Київський національний університет
імені Т.Шевченка

Звіт

до лабораторної роботи №2
з дисципліни:

«Комп'ютерне Моделювання»

Студента третього курсу
Групи МІ-32
Факультету комп'ютерних наук
та кібернетики
Федорича Андрія

Київ-2023

Варіанти: В1 (Завдання 1 стор. 47 - номер 3, **Завдання 2** стор. 57+ - номер 5, **Завдання 3** стор. 72 - номер 1а **)**

Мова виконання: Python

GitHub Code: *StopFuture*

Постановка задачі 1:

3. Показати, що для $f(x) \in C^3[a,b]$ $\left| f'(x_1) - \dfrac{f_2 - f_0}{2h} \right| \le \dfrac{M_3 h^2}{6}$.

Теоретичні відомості:

5. Чисельне диференціювання

1^0. Задача чисельного диференціювання виникає тоді, коли потрібно обчислити похідну функції, значення якої задані таблицею. Нехай $f_i = f(x_i)$, $i = \overline{0,n}$, $x_i \in [a,b]$. Проінтерполюємо ці значення. Дістанемо $f(x) = L_n(x) + r_n(x)$, де $r_n(x)$ — залишковий член, який запишемо у формі Ньютона:

$$r_n(x) = f(x; x_0, \ldots, x_n)\omega_n(x), \ \ \omega_n(x) = \prod_{i=0}^{n}(x - x_i).$$

Звідси $f^{(k)}(x) = L_n^{(k)}(x) + r_n^{(k)}(x)$. За наближене значення похідної в точці x виберемо $f^{(k)}(x) \approx L_n^{(k)}(x), x \in [a,b]$. Оцінка похибки матиме такий вигляд:

$$\left| f^{(k)}(x) - L_n^{(k)}(x) \right| \le M \sum_{j=0}^{k} \frac{k!}{(k-j)!(n+j+1)!} \left| \omega_n^{(k-j)}(x) \right|,$$

де $M = \max\limits_{0 \le j \le k} \max\limits_{x \in [a,b]} \left| f^{(n+j+1)}(x) \right|$.

Зауважимо, що процес інтерполювання розбіжний. Крім того, якщо $k > n$, то $L_n^{(k)} \equiv 0$. Тому не можна брати великі значення n та k. Як правило, $k = 1,2$. Відповідно $n = k$ або $n = k+1$, або $n = k+2$. Нехай $x_i = x_0 + ih$, $h > 0$ — деякий малий крок сітки. Тоді $\omega_n(x) = (x - x_0)\ldots(x - x_n) = \left| x_n - x_0 = O(h) \right| = O(h^{n+1})$, $x \in [x_0, x_n]$. Порядок першої похідної на одиницю менший, тобто $\omega_n'(x) = O(h^n)$, а $r_n^{(k)}(x) = O(h^{n+1-k}) \Rightarrow f^{(k)}(x) - L_n^{(k)}(x) = O(h^{n+1-k})$. За умови $n \ge k$ останній вираз прямує до нуля, тобто $f^{(k)}(x) - L_n^{(k)}(x) \underset{h \to 0}{\to} 0$,

$$r_n^{(k)}(x) = \underbrace{f(x; x_0; \ldots; x_n)\omega_n^{(k)}(x)}_{O(h^{n+1-k})} + \underbrace{\sum_{j=1}^{k} C_k^j f^{(j)}(x; x_0; \ldots; x_n)\omega_n^{(k-j)}(x)}_{O(h^{n+2-k})}.$$

Якщо $\omega_n^{(k)}(\bar{x}) = 0$, то $r_n^{(k)}(\bar{x}) = O(h^{n+2-k})$. Точки $x = \bar{x}$ називаються точками підвищеної точності формул чисельного диференціювання.

Побудуємо формули чисельного диференціювання для $k = 1$, $n = 1$. Виберемо точки x_0, $x_1 = x_0 + h$. Тоді інтерполяційний многочлен має вигляд

$$L_1(x) = f_0 + (x - x_0)\frac{f_1 - f_0}{h}.$$

Для похідної дістанемо формулу

$$f'(x) \approx L_1'(x) = \frac{f_1 - f_0}{h}, \ x \in [x_0, x_1],$$

а похибка матиме вигляд

$$r_1'(x) = \frac{f^{(3)}(\xi_1)}{3!}(x - x_0)(x - x_1) + \frac{f^{(2)}(\xi_0)}{2!}(2x - x_1 - x_0) = O(h).$$

Якщо $2\bar{x} - x_1 - x_0 = 0$, то $r_1'(\bar{x}) = \dfrac{f^{(3)}(\xi_1)}{3!}(\bar{x} - x_0)(\bar{x} - x_1) = O(h^2)$, тобто $\bar{x} = \dfrac{x_1 + x_0}{2}$ — точка підвищеної точності. Точніше $r_1'(\bar{x}) \le \dfrac{h^2}{24}M_3$, де $M_3 = \max f^{(3)}(x)$.

Для $x \in [x_i, x_{i+1}]$ позначимо $f'(x) \approx \dfrac{f_{i+1} - f_i}{h} = f_{x,i}$; для $x \in [x_{i-1}, x_i]$ $f'(x) \approx \dfrac{f_i - f_{i-1}}{h} = f_{\bar{x},i}$ і для $x \in [x_{i-1}, x_{i+1}]$ $f'(x) = \dfrac{f_{i+1} - f_{i-1}}{2h} = f_{\overset{0}{x},i}$. Замість $f'(x)$ можна взяти будь-яке зі значень $f_{x,i}$, $f_{\bar{x},i}$ або $f_{\overset{0}{x},i}$. Для похибки $f_{\overset{0}{x},i}$ маємо оцінку $r_2'(x) = O(h^2)$.

