Решение краевых задач методом разностных аппроксимаций

Цель работы:

- изучить метод разностных аппроксимаций, составить алгоритм метода и программу их реализации, получить численное решение заданной краевой задачи;
- составить алгоритм решения краевых задач указанными методами, применимыми для организации вычислений на ПЭВМ;
- составить программу решения краевых задач по разработанному алгоритму;
- выполнить тестовые примеры и проверить правильность работы программ.

Краткие теоретические сведения

Разностный метод решения краевых задач

Рассмотрим краевую задачу

$$\begin{cases} y'' = f(x, y, y'), \ x \in [a,b], \\ y(a) = A, \\ y(b) = B. \end{cases} \qquad (2.6)$$

Разобьем отрезок [*a, b*] на *n* одинаковых частей с шагом $h = \frac{b-a}{n}$ точками:

$a = x_0 < x_1 < ... < x_n = b$.

Заменим производные на разностные отношения

$$y'(x_k) \approx \frac{y_{k+1} - y_k}{2h},$$
$$y''(x_k) \approx \frac{y_{k+1} - 2y_k + y_{k-1}}{h^2}, \qquad k = \overline{1, n-1},$$

где $y_k = y(x_k)$.

Получим для любого внутреннего узла x_k, $k = \overline{1, n-1}$ уравнение

$$\frac{y_{k+1} - 2y_k + y_{k-1}}{h^2} = f\left(x_k, y_k, \frac{y_{k+1} - y_{k-1}}{2h}\right) \qquad 2.7)$$

и для граничных узлов

$y_0 = A, \qquad\qquad y_n = B$.

То есть, мы имеем систему из ($n+1$) уравнений с ($n+1$) неизвестными y_k. Ее решение дает нам приближенное решение краевой задачи. Рассмотрим частный случай линейной краевой задачи:

$$y'' - p(x)y = f(x), \qquad p(x) > 0, \qquad a \le x \le b, \tag{2.8}$$

$y(a) = A \quad , \qquad y(b) = B$.

В этом случае получаем

$$\frac{y_{k+1} - 2y_k + y_{k-1}}{h^2} - p(x_k)y_k = f(x_k), \qquad k = \overline{1, n-1}, \tag{2.9}$$

$y_0 = A, \qquad\qquad y_n = B$.

Домножая (2.9) на h^2, получим трехдиагональную систему линейных уравнений

$$y_{k-1} - (2 + h^2 p(x_k))y_k + y_{k+1} = h^2 f(x_k), \qquad k = \overline{1, n-1},$$

в которой выполнено условие преобладания диагональных элементов

$2 + p(x_k) > 1 + 1$.

Такая система легко решается методом прогонки.

Задача 1. Составить разностную схему и получить численное решение краевой задачи с точностью 10^{-3}

$$ay'' + (1 + bx^2)y = -1, \qquad -1 \le x \le 1,$$

Исходные данные:

$a = \sin(k), b = \cos(k)$,

где k-номер варианта.

Граничные условия выбрать однородными:

$y(-1) = 0,$
$y(1) = 0.$

Программная реализация

Записываем:

```
def q(x):
    return (1 + b * x * x) / a

def f(x):
    return -1 / a

def p(x):
    return 0
```

