РГР 2 по функциональному анализу

Выполнил студент

группы М8О-307Б-22

Кострюков Евгений Сергеевич

Задание

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.

Номер в списке – 15. Группа 307 => Вариант 5. l = 10, k = 7

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

Код программы

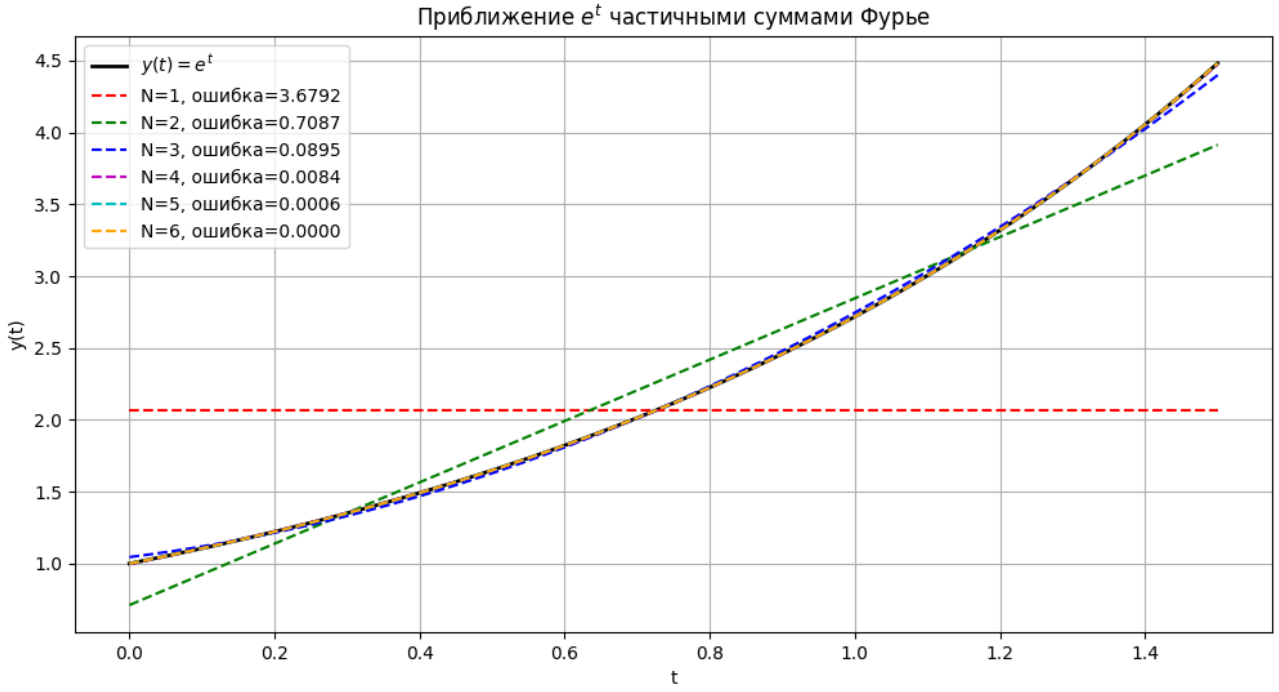
```python
import numpy as np
from scipy.integrate import quad
import matplotlib.pyplot as plt
from typing import Callable, List

class FourierApproximator:
    """
    Класс для ортогонализации системы функций и приближения заданной функции
    частичными суммами Фурье в пространстве с весом f(t).
    """
    def __init__(self,
                 interval: tuple,
                 weight_func: Callable[[float], float],
                 target_func: Callable[[float], float],
                 basis_size: int):
        self.a, self.b = interval
        self.f = weight_func
        self.y = target_func
        self.N = basis_size

        self.basis_funcs = [lambda t, n=n: t ** (n - 1) for n in range(1, self.N + 1)]
        self.ortho_basis = self._gram_schmidt()
        self.coefficients = self._compute_coefficients()

    def _inner_product(self, func1: Callable[[float], float],
                       func2: Callable[[float], float]) -> float:
        """Скалярное произведение с весом f(t)."""
        integrand = lambda t: func1(t) * func2(t) * self.f(t)
        return quad(integrand, self.a, self.b)[0]

    def _gram_schmidt(self) -> List[Callable[[float], float]]:
        """Ортогонализация базисных функций методом Грама-Шмидта."""
        ortho_funcs = []
        for i, func in enumerate(self.basis_funcs):
            def current_func(t, f=func): return f(t)
            for j in range(i):
                phi_j = ortho_funcs[j]
                proj = self._inner_product(current_func, phi_j) / self._inner_product(phi_j, phi_j)
                current_func = lambda t, cf=current_func, pj=phi_j, p=proj: cf(t) - p * pj(t)
            ortho_funcs.append(current_func)
        return ortho_funcs

    def _compute_coefficients(self) -> List[float]:
        """Вычисление коэффициентов Фурье по ортогональному базису."""
        coeffs = []
        for phi in self.ortho_basis:
            num = self._inner_product(self.y, phi)
            denom = self._inner_product(phi, phi)
```