2^0. На практиці для побудови формул чисельного диференціювання часто використовують метод невизначених коефіцієнтів. Він полягає в такому: шукану функцію записуємо у вигляді $f^k(x_0) = \sum\limits_{i=0}^{n} c_i f(x_i) + R(f)$. Коефіцієнти c_i визначаємо із системи лінійних рівнянь $R(f) = 0$, де функцію $f(x)$ послідовно беремо такою, що дорівнює $1, x, x^2, \ldots, x^{n-1}$.

3^0. Як правило, значення функції задані з похибкою, що зумовлена вимірюванням чи обчисленням цих значень. Як така похибка вплине на результат чисельного диференціювання?

Нехай є деякі збурення функції $f(x) \in C^1[a,b]$, тобто $\tilde{f}(x) = f(x) + \dfrac{1}{n}\sin(\omega x)$. При $n \to \infty$ дістаємо $\left\| f(x) - \tilde{f}(x) \right\| = \dfrac{1}{n} \to 0$.

Звідси отримуємо рівномірну збіжність $\tilde{f}(x) \underset{n\to\infty}{\Rightarrow} f(x)$. Таким чином, ці обурення малі. Обчислимо похідну: $\tilde{f}'(x) = f'(x) + \frac{\omega}{n}\cos(\omega x)$. Покладемо $\omega = n^2$. Тоді $\left\|\tilde{f}' - f'\right\| = \frac{|\omega|}{n} = n \xrightarrow[n\to\infty]{} \infty$, $\tilde{f}' \nrightarrow f'$. Похибка диференціювання збуреної функції може бути як завгодно великою. Це є наслідком некоректності операції диференціювання.

Нехай функція задана збуреними значеннями $\tilde{f}_i = f_i + \delta_i$, $f_i = f(x_i)$, $i = \overline{0,n}$ $|\delta_i| \le \delta$. Дослідимо, як впливають похибки на значення похідних при $n = 1$, $k = 1$:

$$f_i' - \frac{\tilde{f}_i - \tilde{f}_{i-1}}{h} = f_i' - \frac{f_i - f_{i-1}}{h} - \frac{\delta_i - \delta_{i-1}}{h},$$

$$\left|f_i' - \frac{\tilde{f}_i - \tilde{f}_{i-1}}{h}\right| \le \left|f_i' - \frac{f_i - f_{i-1}}{h}\right| + \left|\frac{\delta_i - \delta_{i-1}}{h}\right| \le \frac{M_2 h}{2} + \frac{2\delta}{h} \xrightarrow[h\to\infty]{} \infty.$$

Розглянемо оцінку похибки $\varphi(h) = \frac{M_2 h}{2} + \frac{2\delta}{h}$. Постає питання: як обрати h, щоб $\varphi(h) \to \min$? Маємо $\varphi'(h) = \frac{M_2}{2} - \frac{2\delta}{h^2} = 0$. Отже,

$$h_0 = 2\sqrt{\frac{\delta}{M_2}}. \tag{5.1}$$

Попередні розрахунки/ Перевірка:

Щоб довести дану нерівність

$$\left|f'(x_1) - \frac{f_2 - f_0}{2h}\right| \le \frac{M_3 h^2}{6}$$

для функції $f(x)$, яка є тричі безперервно диференційованою на інтервалі $[a, b]$, ми можемо скористатися Теоремою про середнє значення або Формулою Тейлора.

Вираз $\frac{f_2 - f_0}{2h}$ нагадує центральну різницю для наближення першої похідної f в деякій точці x_1 на інтервалі (a, b), де $f_2 = f(x_1 + h)$ і $f_0 = f(x_1 - h)$.

Член праворуч, $\frac{M_3 h^2}{6}$, виглядає як оцінка залишкового члену в розкладі Тейлора функції f навколо x_1, де M_3 є оцінкою третьої похідної f по інтервалу $[a, b]$.

Використовуючи Теорему Тейлора, ми можемо розкласти f навколо x_1, а потім виразити $f(x_1 + h)$ і $f(x_1 - h)$ через ряд Тейлора з залишковим членом. Дана нерівність насправді є виразом оцінки помилки центрального методу чисельного диференціювання.

Щоб продемонструвати цю нерівність для конкретної функції $f(x)$ треба в python коді:

1. Визначити таку функцію f, яка є тричі диференційованою на $[a, b]$.
2. Обчислити f_2, f_0, і фактичну $f'(x_1)$.
3. Вибрати значення для h і обчислити центральну різницю.
4. Знайти відповідну оцінку M_3 для третьої похідної на $[a, b]$.
5. Перевірити нерівність.

Візьмемо:

```
a, b = 0, 1
x1 = 0.5
f(x) = x ** 4
```

Отримаємо такі результати:

```
35  results
36

{'f_prime(x1)': 0.5,
 'Central Difference Approximation': 0.5199999999999999,
 'Actual Error': 0.0199999999999999,
 'Error Bound': 0.04000000000000001}
```

Нерівність працює.

Код програми:

```
from sympy import symbols, diff, lambdify, Abs

x = symbols('x')
f = x**4

f_prime = diff(f, x)
f_third = diff(f_prime, x, x)

f_lambdified = lambdify(x, f)
f_prime_lambdified = lambdify(x, f_prime)
f_third_lambdified = lambdify(x, f_third)

a, b = 0, 1
x1 = 0.5
h = 0.1

f_2 = f_lambdified(x1 + h)
f_0 = f_lambdified(x1 - h)
actual_f_prime = f_prime_lambdified(x1)

central_diff = (f_2 - f_0) / (2 * h)

M_3 = max(abs(f_third_lambdified(a)), abs(f_third_lambdified(b)))

error_bound = (M_3 * h**2) / 6

actual_error = Abs(actual_f_prime - central_diff)

results = {
    'f_prime(x1)': actual_f_prime,
    'Central Difference Approximation': central_diff,
    'Actual Error': actual_error,
    'Error Bound': error_bound
}
results
```

Постановка задачі 2:

5. Побудувати многочлен n-го степеня найкращого рівномірного наближення для функції $f(x) = a_0 + \ldots + a_{n+1}x^{n+1}$, $x \in [a, b]$.

