	-3

Пункты 1-2

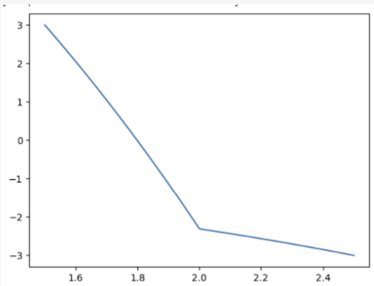
```
import numpy as np
import matplotlib.pyplot as plt
```

```
# Начальные условия
UA, UB = 3, -3
a, b = 1.5 ,2.5
h = (b - a)/150
N = 150

def k(x):
    return x**(-1/3)

def f(x):
    return x + np.sqrt(x)
```

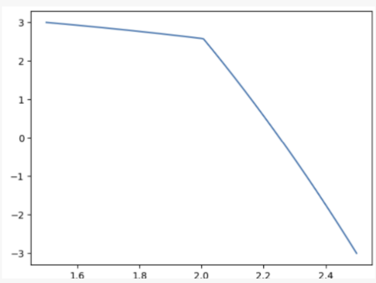
```
def finite_difference_variable_k(N, k, f):
    x = np.linspace(a, b, N+1)
    A = np.zeros((N+1, N+1))  # Матрица коэффициентов
    B = np.zeros(N+1)  # Правая часть

    # Заполняем систему уравнений
    for i in range(1, N):
        k_i_p_half = (k(x[i]) + k(x[i+1])) / 2  # k_{i+1/2}
        k_i_m_half = (k(x[i]) + k(x[i-1])) / 2  # k_{i-1/2}

        A[i, i-1] = -k_i_m_half / h**2
        A[i, i] = (k_i_p_half + k_i_m_half) / h**2
        A[i, i+1] = -k_i_p_half / h**2

        B[i] = f(x[i])

    # Граничные условия
    A[0, 0] = 1
    B[0] = UA
    A[N, N] = 1
    B[N] = UB

    u = np.linalg.solve(A, B)
    return x, u
```

Пункт 3.1

```
# Зададим переменную теплопроводность k(x)
# 3.1 a)
def k_3_1_a(x):
    k1 = 1*k(x)
    k2 = 10*k(x)
    if a <= x <= 0.5*(b + a):
        return k1
    if 0.5*(a+b) < x <= b:
        return k2

x, U = finite_difference_variable_k(N, k_3_1_a, f)
plt.plot(x, U)
# plt.axis('equal')
```

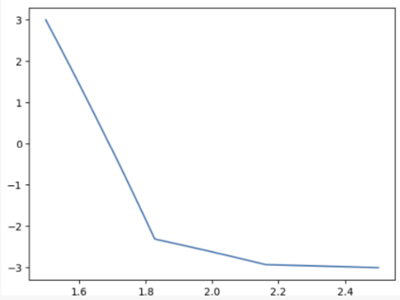



```
# 3.1 б)
def k_3_1_b(x):
    k1 = 10*k(x)
    k2 = 1*k(x)
    if a <= x <= 0.5*(b + a):
        return k1
    if 0.5*(a+b) < x <= b:
        return k2

x, U = finite_difference_variable_k(N, k_3_1_b, f)
plt.plot(x, U)
# plt.axis('equal')
```

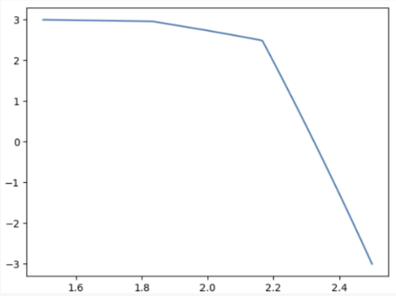


Пункт 3.2

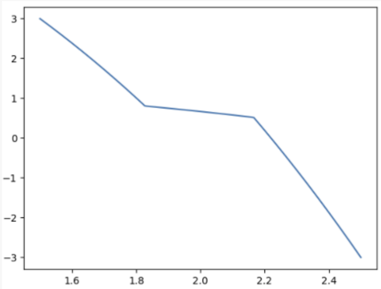
```
# 3.2. а) k1 < k2 < k3
def k_3_2_a(x):
    k1 = 1*k(x)
    k2 = 10*k(x)
    k3 = 100*k(x)
    if a <= x <= a+(b-a)/3:
        return k1
    if a+(b-a)/3 <= x <= a + 2*(b-a)/3:
        return k2
    if a+2*(b-a)/3 < x <= b:
        return k3

x, U = finite_difference_variable_k(N, k_3_2_a, f)
plt.plot(x, U)
```



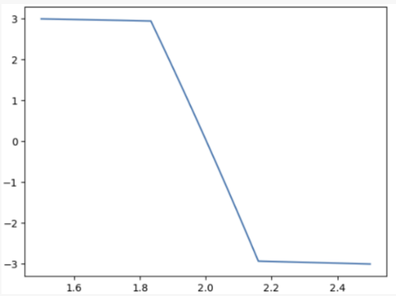

```
# 3.2. б) k1 > k2 > k3
def k_3_2_b(x):
    k1 = 100*k(x)
    k2 = 10*k(x)
    k3 = 1*k(x)
    if a <= x <= a+(b-a)/3:
        return k1
    if a+(b-a)/3 <= x <= a + 2*(b-a)/3:
        return k2
    if a+2*(b-a)/3 < x <= b:
        return k3

x, U = finite_difference_variable_k(N, k_3_2_b, f)
plt.plot(x, U)
```




```
# 3.2. в) k1 = k3 = k, k2 = 10k
def k_3_2_c(x):
    k1 = 1*k(x)
    k2 = 10*k(x)
    k3 = 1*k(x)
    if a <= x <= a+(b-a)/3:
        return k1
    if a+(b-a)/3 <= x <= a + 2*(b-a)/3:
        return k2
    if a+2*(b-a)/3 < x <= b:
        return k3

x, U = finite_difference_variable_k(N, k_3_2_c, f)
plt.plot(x, U)
```

