РГР №2

Былькова Кристина Алексеевна

Группа М8О-308Б-22

Задание:

Задание II

Проведите ортогонализацию системы функций $x_n(t) = t^{n-1}$ в пространстве квадратично суммируемых функций относительно склярного произведения $\langle x, y\rangle = \int\limits_a^b x(t)y(t)f(t)\,dt$. Найдите приближение функции y частичной суммой ряда Фурье, обеспечивающее среднеквадратичную точность разложения $\varepsilon \in \{10^{-1}, 10^{-2}, 10^{-3}\}$ (при достаточных вычислительных ресурсах). Постройте график функции $y(t)$ и его приближения частичными суммами ряда Фурье. Продемонстрируйте несколько графиков, получающихся при промежуточных вычислениях.

Вариант 2:

2) $[a,b] = \left[0; 0{,}8 + \frac{k}{10}\right]$, $f(t) = \left(\frac{2l}{5} - t\right)t$, $y(t) = \cos(2t)$;

Решение:

Для решения задачи для начала необходимо провести ортогонализацию системы функций $x_n(t) = t^{n-1}$ в пространстве квадратично суммируемых функций относительно скалярного произведения: $< x, y >= \int_a^b x(t)y(t)f(t)dt$.

Для первой функции: $e_0 = x_0(t)$

Для последующих: $e_n = x_n(t) - \sum_{k=0}^{n-1} \frac{(x_k, e_k)}{(e_k, e_k)} \times e_k$

Далее нормируем: $z_n = \frac{e_n}{||e_n||}$, где $\left||e_n|\right| = \sqrt{(e_n, e_n)}$

Следующий шаг – разложение функции $y(t) = \cos(2t)$ в ряд Фурье:

$\hat{y}(t) = \sum_{k=0}^{N} \propto_k z_k$, где $\propto_k = \frac{(y, z_k)}{(z_k, z_k)}$

Необходимо найти такое N, чтобы $\left||y - \hat{y}(t)|\right| < \varepsilon$

Код:

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

a = 0
b = 1.6
f = lambda t: (6 - t)
y = lambda t: np.cos(2*t)

# Скалярное произведение
def scalar_product(x, y):
    result, _ = quad(lambda t: x(t) * y(t) * f(t), a, b)
    return result

# Ортогонализация Грама-Шмидта
def gram_schmidt(max_n):
    basis = [lambda t, n=n: t**n for n in range(max_n)]
    ortho_basis = []

    for n in range(max_n):
        x_n = basis[n]
        for k in range(n):
            coeff = scalar_product(basis[n], ortho_basis[k]) /
scalar_product(ortho_basis[k], ortho_basis[k])
            x_n = lambda t, n=n, k=k, x_n=x_n, coeff=coeff: x_n(t) - coeff *
ortho_basis[k](t)

        norm = np.sqrt(scalar_product(x_n, x_n))
        x_n_norm = lambda t, x_n=x_n, norm=norm: x_n(t) / norm if norm != 0 else
x_n(t)
        ortho_basis.append(x_n_norm)

    return ortho_basis

# Ошибка проектирования
def calculate_error(original_func, approx_func):
    error_func = lambda t: (original_func(t) - approx_func(t))**2
    return np.sqrt(quad(error_func, a, b)[0])

# Нахождение приближения
def find_approximation(y, epsilons, max_n):
    ortho_basis = gram_schmidt(max_n)
    t_vals = np.linspace(a, b, 1000)

    # Коэффициенты Фурье
```

```python
    coeffs = [scalar_product(y, ortho_basis[n]) / scalar_product(ortho_basis[n],
ortho_basis[n]) for n in range(max_n)]

    # Определяем число членов ряда Фурье
    N_for_eps = {}

    for epsilon in epsilons:
        print(f"\nepsilon = {epsilon}:")
        for N in range(1, max_n + 1):
            approx = lambda t, N=N: sum(coeffs[n] * ortho_basis[n](t) for n in
range(N))
            error = calculate_error(y, approx)
            print(f"Итерация {N}: error = {error:.6f}")
            if error < epsilon:
                N_for_eps[epsilon] = N
                print(f"Достигнута требуемая точность на итерации {N}")
                break
        else:
            print(f"Требуемая точность не достигнута за {max_n} итераций")
            N_for_eps[epsilon] = max_n

    return ortho_basis, coeffs, N_for_eps

def main():
    max_n = 10
    epsilons = [1e-1, 1e-2, 1e-3]
    ortho_basis, coeffs, N_for_eps = find_approximation(y, epsilons, max_n)

    t_vals = np.linspace(a, b, 1000)
    y_vals = y(t_vals)

    for epsilon in epsilons:
        N = N_for_eps[epsilon]
        plt.figure(figsize=(12, 6))