Теоретичні відомості:

Найзагальніший підхід полягає в наближенні $f(x)$ функцією $\Phi(x)$ так, щоб досягалася деяка задана точність ε: $\|f(x) - \Phi(x)\| < \varepsilon$. Але розв'язок у такій постановці може не існувати або бути не єдиним.

Загальна постановка задачі наближення має такий вигляд. Нехай R — лінійний нормований простір та елемент $f \in R$; M_n — підпростір усіх можливих лінійних комбінацій

$$\Phi = \sum_{i=0}^{n} c_i \varphi_i \in M_n \subset R \qquad (6.1)$$

за елементами лінійно-незалежної системи $\{\varphi_i\}_{i=0}^{\infty}$, $\varphi_i \in R$.

Відхилення $\Phi \in M_n$ від $f \in R$ є числовою множиною $\Delta(f, \Phi) = \|f - \Phi\|$.

Позначимо $\inf\limits_{\Phi \in M_n} \|f - \Phi\| = \Delta(f)$.

Елемент $\Phi_0 \in M_n$, для якого виконується умова

$$\Delta(f, \Phi_0) = \Delta(f), \qquad (6.2)$$

називається елементом найкращого наближення (ЕНН).

Теорема 1. Для будь-якого f з лінійного нормованого простору R існує елемент найкращого наближення, причому множина елементів найкращого наближення опукла.

Теорема 2. Для будь-якого f з гільбертового простору H існує єдиний елемент найкращого наближення.

Найкраще рівномірне наближення — це наближення у просторі $R = C[a,b]$, де $\|f\|_{C[a,b]} = \max\limits_{x \in [a,b]} |f(x)|$.

Теорема 3 (Хаара). Для того щоб на $\forall f \in C[a,b]$ існував єдиний елемент найкращого рівномірного наближення, необхідно і достатньо, щоб система $\{\varphi_i\}_{i=0}^{\infty}$ була системою функцій Чебишова.

Означення системи функцій Чебишова наведено в розділі про інтерполювання.

Позначимо $Q_n^0(x)$ — многочлен найкращого рівномірного наближення (МНРН). Тоді

$$\Delta(f) = \|Q_n^0(x) - f(x)\|_C = \inf_{Q_n(x)} \|Q_n(x) - f(x)\|.$$

Теорема 4 МНРН для неперервної функції єдиний.

Теорема 5 (Чебишова). $Q_n^0(x)$ буде МНРН для неперервної функції $f(x)$ тоді і тільки тоді, коли на відрізку $[a,b]$ існує хоча б $n+2$ точки $a \le x_0 \ldots \le x_m \le b$, $m \ge n+1$ такі, що

$$f(x_i) - Q_n^0(x_i) = \alpha(-1)^i \Delta(f), \qquad (6.3)$$

де $i = \overline{0, m}$, $\alpha = \pm 1$ для всіх i.

Точки $\{x_i\}_{i=0}^{m}$, які задовольняють умови теореми Чебишова, називаються точками чебишовського альтернансу.

Розглянемо алгоритм, який називають ще телескопічним методом побудови многочлена, близького до МНРН. Якщо точний МНРН знайти не вдається, то в таких випадках будується многочлен, близький до нього. Бажано, щоб цей многочлен був невисокого степеня (менше арифметичних операцій на його обчислення). Спочатку будують такий многочлен $P_n(x) = \sum_{j=0}^{n} a_j x^j$, щоб похибка була доволі малою (наприклад, $< \frac{\varepsilon}{2}$ за формулою Тейлора). Потім наближають многочлен $P_n(x)$ многочленом найкращого рівномірного наближеня $P_{n-1}(x)$ (для простоти $x \in [-1, 1]$): $P_{n-1}(x) = P_n(x) - a_n T_n(x) 2^{1-n}$.

Оскільки $|T_n(x)| \le 1$ на відрізку $[-1,1]$, то $|P_{n-1}(x) - P_n(x)| \le |a_n| 2^{1-n}$.

Далі наближають многочлен $P_{n-1}(x)$ многочленом найкращого рівномірного наближення $P_{n-2}(x)$ і т. д. Зниження степеня триває доти, поки похибка від таких послідовних апроксимацій залишається меншою від заданого ε.

Приклади розв'язання задач

1. Наблизити $f(x) \in C[a,b]$ многочленом НРН нульового степеня.

Розв'язання. Нехай $M = \max\limits_{[a,b]} f(x) = f(x_0)$, $m = \min\limits_{[a,b]} f(x) = f(x_1)$. Тоді $Q_0(x)$ – МНРН — матиме вигляд $Q_0(x) = \frac{M+m}{2}$, а в точках x_0, x_1 отримаємо

$$Q_0(x_0) - f(x_0) = \frac{M+m}{2} - M = -\frac{M-m}{2},$$

$Q_0(x_0) - f(x_1) = m - \frac{M+m}{2} = \frac{M-m}{2}$, $\Delta(f) = \frac{M-m}{2}$, де x_0, x_1 — точки чебишовського альтернансу.

