Вариант 1.

Задача 8.1. Найти аналитическое и приближенное решения краевой задачи

$$\begin{cases} -u'' + pu' + qu = f(x), \quad x \in (a,b); \\ \quad u(a) = ua, \quad u(b) = ub; \end{cases}$$

с заданным шагом h. Решение системы разностных уравнений найти с помощью метода прогонки.

№	a	b	p	q	f(x)	Ua	Ub
8.1.1	1	2	0	0.25	$0.5x^2 - 0.5x - 3.25$	2.5	5

Решение задачи

1. Найти аналитическое решение задачи (см. ПРИЛОЖЕНИЕ 8.В)

- $\begin{cases} -u'' + 0.25u = 0.5x^2 - 0.5x - 3.25 \\ \quad u(1) = 2.5, \; u(2) = 5 \end{cases}$

- Исходное ДУ является линейным неоднородным ДУ, поэтому его общее решение имеет вид $u_{\text{он}} = u_{\text{оо}} + u_{\text{чн}}$, где $u_{\text{оо}}$ – общее решение соответствующего линейного однородного ДУ, $u_{\text{чн}}$ – частное решение линейного неоднородного ДУ.

- Решим линейное однородное ДУ
  $-u'' + 0.25u = 0$ –уравнение с постоянными коэффициентами, его можно решать с помощью характеристического уравнения
  $\lambda^2 - 0.25 = 0$
  $\lambda = \pm\dfrac{1}{2}$
  $u_{\text{оо}} = C_1 e^{\frac{1}{2}x} + C_2 e^{-\frac{1}{2}x}$

- Частное решение неоднородного ДУ найдём методом неопределённых коэффициентов: поскольку правая часть $f(x) = 0.5x^2 - 0.5x - 3.25$ уравнения представляет собой многочлен второй степени, то решение можно искать также в виде многочлена второй степени с неизвестными (неопределёнными) коэффициентами

  $u_{\text{чн}} = Ax^2 + Bx + C$

  $u'_{\text{чн}} = 2Ax + B$

  $u''_{\text{чн}} = 2A$

  Подставим частное решение в исходное уравнение, получим

  $-2A + 0.25(Ax^2 + Bx + C) = 0.5x^2 - 0.5x - 3.25$

  $x^2 : \; 0.25A = 0.5 => A = 2$

  $x^1 : \; 0.25B = -0.5 => B = -2$

  $x^0 : \; -2A + 0.25C + 3.25 = 0 \; => \; C = 3$
  $u_{\text{чн}} = 2x^2 - 2x + 3$

- $u_{\text{он}} = C_1 e^{\frac{1}{2}x} + C_2 e^{-\frac{1}{2}x} + 2x^2 - 2x + 3$
- Решение исходной краевой задачи получается подстановкой вместо C_1 и C_2 числовых значений, при которых это решение удовлетворяет граничным условиям: $u(1) = 2.5,\ u(2) = 5$

$$u_{\text{он}} = C_1 e^{\frac{1}{2}x} + C_2 e^{-\frac{1}{2}x} + 2x^2 - 2x + 3$$

$$\begin{cases} u(1) = C_1 e^{\frac{1}{2}} + C_2 e^{-\frac{1}{2}} + 2 - 2 + 3 = 2.5 \\ u(2) = C_1 e^{1} + C_2 e^{-1} + 8 - 4 + 3 = 5 \end{cases}$$

Получаем систему

$$\begin{cases} C_1 e^{\frac{1}{2}} + C_2 e^{-\frac{1}{2}} = -\frac{1}{2} \\ C_1 e^{1} + C_2 e^{-1} = -2 \end{cases}$$

