ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені ІВАНА ФРАНКА
Факультет прикладної математики та інформатики

Методи комп’ютерних обчислень

Лабораторна робота

«Метод скінченних елементів»

Виконав:
Ст. Юрас Назар
Група: ПМІ-32
Оцінка_____
Прийняв:
Василишин Б. Б.

Львів 2022

Варіант-16

Формулювання задачі:

Знайти $u = u(x),\ x \in [0,\ 1]$, таку що

$$\begin{cases} -(\mu\, u')' + \beta u' + \sigma u = f \quad \forall x \in [0,\ 1] \\ u(0) = u_0 \\ -\mu\, u'|_{x=1} = g - k\, u(1) \end{cases}$$

$\mu, \beta,\ \sigma,\ f \in C([0,1]); \quad k, g,\ u_0 \in \mathbb{R}$

Крайові умови:

16. $\mu = x - 1,\ \ \beta = 1,\ \ \sigma = \dfrac{\pi^2}{4},\ \ f = \dfrac{\pi^2}{4} x\, sin\left(\dfrac{\pi x}{2}\right)$

$u(0) = 0 \qquad u(1) = 1$

Хід роботи

1. **Формування варіаційної задачі**

Спочатку нам потрібно сформувати варіаційну задачу. Визначаємо простір V:

$$V = \{u \in C^1([0,1]) : u(0) = u_0\}$$

Домножуємо обидві частини рівняння на тестову функцію v(x) і проінтегруємо на проміжку [0, 1]:

$$\int_0^1 -(\mu u')' v + \beta u' v + \sigma u v dx = \int_0^1 f v dx$$

Використовуючи інтегрування частинами, ми спростимо перший доданок у лівій частині рівняння:

$$\int_0^1 -(\mu u')' v dx = [-\mu u' v]_0^1 + \int_0^1 \mu u' v' dx$$

Тепер, застосовуючи задані граничні умови, маємо:

$$[-\mu u'v]_0^1 = -\mu(1)u_1'v(1) + \mu(0)u_0'v(0) = g - ku_1v(1)$$

враховуючи, що u(0) = u0 і u(1) = 1

Підставивши це у інтегральне рівняння, отримаємо:

$$\int_0^1 \mu u'v' - \beta u'v + \sigma uvdx = \int_0^1 fvdx + ku_1v(1)$$

Нарешті, можемо сформулювати варіаційну задачу:

Знайти u = u(x), x ∈ [0, 1], таку, що u(0) = u0, $-\mu(1)u_1'$ + ku_1 = g для всіх тестових функцій v(x) у просторі Соболєва $H^1([0; 1])$ виконується:

$$\int_0^1 \mu u'v' - \beta u'v + \sigma uvdx = \int_0^1 fvdx + ku_1v(1)$$

where $\mu = x - 1, \beta = 1, \sigma = \frac{\pi^2}{4}$, and $f = \frac{\pi^2}{4} x \sin(\frac{\pi x}{2})$ for the given problem.

2. **Знайдемо білінійну форму:**

Щоб записати білінійну форму задачі, ми починаємо з множення диференціального рівняння в частинних похідних на тестову функцію v(x) та інтегрування на проміжку [0, 1]:

$$-\int_0^1 (mu'(x)v'(x) + \beta u'(x)v(x) + \sigma u(x)v(x))dx = \int_0^1 f(x)v(x)dx$$

Проінтегруємо частинами перший доданок і використаємо граничні умови для отримання:

$$\int_0^1 mu(x)u'(x)v'(x)dx + (g - ku(1)) \int_0^1 mu'(x)v(x)dx + \int_0^1 (\beta u'(x) + \sigma u(x))v(x)dx = \int_0^1 f(x)v(x)dx$$

Тепер визначимо білінійну форму:

$$a(u,v) = \int_0^1 mu(x)u'(x)v'(x)dx + (\beta u'(x) + \sigma u(x))v(x)dx$$

Також додатково визначимо лінійну форму, яка знадобиться нам надалі для визначення лінійного функціоналу:

$$L(v) = \int_0^1 f(x)v(x)dx - (g - ku(1)) \int_0^1 mu'(x)v(x)dx$$

3. **Визначимо лінійний функціонал:**

$$L(u) = \int_0^1 f(x)u(x)dx - (g - ku(1)) \int_0^1 mu'(x)dx$$

і, підставивши коефіцієнти моєї умови, отримаємо

$$L(u) = \int_0^1 \frac{\pi^2}{4} x \sin(\frac{\pi x}{2})u(x)dx - (1 - u(1)) \int_0^1 (x-1)u'(x)dx$$

4. **Знайдемо рішення за допомогою методу скінченних елементів.**

Застосуємо безпосередньо наш метод скінченних елементів: розбиваємо область [0,1] на N підінтервалів розміром h, h = 1 / N:

Далі визначаємо простір функцій, які апроксимують розв'язок задачі. Як я вже писав вище, використовуємо лінійні кусково-лінійні функції:

$$V_h = \{v_h \in C^0[0,1] : v_h|_{[x_{i-1},x_i]} \in \mathbb{P}^1, i = 1,2,...,N\}$$

Тепер нам треба підставити апроксимуючі функції та функцію розв'язку у визначений раніше білінійний та лінійний функціонали:

$$a(u_h, v_h) = \int_0^1 m u_h'(x) v_h'(x) dx + (\beta u_h'(x) + \sigma u_h(x)) v_h(x) dx$$

$$a(u_h, v_h) = \sum_{i=1}^{N-1} \int_{x_{i-1}}^{x_i} m(x) u_h'(x) v_h'(x) dx + \int_0^1 (\beta u_h'(x) + \sigma u_h(x)) v_h(x) dx$$

$$L(v_h) = \int_0^1 f(x) v_h(x) dx + g v_h(1) - k u_0 v_h'(1)$$

$$L(v_h) = \sum_{i=1}^{N-1} \int_{x_{i-1}}^{x_i} f(x) v_h(x) dx + g v_h(1) - k u_0 v_h'(1)$$

Тепер, розв'язок знаходимо у вигляді лінійної комбінації базисних функцій:

$$u_h(x) = \sum_{j=1}^{N} u_j \varphi_j(x)$$

, де $\boldsymbol{\varphi}_j$ – базисна функція

Підставимо вираз для $u_h(x)$ у визначений раніше білінійний та лінійний функціонали та після інтегрування частинами одержимо систему лінійних алгебраїчних рівнянь:

Au = f

Оскільки коефіцієнти утворюють тридіагональну матрицю, можна застосувати будь-який ефективний алгоритм розв’язування СЛАР, проте ми застосуємо метод Томаса (прогонки), оскільки у нього досить хороша алгоритмічна складність (O(n)).

