|  |  |
| --- | --- |
| ***Вариант задания №2*** | |
| *Код программы:* | *Код оптимизированной программы:* |
| ***import numpy as np  # Квадратичная функция def f1(x1, x2):  return 100\*(x2 - x1)\*\*2 + (1 - x1)\*\*2  # Функция Розенброка def f2(x1, x2):  return 100\*(x2 - x1\*\*2)\*\*2 + (1 - x1)\*\*2  # Тестовая функция варианта def f3(x, y):  return (1/(1 + ((x-2)/3)\*\*2 + ((y-2)/3)\*\*2) +  3/(1 + (x-1)\*\*2 + ((y-1)/2)\*\*2))  # Градиентный спуск (упрощенная реализация) def gradient\_descent(func, start\_x, start\_y, lr=0.001, max\_iter=1000):  x, y = start\_x, start\_y  for \_ in range(max\_iter):  grad\_x = (func(x + 0.001, y) - func(x - 0.001, y)) / 0.002  grad\_y = (func(x, y + 0.001) - func(x, y - 0.001)) / 0.002  x -= lr \* grad\_x  y -= lr \* grad\_y  return x, y, func(x, y)  # Исследование для f1 points = [(0,0), (2,2)] for point in points:  result = gradient\_descent(f1, \*point)  print(f"f1: Начало {point} -> Минимум ({result[0]:.4f}, {result[1]:.4f}), значение={result[2]:.4f}")  # Исследование для f2 for point in points:  result = gradient\_descent(f2, \*point)  print(f"f2: Начало {point} -> Минимум ({result[0]:.4f}, {result[1]:.4f}), значение={result[2]:.4f}")  # Исследование для f3 points = [(0,0), (3,3)] for point in points:  result = gradient\_descent(f3, \*point)  print(f"f3: Начало {point} -> Минимум ({result[0]:.4f}, {result[1]:.4f}), значение={result[2]:.4f}")*** | ***import numpy as np  def quadratic(x1, x2):  return 100\*(x2 - x1)\*\*2 + (1 - x1)\*\*2  def rosenbrock(x1, x2):  return 100\*(x2 - x1\*\*2)\*\*2 + (1 - x1)\*\*2  def custom\_func(x, y):  return (1/(1 + ((x-2)/3)\*\*2 + ((y-2)/3)\*\*2) +  3/(1 + (x-1)\*\*2 + ((y-1)/2)\*\*2))  def compute\_gradient(func, x, y, h=1e-5):  dx = (func(x+h, y) - func(x-h, y)) / (2\*h)  dy = (func(x, y+h) - func(x, y-h)) / (2\*h)  return dx, dy  def optimize(func, start\_points, lr=0.001, max\_iter=1000):  results = []  for x0, y0 in start\_points:  x, y = x0, y0  for \_ in range(max\_iter):  grad\_x, grad\_y = compute\_gradient(func, x, y)  x -= lr \* grad\_x  y -= lr \* grad\_y  results.append((x, y, func(x, y)))  return results  # Исследование всех функций functions = [quadratic, rosenbrock, custom\_func] start\_points = [(0,0), (2,2), (3,3)]  for idx, func in enumerate(functions, 1):  res = optimize(func, start\_points[:2] if idx <3 else start\_points[1:])  print(f"\nФункция {idx}:")  for (x, y, val), point in zip(res, start\_points):  print(f"Старт {point} -> ({x:.4f}, {y:.4f}), f={val:.4f}")*** |
| *Расчет метрики:* | *Расчет метрики:* |
|  | |