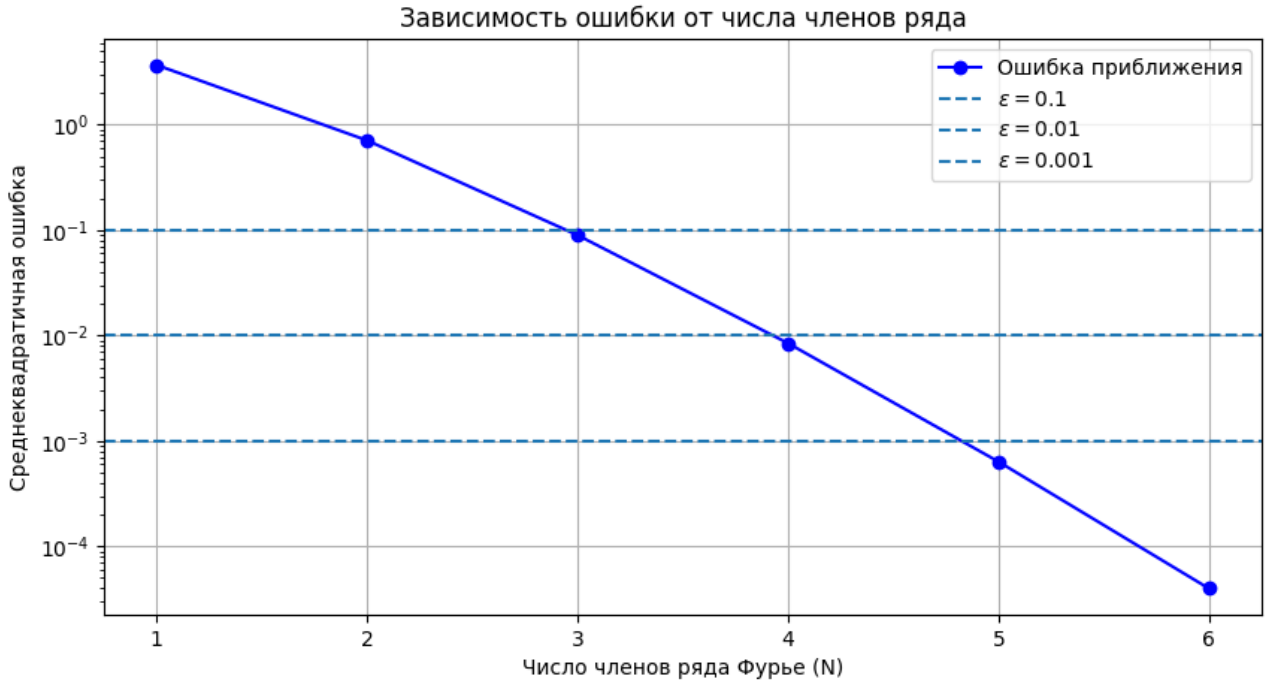
```
            coeffs.append(num / denom)
        return coeffs

    def partial_sum(self, t: float, n_terms: int) -> float:
        """Частичная сумма ряда Фурье по первым n_terms членам."""
        return sum(
            c * phi(t)
            for c, phi in zip(self.coefficients[:n_terms], self.ortho_basis[:n_terms])
        )

    def projection_error(self, n_terms: int) -> float:
        """Среднеквадратичная ошибка приближения."""
        approx = lambda t: self.partial_sum(t, n_terms)
        integrand = lambda t: (self.y(t) - approx(t)) ** 2 * self.f(t)
        return np.sqrt(quad(integrand, self.a, self.b)[0])

    def plot_approximations(self, epsilons: List[float]):
        """График y(t) и приближений частичными суммами Фурье для заданных ε."""
        t_vals = np.linspace(self.a, self.b, 500)
        y_vals = self.y(t_vals)

        plt.figure(figsize=(12, 6))
        plt.plot(t_vals, y_vals, 'k-', linewidth=2, label='$y(t) = e^t$')

        errors = [self.projection_error(n) for n in range(1, self.N + 1)]

        for eps in epsilons:
            for n, err in enumerate(errors, 1):
                if err < eps:
                    break
            else:
                n = self.N

            approx_vals = [self.partial_sum(t, n) for t in t_vals]
            plt.plot(t_vals, approx_vals, '--', label=f'N={n}, $\\epsilon$={eps}')

        plt.xlabel('t')
        plt.ylabel('y(t)')
        plt.title('Приближение $e^t$ с весом $f(t) = (4 - t)^2$')
        plt.legend()
        plt.grid()
        plt.show()

    def plot_error_curve(self):
        """График зависимости ошибки от числа членов ряда."""
```