Після знаходження вектора невідомих розв’язок задачі можна знайти як

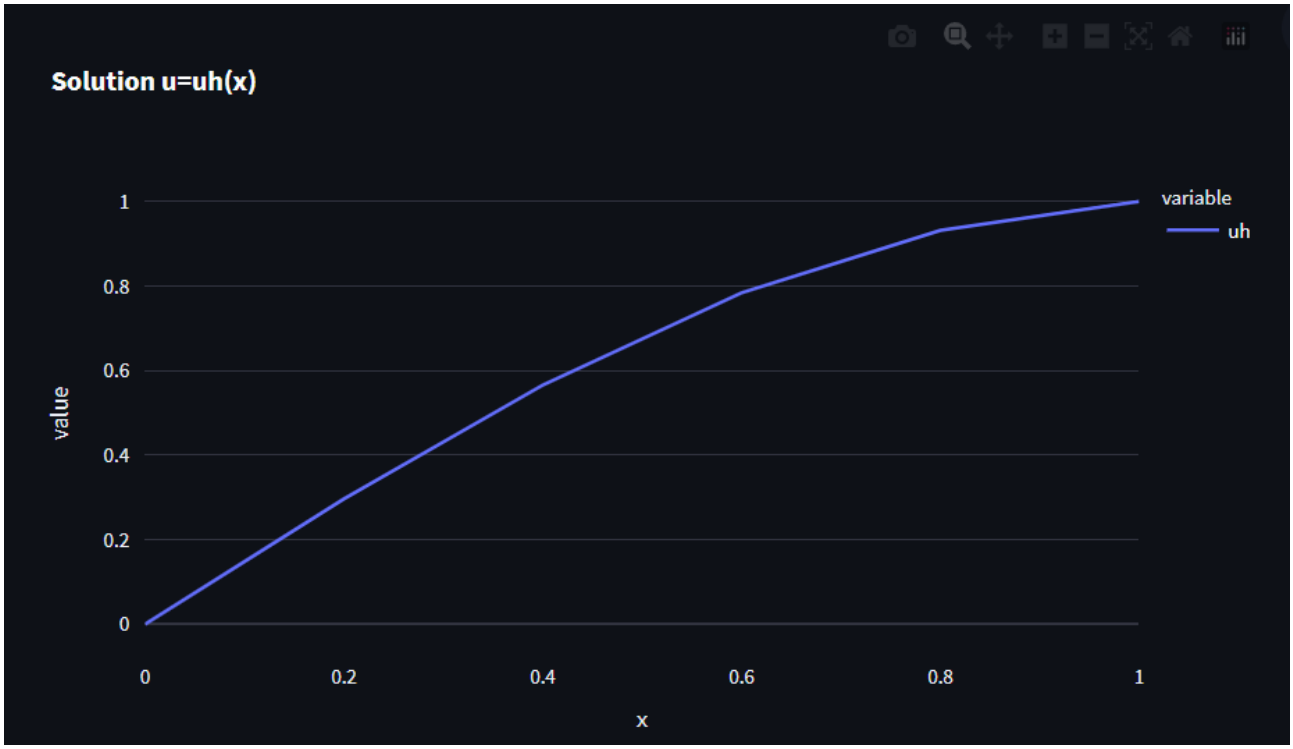
$$u_h(x) = \sum_{j=1}^{N} u_j \varphi_j(x)$$

, де $u_h(x)$ – функція апроксимації

Далі просто будуємо її графік для певної кількості вузлів

Аналіз результатів роботи програми (графіки, таблиці)

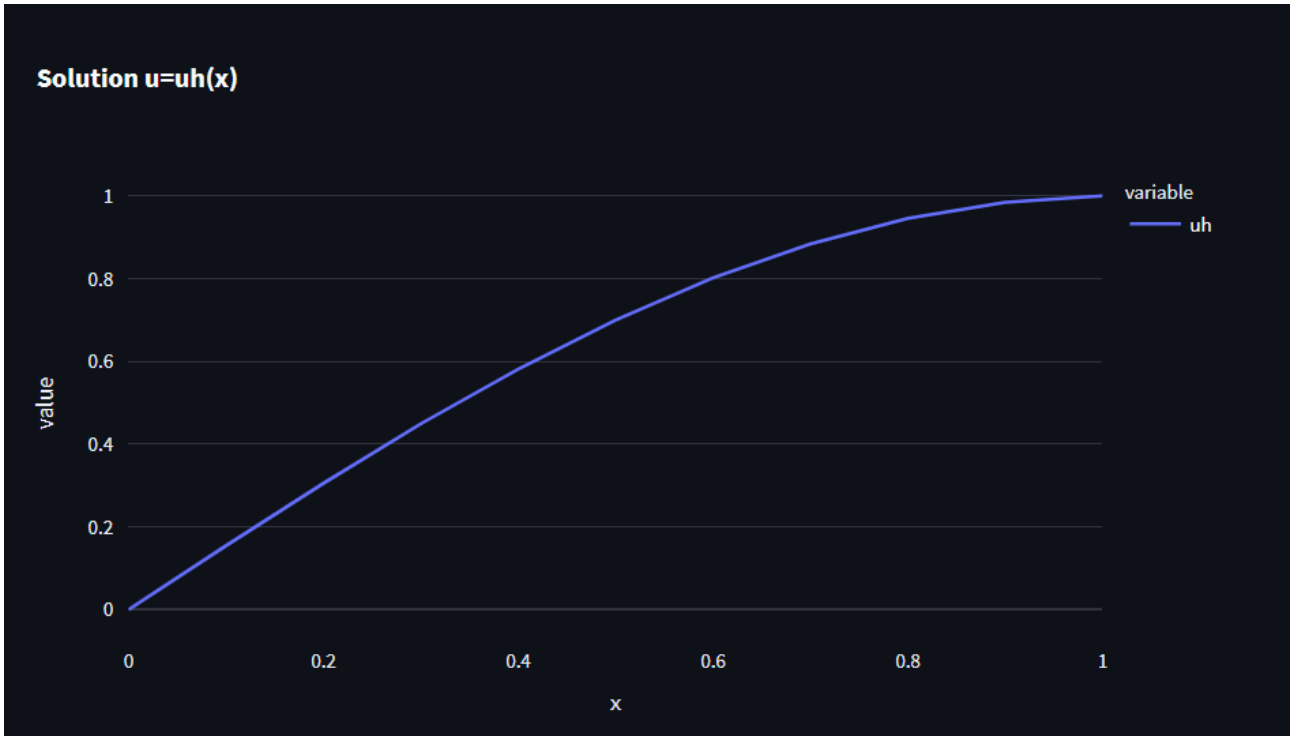
Графік функції u_h при n = 5 вузлах:



Таблиця похибок та норм при n = 5 вузлах:

	n	Incoherence	Dirichlet Estimator	Neumann Estimator	Energy Norm	Sobolev Norm	Order of convergence
0	5	0.3222	-0.0347	-0.0256	0.8573	1.6424	None
1	10	0.1626	-0.012	-0.0089	0.864	1.7063	None
2	20	0.0815	-0.0038	-0.0029	0.8661	1.7264	None
3	40	0.0408	-0.0012	-0.0009	0.8667	1.7318	1.1135
4	80	0.0204	-0.0003	-0.0003	0.8668	1.7332	1.0384
5	160	0.0102	-0.0001	-0.0001	0.8668	1.7336	1.0132
6	320	0.0051	0	0	0.8668	1.7337	1.0048
7	640	0.0025	0	0	0.8668	1.7337	1.0019
8	1,280	0.0013	0	0	0.8669	1.7337	1.0008
9	2,560	0.0006	0	0	0.8669	1.7337	1.0005

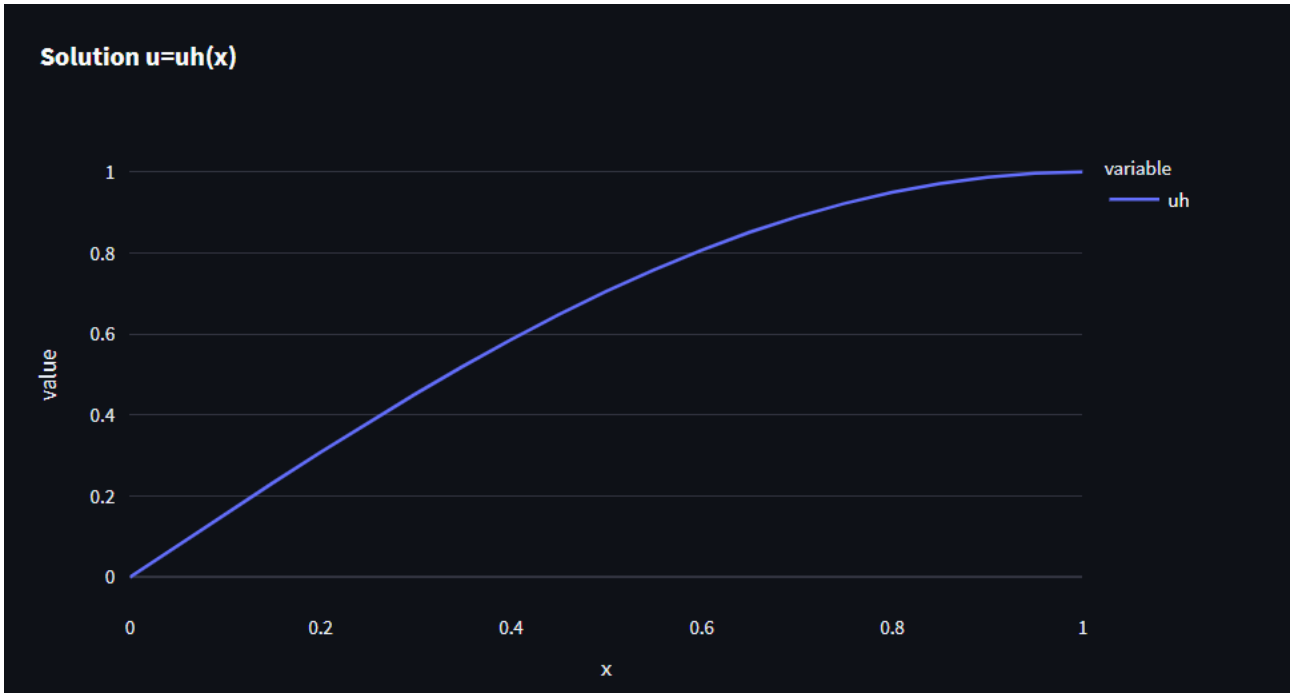
Графік функції u_h при n = 10 вузлах:



Таблиця похибок та норм при n = 10 вузлах:

	n	Incoherence	Dirichlet Estimator	Neumann Estimator	Energy Norm	Sobolev Norm	Order of convergence
0	10	0.1626	-0.012	-0.0089	0.864	1.7063	None
1	20	0.0815	-0.0038	-0.0029	0.8661	1.7264	None
2	40	0.0408	-0.0012	-0.0009	0.8667	1.7318	None
3	80	0.0204	-0.0003	-0.0003	0.8668	1.7332	1.0384
4	160	0.0102	-0.0001	-0.0001	0.8668	1.7336	1.0132
5	320	0.0051	0	0	0.8668	1.7337	1.0048
6	640	0.0025	0	0	0.8668	1.7337	1.0019
7	1,280	0.0013	0	0	0.8669	1.7337	1.0008
8	2,560	0.0006	0	0	0.8669	1.7337	1.0005
9	5,120	0.0003	0	0	0.8669	1.7337	1.0017

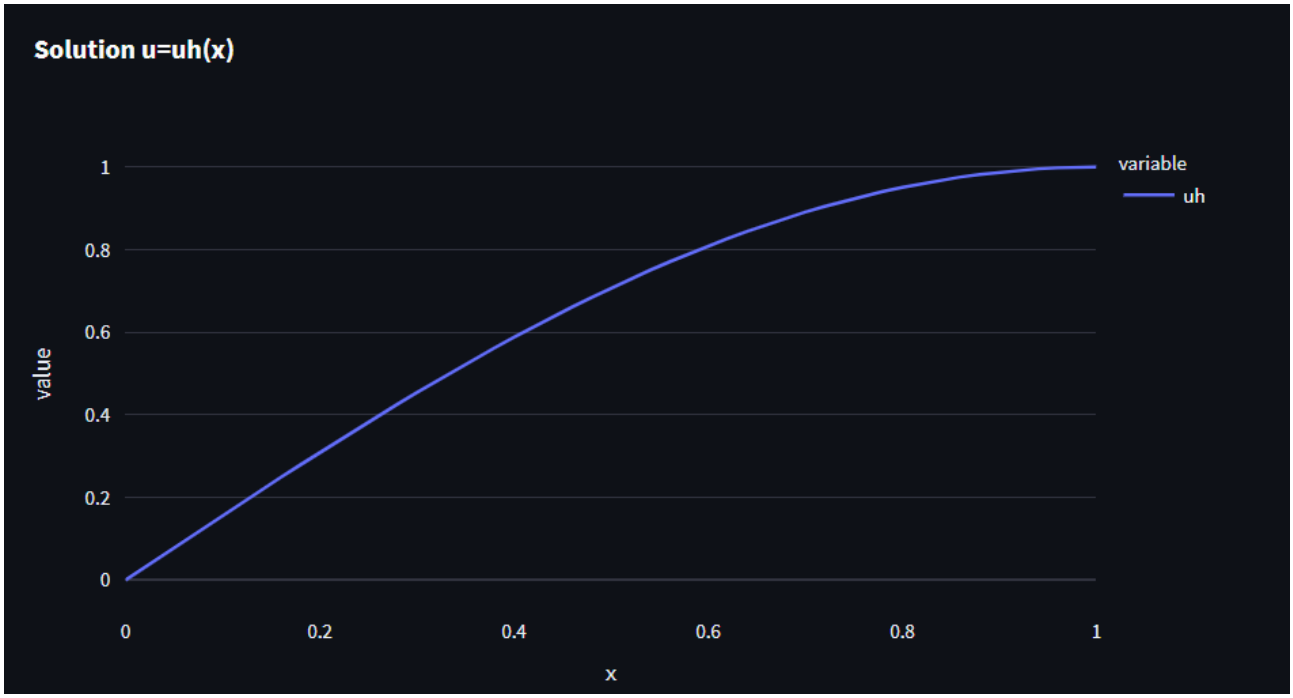
Графік функції u_h при n = 20 вузлах:



Таблиця похибок та норм при n = 20 вузлах:

	n	Incoherence	Dirichlet Estimator	Neumann Estimator	Energy Norm	Sobolev Norm	Order of convergence
0	20	0.0815	-0.0038	-0.0029	0.8661	1.7264	None
1	40	0.0408	-0.0012	-0.0009	0.8667	1.7318	None
2	80	0.0204	-0.0003	-0.0003	0.8668	1.7332	None
3	160	0.0102	-0.0001	-0.0001	0.8668	1.7336	1.0132
4	320	0.0051	0	0	0.8668	1.7337	1.0048
5	640	0.0025	0	0	0.8668	1.7337	1.0019
6	1,280	0.0013	0	0	0.8669	1.7337	1.0008
7	2,560	0.0006	0	0	0.8669	1.7337	1.0005
8	5,120	0.0003	0	0	0.8669	1.7337	1.0017
9	10,240	0.0002	0	0	0.8669	1.7337	0.9772

Графік функції u_h при n = 50 вузлах:



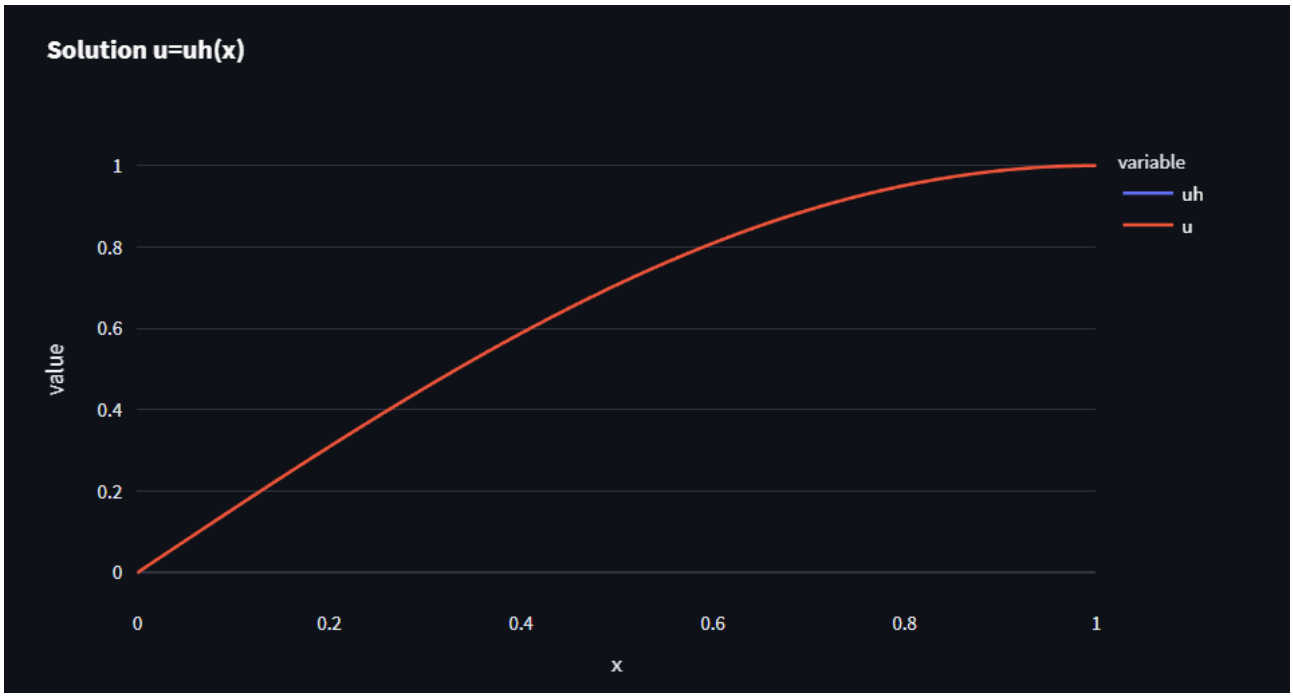
Таблиця похибок та норм при n = 50 вузлах:

	n	Incoherence	Dirichlet Estimator	Neumann Estimator	Energy Norm	Sobolev Norm	Order of convergence
0	50	0.0326	-0.0008	-0.0006	0.8667	1.7325	None
1	100	0.0163	-0.0002	-0.0002	0.8668	1.7334	None
2	200	0.0082	-0.0001	-0.0001	0.8668	1.7336	None
3	400	0.0041	0	0	0.8668	1.7337	1.0035
4	800	0.002	0	0	0.8668	1.7337	1.0014
5	1,600	0.001	0	0	0.8669	1.7337	1.0006
6	3,200	0.0005	0	0	0.8669	1.7337	1.0004
7	6,400	0.0003	0	0	0.8669	1.7337	0.9951
8	12,800	0.0001	0	0	0.8669	1.7337	1.1137
9	25,600	0.0001	0	0	0.8669	1.7337	0.789

Як видно на скрінах графіків функції апроксимації, n = 5 трохи замало для нашого розв’язку, оскільки графік зростає не дуже плавно і в ньому видно переломи, а вже при n = 10 вузлах наш графік вже виглядає плавним і досить точно відтворює розв’язок нашої задачі. В залежності від бажаної точності вже можна задуматись, чи варто обчислювати задачу для більшої кількості вузлів і чи варто витрачати обчислювальні ресурси системи.