Щоб побудувати поліном степеня n, який забезпечує найкраще рівномірне наближення для функції

$$f(x) = a_0 + a_1x + \ldots + a_{n+1}x^{n+1}$$

на інтервалі $[a, b]$, де $a_0, a_1, \ldots, a_{n+1}$ є заданими коефіцієнтами, ми зазвичай використовуємо поліноми Чебишева першого роду, які відомі як найкращі рівномірні наближення для функцій на певному інтервалі, коли функція, яку потрібно наблизити, є неперервною.
Поліноми Чебишева першого роду визначаються за допомогою рекурентного співвідношення:

$$T_0(x) = 1$$
$$T_1(x) = x$$
$$T_{k+1}(x) = 2xT_k(x) - T_{k-1}(x)$$

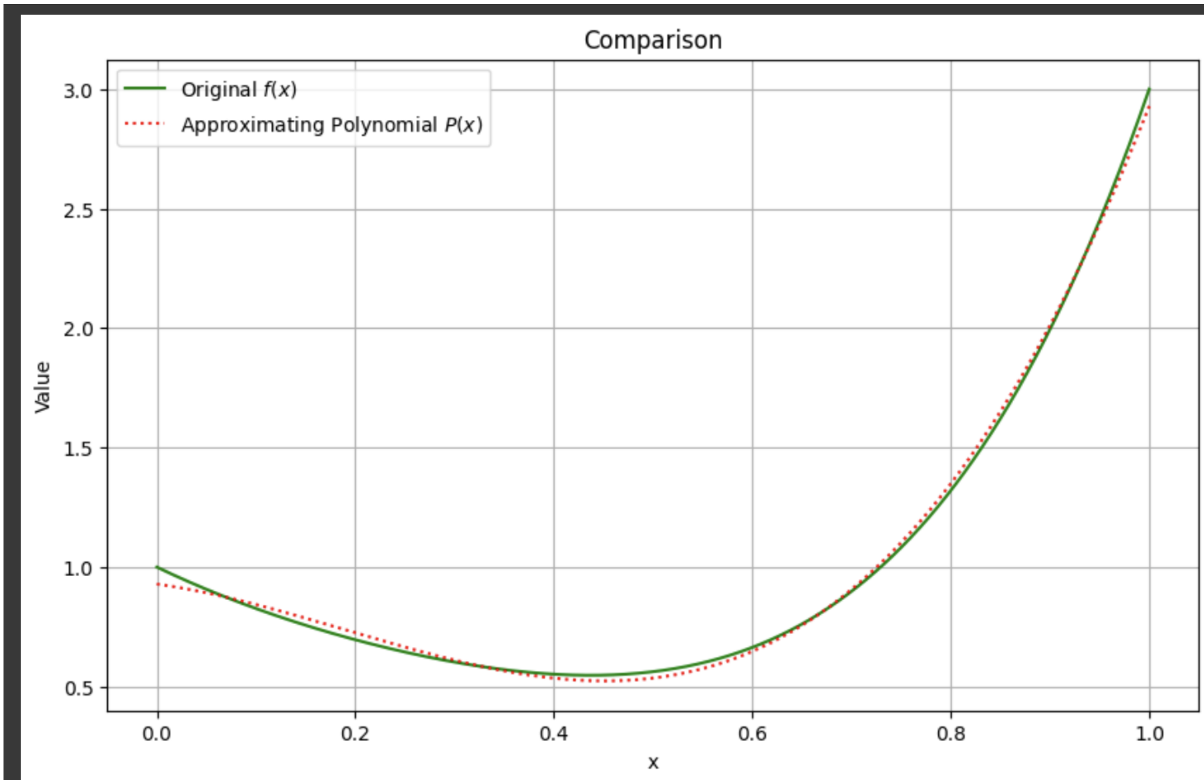
для $k \geq 1$.

$$
\begin{aligned}
T_0(x) &= 1 \\
T_1(x) &= x \\
T_2(x) &= 2x^2 - 1 \\
T_3(x) &= 4x^3 - 3x \\
T_4(x) &= 8x^4 - 8x^2 + 1 \\
T_5(x) &= 16x^5 - 20x^3 + 5x \\
T_6(x) &= 32x^6 - 48x^4 + 18x^2 - 1 \\
T_7(x) &= 64x^7 - 112x^5 + 56x^3 - 7x \\
T_8(x) &= 128x^8 - 256x^6 + 160x^4 - 32x^2 + 1 \\
T_9(x) &= 256x^9 - 576x^7 + 432x^5 - 120x^3 + 9x
\end{aligned}
$$

Найкращий рівномірний наближений поліном $P_n(x)$ степеня n можна знайти, мінімізуючи максимальну помилку між $P_n(x)$ та $f(x)$ на інтервалі $[a, b]$.

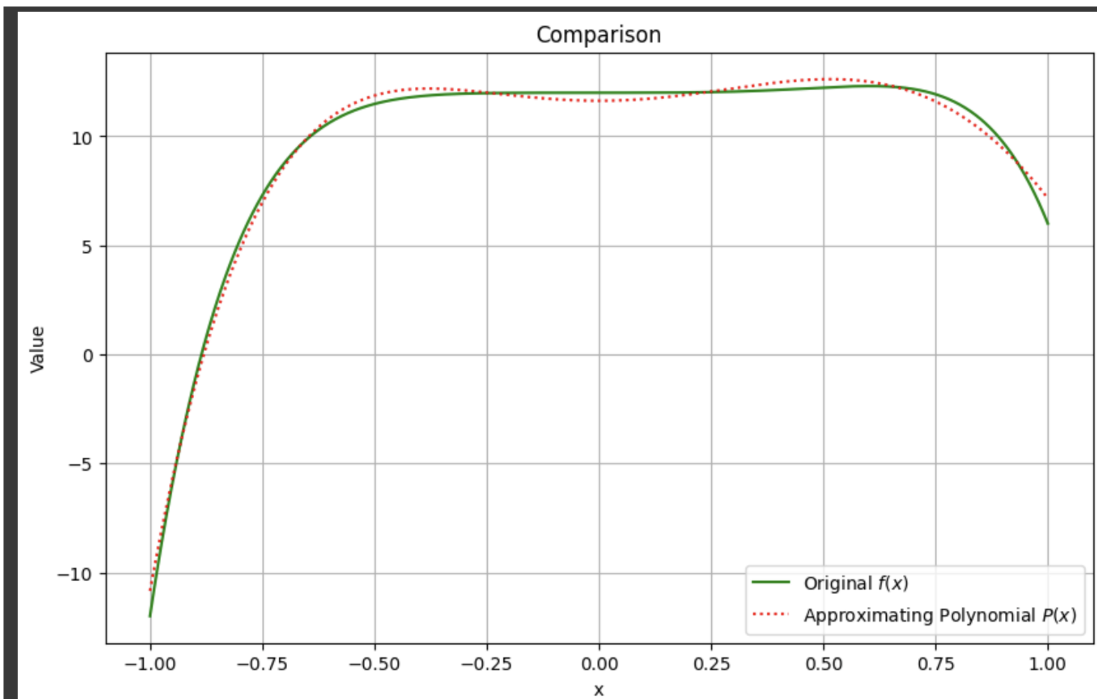
Обчислення будемо робити кодом, оскільки записувати розрахунки коефіцієнтами не дуже зручно, адже ми можемо мати величезну кількість різних поліноміальних функцій та відрізків $[a, b]$. Візьмемо декілька різних інпутів, щоб побачити наскільки сильним буде відхилення в різних випадках.