        # Левый подграфик: исходная функция и приближение
        plt.subplot(1, 2, 1)
        plt.plot(t_vals, y_vals, label='y(t) = cos(2t)', linewidth=2)

        approx = lambda t: sum(coeffs[n] * ortho_basis[n](t) for n in range(N))
        approx_vals = np.array([approx(t) for t in t_vals])
        plt.plot(t_vals, approx_vals, '--',
                 label=f'Приближение (epsilon={epsilon}, N={N})',
                 linewidth=1.5, color='red')

        plt.title(f'Аппроксимация для epsilon = {epsilon}')
        plt.xlabel('t')
```

```python
        plt.ylabel('y(t)')
        plt.legend()
        plt.grid(True)

        # Правый подграфик: промежуточные приближения
        plt.subplot(1, 2, 2)
        plt.plot(t_vals, y_vals, label='y(t) = cos(2t)', linewidth=2)

        for n in range(1, N + 1):
            partial_approx = lambda t: sum(coeffs[k] * ortho_basis[k](t) for k in
range(n))
            partial_vals = np.array([partial_approx(t) for t in t_vals])
            plt.plot(t_vals, partial_vals, '--',
                     label=f'N={n}',
                     linewidth=1, alpha=0.7)

        plt.title(f'Промежуточные приближения')
        plt.xlabel('t')
        plt.ylabel('y(t)')
        plt.legend()
        plt.grid(True)

        plt.tight_layout()
        plt.show()

