Федеральное государственное автономное образовательное учреждение высшего образования **«Национальный исследовательский университет ИТМО»**

Факультет Программной Инженерии и Компьютерной Техники

Лабораторная работа **№4**

**«**АППРОКСИМАЦИЯ ФУНКЦИИ МЕТОДОМ

НАИМЕНЬШИХ КВАДРАТОВ**»**

по дисциплине «Вычислительная математика**»**

Вариант: **18**

**Преподаватель:**

**Выполнил:**

Шаматульский Роман Константинович

**Группа:** Р3212

Санкт-Петербург, 2025 г.

Оглавление

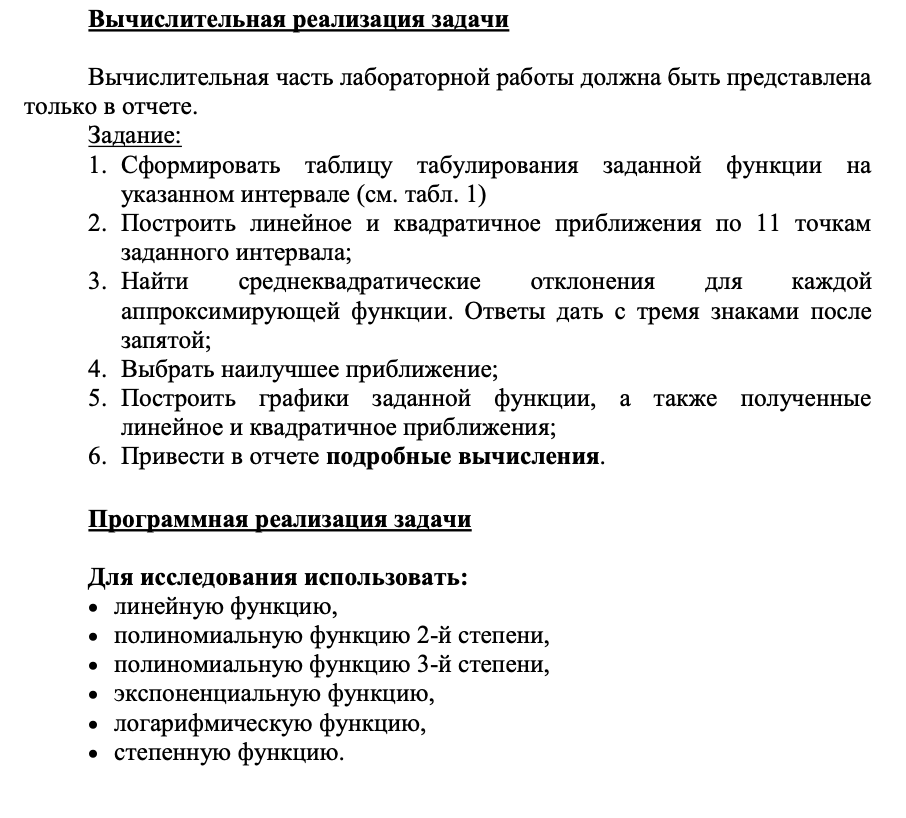
[Текст задания 3](#_Toc196928063)

[Вычислительная часть 4](#_Toc196928064)

[Программная часть 6](#_Toc196928065)

[Вывод 10](#_Toc196928066)

# Текст задания



# Вычислительная часть

**Таблица табулирования функции:**

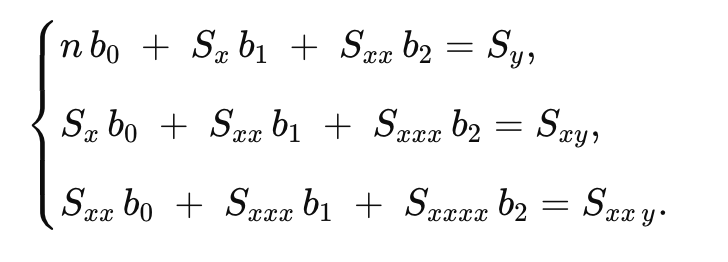
|  |  |
| --- | --- |
| x | f(x) |
| 0,0 | 0.0 |
| 0,4 | |  | | --- | |  |   0.665720 |
| 0,8 | 1.303668 |
| 1,2 | 1.793400 |
| 1,6 | 1.954907 |
| 2,0 | 1.764706 |
| 2,4 | 1.406866 |
| 2,8 | 1.057061 |
| 3,2 | 0.781392 |
| 3,6 | 0.580765 |
| 4,0 | 0.437956 |