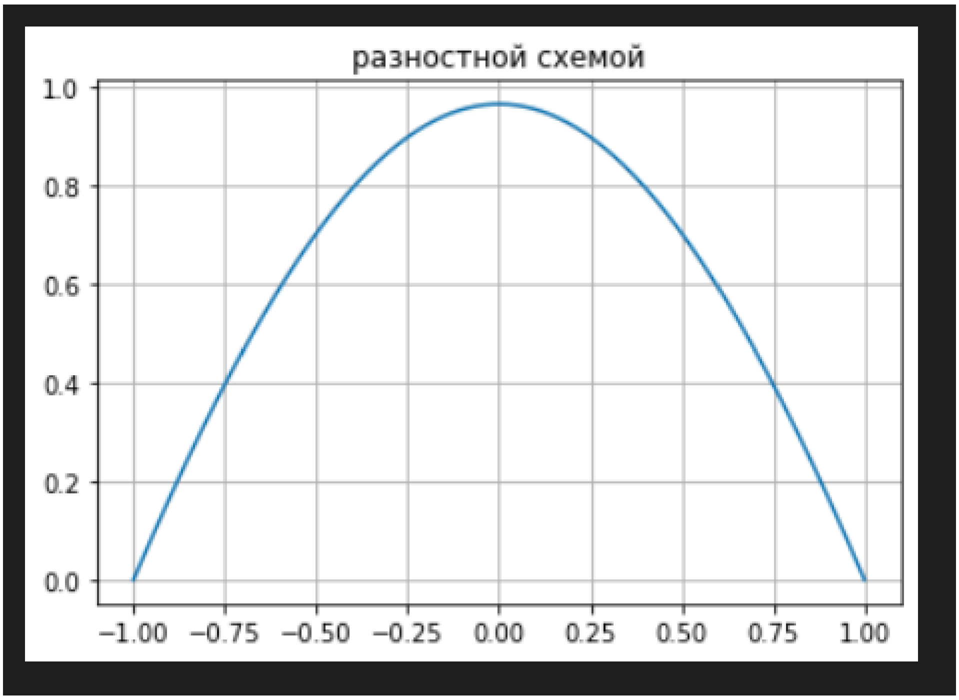
Дописываем:

```
def решить_разностной_схемой(p, q, f, A, B, Ua, Ub, n=2_000, *, check=True):
    if check and n > 2_000:
        n = 2_000

    h = (B - A) / n
    xs = [(round(x, 3) if check else x) for x in np.linspace(A, B, n + 1)]

    diags = [
        [(1 + h * p(x) / 2) for x in xs],
        [(1*h*h*q(x) - 2) for x in xs],
        [(1 - h * p(x) / 2) for x in xs]
    ]

    syst = []
    for _ in range(n+1):
        syst.append([0]*(n+1))

    syst[0][0] = 1
    syst[-1][-1] = 1

    i = 1
    j = 1
    while i < n:
        syst[i][j+1] = diags[2][i+1]    # k+1
        syst[i][j] = diags[1][i]      # k
        syst[i][j-1] = diags[0][i-1]    # k-1
        i += 1
        j += 1

    vec = [h * h * f(x) for x in xs]
    vec[0] = Ua
    vec[-1] = Ub

    ys = np.linalg.solve(syst, vec)

    return xs, ys
```

```python
xs, ys = решить_разностной_схемой(p, q, f, A, B, Ua=0, Ub=0)
n = len(xs)

grid_func = list(zip(xs, [round(y, 3) for y in ys]))

print('(x: y(x))')
print(f'''
{grid_func[:11]}...

...{grid_func[(n+1)//2 - 5: (n+1)//2 + 5 + 1]}...

...{grid_func[-11:]}
''' if len(grid_func) > 10 else grid_func)

plt.plot(xs, ys)
plt.title("разностной схемой")
plt.grid()
plt.show()
```

Фиксируем:

```
[(-1.0, 0.0), (-0.999, 0.002), (-0.998, 0.003), (-0.997, 0.005), (-0.996, 0.007), (-0.995, 0.009), (-0.994, 0.01), (-0.993, 0.012), (-0.992, 0.014), (-0.991, 0.016), (-0.99, 0.017)]...

...[(-0.004, 0.964), (-0.003, 0.964), (-0.002, 0.964), (-0.001, 0.964), (0.0, 0.964), (0.001, 0.964), (0.002, 0.964), (0.003, 0.964), (0.004, 0.964), (0.005, 0.964), (0.006, 0.964)]...

...[(0.99, 0.017), (0.991, 0.016), (0.992, 0.014), (0.993, 0.012), (0.994, 0.01), (0.995, 0.009), (0.996, 0.007), (0.997, 0.005), (0.998, 0.003), (0.999, 0.002), (1.0, 0.0)]
```



Задача 2. Найти приближенное решение краевой задачи методом конечных разностей:

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

с заданной точностью ε и построить его график. Исходные данные указаны в таблице 2.1.

Программная реализация:

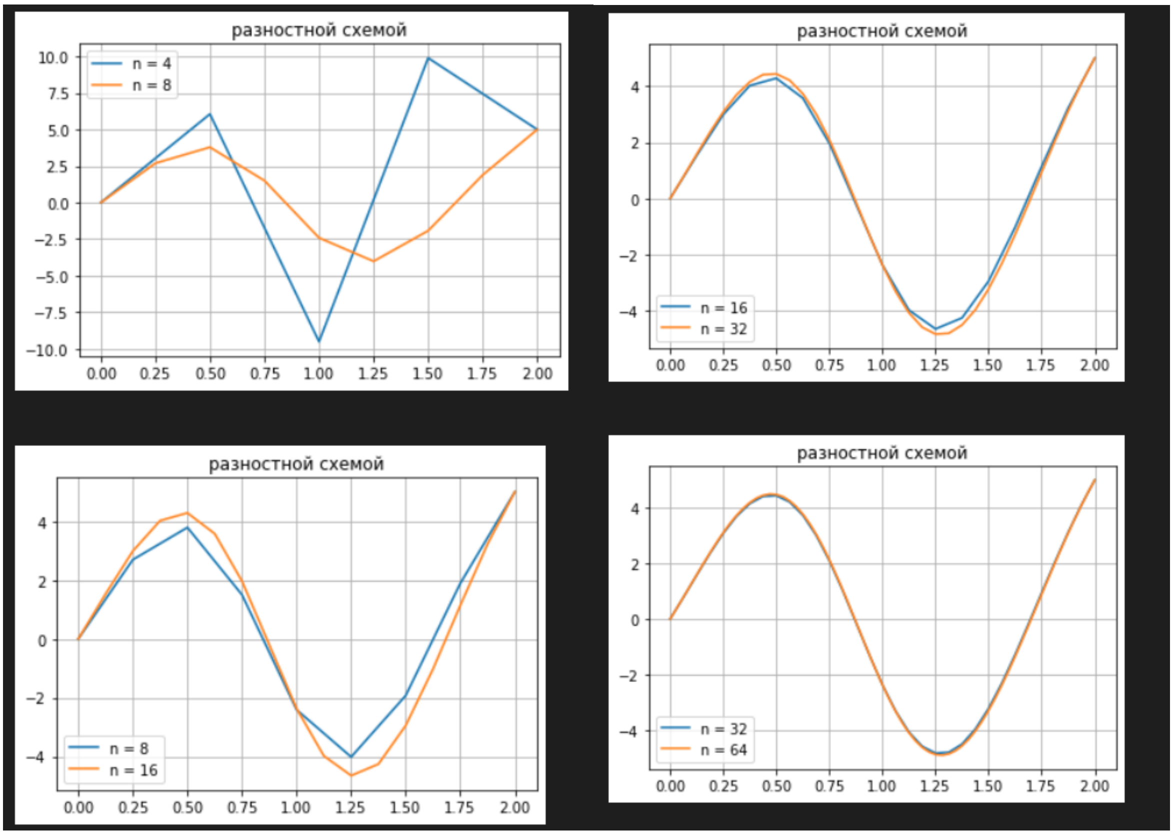
Записываем:

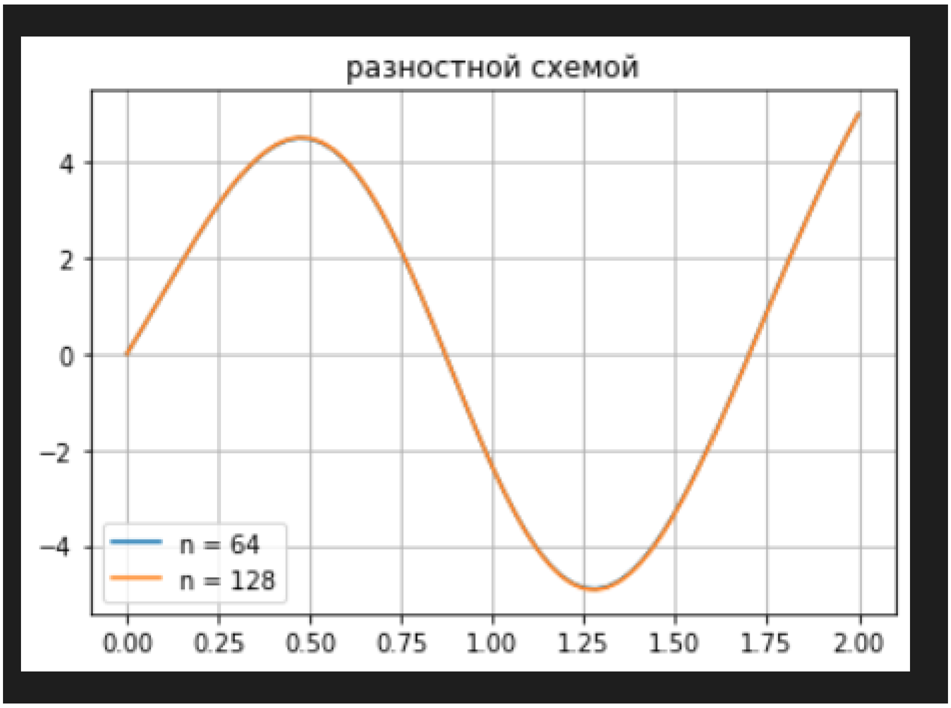
```
# u(a) = Ua
# u(b) = ub

A = 0
B = 2
Ua = 0
Ub = 5
eps = 0.05

def p(x):
    return np.exp(-x*x)

def q(x):
    return 5 * (2 + np.sin(2 * x))

def f(x):
    return np.exp(x) * (1 + np.sin(2 * x))
```

Дописываем:

```
n = 4

xs_pre, ys_pre = решить_разностной_схемой(p, q, f, A, B, Ua, Ub, n=n, check=False)

found = False
while not found:
    n *= 2
    xs_post, ys_post = решить_разностной_схемой(p, q, f, A, B, Ua, Ub, n=n, check=False)

    tmp = ys_post[::2]

    for i in range(1, len(ys_pre) - 1):
        if abs(ys_pre[i] - tmp[i]) > eps:
            found = False
            break
        found = True

    plt.plot(xs_pre, ys_pre, label=f"n = {n//2}")
    plt.plot(xs_post, ys_post, label=f"{n = }")
    plt.title("разностной схемой")
    plt.legend()
    plt.grid()
    plt.show()

    xs_pre = xs_post[:]
    ys_pre = ys_post[:]

    if n > 2000:
        break
```

Фиксируем:





Задача 3. Методом конечных разностей найти приближенное решение указанной в индивидуальном варианте краевой задачи смотри таблицу 2.4 с точностью ε и построить его график. Решение системы разностных уравнений найти, используя метод прогонки.