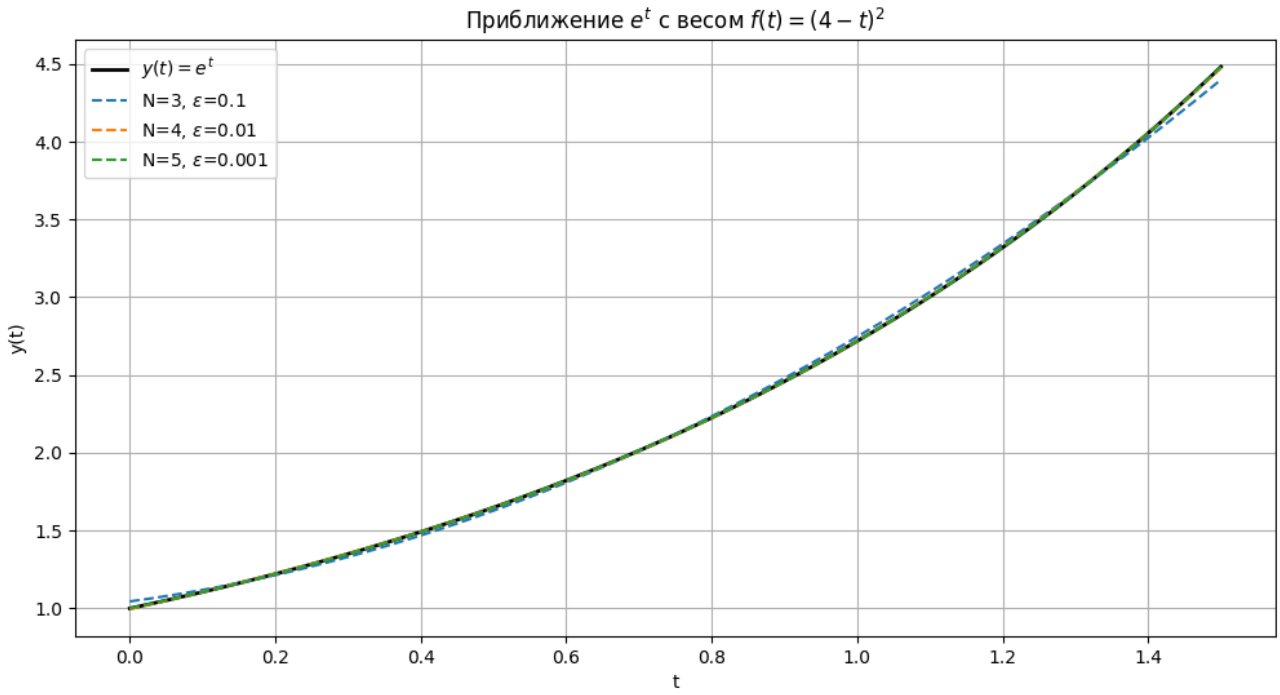
```python
        errors = [self.projection_error(n) for n in range(1, self.N + 1)]

        plt.figure(figsize=(10, 5))
        plt.plot(range(1, self.N + 1), errors, 'bo-', label='Ошибка приближения')
        for eps in [0.1, 0.01, 0.001]:
            plt.axhline(y=eps, linestyle='--', label=f'$\\epsilon = {eps}$')
        plt.xlabel('Число членов ряда Фурье (N)')
        plt.ylabel('Среднеквадратичная ошибка')
        plt.title('Зависимость ошибки от числа членов ряда')
        plt.yscale('log')
        plt.legend()
        plt.grid()
        plt.show()

        for n, err in enumerate(errors, 1):
            print(f"N = {n}: ошибка = {err:.6f}")
        return errors

    def plot_all_partial_sums(self):
        """График всех частичных сумм Фурье от 1 до N с подписями ошибок."""
        t_vals = np.linspace(self.a, self.b, 500)
        y_vals = self.y(t_vals)

        plt.figure(figsize=(12, 6))
        plt.plot(t_vals, y_vals, 'k-', linewidth=2, label='$y(t) = e^t$')

        colors = ['r', 'g', 'b', 'm', 'c', 'orange', 'purple', 'brown']
        errors = []

        for n in range(1, self.N + 1):
            approx_vals = [self.partial_sum(t, n) for t in t_vals]
            error = self.projection_error(n)
            errors.append(error)
            color = colors[(n - 1) % len(colors)]
            plt.plot(t_vals, approx_vals, '--', color=color, label=f'N={n}, ошибка={error:.4f}')

        plt.xlabel('t')
        plt.ylabel('y(t)')
        plt.title('Приближение $e^t$ частичными суммами Фурье')
        plt.legend()
        plt.grid()
        plt.show()

        return errors
```

```python

if __name__ == "__main__":
    # Настройка параметров задачи
    a, b = 0, 1.5