Решим систему методом Крамера

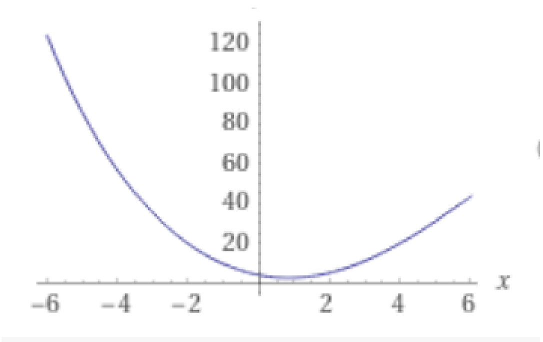
$$\Delta = \begin{vmatrix} e^{\frac{1}{2}} & e^{-\frac{1}{2}} \\ e^{1} & e^{-1} \end{vmatrix} = e^{-\frac{1}{2}} - e^{\frac{1}{2}}$$

$$C_1 = \frac{\Delta_1}{\Delta} = \frac{\begin{vmatrix} -\frac{1}{2} & e^{-\frac{1}{2}} \\ -2 & e^{-1} \end{vmatrix}}{e^{-\frac{1}{2}} - e^{\frac{1}{2}}} = \frac{-\frac{1}{2}e^{-1} + 2e^{-\frac{1}{2}}}{e^{-\frac{1}{2}} - e^{\frac{1}{2}}} \approx -0.98746$$

$$C_2 = \frac{\Delta_2}{\Delta} = \frac{\begin{vmatrix} e^{\frac{1}{2}} & -\frac{1}{2} \\ e^{1} & -2 \end{vmatrix}}{e^{-\frac{1}{2}} - e^{\frac{1}{2}}} = \frac{-2e^{\frac{1}{2}} + \frac{1}{2}e}{e^{-\frac{1}{2}} - e^{\frac{1}{2}}} \approx 1.85983$$

$$u = u_{\text{он}} = -0.98746 e^{\frac{1}{2}x} + 1.85983 e^{-\frac{1}{2}x} + 2x^2 - 2x + 3$$

График решения $u(x)$



2. Составить разностную схему и выписать коэффициенты матрицы системы уравнений и коэффициенты правой части.

- $\begin{cases} -u'' + 0.25u = 0.5x^2 - 0.5x - 3.25 \\ u(1) = 2.5,\ u(2) = 5 \end{cases}$

  Заменим отрезок [a, b]=[1, 2] сеткой $\overline{\omega}^h$. Будем считать сетку равномерной с шагом $h = \frac{b-a}{N},\ x_i = a + ih$, $i = 0, 1, 2, \dots, N$

  Заменим вторую производную разностной аппроксимацией:

  $$u_i'' = \frac{u_{i-1} - 2u_i + u_{i+1}}{h^2}$$

  Так как дифференциальное уравнение в исходной задаче рассматривается во внутренних точках отрезка, то в каждой точке x_i, $i = 1, 2, \dots, N-1$ потребуем выполнения уравнения:

  $$-\frac{u_{i-1} - 2u_i + u_{i+1}}{h^2} + 0.25u_i = 0.5{x_i}^2 - 0.5x_i - 3.25\ ,\ i = 1, 2, \dots, N-1 \qquad (1)$$

  В результате ДУ оказалось аппроксимированным его дискретным аналогом – разностным уравнением (1). Потребуем выполнения граничных условий

  $u_0 = 2.5, \qquad u_N = 5$

  Таким образом, пришли к системе линейных алгебраических уравнений, в которой число уравнения совпадает с числом неизвестных и равно $N + 1$

- Преобразуем уравнение (1) к следующему виду:

  $-u_{i-1} + 2u_i - u_{i+1} + 0.25h^2u_i = h^2(0.5{x_i}^2 - 0.5x_i - 3.25\ )$

  $-u_{i-1} + (2{+}0.25h^2)u_i - u_{i+1} = h^2(0.5{x_i}^2 - 0.5x_i - 3.25\ )$

  Запишем систему более подробно:

  $$\begin{cases} u_0 = 2.5 \\ -u_0 + (2{+}0.25h^2)u_1 - u_2 = h^2(0.5{x_1}^2 - 0.5x_1 - 3.25\ ) \\ -u_1 + (2{+}0.25h^2)u_2 - u_3 = h^2(0.5{x_2}^2 - 0.5x_2 - 3.25\ ) \\ \vdots \\ -u_{N-2} + (2{+}0.25h^2)u_{N-1} - u_N = h^2(0.5{x_{N-1}}^2 - 0.5x_{N-1} - 3.25\ ) \\ u_N = 5 \end{cases}$$

- $$\begin{pmatrix} 1 & 0 & \cdots & & & & 0 \\ -1 & (2+0.25h^2) & -1 & 0 & \cdots & & 0 \\ 0 & -1 & (2+0.25h^2) & -1 & \cdots & & 0 \\ 0 & & \vdots & & & & \\ 0 & & \cdots & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} u_0 \\ u_1 \\ u_2 \\ \vdots \\ u_N \end{pmatrix} = \begin{pmatrix} u_0 \\ h^2(0.5{x_1}^2 - 0.5x_1 - 3.25\ ) \\ h^2(0.5{x_2}^2 - 0.5x_2 - 3.25\ ) \\ \vdots \\ u_N \end{pmatrix}$$

3.Найти решение задачи по разностной схеме с точностью 0.001.
Метод прогонки:

```
def  tridiagonal_matrix(arr_a, arr_b, arr_c, arr_d):
    y=[arr_b[0]]
    alfa=[-arr_c[0]/y[0]]
    betta = [arr_d[0]/y[0]]

    n = len(arr_a)
    for i in range(1, n-1):
        y.append(arr_b[i]+arr_a[i]*alfa[i-1])
        alfa.append(-arr_c[i]/y[i])
        betta.append((arr_d[i]-arr_a[i]*betta[i-1])/y[i])



    alfa.append(0)
    betta.append((arr_d[n-1]-arr_a[n-1]*betta[n-2])/(arr_b[n-1]+arr_a[n-1]*alfa[n-2]))
    #print("alfa", alfa)
    #print("betta", betta)


    x = [betta[n-1]]
    for i in range (n-2, -1, -1):
        x.append(alfa[i]*x[n-2-i]+betta[i])
    return x[:: -1]
```

Подпрограмма, которая вычисляет элементы “матрицы”, на самом деле хранится не матрица, так как это занимает много памяти, а одномерные массивы.

$$\begin{pmatrix} 1 & 0 & \cdots & & & 0 & \\ -1 & (2+0.25h^2) & -1 & 0 & \cdots & 0 & \\ 0 & -1 & (2+0.25h^2) & -1 & \cdots & & 0 \\ 0 & & \vdots & & & & \\ 0 & & \cdots & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} u_0 \\ u_1 \\ u_2 \\ \vdots \\ u_N \end{pmatrix} = \begin{pmatrix} u_0 \\ h^2(0.5{x_1}^2 - 0.5x_1 - 3.25\,) \\ h^2(0.5{x_2}^2 - 0.5x_2 - 3.25\,) \\ \vdots \\ u_N \end{pmatrix} (1)$$

В массиве arr_b хранятся элементы главной диагонали, в массиве arr_ a = [-1, -1, …, 0] ,