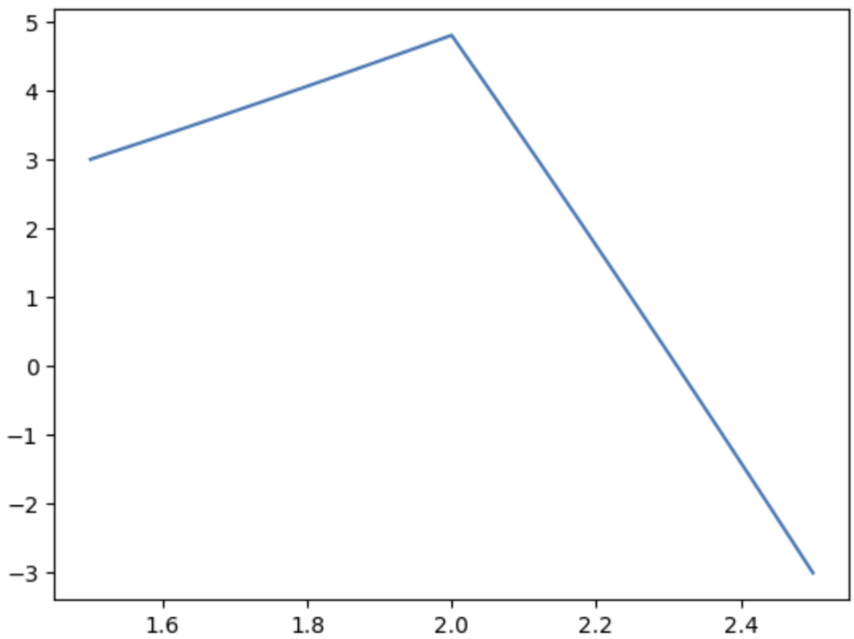



```
# 3.2. г) k1 = 100k, k2 = k, k3 = 100k
def k_3_2_d(x):
    k1 = 100*k(x)
    k2 = 1*k(x)
    k3 = 100*k(x)
    if a <= x <= a+(b-a)/3:
        return k1
    if a+(b-a)/3 <= x <= a + 2*(b-a)/3:
        return k2
    if a+2*(b-a)/3 < x <= b:
        return k3

x, U = finite_difference_variable_k(N, k_3_2_d, f)
plt.plot(x, U)
```