Программная реализация:

Записываем:

```
A = 0.4
B = 1.4
Ua = 1.2
Ub = ...

eps = 0.05
```

```
def p(x):
    return -0.5 * x

def q(x):
    return 1

def f(x):
    return 2
```

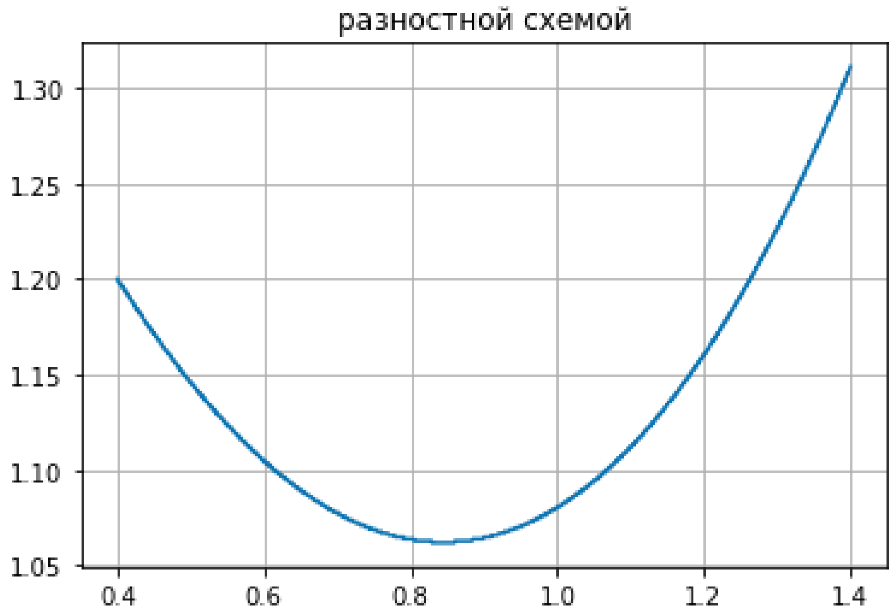
Дописываем:

```python
def решить_разностной_схемой3(p, q, f, A, B, Ua, Ub, n=2_000, *, check=True):
    if check and n > 2_000:
        n = 2_000

    h = (B - A) / n
    xs = [(round(x, 3) if check else x) for x in np.linspace(A, B, n + 1)]

    diags = [
        [(1 - h * p(x) / 2) for x in xs],
        [(1*h*h*q(x) - 2) for x in xs],
        [(1 + h * p(x) / 2) for x in xs],
    ]

    syst = []
    for _ in range(n+1):
        syst.append([0]*(n+1))

    syst[0][0] = 1
    syst[-1][-1] = 1

    i = 1
    j = 1
    while i < n:
        syst[i][j+1] = diags[2][i+1]    # k+1
        syst[i][j] = diags[1][i]      # k
        syst[i][j-1] = diags[0][i-1]    # k-1
        i += 1
        j += 1
    syst[-1][-1] = 1+h
    syst[-1][-2] = -1

    vec = [h * h * f(x) for x in xs]
    vec[0] = Ua
    vec[-1] = 3.2 * h

    ys = np.linalg.solve(syst, vec)

    return xs, ys
```

```python
xs, ys = решить_разностной_схемой3(p, q, f, A, B, Ua, Ub)

plt.plot(xs, ys)
plt.title("разностной схемой")
plt.grid()
plt.show()
```

Фиксируем:

```
(x: y(x))

[(0.4, 1.2), (0.4, 1.2), (0.401, 1.199), (0.402, 1.199), (0.402, 1.199), (0.402, 1.198), (0.403, 1.198), (0.404, 1.198), (0.404, 1.198), (0.404, 1.197), (0.405, 1.197)]...

...[(0.43, 1.183), (0.43, 1.182), (0.43, 1.182), (0.431, 1.182), (0.432, 1.182), (0.432, 1.181), (0.432, 1.181), (0.433, 1.181), (0.434, 1.18), (0.434, 1.18), (0.434, 1.18)]...

...[(1.395, 1.306), (1.395, 1.307), (1.396, 1.307), (1.396, 1.308), (1.397, 1.308), (1.398, 1.309), (1.398, 1.309), (1.398, 1.309), (1.399, 1.31), (1.399, 1.31), (1.4, 1.311)]
```



Задача 4. Методом конечных разностей найти приближенное решение краевой задачи с тремя верными значащими цифрами. Решение системы разностных уравнений найти, используя метод прогонки. Исходные данные указаны в таблице 2.3.