Також, проаналізувавши скріни табличок похибок та норм, можна зробити висновок, що вже при n = 160 вузлах оцінювач Діріхле, оцінювач Неймана, а також енергетична норма та норма Соболєва досягають своїх пікових значень для моєї задачі і якщо нам не потрібно обчислювати більш точно нев’язку та порядок збіжності, то можна зупинитись на n = 160 для економії обчислювальних ресурсів нашої системи.

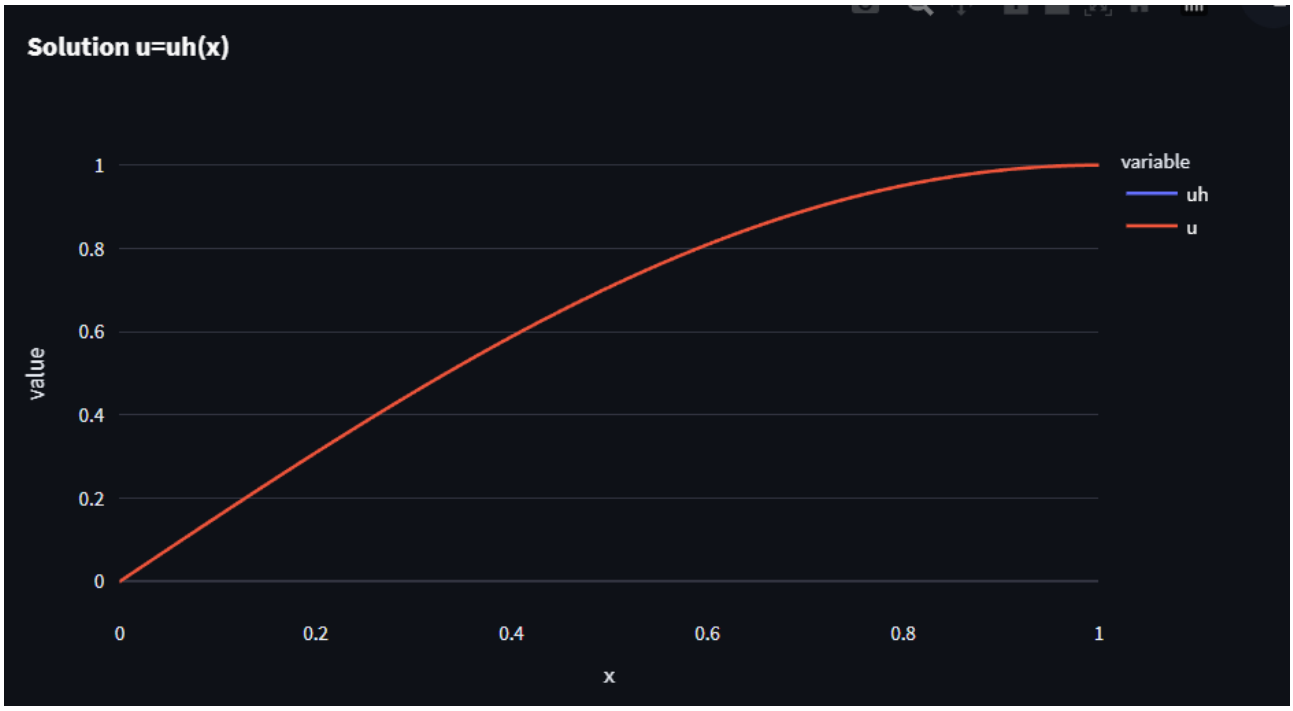
Графік функції u_h при n = 5 вузлах з точним розв’язком:



Таблиця похибок та норм при n = 5 вузлах з точним розв’язком:

	n	Incoherence	Dirichlet Estimator	Neumann Estimator	Energy Norm	Sobolev Norm	[EXACT] Energy Norm	[EXACT] Sobolev Norm	Energy Norm Diff	Sobolev Norm Diff	Order of convergence
0	5	0.3222	-0.0347	-0.0256	0.8573	1.6424	0.8581	1.7154	0	0	None
1	10	0.1626	-0.012	-0.0089	0.864	1.7063	0.8646	1.7291	0	0	None
2	20	0.0815	-0.0038	-0.0029	0.8661	1.7264	0.8663	1.7326	0	0	None
3	40	0.0408	-0.0012	-0.0009	0.8667	1.7318	0.8667	1.7334	0	0	1.1135
4	80	0.0204	-0.0003	-0.0003	0.8668	1.7332	0.8668	1.7336	0	0	1.0384
5	160	0.0102	-0.0001	-0.0001	0.8668	1.7336	0.8668	1.7337	0	0	1.0132
6	320	0.0051	0	0	0.8668	1.7337	0.8668	1.7337	0	0	1.0048
7	640	0.0025	0	0	0.8668	1.7337	0.8668	1.7337	0	0	1.0019
8	1,280	0.0013	0	0	0.8669	1.7337	0.8669	1.7337	0	0	1.0008
9	2,560	0.0006	0	0	0.8669	1.7337	0.8669	1.7337	0	0	1.0005

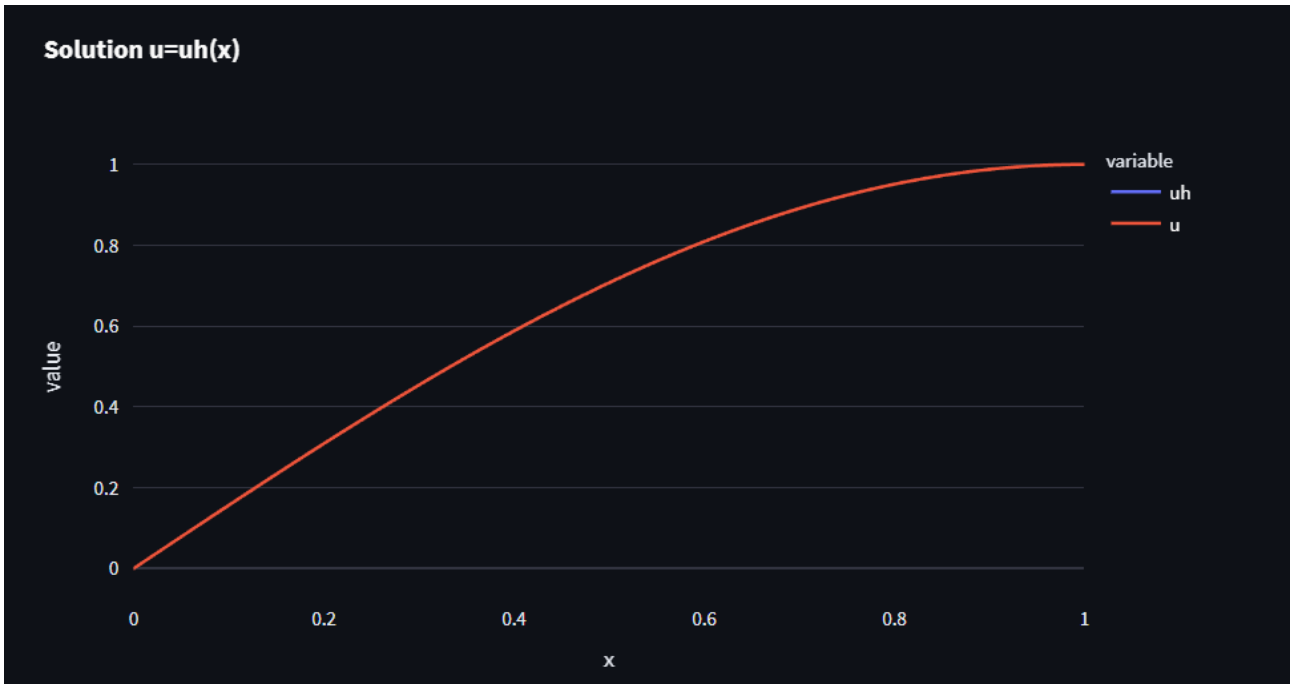
Графік функції u_h при n = 10 вузлах з точним розв’язком:



Таблиця похибок та норм при n = 10 вузлах з точним розв’язком:

	n	Incoherence	Dirichlet Estimator	Neumann Estimator	Energy Norm	Sobolev Norm	[EXACT] Energy Norm	[EXACT] Sobolev Norm	Energy Norm Diff	Sobolev Norm Diff	Order of convergence
0	10	0.1626	-0.012	-0.0089	0.864	1.7063	0.8646	1.7291	0	0	None
1	20	0.0815	-0.0038	-0.0029	0.8661	1.7264	0.8663	1.7326	0	0	None
2	40	0.0408	-0.0012	-0.0009	0.8667	1.7318	0.8667	1.7334	0	0	None
3	80	0.0204	-0.0003	-0.0003	0.8668	1.7332	0.8668	1.7336	0	0	1.0384
4	160	0.0102	-0.0001	-0.0001	0.8668	1.7336	0.8668	1.7337	0	0	1.0132
5	320	0.0051	0	0	0.8668	1.7337	0.8668	1.7337	0	0	1.0048
6	640	0.0025	0	0	0.8668	1.7337	0.8668	1.7337	0	0	1.0019
7	1,280	0.0013	0	0	0.8669	1.7337	0.8669	1.7337	0	0	1.0008
8	2,560	0.0006	0	0	0.8669	1.7337	0.8669	1.7337	0	0	1.0005
9	5,120	0.0003	0	0	0.8669	1.7337	0.8669	1.7337	0	0	1.0017

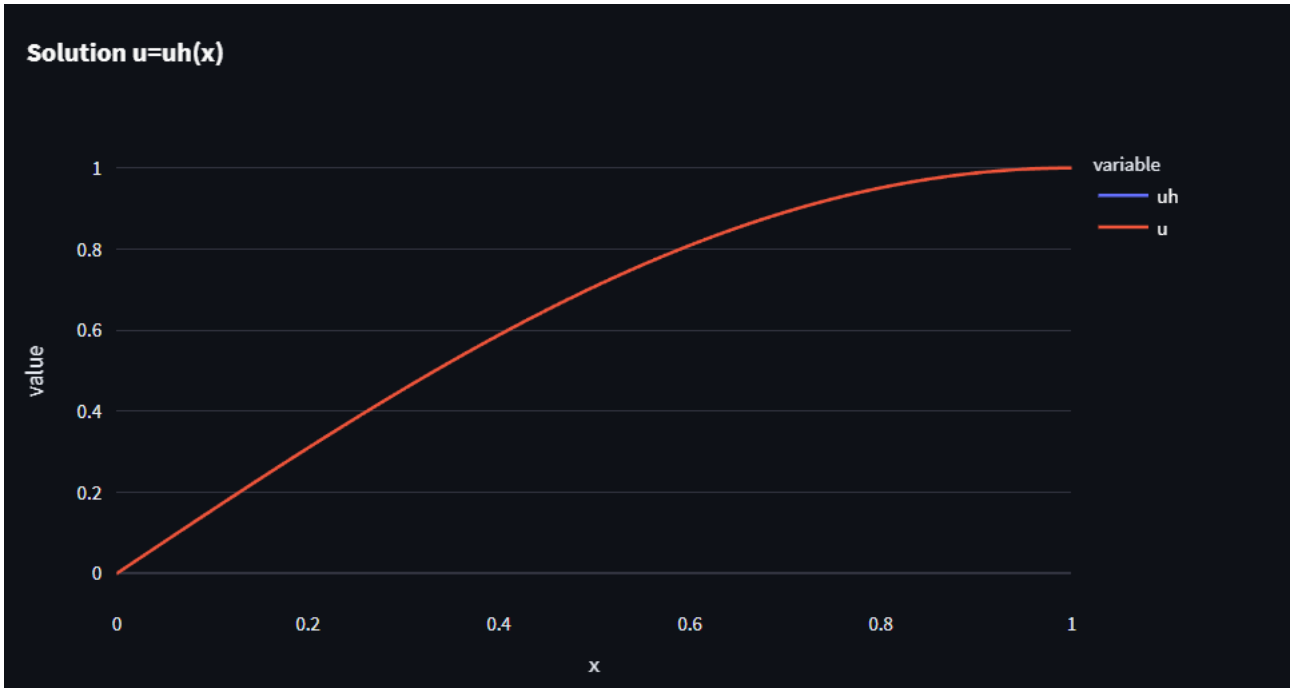
Графік функції u_h при n = 20 вузлах з точним розв’язком:



Таблиця похибок та норм при n = 20 вузлах з точним розв’язком:

	n	Incoherence	Dirichlet Estimator	Neumann Estimator	Energy Norm	Sobolev Norm	[EXACT] Energy Norm	[EXACT] Sobolev Norm	Energy Norm Diff	Sobolev Norm Diff	Order of convergence
0	20	0.0815	-0.0038	-0.0029	0.8661	1.7264	0.8663	1.7326	0	0	None
1	40	0.0408	-0.0012	-0.0009	0.8667	1.7318	0.8667	1.7334	0	0	None
2	80	0.0204	-0.0003	-0.0003	0.8668	1.7332	0.8668	1.7336	0	0	None
3	160	0.0102	-0.0001	-0.0001	0.8668	1.7336	0.8668	1.7337	0	0	1.0132
4	320	0.0051	0	0	0.8668	1.7337	0.8668	1.7337	0	0	1.0048
5	640	0.0025	0	0	0.8668	1.7337	0.8668	1.7337	0	0	1.0019
6	1,280	0.0013	0	0	0.8669	1.7337	0.8669	1.7337	0	0	1.0008
7	2,560	0.0006	0	0	0.8669	1.7337	0.8669	1.7337	0	0	1.0005
8	5,120	0.0003	0	0	0.8669	1.7337	0.8669	1.7337	0	0	1.0017
9	10,240	0.0002	0	0	0.8669	1.7337	0.8669	1.7337	0	0	0.9772