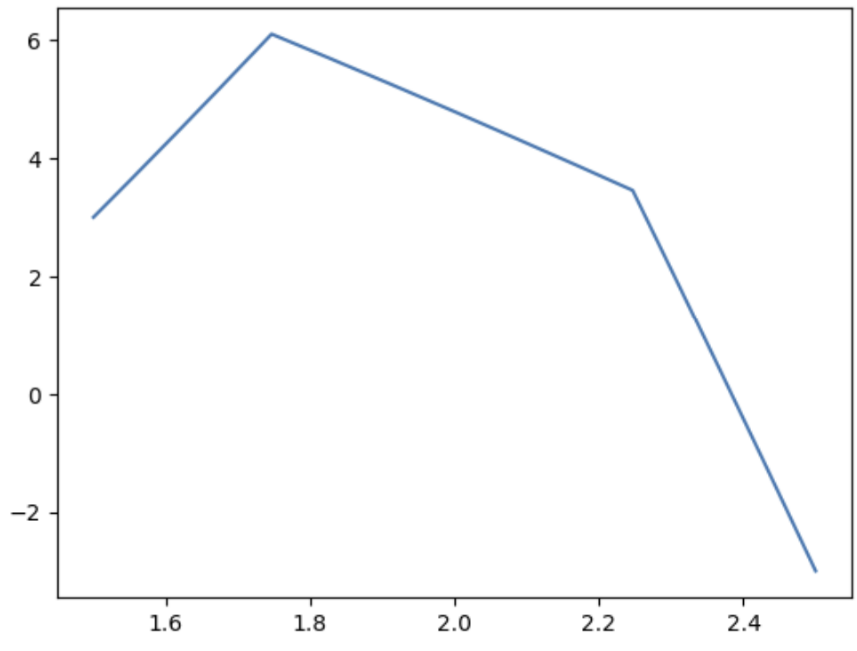


Пункт 4

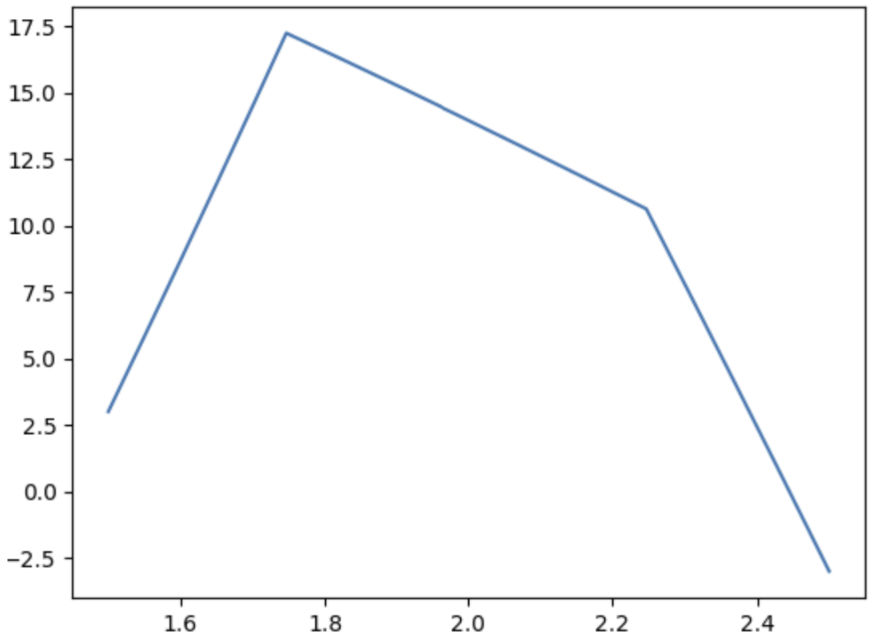
```
# 4 a) x0 = 2

c = 0.1 # Пусть будет такой

def finite_difference_variable_k_source(N, k):
    x = np.linspace(a, b, N+1)

    A = np.zeros((N+1, N+1))  # Матрица коэффициентов
    B = np.zeros(N+1)  # Правая часть

    # Заполняем систему уравнений
    for i in range(1, N):
        k_i_p_half = (k(x[i]) + k(x[i+1])) / 2  # k_{i+1/2}
        k_i_m_half = (k(x[i]) + k(x[i-1])) / 2  # k_{i-1/2}

        A[i, i-1] = -k_i_m_half / h**2
        A[i, i] = (k_i_p_half + k_i_m_half) / h**2
        A[i, i+1] = -k_i_p_half / h**2

        x_0 = (a + b) / 2
        i_0 = np.argmin(np.abs(x - x_0))  # Индекс ближайшего узла
        B[i_0] += c / h  # Добавляем источник тепла

    # Граничные условия
    A[0, 0] = 1
    B[0] = UA

    A[N, N] = 1
    B[N] = UB

    u = np.linalg.solve(A, B)

    return x, u  # Возвращаем значения u (решение) и x (узлы)

x, U = finite_difference_variable_k_source(N, k)
plt.plot(x,U)
```