в массиве arr_c = [0, -1, …, 0], в массиве arr_d – правая часть (1)

```
import numpy as np
u0 = 2.5
un= 5
a = 1
b = 2
n = 100
def func1(a, b,u0, un, n):
    h = (b-a)/n
    x = np.linspace(a, b, n+1)
    arr_a = [0]
    arr_b = [1]
    arr_c = [0]
    arr_d = [u0]
    for i in range (0 , n -1):
        arr_a .append(-1)
        arr_b.append(2+0.25*h**2)
        arr_c .append(-1)
        xi = a+ i*h
        arr_d.append ((h**2)*(0.5*xi**2 - 0.5*xi-3.25))
    arr_a .append(0)
    arr_b.append(1)
    arr_c .append(0)
    arr_d.append (un)
    u = tridiagonal_matrix(arr_a, arr_b, arr_c, arr_d)
    return u
u = func1(a, b,u0, un, n)
print (u)
```

```
[2.5, 2.506373035206741, 2.5131337297393626, 2.5202817476152273, 2.5278167525347826, 2.5357384078731515, 2.544046376671717, 2.5
527403216296993, 2.561819905095722, 2.571284789059373, 2.5811346351427495, 2.5913691045920046, 2.6019878582688745, 2.6129905566
422007, 2.624376859779443, 2.6361464273381796, 2.6482989185575994, 2.660833992249983, 2.6737513067921728, 2.687050520117032, 2.
700731289704894, 2.7147932725749984, 2.729236125276917, 2.7440595038819673, 2.7592630639746143, 2.77484646064386, 2.7908093484
74622, 2.8071513815390956, 2.823872213388108, 2.8409714970424544, 2.858448884984227, 2.876304029148123 7, 2.8945365809127495, 2.9
13146191091898, 2.932132509925 8235, 2.951495187072497, 2.971233871598848, 2.9913482119719883, 3.0118378560504278, 3.03270245107
52683, 3.053941643661386, 3.0755550797885944, 3.0975424047927973, 3.11990326335712, 3.1426372995030265, 3.1657441565814204, 3.1
892234772637282, 3.213074903532967, 3.2372980766747936, 3.2618926372685366, 3.2868582251782117, 3.312194479543516, 3.3379010387
70808, 3.3639775405240693, 3.390423621715843, 3.4172389184981595, 3.444423066253438, 3.4719756995853728, 3.4998964523097973, 3.
528184957445 5295, 3.556840847205198, 3.585863752986047, 3.615253305360721, 3.6450091340680295, 3.675130868003689, 3.70561813521
10484, 3.7364705628717876, 3.767687777296598, 3.799269403915841, 3.831215067270182, 3.863524391001205, 3.8961969978420035, 3.92
9232509607747, 3.96263054718623, 3.996390730528393, 4.030512678638819, 4.06499600956621, 4.09984034039384, 4.13504528722997 9,
4.17061046519829 9, 4.206535488428249, 4.2428199700454075, 4.279463522161818, 4.316465755866281, 4.353826281214641, 4.3915447072
20031, 4.4296206418431, 4.468053691982217, 4.506843463463632, 4.545989561031634, 4.5854915883386616, 4.625349147935397, 4.66556
1841260831, 4.706129268632296, 4.747051029235476, 4.788326721114388, 4.8299559411613275, 4.871938285106795, 4.914273347509391,
4.9569607217456735, 5.0]
```

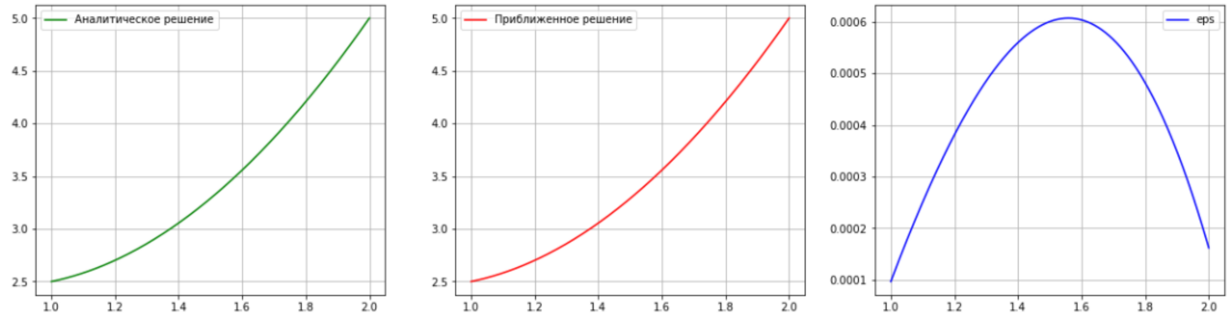
```
def analit_func(x):
    return (-0.9874*np.exp(0.5*x)+1.85983*np.exp(-0.5*x)+2*x**2-2*x+3)
def tridiagonal_matrix_with_eps(a, b,u0, un, eps ):
    n1 = 2
    h =(b-a)/n1
    while  (np.max(np.abs(analit_func(np.linspace(a, b, n1+1))-func1(a, b,u0, un, n1)))>=eps):
        #print(np.max(np.abs(analit_func(np.linspace(a, b, n1+1))-func1(a, b,u0, un, n1))))
        n1*= 2
        h =(b-a)/n1
    return func1(a, b,u0, un, n1)
```

```
eps = 0.001
res = tridiagonal_matrix_with_eps(a, b,u0, un, eps )
```