Графік функції u_h при n = 50 вузлах з точним розв’язком:



Таблиця похибок та норм при n = 50 вузлах з точним розв’язком:

	n	Incoherence	Dirichlet Estimator	Neumann Estimator	Energy Norm	Sobolev Norm	[EXACT] Energy Norm	[EXACT] Sobolev Norm	Energy Norm Diff	Sobolev Norm Diff	Order of convergence
0	50	0.0326	-0.0008	-0.0006	0.8667	1.7325	0.8668	1.7335	0	0	None
1	100	0.0163	-0.0002	-0.0002	0.8668	1.7334	0.8668	1.7337	0	0	None
2	200	0.0082	-0.0001	-0.0001	0.8668	1.7336	0.8668	1.7337	0	0	None
3	400	0.0041	0	0	0.8668	1.7337	0.8668	1.7337	0	0	1.0035
4	800	0.002	0	0	0.8668	1.7337	0.8668	1.7337	0	0	1.0014
5	1,600	0.001	0	0	0.8669	1.7337	0.8669	1.7337	0	0	1.0006
6	3,200	0.0005	0	0	0.8669	1.7337	0.8669	1.7337	0	0	1.0004
7	6,400	0.0003	0	0	0.8669	1.7337	0.8669	1.7337	0	0	0.9951
8	12,800	0.0001	0	0	0.8669	1.7337	0.8669	1.7337	0	0	1.1137
9	25,600	0.0001	0	0	0.8669	1.7337	0.8669	1.7337	0	0	0.789

Як видно з результатів, мій графік функції апроксимації та графік точного розв’язку повністю співпадають, також це підтверджують різниці норм у таблицях. Проте хотілося б відмітити, що при малій кількості вузлів, наприклад n = 5, різниця між графіками все ж є, хоча її на перший погляд і не видно. Наглядно продемонструю це, збільшивши масштаб графіків:

Різниця аж у шостому значенні після коми, що, на мою думку, не дуже критично, але все ж вона є.

Трохи деталей, як реалізовані деякі формули величин з таблиці у коді:

1. **Обчислення норми Соболєва:**

```python
def calculate(self, nodes, values) -> np.ndarray:
    n = len(nodes)
    h = 1.0 / (n - 1)
    norm = np.zeros(n-1, dtype=np.float64)

    for i in range(n-1):
        el_vals = values[i:i+2]

        norm[i] += 1/h * np.dot(np.dot(el_vals, self._mu_matrix), el_vals)
        norm[i] += h/6 * np.dot(np.dot(el_vals, self._sigma_matrix), el_vals)

    return norm
```

2. **Обчислення енергетичної норми:**

```python
def calculate(self, nodes, values) -> np.ndarray:
    n = len(nodes)
    h = 1.0 / (n - 1)
    norm = np.zeros(n - 1, dtype=np.float64)

    for i in range(n - 1):
        el_cent = nodes[i:i + 2].mean()
        el_vals = values[i:i + 2]

        mu = self._pdeproblem.mu(el_cent)
        beta = self._pdeproblem.beta(el_cent)
        sigma = self._pdeproblem.sigma(el_cent)
        norm[i] += mu * 1 / h * np.dot(np.squeeze(np.dot(el_vals, np.squeeze(self._mu_matrix))), el_vals)
        norm[i] += beta * 1 / 2 * np.dot(np.squeeze(np.dot(el_vals, np.squeeze(self._beta_matrix))), el_vals)
        norm[i] += sigma * h / 6 * np.dot(np.squeeze(np.dot(el_vals, np.squeeze(self._sigma_matrix))), el_vals)
    return norm
```

3. Обчислення порядку збіжності:

```
if i >= 3:
    numerator = math.log2(abs((energy_norms[i] - energy_norms[i - 1]) / (
            energy_norms[i - 1] - energy_norms[i - 2])))
    denominator = math.log2(abs((energy_norms[i - 1] - energy_norms[i - 2]) / (
            energy_norms[i - 2] - energy_norms[i - 3])))
    q = numerator / denominator
    q_list[i] = q
```

4. Обчислення нев'язки:

```
def calculate(self, nodes, values) -> np.ndarray:
    n = len(nodes)
    h = 1. / (n-1)
    incoherence = np.zeros(n-1, dtype=np.float64)

    for i in range(n-1):
        el_cent = nodes[i:i+2].mean()
        el_vals = values[i:i+2]

        beta = self._pdeproblem.beta(el_cent)
        sigma = self._pdeproblem.sigma(el_cent)
        f = self._pdeproblem.f(el_cent)

        incoherence[i] += beta * 1 / h * np.dot(self._beta_matrix, el_vals)
        incoherence[i] += sigma * 1 / 2 * np.dot(self._sigma_matrix, el_vals)
        incoherence[i] += f - incoherence[i]

        incoherence[i] = np.power(incoherence[i], 2)
        incoherence[i] = incoherence[i] * (h * h)

    return incoherence
```

5. Обчислення оцінювача Діріхле:

```
def calculate(self, nodes, elements, values) -> np.ndarray:
    n = len(nodes)
    h = 1.0 / (n - 1)
    err = np.zeros(n - 1, dtype=np.float64)
    denom = np.zeros(n - 1, dtype=np.float64)

    for i, elem in enumerate(elements):
        el_cent = nodes[elem].mean()
        el_vals = values[elem]

        mu = self._pdeProblem.mu(el_cent)
        beta = self._pdeProblem.beta(el_cent)
        sigma = self._pdeProblem.sigma(el_cent)
        f = self._pdeProblem.f(el_cent)

        denom[i] = (8 * mu) / (15 * h) * (10 + h**2 * sigma / mu)
        err[i] += beta * 1/h * np.dot(self._beta_matrix, el_vals)
        err[i] += sigma * 1/2 * np.dot(self._sigma_matrix, el_vals)
        err[i] = (f - err[i]) * h * 2/3

    return np.power(err, 2) / denom
```