Приклад 1:



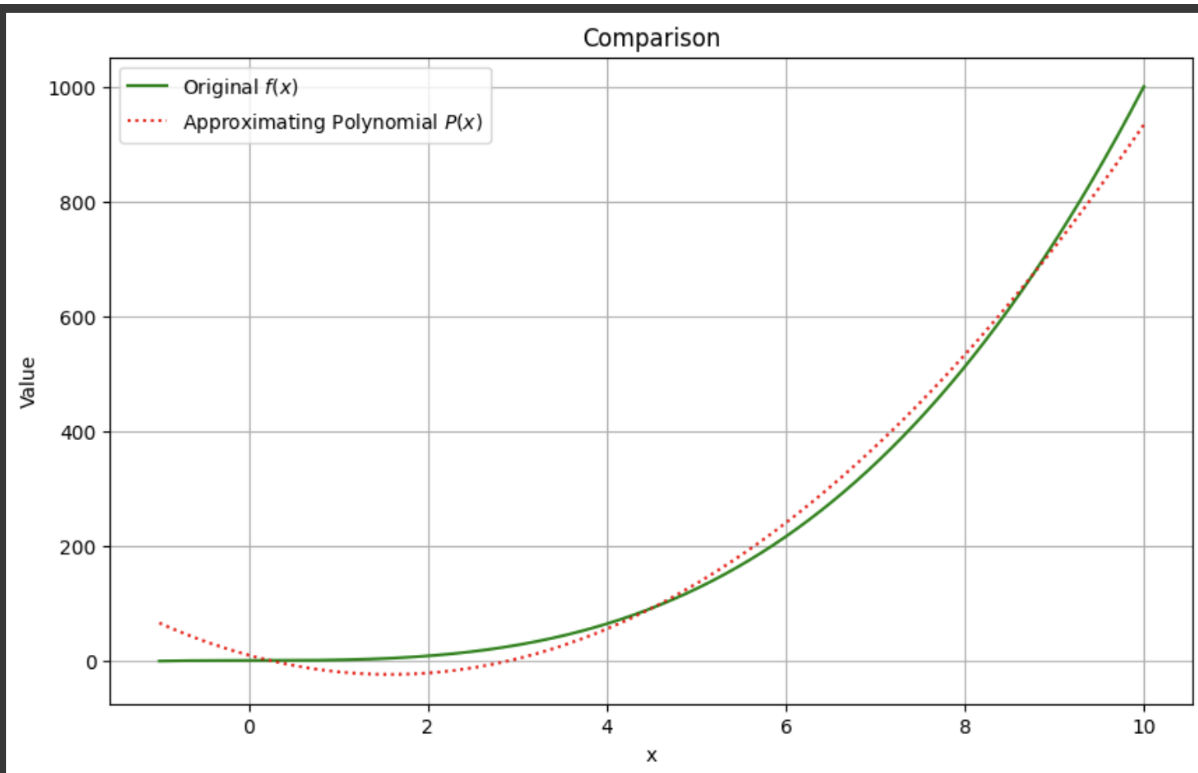

```
Original Coefficients: [1, -2, 3, -4, 5]
Max Residual:  0.07100035837740215
Approximating Polynomial Coefficients: [-0.78421535  3.92642999 -1.71321499  1.5        ]
```

Приклад 2:



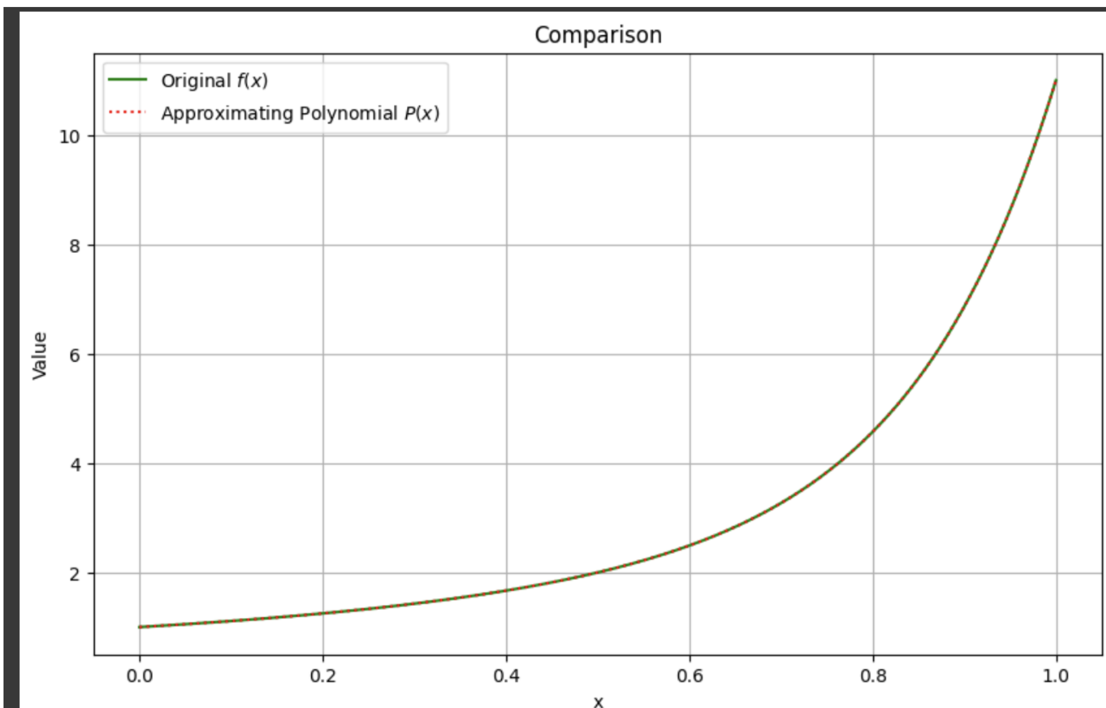

```
Original Coefficients: [12, 0, 0, 1, 2, 8, -17]
Max Residual:  1.1599091821390832
Approximating Polynomial Coefficients: [ 7.54837204  5.75       -6.73496143  2.75       -2.65350143  0.5        ]
```

Приклад 3:




```
Original Coefficients: [1, 0, 0, 1]
Max Residual:  66.35028483826316
Approximating Polynomial Coefficients: [  17.03659656 -425.63688265 1343.25000284]
```

Приклад 4:




```
Original Coefficients: [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
Max Residual:  5.174861158252497e-06
Approximating Polynomial Coefficients: [-3.72798621 15.47452022 -7.59364137  8.32623316 -3.64666878  2.65214618
 -0.85644853  0.42384306 -0.07544008  0.02343716]
```

Код програми:

```
def normalized_chebyshev_polys(x):
    T = chebyshev_polys(x)
    normalized_T = []
    for poly in T:
        max_coeff = np.max(np.abs(poly))
        normalized_poly = poly / max_coeff if max_coeff != 0 else poly
        normalized_T.append(normalized_poly)
    return normalized_T

def chebyshev_fit(coeffs, x):
    T = normalized_chebyshev_polys(x)
    return sum(c * Tn for c, Tn in zip(coefficients, T))

def residual(coeffs, x, y):
    T = normalized_chebyshev_polys(x)
    approx = sum(c * Tn for c, Tn in zip(coeffs, T))
    return approx - y


coeffs = [1, -2, 3, -4, 5]
#coeffs = [12, 0, 0, 1, 2, 8, -17]
#coeffs = [1, 0, 0, 1]
coeffs = [1,1,1,1,1,1,1,1,1,1,1]
n = len(coeffs) - 1
interval = [0, 1]

target_function = Polynomial(coeffs)
x_vals = np.linspace(interval[0], interval[1], 1000)
y_vals = target_function(x_vals)
initial_coeffs = [0]*n
coefficients = least_squares(residual, initial_coeffs, args=(x_vals, y_vals)).x



fit_vals = chebyshev_fit(coefficients, x_vals)
max_residual = np.max(np.abs(y_vals - fit_vals))


plt.figure(figsize=(10, 6))
plt.plot(x_vals, y_vals, label='Original $f(x)$', color='green')
plt.plot(x_vals, fit_vals, label='Approximating Polynomial $P(x)$', linestyle='dotted', color='red')
plt.title("Comparison")
plt.xlabel("x")
plt.ylabel("Value")
plt.legend()
plt.grid(True)
plt.show()

print("Original Coefficients:",coeffs , "\nMax Residual: ", max_residual, "\nApproximating Polynomial Coefficients:", coefficients)
```

Постановка задачі 3:

Задачі для самостійного розв'язання

1. Побудувати лінійний многочлен найкращого середньоквадратичного наближення для функції:

а) $f(x)=\sqrt[3]{x}$, $x\in[0,1]$;

Теоретичні відомості:

Найкраще середньоквадратичне наближення — це наближення у просторі $R=H$, де H — гільбертів простір зі скалярним добутком (u,v), норма і відстань для якого визначаються формулами $\|u\|=\sqrt{(u,u)}$, $\Delta(u,v)=\|u-v\|$.

Побудуємо елемент найкращого середньоквадратичного наближення (ЕНСКН).

Теорема 1. Нехай $f\in H$, $\Phi_0\in M_n$ — ЕНСКН. Тоді

$$(f-\Phi_0,\Phi)=0,\ \forall\Phi\in M_n. \tag{7.1}$$

Наслідок. Функцію $f\in H$ можна подати у вигляді $f=\Phi_0+v$, де $\Phi_0\in M_n$; $v\perp M_n$.

Знайти ЕНСКН

$$\Phi_0=\sum_{i=0}^{n}c_i\varphi_i \tag{7.2}$$

означає знайти коефіцієнти c_i.

Для виконання (7.1) достатньо, щоб $(f-\Phi_0,\varphi_k)=0$, $k=\overline{0,n}$. Разом з формулою (7.2) це приводить до СЛАР для знаходження c_i:

$$\sum_{i=0}^{n}c_i(\varphi_i,\varphi_k)=(f,\varphi_k),\ k=\overline{0,n}. \tag{7.3}$$

Матриця СЛАР (7.3) $G=\{(\varphi_i,\varphi_k)\}_{i,k=0}^{n}$ є матрицею Грамма лінійно незалежної системи функцій, тобто $\det G\neq 0$, що доводить існування та єдиність ЕНСКН. Оскільки $G^T=G$, то для розв'язку цієї системи використовують метод квадратних коренів.