4. Построить на одном чертеже графики приближенного и аналитического решений, и график погрешности.

```
fig, axs = plt.subplots(1, 3, figsize = (20, 5))
axs[0].plot(x, analit_func(x), label = 'Аналитическое решение', color = 'g')
axs[0].legend()
axs[0].grid()
axs[1].plot(x, res, label = 'Приближенное решение', color = 'r')
axs[1].legend()
axs[1].grid()


axs[2].plot(x,abs(analit_func(x) -res),color = 'b', label = 'eps')
axs[2].legend()
axs[2].grid()
```



По графикам видно, что аналитическое решение и решение, найденное методом прогонки, визуально совпадают, по графику eps, видно, что требуемая точность 0.001 достигается.

Задача 8.2. Стержень составляется из трех частей одинаковой длины 1 и с разными коэффициентами теплопроводности. Концы стержня поддерживаются при постоянной температуре. В каком порядке следует составить части стержня, чтобы указанная точка x0 стержня имела максимальную температуру?

№	x0	$k1(x)$	$k2(x)$	$k3(x)$	Ua	Ub
8.2.1	1,6	$7-x$	6	15	2	6

Математически задача формулируется следующим образом: найти приближенное решение краевой задачи

$$\begin{cases} -(k(x)u')' + q(x)u = f(x), & x \in (a,b); \\ u(a) = Ua, \quad u(b) = Ub, \end{cases} \quad \text{где} \quad k(x) = \begin{cases} k1(x), & \text{если } 0 \le x \le 1, \\ k2(x), & \text{если } 1 < x \le 2, \\ k3(x), & \text{если } 2 < x \le 3 \end{cases}$$

при каждой конфигурации стержня. Значения $q(x)$и $f(x)$ взять из таблицы 8.1.

Сравнить полученные значения температуры в фиксированной точке в каждом варианте. Выбрать оптимальный результат.

- $\begin{cases} -(k(x)u')' + 0.25u = 0.5x^2 - 0.5x - 3.25 \\ u(0) = 2, \ u(3) = 6 \end{cases}$

ПОРЯДОК РЕШЕНИЯ ЗАДАЧИ

1. Составить подпрограмму, вычисляющую функцию $k(x)$ из индивидуального варианта.

```
def k1(x):
    return 7-x
def k2(x):
    return 6
def k3(x):
    return 15
def k(x):
    if x >= 0 and x <= 1:
        return k1(x)
    elif x <= 2:
        return k2(x)
    elif x <= 3:
        return k3(x)
print(k(1))
```

2. Для каждого варианта конфигурации стержня произвести расчет по разностной схеме с шагом $h = \frac{b-a}{100}$

3. Построить на одном чертеже графики приближенного решения для каждой конфигурации стержня.

4. Сравнив полученные решения, выбрать оптимальный результат.

5. Оформить отчет по задаче.