**Линейное и квадратичное приближение по 11 точкам:**Линейное приближение строим по формуле

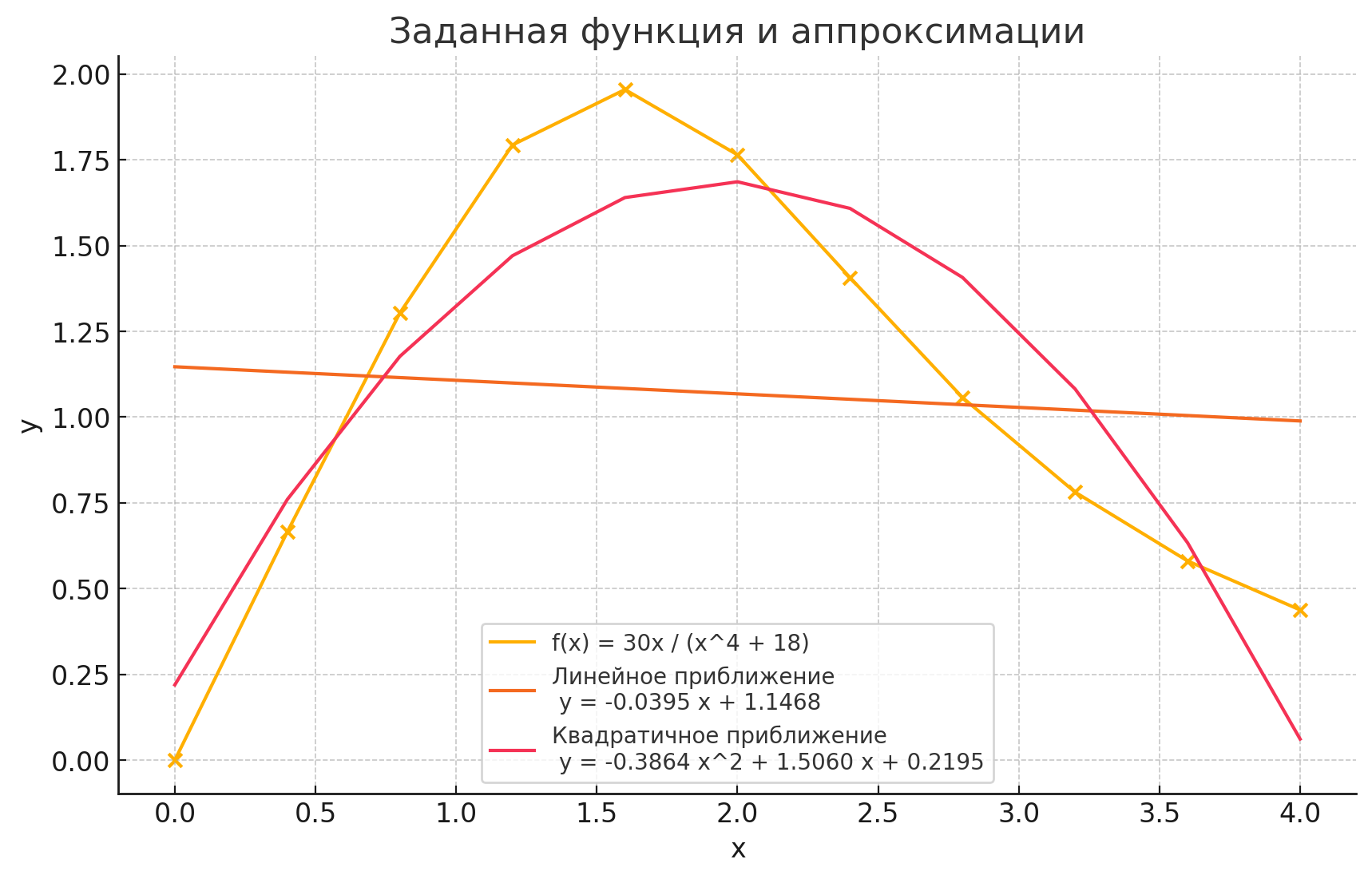
, где ,

y = -0.039491x + 1.146840

Квадратичное:



y = -0.386383x² + 1.506039·x + 0.219521



**Среднеквадратичное отклонение:**

Линейная аппроксимация: 0.6

Квадратичная аппроксимация: 0.249

Соответственно, квадратичное приближение лучше, так как отклонение у него меньше.