if __name__ == "__main__":
    main()
```

Вывод:

$ python main.py

epsilon = 0.1:

Итерация 1: error = 0.905735

Итерация 2: error = 0.113655

Итерация 3: error = 0.113398

Итерация 4: error = 0.003892

Достигнута требуемая точность на итерации 4

epsilon = 0.01:

Итерация 1: error = 0.905735

Итерация 2: error = 0.113655

Итерация 3: error = 0.113398

Итерация 4: error = 0.003892

Достигнута требуемая точность на итерации 4

epsilon = 0.001:

Итерация 1: error = 0.905735

Итерация 2: error = 0.113655

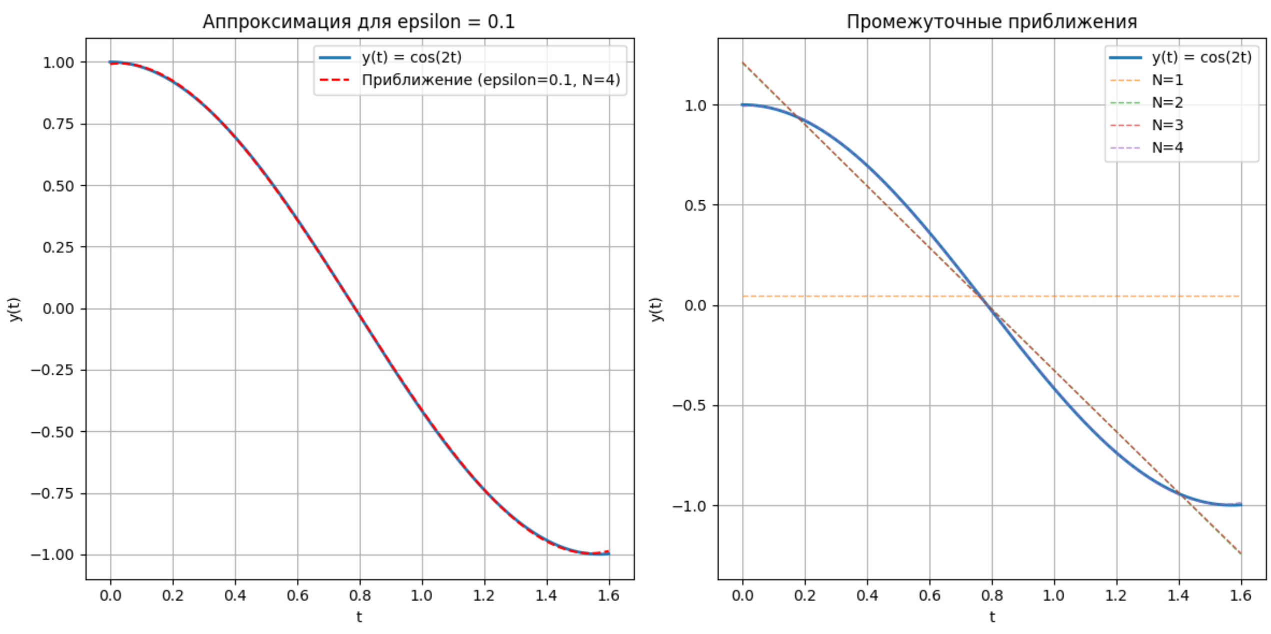
Итерация 3: error = 0.113398

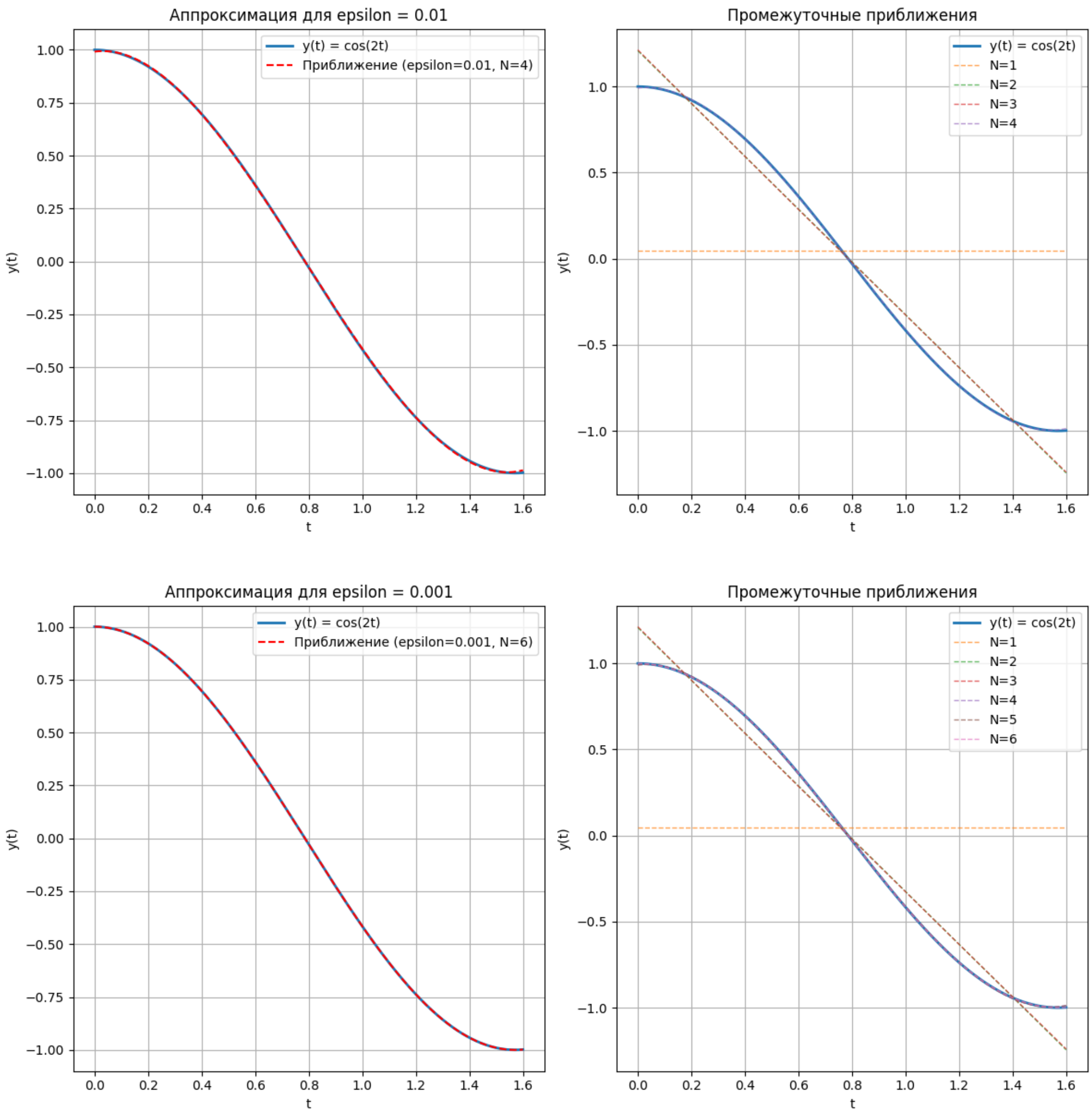
Итерация 4: error = 0.003892

Итерация 5: error = 0.003843

Итерация 6: error = 0.000062

Достигнута требуемая точность на итерации 6





На левых подграфиках изображены график функции y(t) и его приближения частичными суммами ряда Фурье. Для epsilon = 0,1 сходимость достигается за 4 итерации, для epsilon = 0,01 – за 4 итерации, а для epsilon = 0,001 – за 6 итераций.

На правых подграфиках представлены промежуточные приближения (с первой итерации до итерации, на которой достигается заданная точность разложения).