У багатьох випадках матриця G погано обумовлена. У таких випадках доцільно вибирати систему $\{\varphi_i\}$ ортонормованою, тобто

1. На проміжку [0, 1] побудувати алгебраїчний многочлен першого степеня НСКН для функції $f(x)=\sqrt{x}$ та знайти відхилення.

Розв'язання. Розв'язок шукаємо у вигляді

$\Phi_1(x)= c_1\varphi_0(x)+ c_2\varphi_1(x)$,

$\varphi_0(x)= 1,\ \varphi_1(x) = x.$

Тоді $(\varphi_0, \varphi_0) = \int_0^1 1\,dx = 1;\ (\varphi_0, \varphi_1) = (\varphi_1, \varphi_0) = \int_0^1 x\,dx = \frac{1}{2}$,

$(\varphi_1, \varphi_1) = \int_0^1 x^2\,dx = \frac{1}{3},\ (f, \varphi_0) = \int_0^1 \sqrt{x}\,dx = \frac{2}{3}$,

$(f, \varphi_1) = \int_0^1 x\sqrt{x}\,dx = \frac{2}{5}$.

Розв'язуючи систему рівнянь

$$\begin{cases} c_0 + \frac{1}{2}c_1 = \frac{2}{3}; \\ \frac{1}{2}c_0 + \frac{1}{3}c_1 = \frac{2}{5}, \end{cases}$$

дістаємо $c_0 = \frac{4}{15},\quad c_1 = \frac{4}{5},\ \Phi_1(x) = \frac{4}{15} + \frac{4}{5}x = 0.2667 + 0.8x.$

Середньоквадратичне відхилення $\Delta(f) = \frac{\sqrt{2}}{30} = 0.0471.$

Попередні розрахунки:
Абсолютно аналогічні до даного прикладу.

Розглянемо весь процес покроково для функції $f(x) = x^{1/3}$ на інтервалі [0, 1], використовуючи ортогональний базис $\Phi_0(x) = 1$ та $\Phi_1(x) = x$.

Розглянемо весь процес покроково для функції $f(x) = x^{1/3}$ на інтервалі [0, 1], використовуючи ортогональний базис $\Phi_0(x) = 1$ та $\Phi_1(x) = x$.

1. Визначаємо внутрішні добутки між функцією $f(x)$ і базисними функціями:

$$(f, \Phi_0) = \int_0^1 x^{1/3} \cdot 1 \, dx$$

$$(f, \Phi_1) = \int_0^1 x^{1/3} \cdot x \, dx$$

2. Визначаємо внутрішні добутки між базисними функціями:

$$(\Phi_0, \Phi_0) = \int_0^1 1 \cdot 1 \, dx$$

$$(\Phi_1, \Phi_1) = \int_0^1 x \cdot x \, dx$$

3. Знаходимо коефіцієнти c_0 та c_1, використовуючи формули з методу найменших квадратів:

$$c_0 = \frac{2(f, \Phi_0)}{(\Phi_0, \Phi_0)}$$

$$c_1 = \frac{2(f, \Phi_1)}{(\Phi_1, \Phi_1)}$$

4. Складаємо многочлен найкращого середньоквадратичного наближення:

$$P(x) = c_0\Phi_0(x) + c_1\Phi_1(x)$$

5. Розраховуємо середньоквадратичне відхилення $\Delta(f)$, використовуючи норми базисних функцій і внутрішні добутки з $f(x)$:

$$\Delta(f) = \sqrt{\int_0^1 [f(x) - P(x)]^2 \, dx}$$

$$\Delta(f) = \sqrt{\int_0^1 f(x)^2 \, dx - 2c_0(f, \Phi_0) - 2c_1(f, \Phi_1) + c_0^2(\Phi_0, \Phi_0) + c_1^2(\Phi_1, \Phi_1)}$$

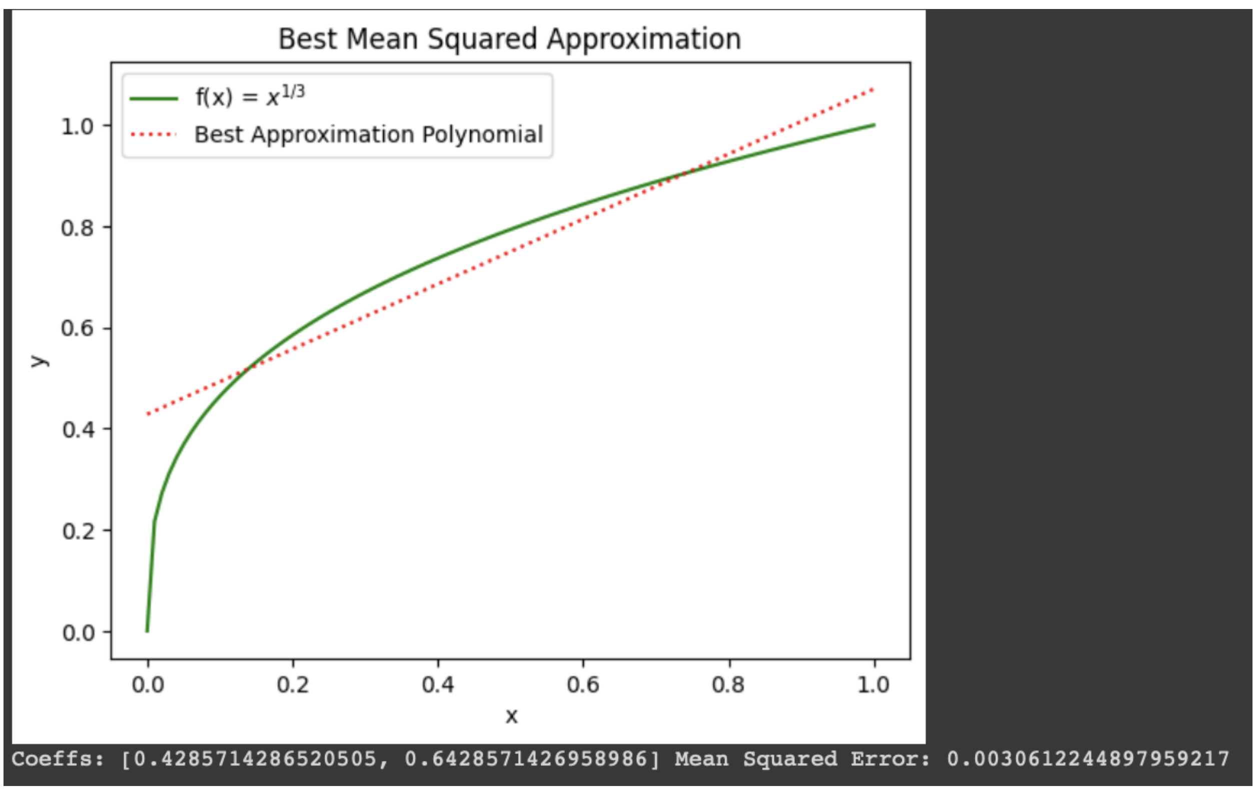
Так як базис ортогональний, члени, що містять перемноження різних базисних функцій, рівні нулю.