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

Программная реализация:

Записываем:

```
a = 0
b = 1.8
c = 1.275
```

```
def f(x):
    return 8*x*(2 - x*x)
```

```
def k(x):
    if a <= x < c:
        return 0.4
    if c <= x <= b:
        return 1.4
    return 0
```

```
def q(x):
    if a < x < c:
        return 3.2
    if c < x < b:
        return 12
    return 0
```

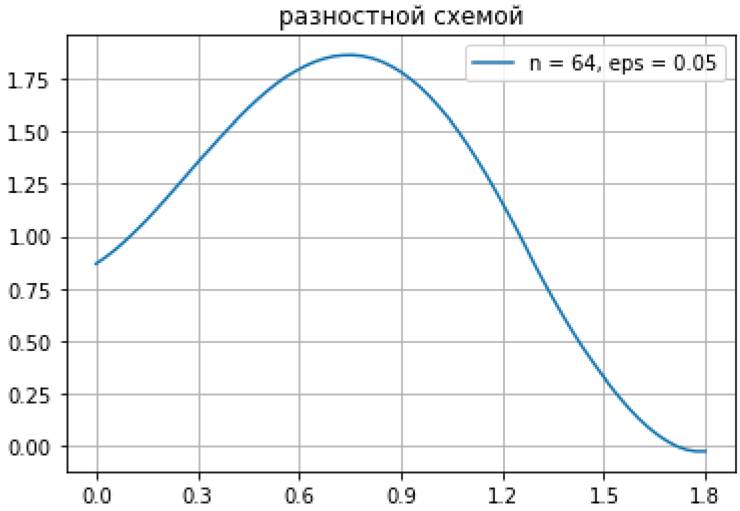
Дописываем:

```
def решить_разностной_схемой4(k, p, q, f, A, B, n=2000):
    if n > 2000:
        n = 2000

    h = (B - A)/n

    xs = [round(x, 3) for x in np.linspace(A, B, n + 1)]

    diags = [
        [(1 - h * p(x) / 2) for x in xs],
        [(1*h*h*q(x) - 2) for x in xs],
        [(1 + h * p(x) / 2) for x in xs],
    ]

    syst = []
    for _ in range(n+1):
        syst.append([0]*(n+1))

    syst[0][0] = h/2+k(A)
    syst[0][1] = -k(A)
    i = 1
    j = 1
    while i < n:
        syst[i][j+1] = diags[2][i+1]    # k+1
        syst[i][j] = diags[1][i]      # k
        syst[i][j-1] = diags[0][i-1]    # k-1
        i += 1
        j += 1
    syst[-1][-1] = k(B) + h
    syst[-1][-2] = -k(B)

    vec = [h * h * f(x) for x in xs]
    vec[0] = 0
    vec[-1] = 0

    ys = [round(y, 3) for y in np.linalg.solve(syst, vec)]

    return xs, ys
```

```
xs, ys = решить_разностной_схемой4(k, q, f, A, B)

plt.plot(xs, ys)
plt.title("разностной схемой")
plt.grid()
plt.show()
```

Фиксируем:

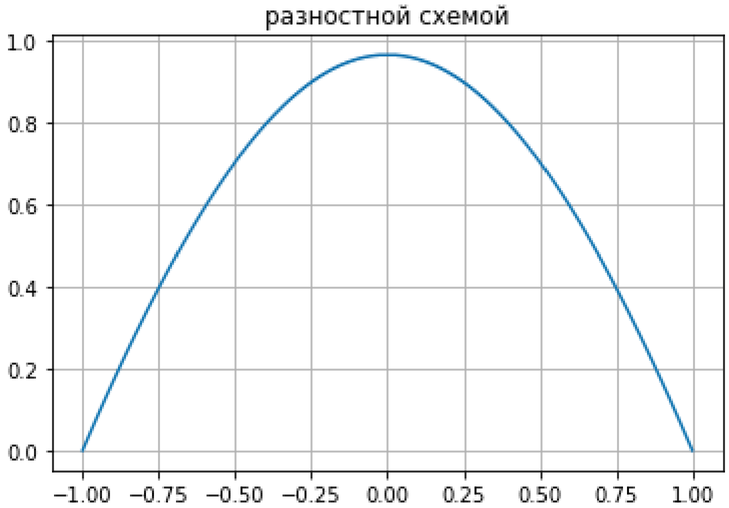
```
(x: y(x))

[(0.0, 0.868), (0.028, 0.899), (0.056, 0.934), (0.084, 0.974), (0.112, 1.017), (0.141, 1.063), (0.169, 1.111), (0.197, 1.161), (0.225, 1.212), (0.253, 1.264), (0.281, 1.316)]...

...[(0.759, 1.863), (0.788, 1.858), (0.816, 1.847), (0.844, 1.831), (0.872, 1.809), (0.9, 1.782), (0.928, 1.749), (0.956, 1.711), (0.984, 1.667), (1.012, 1.617), (1.041, 1.562)]...

...[(1.519, 0.286), (1.547, 0.23), (1.575, 0.178), (1.603, 0.131), (1.631, 0.089), (1.659, 0.053), (1.688, 0.022), (1.716, -0.002), (1.744, -0.018), (1.772, -0.026), (1.8, -0.026)]
```