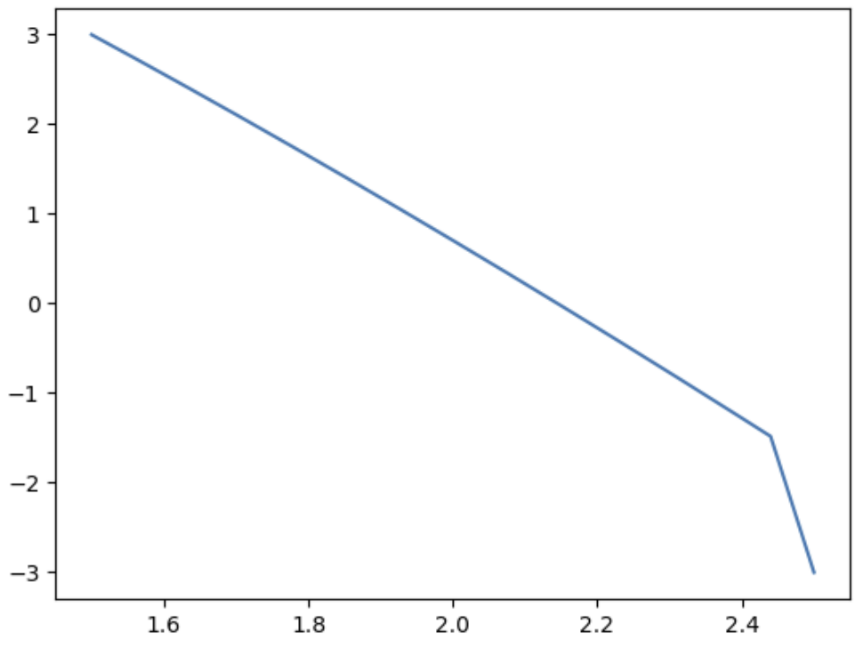
```python
# 4 б) c1=c2, x01 и xo2 в симметричных относительно центра точках отрезка

def finite_difference_variable_k_source(N, k):
    x = np.linspace(a, b, N+1)

    A = np.zeros((N+1, N+1))  # Матрица коэффициентов
    B = np.zeros(N+1)  # Правая часть

    # Заполняем систему уравнений
    for i in range(1, N):
        k_i_p_half = (k(x[i]) + k(x[i+1])) / 2  # k_{i+1/2}
        k_i_m_half = (k(x[i]) + k(x[i-1])) / 2  # k_{i-1/2}

        A[i, i-1] = -k_i_m_half / h**2
        A[i, i] = (k_i_p_half + k_i_m_half) / h**2
        A[i, i+1] = -k_i_p_half / h**2

        x_1 = a + (b - a) / 4
        x_2 = b - (b - a) / 4
        i_1 = np.argmin(np.abs(x - x_1))
        i_2 = np.argmin(np.abs(x - x_2))
        B[i_1] += c / h  # Источник в точке x_1
        B[i_2] += c / h  # Источник в точке x_2

    # Граничные условия
    A[0, 0] = 1
    B[0] = UA

    A[N, N] = 1
    B[N] = UB

    u = np.linalg.solve(A, B)

    return x, u  # Возвращаем значения u (решение) и x (узлы)


x, U = finite_difference_variable_k_source(N, k)
plt.plot(x,U)
```



```
#4 в) c1 != c2, расставлены симметрично

def finite_difference_variable_k_source(N, k):
    x = np.linspace(a, b, N+1)

    A = np.zeros((N+1, N+1))  # Матрица коэффициентов
    B = np.zeros(N+1)  # Правая часть

    # Заполняем систему уравнений
    for i in range(1, N):
        k_i_p_half = (k(x[i]) + k(x[i+1])) / 2  # k_{i+1/2}
        k_i_m_half = (k(x[i]) + k(x[i-1])) / 2  # k_{i-1/2}

        A[i, i-1] = -k_i_m_half / h**2
        A[i, i] = (k_i_p_half + k_i_m_half) / h**2
        A[i, i+1] = -k_i_p_half / h**2

        x_1 = a + (b - a) / 4
        x_2 = b - (b - a) / 4
        i_1 = np.argmin(np.abs(x - x_1))
        i_2 = np.argmin(np.abs(x - x_2))
        c1 = 4 * c
        c2 = 2 * c
        B[i_1] += c1 / h  # Источник в точке x_1 с мощностью c1
        B[i_2] += c2 / h  # Источник в точке x_2 с мощностью c2

    # Граничные условия
    A[0, 0] = 1
    B[0] = UA

    A[N, N] = 1
    B[N] = UB

    u = np.linalg.solve(A, B)

    return x, u  # Возвращаем значения u (решение) и x (узлы)

x, U = finite_difference_variable_k_source(N, k)
plt.plot(x,U)
```



```python
#4 в) свой вариант расположения источника

def finite_difference_variable_k_source(N, k):
    x = np.linspace(a, b, N+1)

    A = np.zeros((N+1, N+1))  # Матрица коэффициентов
    B = np.zeros(N+1)  # Правая часть

    # Заполняем систему уравнений
    for i in range(1, N):
        k_i_p_half = (k(x[i]) + k(x[i+1])) / 2  # k_{i+1/2}
        k_i_m_half = (k(x[i]) + k(x[i-1])) / 2  # k_{i-1/2}

        A[i, i-1] = -k_i_m_half / h**2
        A[i, i] = (k_i_p_half + k_i_m_half) / h**2
        A[i, i+1] = -k_i_p_half / h**2

        B[-10] += c / h  # Источник в точке x_0 = b - 10/N

    # Граничные условия
    A[0, 0] = 1
    B[0] = UA

    A[N, N] = 1
    B[N] = UB

    u = np.linalg.solve(A, B)

    return x, u  # Возвращаем значения u (решение) и x (узлы)

x, U = finite_difference_variable_k_source(N, k)
plt.plot(x,U)
```



Задача 10.5.15

Задача 10.5. Методом конечных разностей найти приближенное решение краевой задачи

$$\begin{cases} -(k(x)u')' + q(x)u = f(x), \quad x \in (a,b), \\ -k(a)u'(a) + 0.5u(a) = 0, \\ \ k(b)u'(b) + 0.5u(b) = 0. \end{cases}$$

с тремя верными значащими цифрами. Решение системы разностных уравнений найти, используя метод прогонки.

УКАЗАНИЯ.

1. Использовать разностную схему второго порядка точности.
2. При аппроксимации производных в граничных условиях использовать метод баланса.

№	*a*	*b*	*c*	*k(x)*		*q (x)*		*f (x)*
				$a<x<c$	$c<x<b$	$a<x<c$	$c<x<b$	
10.5.15	0	1.5	0.875	0.5	1.8	5.6	8.5	$9x(3.5-x)$