# Программная часть

Листинг:

import sys  
import math  
from pathlib import Path  
  
import numpy as np  
import matplotlib.pyplot as plt  
  
# -----------------------------------------------------  
# helpers ------------------------------------------------  
  
def read\_points\_interactive():  
 print("Введите пары x y (пустая строка — конец ввода):")  
 pts = []  
 while True:  
 line = input().strip()  
 if not line:  
 break  
 parts = line.replace(",", ".").split()  
 if len(parts) != 2:  
 print("Ожидалось два числа через пробел")  
 continue  
 pts.append((float(parts[0]), float(parts[1])))  
 return pts  
  
  
def read\_points\_from\_file() -> list:  
 fname = input("Имя файла с данными: ").strip()  
 path = Path(fname)  
 if not path.exists():  
 print("Файл не найден"); sys.exit(1)  
 pts = []  
 for line in path.read\_text().splitlines():  
 line = line.strip()  
 if not line:  
 continue  
 parts = line.replace(",", ".").split()  
 if len(parts) != 2:  
 continue  
 pts.append((float(parts[0]), float(parts[1])))  
 return pts  
  
  
def request\_points():  
 mode = input("Читать точки из файла? (y/n): ").strip().lower()  
 pts = read\_points\_from\_file() if mode == "y" else read\_points\_interactive()  
 if not (8 <= len(pts) <= 12):  
 print("Нужно 8–12 точек"); sys.exit(1)  
 xs, ys = map(np.array, zip(\*pts))  
 return xs, ys  
  
  
# -----------------------------------------------------  
# метрики ------------------------------------------------  
  
def rms(pred, real):  
 return math.sqrt(np.mean((pred - real) \*\* 2))  
  
def r2\_score(pred, real):  
 ss\_res = np.sum((real - pred) \*\* 2)  
 ss\_tot = np.sum((real - real.mean()) \*\* 2)  
 return 1 - ss\_res / ss\_tot if ss\_tot else 0.0  
  
  
# -----------------------------------------------------  
# модели ------------------------------------------------  
  
def model\_linear(xs, ys):  
 c = np.polyfit(xs, ys, 1)  
 return np.polyval(c, xs), c  
  
def model\_poly2(xs, ys):  
 c = np.polyfit(xs, ys, 2)  
 return np.polyval(c, xs), c  
  
def model\_poly3(xs, ys):  
 c = np.polyfit(xs, ys, 3)  
 return np.polyval(c, xs), c  
  
def model\_expo(xs, ys):  
 if np.any(ys <= 0):  
 return None, None  
 b, ln\_a = np.polyfit(xs, np.log(ys), 1)  
 a = math.exp(ln\_a)  
 return a \* np.exp(b \* xs), (a, b)  
  
def model\_log(xs, ys):  
 if np.any(xs <= 0):  
 return None, None  
 b, a = np.polyfit(np.log(xs), ys, 1)  
 return a + b \* np.log(xs), (a, b)  
  
def model\_power(xs, ys):  
 if np.any(xs <= 0) or np.any(ys <= 0):  
 return None, None  
 b, ln\_a = np.polyfit(np.log(xs), np.log(ys), 1)  
 a = math.exp(ln\_a)  
 return a \* xs \*\* b, (a, b)  
  
MODELS = {  
 "linear": (model\_linear, "y = a₀ + a₁ x"),  
 "poly2": (model\_poly2, "y = a₂ x² + a₁ x + a₀"),  
 "poly3": (model\_poly3, "y = a₃ x³ + a₂ x² + a₁ x + a₀"),  
 "expo": (model\_expo, "y = A · e^{Bx}"),  
 "log": (model\_log, "y = A + B · ln x"),  
 "power": (model\_power, "y = A · x^{B}"),  
}  
  