Тестовые примеры:

```
def q(x):
    return (1 + b * x * x) / a

def f(x):
    return -1 / a

def p(x):
    return 0
```

```
a = np.sin(k)
b = np.cos(k)

A = -1
B = 1
```

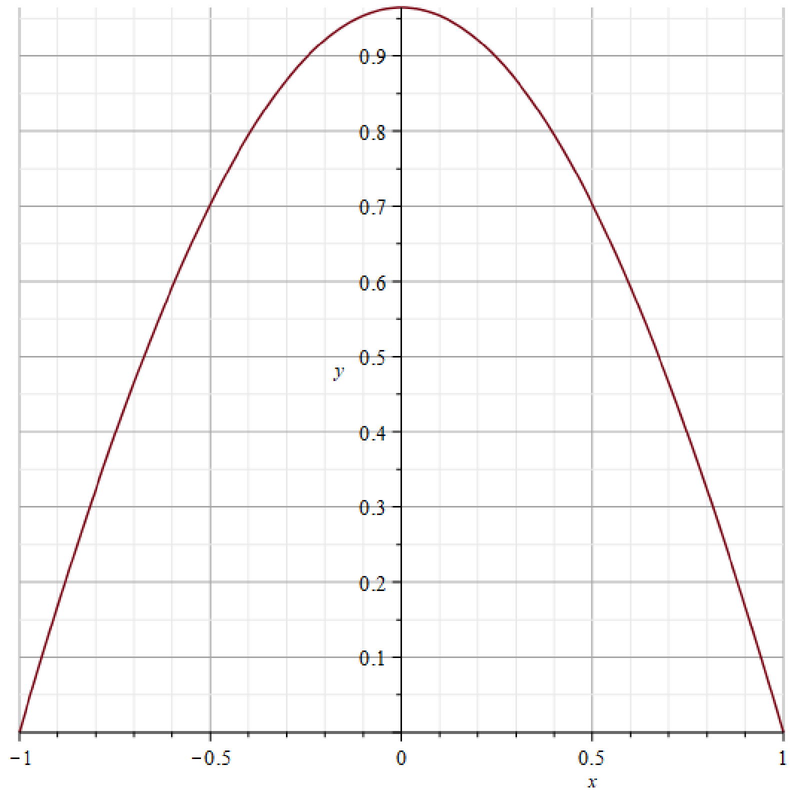


$a := \sin(2)$:

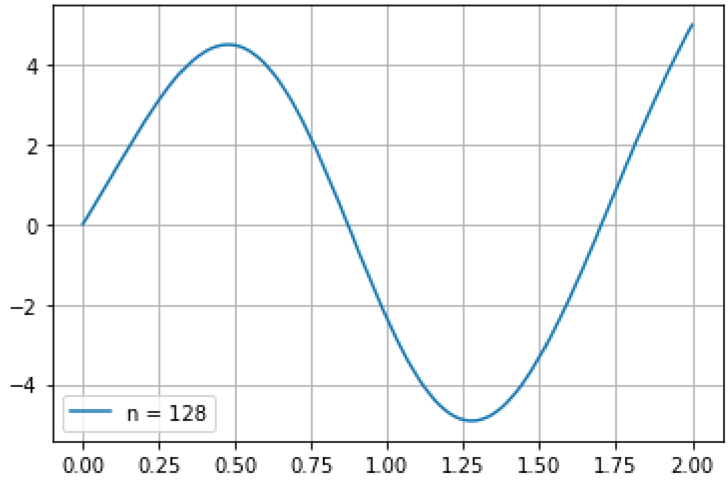
$b := \cos(2)$:

$r := dsolve\left(\left\{y''(x) + \frac{1 + b \cdot x^2}{a} \cdot y(x) = -\frac{1}{a}, y(-1) = 0, y(1) = 0\right\}, numeric\right)$:

$odeplot(r, gridlines)$

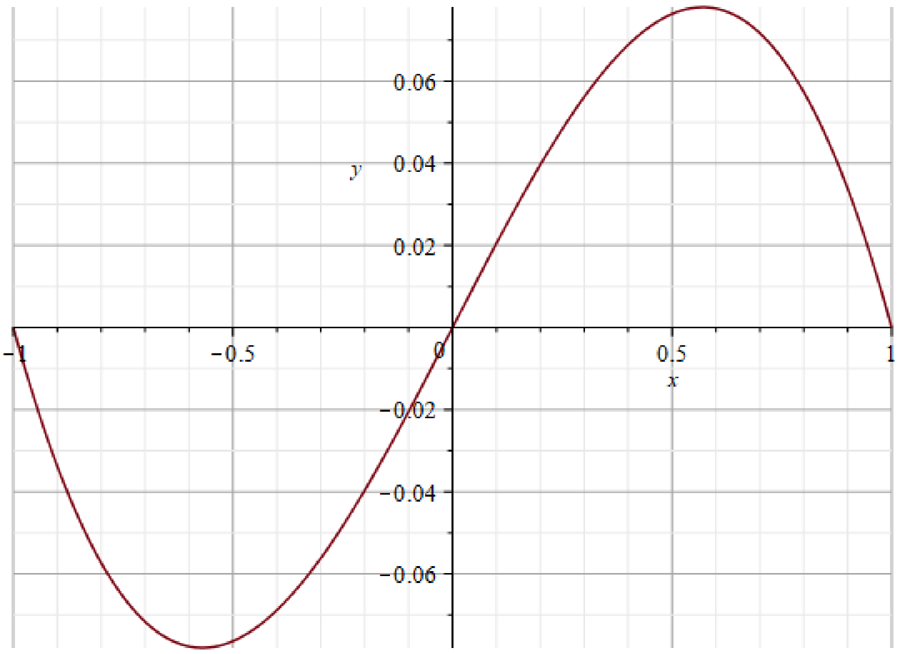


```python
A = 0
B = 2
Ua = 0
Ub = 5
eps = 0.05

def p(x):
    return np.exp(-x*x)

def q(x):
    return 5 * (2 + np.sin(2 * x))

def f(x):
    return np.exp(x) * (1 + np.sin(2 * x))
```



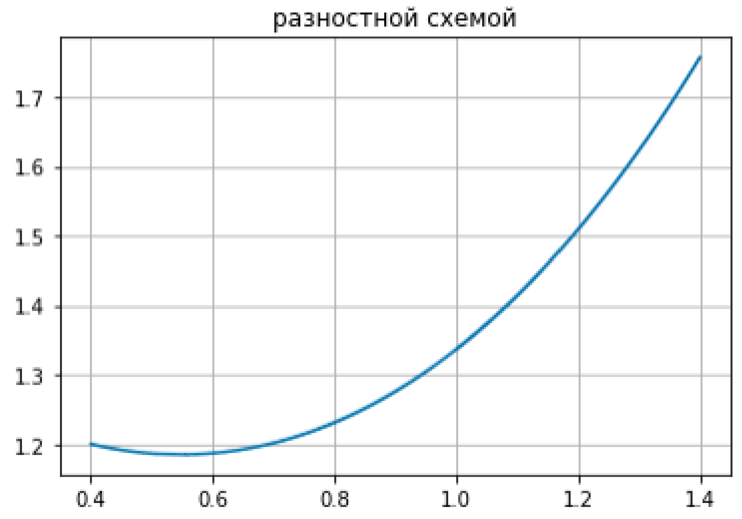
$$r := dsolve\left(\left\{y''(x) + \frac{1 + b\cdot x^2}{a}\cdot y(x) = -\frac{x}{a}, y(-1) = 0, y(1) = 0\right\}, numeric\right):$$

$odeplot(r, gridlines)$



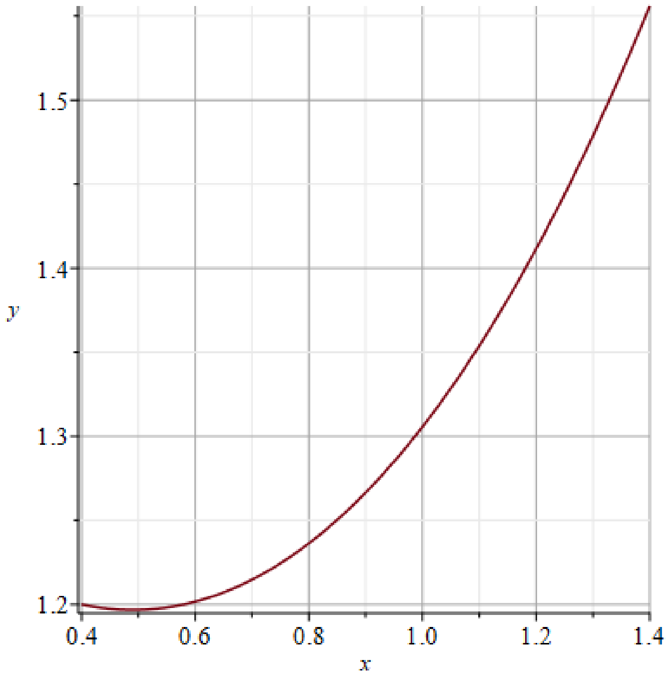
```
A = 0.4
B = 1.4
Ua = 1.2
Ub = ...

eps = 0.05
```

```
def p(x):
    return -0.5 * x

def q(x):
    return 1

def f(x):
    return 2
```