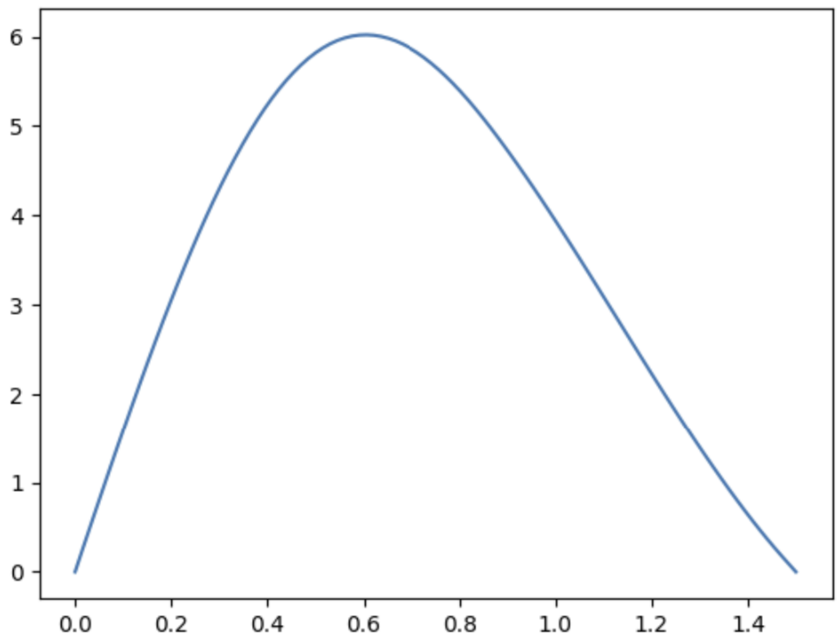
Пункты 1-2

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

```
# Задаем параметры задачи
a, b, c = 0, 1.5, 0.875
n = 100  # количество узлов сетки
h = (b - a) / (n - 1)  # шаг сетки
x = np.linspace(a, b, n)

def k(x):
    if a <= x < c:
        return 0.5
    elif c <= x <= b:
        return 1.8

def q(x):
    if a <= x < c:
        return 5.6
    elif c <= x <= b:
        return 8.5

def f(x):
    return 9*x*(3.5 - x)
```

```
# Матрицы для прогонки
A = np.zeros(n-2)
B = np.zeros(n-2)
C = np.zeros(n-2)
D = np.zeros(n-2)
```

```
# Построение матрицы для внутренней части
for i in range(1, n-1):
    A[i-1] = k(x[i]) / h**2
    B[i-1] = -2 * k(x[i]) / h**2 + q(x[i])
    C[i-1] = k(x[i]) / h**2
    D[i-1] = f(x[i])

# Учет краевых условий методом баланса
# Левое краевое условие: -k(a)u'(a) + 0.5*u(a) = 0
B[0] += 1 / h * k(x[0]) + 0.5
D[0] -= 0.5 * f(x[0])

# Правое краевое условие: k(b)u'(b) + 0.5*u(b) = 0
B[-1] += 1 / h * k(x[-1]) + 0.5
D[-1] -= 0.5 * f(x[-1])

# Прямой проход метода прогонки
alpha = np.zeros(n-2)
beta = np.zeros(n-2)
alpha[0] = -C[0] / B[0]
beta[0] = D[0] / B[0]

for i in range(1, n-2):
    alpha[i] = -C[i] / (B[i] + A[i] * alpha[i-1])
    beta[i] = (D[i] - A[i] * beta[i-1]) / (B[i] + A[i] * alpha[i-1])

# Обратный проход
u = np.zeros(n)
u[-2] = beta[-1]

for i in range(n-3, -1, -1):
    u[i+1] = alpha[i] * u[i+2] + beta[i]

plt.plot(x, u)
```



Задача 10.6.15

Задача 10.6. Промоделировать нестационарные процессы теплопроводности в зависимости от входных данных задачи - коэффициента теплопроводности *k(x)* и начальной температуры $\phi(x)$:

$$\begin{cases} \dfrac{\partial u}{\partial t} = \dfrac{\partial}{\partial x}\left(k(x)\dfrac{\partial u}{\partial x}\right) + f(x)(1-e^{-t}), & 0<x<l, \quad 0<t<T, \\ u(0,t)=UA, \quad u(l,t)=UB, & 0 \le t \le T, \\ u(x,0)=\phi(x), & 0 \le x \le l. \end{cases}$$

ПОРЯДОК РЕШЕНИЯ ЗАДАЧИ:

1. Найти приближенное решение задачи с шагами $\tau = 0.05$ и h=0.1, используя явную разностную схему. Построить графики решений при значениях $t = 0.5\tau$, 10τ, 20τ.

2. Используя результаты задачи 10.1, экспериментально определить момент времени *t*, при котором происходит установление процесса (визуально).
3. Произвести анимацию процесса установления.
4. Исследовать, как влияет начальная температура на процесс установления, взяв другие функции $\phi(x)$ (согласованные с граничными условиями).

УКАЗАНИЕ.

Для создания анимационного клипа нужно:
- выбрать пункт меню **Animate**,
- заключить в выделяющий пунктирный прямоугольник поле графика, который нужно анимировать,
- в диалоговом окне установить значение переменной **FRAME**, например,10,
- нажать кнопку **Create** (или **Animate**),
- воспроизвести анимацию.

Таблица к задаче 10.6

В задаче 10.6 взять входные данные *k(x), f(x), ua, ub* из задачи 10.1,
$\phi(x) = (ub - ua)(x-a)/l + ua$, $l = b - a$.

№	k(x)	f(x)	a	UA	b	UB
10.1.15	$x^{-1/3}$	$x+\sqrt{x}$	1.5	3	2.5	-3

Пункт 1-2

Для аппроксимации уравнения (1) используем явную разностную схему

$$\frac{y^{n+1}-y^n}{\tau} = -Ay^n + \varphi^n, \quad n = 0, 1, \ldots, N_t. \qquad (8)$$