# -----------------------------------------------------  
# main --------------------------------------  
  
def main():  
 print("Будут исследоваться следующие формы зависимостей:")  
 for name, (\_, formula) in MODELS.items():  
 print(f" {name:6s}: {formula}")  
 print()  
  
 xs, ys = request\_points()  
  
 results = {}  
 for name, (fn, \_) in MODELS.items():  
 pred, coeffs = fn(xs, ys)  
 if pred is None:  
 continue  
 results[name] = {  
 "sigma": rms(pred, ys),  
 "coeffs": coeffs,  
 "pred": pred,  
 "r2": r2\_score(pred, ys),  
 }  
  
 if not results:  
 print("Не удалось построить ни одной модели"); sys.exit(1)  
  
 best\_name = min(results, key=lambda n: results[n]["sigma"])  
  
 out\_mode = input("Сохранить результаты в файл? (y/n): ").strip().lower()  
 if out\_mode == "y":  
 out\_path = Path(input("Имя выходного файла: ").strip())  
 with out\_path.open("w", encoding="utf-8") as f:  
 for name, res in results.items():  
 f.write(f"{name}: σ={res['sigma']:.3f} R²={res['r2']:.3f} coeffs={res['coeffs']}\n")  
 f.write(f"Лучшее приближение: {best\_name} (σ={results[best\_name]['sigma']:.3f})\n")  
 if "linear" in results:  
 r = np.corrcoef(xs, ys)[0, 1]  
 f.write(f"Корреляция Пирсона (linear): r={r:.3f}\n")  
 print(f"Результаты сохранены в {out\_path}")  
 else:  
 for name, res in results.items():  
 print(f"{name:6s}: σ={res['sigma']:.3f} R²={res['r2']:.3f} coeffs={res['coeffs']}")  
 print(f"Лучшее приближение => {best\_name} (σ={results[best\_name]['sigma']:.3f})")  
 if "linear" in results:  
 r = np.corrcoef(xs, ys)[0, 1]  
 print(f"Корреляция Пирсона (linear): r={r:.3f}")  
  
 plt.scatter(xs, ys, label="data", zorder=3)  
 pu = np.linspace(xs.min(), xs.max(), 400)  
 for name, res in results.items():  
 if name.startswith("poly"):  
 y\_plot = np.polyval(res["coeffs"], pu)  
 elif name == "linear":  
 y\_plot = res["coeffs"][0] \* pu + res["coeffs"][1]  
 elif name == "expo":  
 y\_plot = res["coeffs"][0] \* np.exp(res["coeffs"][1] \* pu)  
 elif name == "log":  
 y\_plot = res["coeffs"][0] + res["coeffs"][1] \* np.log(pu)  
 elif name == "power":  
 y\_plot = res["coeffs"][0] \* pu \*\* res["coeffs"][1]  
 else:  
 continue  
 plt.plot(pu, y\_plot, label=name)  
 plt.title("Аппроксимация точек")  
 plt.legend()  
 plt.grid(True)  
 plt.show()  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()

Результаты работы:

linear: σ=0.091 R²=0.999 coeffs=[0.99030303 0.04363636]

poly2 : σ=0.090 R²=0.999 coeffs=[9.46969697e-04 9.81780303e-01 5.50000000e-02]

poly3 : σ=0.087 R²=0.999 coeffs=[ 0.00135004 -0.01727855 1.04401709 0.02097902]

Лучшее приближение => poly3 (σ=0.087)

Корреляция Пирсона (linear): r=0.999

linear: σ=7.289 R²=0.927 coeffs=[ 9.01939394 -12.02727273]

poly2 : σ=0.113 R²=1.000 coeffs=[ 1.0030303 -0.00787879 0.00909091]

poly3 : σ=0.102 R²=1.000 coeffs=[-0.0027972 1.04079254 -0.13682984 0.07958042]

Лучшее приближение => poly3 (σ=0.102)

Корреляция Пирсона (linear): r=0.963

linear: σ=10.203 R²=0.811 coeffs=[ 7.35981818 -10.43618182]