6. Обчислення оцінювача Неймана:

```
def calculate(self, nodes: np.ndarray, elements: np.ndarray, values: np.ndarray):
    n = len(nodes)
    h = 1.0 / (n - 1)
    err = np.zeros(n - 1, dtype=np.float64)
    denom = np.zeros(n - 1, dtype=np.float64)

    for i, elem in enumerate(elements):
        el_cent = nodes[elem].mean()
        el_vals = values[elem]

        mu = self._pdeProblem.mu(el_cent)
        beta = self._pdeProblem.beta(el_cent)
        sigma = self._pdeProblem.sigma(el_cent)
        f = self._pdeProblem.f(el_cent)

        denom[i] = mu / (3 * h) * (12 + h**2 * sigma / mu)
        err[i] += beta * 1/h * np.dot(self._beta_matrix, el_vals)
        err[i] += sigma * 1/2 * np.dot(self._sigma_matrix, el_vals)
        err[i] = (f - err[i]) * h/2

    return np.power(err, 2) / denom
```

Обчислення елементів матриці:

```
if problem.leftBC().isFirstType:
    A[0, 1] = 1
else:
    xr = nodes[0] + h/2
    A[0, 1] = mu(xr) / h - beta(xr) / 2 + sigma(xr) * h / 3 - problem.leftBC().qvalue
    A[0, 2] = -mu(xr) / h + beta(xr) / 2 + sigma(xr) * h / 6
# From 2nd to n-1
for i in tqdm(range(1, n-1), desc='Lefthand matrix creation:'):
    xl = nodes[i] - h / 2
    xr = nodes[i] + h / 2
    A[i, 0] = -mu(xl) / h - beta(xl) / 2 + sigma(xl) * h / 6
    A[i, 1] = (mu(xl) / h + beta(xl) / 2 + sigma(xl) * h / 3) + \
              (mu(xr) / h - beta(xr) / 2 + sigma(xr) * h / 3)
    A[i, 2] = -mu(xr) / h + beta(xr) / 2 + sigma(xr) * h / 6
# Last row
if problem.rightBC().isFirstType:
    A[-1, 1] = 1
else:
    xl = nodes[-1] - h / 2
    A[-1, 0] = -mu(xl) / h - beta(xl) / 2 + sigma(xl) * h / 6
    A[-1, 1] = mu(xl) / h + beta(xl) / 2 + sigma(xl) * h / 3 - problem.rightBC().qvalue
```

Обчислення елементів вектора:

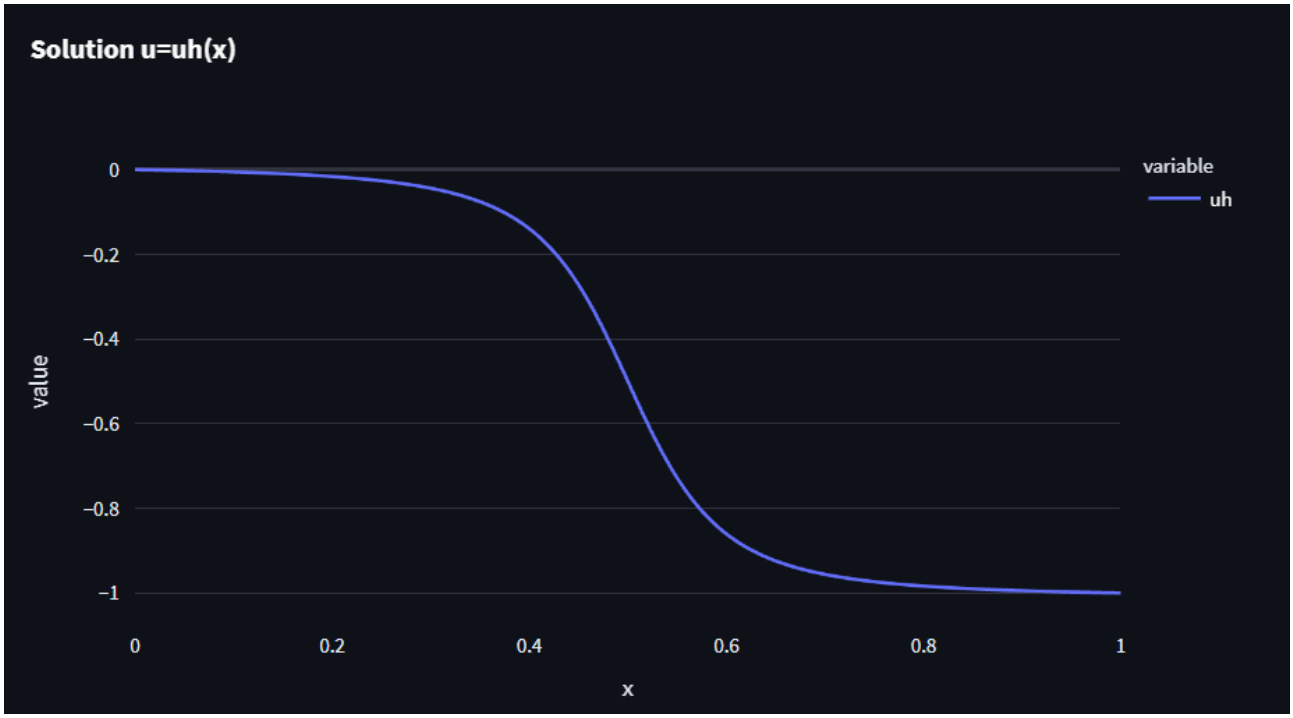
```
if problem.leftBC().isFirstType:
    l[0] = problem.leftBC().gvalue
else:
    xr = nodes[0] + h / 2
    l[0] = h * f(xr) / 2 - problem.leftBC().gvalue
# From 2nd to n-1
for i in tqdm(range(1, n - 1), desc='Righthand vector creation:'):
    xl = nodes[i] - h / 2
    xr = nodes[i] + h / 2
    l[i] = h * (f(xl) + f(xr)) / 2
# Last row
if problem.rightBC().isFirstType:
    l[-1] = problem.rightBC().gvalue
else:
    xl = nodes[-1] - h / 2
    l[-1] = h * f(xl) / 2 - problem.rightBC().gvalue
```

Також, я розв’язав інший варіант завдання (20), щоб перевірити коректність роботи програми:

$$20.\ \mu = 1,\ \ \beta = \frac{-30(10x-5)}{1+(10x-5)^2},\ \ \sigma = 0,\ \ f = 0$$

$$u(0) = 0 \qquad -\mu u'\big|_{x=1} = \frac{1}{26}$$

Графік функції u_h при n = 5 вузлах:



Ця функція відповідає одному з п’яти можливих варіантів графіків функцій апроксимації, тому, я думаю, що все правильно.

Висновок: отже, проаналізувавши результати роботи моєї програми, можна зробити висновок, що метод скінченних елементів дозволяє знайти дуже точний апроксимальний розв'язок для моєї задачі і зі збільшенням кількості вузлів точність обчислень зростає.