Здесь через $y^n = y_i^n$ обозначено приближенное решение в узле сетки $(x_i, t_n) \in \omega$, сеточный оператор A определен для сеточных функций $y = 0$ при $x \in \partial\omega_h$ (граничные узлы сетки ω_h) следующим образом:

$$Ay^n = -(ay^n_{\bar{x}})_{x,i} = -\frac{1}{h}\left(a_{i+1}\frac{y^n_{i+1}-y^n_i}{h} - a_i\frac{y^n_i - y^n_{i-1}}{h}\right), \quad x_i \in \omega_h.$$

n = 10   Количество точек сетки

дополненное граничными (первого рода)

$$u(0,t) = u(l,t) = 0, \quad t \in (0,T],$$

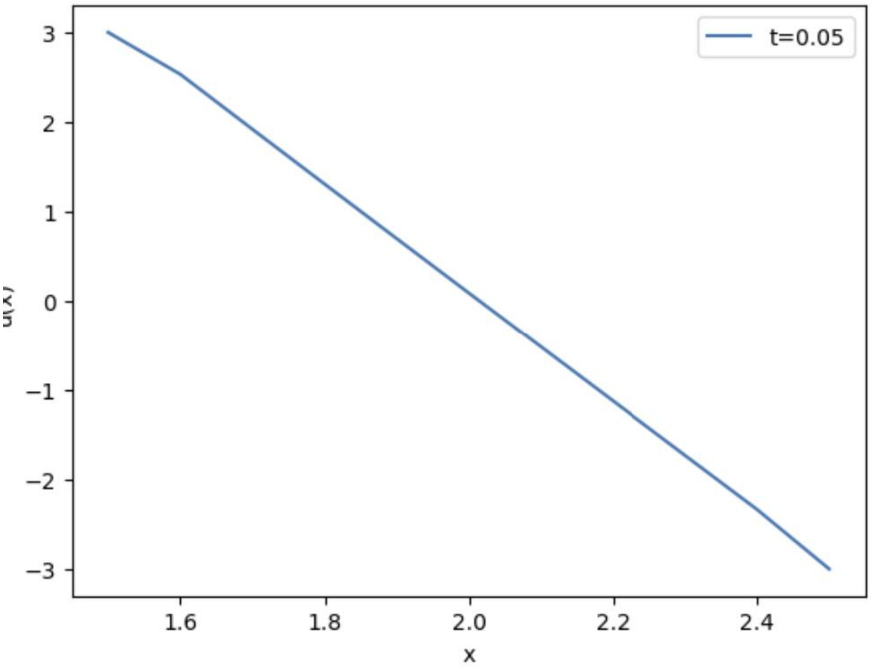
и начальными

$$u(x,0) = I(x), \quad 0, \quad x \in \bar{\Omega},$$

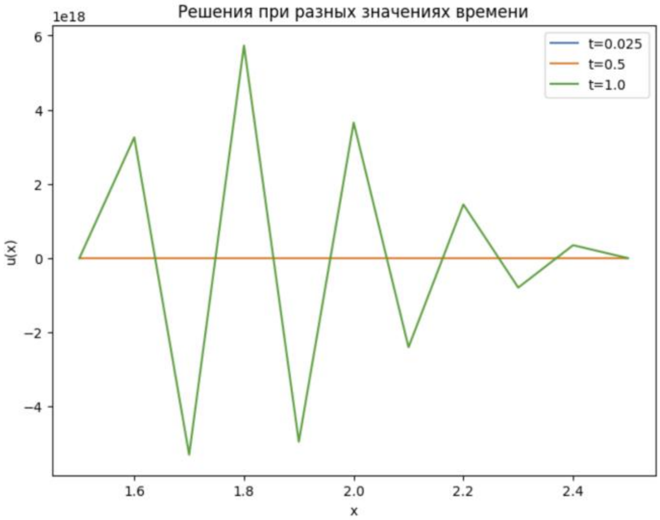
$$y_i^{n+1} = y_i^n + \frac{\tau}{h}\left(a_{i+1}\frac{y_{i+1}^n - y_i^n}{h} - a_i\frac{y_i^n - y_{i-1}^n}{h}\right) + \tau\varphi_i^n$$

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

# Параметры задачи
a, b = 1.5, 2.5  # Границы по x
tau = 0.05  # Шаг по времени
T0, T = 0, 1  # Время начала и конца
ua, ub = 3, -3  # Граничные условия
h = 0.1  # Шаг по пространству
Nx = int(round((b - a)/h))
x = np.linspace(a, b, Nx + 1)  # Массив значений по x
l = b - a

# Функции для уравнения
f = lambda x, t: (x + np.sqrt(x)) * (1 - np.exp(-t))
k = lambda x: x ** (-1/3)
phi = lambda x: (ub - ua)*(x - a)/l + ua

def solver_Ex_simple(tt):
    """
    Реализация явной разностной схемы для приближенного решения
    параболического уравнения для произвольного момента времени tt.
    """

    Nt = int(T / tau)
    u = np.zeros(Nx + 1) # Итоговый результат
    u_n = np.zeros(Nx + 1) # Шаговой результат
    t = np.linspace(T0, T, Nt+1)

    # Устанавливаем начальные условия u(x,0) = phi(x)
    for i in range(0, Nx + 1):
        u_n[i] = phi(x[i])

    # Прокручиваем временные шаги до времени tt
    for n in range(1, Nt):
        current_time = n * tau  # Текущее время
        if current_time > tt:
            break  # Если текущее время больше заданного tt, останавливаем
цикл

        # Вычисляем приближенное решение во внутренних узлах сетки
        for i in range(1, Nx):
            u[i] = u_n[i] + tau * (k(x[i+1]) * u_n[i+1] - 2 * k(x[i]) *
u_n[i] + k(x[i-1]) * u_n[i-1]) / (h**2) + tau * f(x[i], current_time)