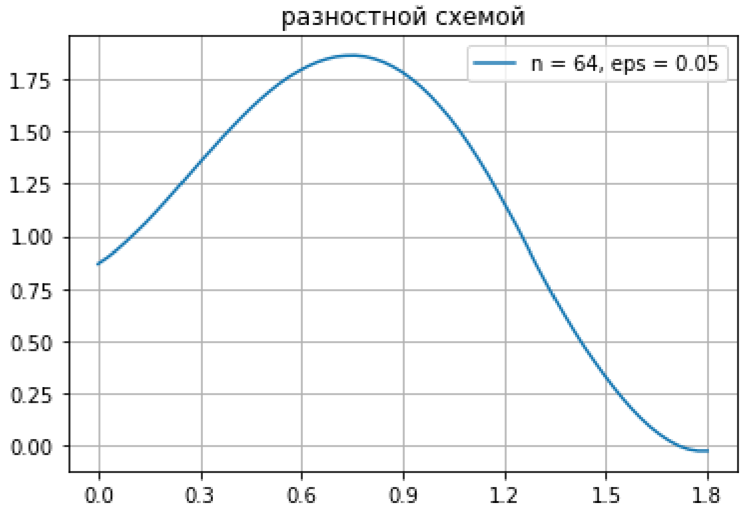
$r := dsolve(\{y''(x) - 0.5 \cdot x \cdot y'(x) + y(x) = 2, y(0.4) = 1.2, y(1.4) + 2 \cdot y'(1.4) = 3.2\}, numeric):$

$odeplot(r, gridlines);$



```python
def k(x):
    if a <= x < c:
        return 0.4
    if c <= x <= b:
        return 1.4
    return 0

def p(x):
    return 0

def q(x):
    res = 0
    if a < x < c:
        res = 3.2
    if c < x < b:
        res =  12
    return res/(-k(x))

def f(x):
    return 8*x*(2 - x*x)/(-k(x))
```

```python
a = 0
b = 1.8
c = 1.275
```



$a := 0:$

$b := 1.8:$

$c := 1.275:$

$k(x) := \mathit{piecewise}(a \le x < c,\ 0.4,\ c \le x \le b,\ 1.4):$

$q(x) := \mathit{piecewise}(a \le x < c,\ 3.2,\ c \le x \le b,\ 12):$

$f(x) := 8 \cdot x \cdot \left(2 - x^2\right):$

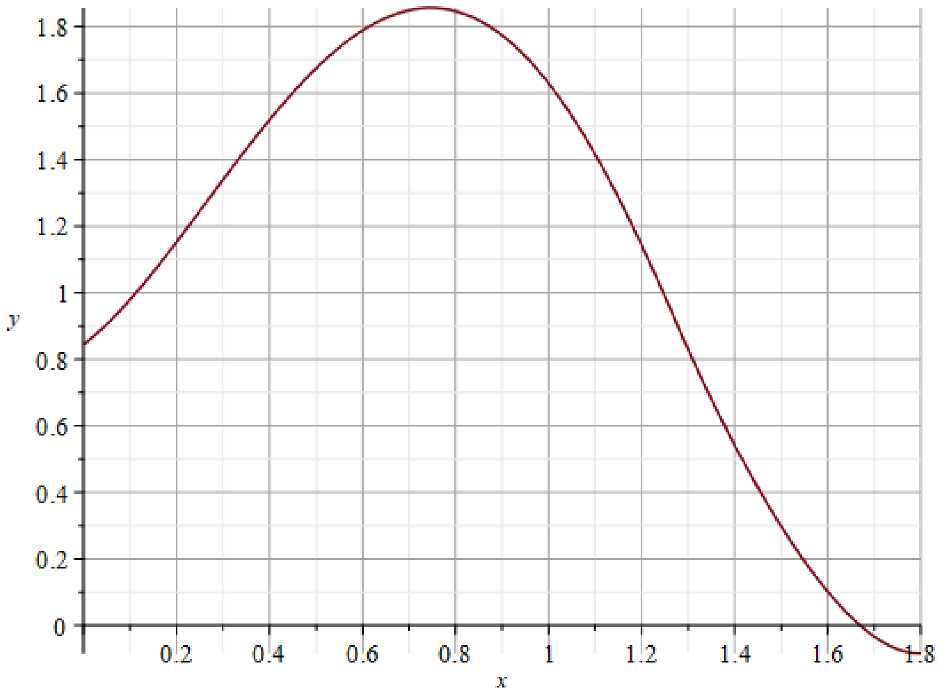
$\mathit{expr} := (-k(x) \cdot y''(x) + q(x) \cdot y(x) = f(x)):$

$c1 := (-k(a) \cdot y'(a) + 0.5 \cdot y(a) = 0):$

$c2 := (k(b) \cdot y'(b) + 0.5 \cdot y(b) = 0):$

$r := \mathit{dsolve}(\{\mathit{expr}, c1, c2\}, \mathit{numeric}, \mathit{abserr} = 0.05):$

$\mathit{odeplot}(r, \mathit{gridlines});$



Заключение: все цели достигнуты.