    weight_function = lambda t: (4 - t)**2
    target_function = lambda t: np.exp(t)
    N = 6

    approximator = FourierApproximator(
        interval=(a, b),
        weight_func=weight_function,
        target_func=target_function,
        basis_size=N
    )

    print("Коэффициенты Фурье:")
    for i, c in enumerate(approximator.coefficients):
        print(f"c_{i + 1} = {c:.6f}")

    print("\nОшибки проектирования для разных N:")
    errors = []
    for n in range(1, N + 1):
        err = approximator.projection_error(n)
        errors.append(err)
        print(f"N = {n}: ||e|| = {err:.6f}")

    epsilons = [0.1, 0.01, 0.001]
    for eps in epsilons:
        for n, err in enumerate(errors, 1):
            if err < eps:
                print(f"\nДля точности ε = {eps} достаточно N = {n} (ошибка = {err:.6f})")
                break
        else:
            print(f"\nДля точности ε = {eps} требуется N > {N} (текущая минимальная ошибка = {errors[-1]:.6f})")

    approximator.plot_all_partial_sums()
    approximator.plot_error_curve()
    approximator.plot_approximations(epsilons)
```

Вывод программы и графики








```
Коэффициенты Фурье:
c_1 = 2.070225
c_2 = 2.135427
c_3 = 1.068986
c_4 = 0.355968
c_5 = 0.088896
c_6 = 0.017763

Ошибки проектирования для разных N:
N = 1: ||e|| = 3.679207
N = 2: ||e|| = 0.708713
N = 3: ||e|| = 0.089506
N = 4: ||e|| = 0.008437
N = 5: ||e|| = 0.000635
N = 6: ||e|| = 0.000040

Для точности ε = 0.1 достаточно N = 3 (ошибка = 0.089506)

Для точности ε = 0.01 достаточно N = 4 (ошибка = 0.008437)

Для точности ε = 0.001 достаточно N = 5 (ошибка = 0.000635)
N = 1: ошибка = 3.679207
N = 2: ошибка = 0.708713
N = 3: ошибка = 0.089506
N = 4: ошибка = 0.008437
N = 5: ошибка = 0.000635
N = 6: ошибка = 0.000040
```