        # Граничные условия
        u[0] = ua
        u[Nx] = ub
```

```
        # Обновляем переменные перед следующим шагом
        u_n, u = u, u_n

    return u_n, x

# Тестирование при разных временных шагах
u, x = solver_Ex_simple(tau)
plt.plot(x, u, label=f't={tau}')

plt.xlabel('x')
plt.ylabel('u(x)')
plt.legend()
plt.show()
```



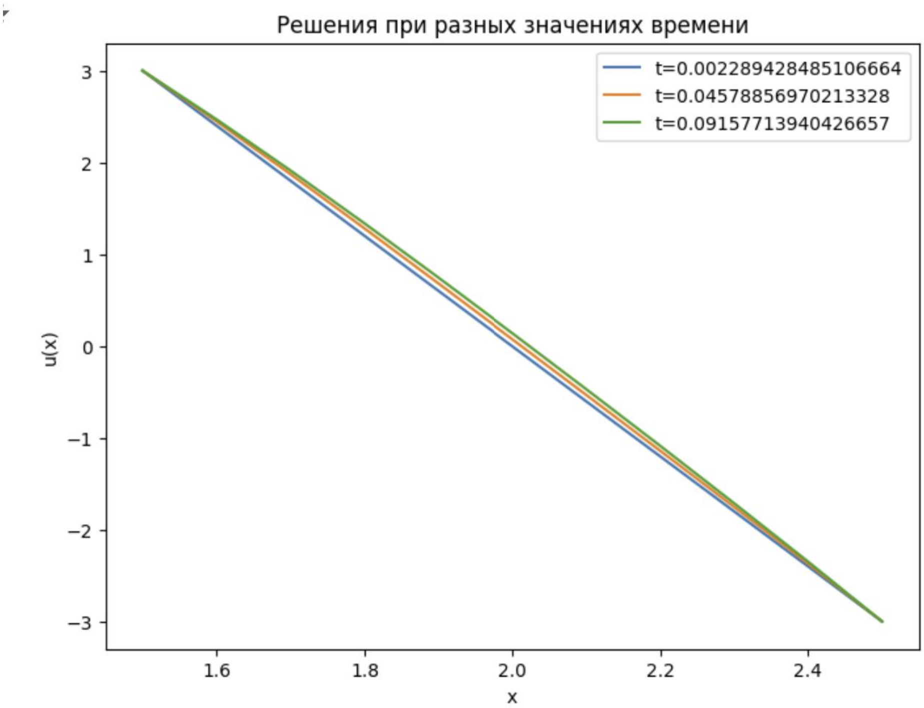
Однако при других t решение сильно осциллирует!



Это связано с тем, что не соблюдается условие устойчивости Куранта явной схемы

$$\tau \leq \frac{h^2}{2k_{\max}}$$

```
# Пересчитаем tau на основе условия Куранта
k_max = np.max(k(x))  # Максимальное значение функции теплопроводности
tau_new = 0.4 * h**2 / k_max  # Скорректированный шаг по времени

# Обновим tau на новое значение
tau = tau_new
print(f"Новый шаг по времени (tau): {tau}")
```



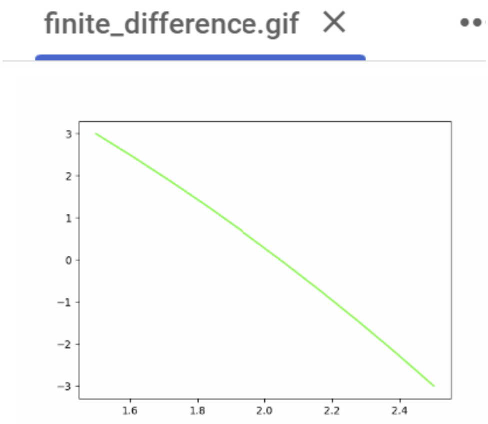
Видим, что чем больше проходит времени, тем сильнее решение движется к устоявшемуся

Пункт 3

Код для создания гифки:

```
import scipy.sparse as sps
import scipy.sparse.linalg
import imageio.v2 as imageio
from tqdm import trange
import numpy as np
import matplotlib.pyplot as plt

# Параметры задачи
T0, T = 0, tau * 100  # Время начала и конца
Nx = int(round((b - a)/h))
x = np.linspace(a, b, Nx)  # Массив значений по x

# Функции для f(x) и k(x)
f = lambda x: x + np.sqrt(x)
k = lambda x: x ** (-1/3)

# Функция для одного временного шага
def iteration(t):
    left = np.zeros((Nx, Nx))  # Матрица коэффициентов
    right = np.zeros(Nx)  # Вектор правой части
```

```python
    # Заполнение системы уравнений
    for i in range(1, Nx - 1):
        k_right = k((x[i] + x[i + 1]) / 2)
        k_left = k((x[i] + x[i - 1]) / 2)
        left[i, i] = (k_right + k_left) / h**2  # Диагональный элемент
        left[i, i - 1] = -k_left / h**2  # Левый сосед
        left[i, i + 1] = -k_right / h**2  # Правый сосед
        right[i] = f(x[i]) * (1 - np.exp(-t))  # Правая часть уравнения без h^2