6. Враховуючи, що $(\Phi_0, \Phi_0) = 1$ та $(\Phi_1, \Phi_1) = \dfrac{1}{3}$, вираз для $\Delta(f)$ спрощується:

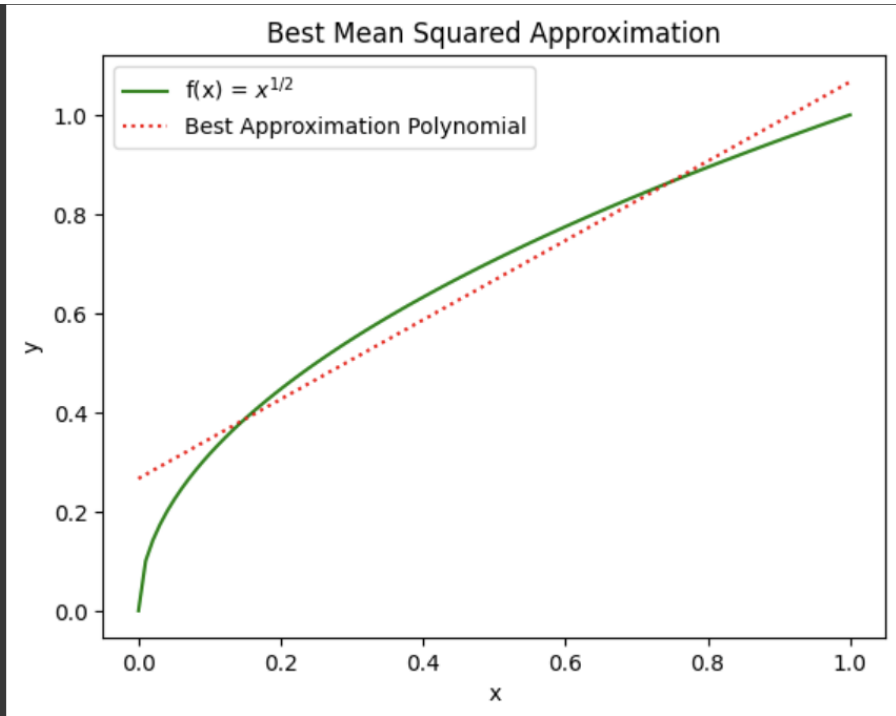
$$\Delta(f) = \sqrt{\int_0^1 f(x)^2\, dx - c_0^2 - \frac{c_1^2}{3}}$$

Сюди потрібно підставити числові значення і

Отримуємо:



Щоб перевірити коректність коду, можна використати наведений приклад із x^(½), результат такий же, як в методичці:




```
Coeffs: [0.26666666668329014, 0.7999999999667534] Mean Squared Error: 0.0022222222222222227
```

Код програми:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import quad

def f(x):
    return x**(1/3)

phi_0 = lambda x: 1
phi_1 = lambda x: x

phi_0_phi_0 = quad(lambda x: phi_0(x)**2, 0, 1)[0]
phi_0_phi_1 = quad(lambda x: phi_0(x)*phi_1(x), 0, 1)[0]
phi_1_phi_1 = quad(lambda x: phi_1(x)**2, 0, 1)[0]

f_phi_0 = quad(lambda x: f(x)*phi_0(x), 0, 1)[0]
f_phi_1 = quad(lambda x: f(x)*phi_1(x), 0, 1)[0]

A = np.array([[phi_0_phi_0, phi_0_phi_1], [phi_0_phi_1, phi_1_phi_1]])
b = np.array([f_phi_0, f_phi_1])
c = np.linalg.solve(A, b)

P = lambda x: c[0]*phi_0(x) + c[1]*phi_1(x)


mse = quad(lambda x: (f(x) - P(x))**2, 0, 1)[0]

x_values = np.linspace(0, 1, 10000)
plt.plot(x_values, f(x_values), label='f(x) = $x^{1/3}$', color = "green")
plt.plot(x_values, P(x_values), label='Best Approximation Polynomial', linestyle='dotted', color = "red")
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.title('Best Mean Squared Approximation')
plt.show()

print("Coeffs:", list(c), "Mean Squared Error:", mse)
```