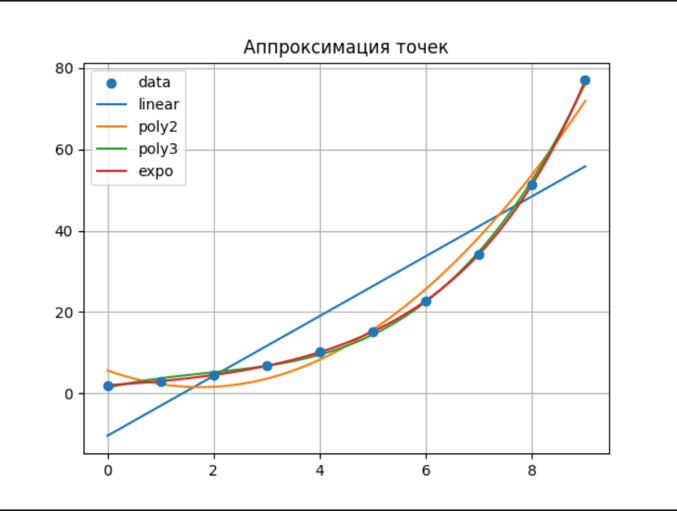
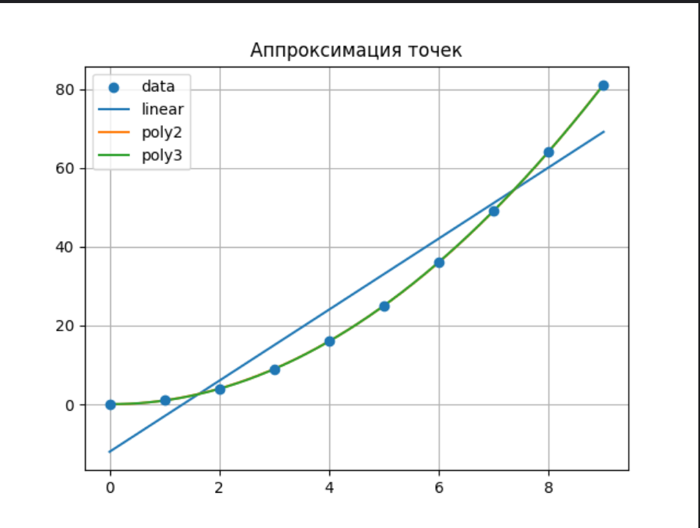
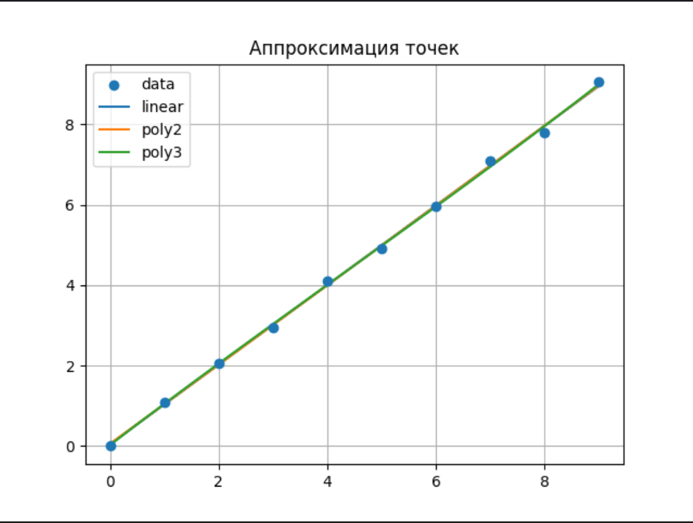
poly2 : σ=3.060 R²=0.983 coeffs=[ 1.33950758 -4.69575 5.63790909]

poly3 : σ=0.679 R²=0.999 coeffs=[ 0.16973776 -0.95195221 3.12916084 1.36051748]

expo : σ=0.005 R²=1.000 coeffs=(1.999826511865379, np.float64(0.4055867684931256))

Лучшее приближение => expo (σ=0.005)

Корреляция Пирсона (linear): r=0.901



# Вывод

В ходе данной лабораторной работы я вручную исследовал функцию, строил ее приближения и находил лучшее. Кроме того, была реализована программа для аппроксимации экспериментальных данных с помощью шести моделей. По значениям среднеквадратического отклонения и коэффициента детерминации определялась наиболее точная модель.