    # Граничные условия
    left[0, 0], right[0] = 1, ua
    left[-1, -1], right[-1] = 1, ub

    # Решение системы уравнений
    u = sps.linalg.spsolve(sps.csr_matrix(left), right)
    return u

# Параметры для визуализации
max_iter = 100
step = 1

# Основной цикл по временным шагам
for i in trange(0, max_iter, step):
    u = iteration(i * tau)  # Вычисление решения на шаге i
    plt.plot(x, u, color=(1 - i / max_iter, i / max_iter, 0))  # Построение графика
    plt.savefig(f'{i}.png')  # Сохранение графика как изображения
    plt.close()

# Создание GIF из сохранённых изображений
filenames = [f'{i}.png' for i in range(0, max_iter, step)]
with imageio.get_writer('finite_difference.gif', mode='I') as writer:
    for filename in filenames:
        writer.append_data(imageio.imread(filename))
```



(гиф приложена к репозиторию)

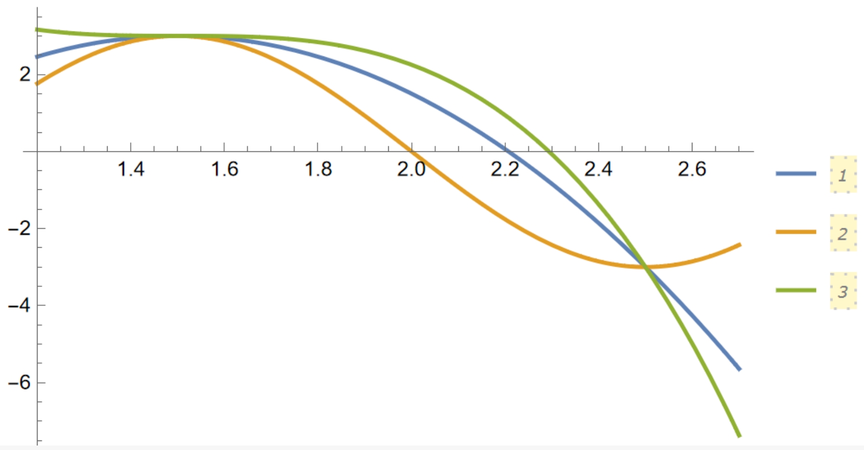
Пункт 4 (при других phi)

Возьму:
Phi1 = 3 – 6*(x – 1.5)^2 – квадратичную функцию, которая тоже подходит под граничные условия.

Phi2 = 3*sin(pi(x – 1))
Phi3 = 3 – 6*(x – 1.5)^3 – полином третьей степени

```
In[97]:= phi1[x_] := 3 - 6 (x - 1.5)^2

In[106]:= phi2[x_] := 3 Sin[Pi (x - 1)]
                        си··· число пи

          phi3[x_] := 3 - 6 (x - 1.5)^3

In[108]:= Plot[{phi1[x], phi2[x], phi3[x]}, {x, 1.2, 2.7}, PlotLegends → Automatic]
          график функции                                  легенды графика автоматически
```

Out[108]=




```python
# Определение начальных функций phi
phi1 = lambda x: ua - 6 * ((x - 1.5) ** 2)
phi2 = lambda x: 3 * np.sin(np.pi * (x - 1))
phi3 = lambda x: ua - 6 * ((x - 1.5) ** 3)

def solver_Ex_simple(phi):
    """
    Реализация явной разностной схемы для приближенного решения
    параболического уравнения для произвольной функции phi(x).
    """
    Nt = int(T / tau)
    u = np.zeros(Nx + 1)  # Итоговый результат
    u_n = np.zeros(Nx + 1)  # Шаговой результат
    t = np.linspace(T0, T, Nt + 1)

    # Устанавливаем начальные условия u(x,0) = phi(x)
    for i in range(0, Nx + 1):
        u_n[i] = phi(x[i])

    # Массив для хранения решений в разные моменты времени
    solutions = [u_n.copy()]

    # Прокручиваем временные шаги
    for n in range(1, Nt):
        current_time = n * tau  # Текущее время

        # Вычисляем приближенное решение во внутренних узлах сетки
        for i in range(1, Nx):
            u[i] = u_n[i] + tau * (k(x[i + 1]) * u_n[i + 1] - 2 * k(x[i]) *
u_n[i] + k(x[i - 1]) * u_n[i - 1]) / (h**2) + tau * f(x[i], current_time)

        # Граничные условия
        u[0] = ua
        u[Nx] = ub

        # Обновляем переменные перед следующим шагом
        u_n, u = u, u_n
```

```
        # Сохраняем решения на каждом шаге времени
        if n % (Nt // 5) == 0:  # Сохраняем решение 5 раз за процесс
            solutions.append(u_n.copy())

    return solutions, x

# Массив начальных функций
phi_array = [phi1, phi2, phi3]
phi_labels = ['phi1: ua - 6*(x - 1.5)^2', 'phi2: 3*sin(pi*(x - 1))', 'phi3: ua - 6*(x - 1.5)^3']

# Построение решений для разных начальных условий
plt.figure(figsize=(12, 8))

for j, phi in enumerate(phi_array):
    solutions, x = solver_Ex_simple(phi)

    # Для каждого phi строим графики через несколько временных шагов
    for i, u in enumerate(solutions):
        plt.plot(x, u, label=f'{phi_labels[j]}, time step {i+1}')

plt.xlabel('x')
plt.ylabel('u(x)')
plt.legend(loc='upper right')
plt.title('Решения для разных начальных условий phi в разные моменты времени')
plt.show()
```