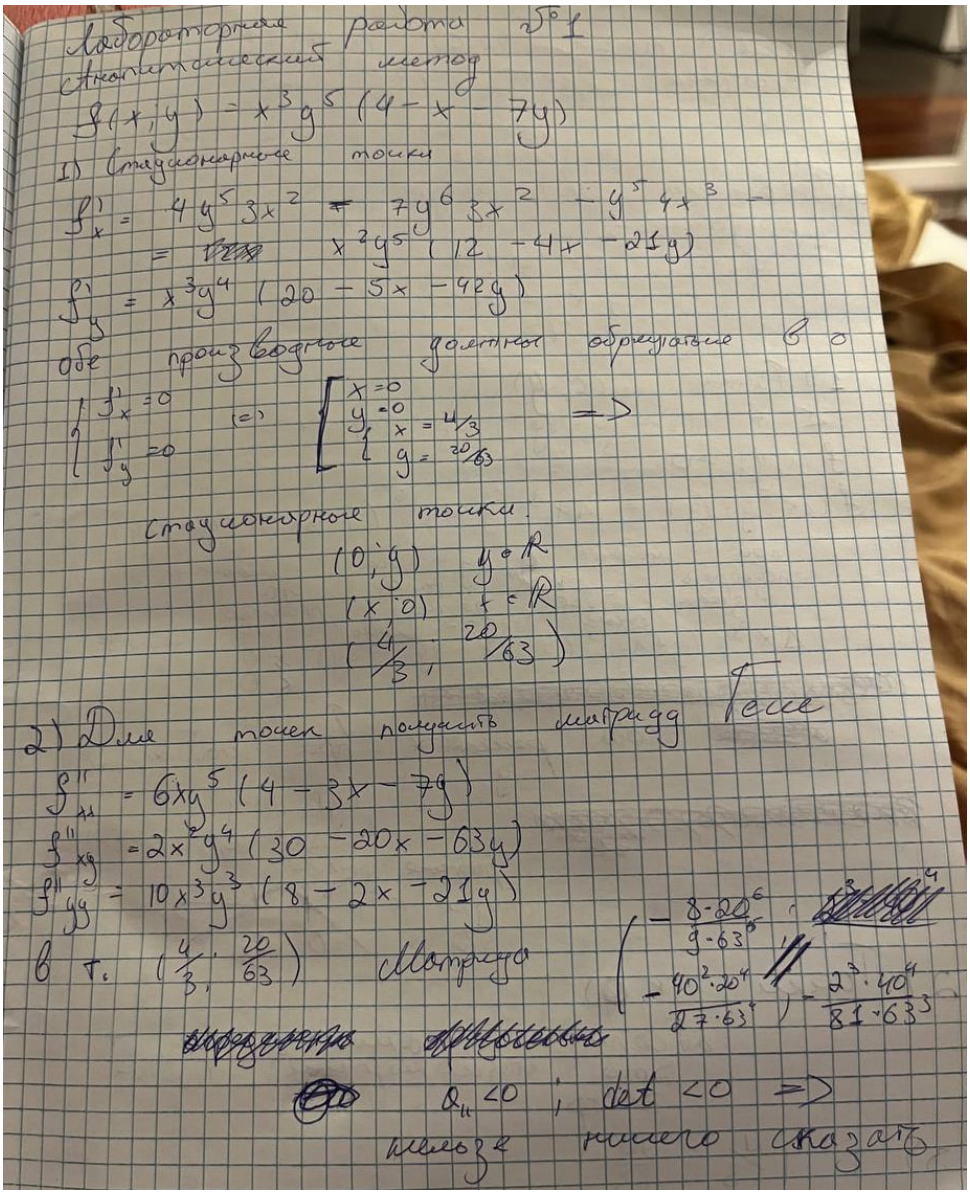
Лабораторная работа №1 по математическому анализу

Веселкова Варвара Артемовна, М3234

Часть 1. Аналитический метод



для точек $(x, 0)$ $(0, y)$
матрица будет $\begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \Rightarrow$

так не нельзя ничего сказать

3) Проверим экстремумы по определению

$\exists\ U(x_0): \forall x \in U(x_0)$

$f(x) \le f(x_0)$

$(\ge$

1) Точка $(0; y)$

Δ т. $(\Delta x; y + \Delta y)$ из окр-ти

$f(0, y) = 0$

$f(\Delta x; y + \Delta y) = \Delta x^3 (y + \Delta y)^5 (4 - \Delta x - 7y - 7\Delta y)$

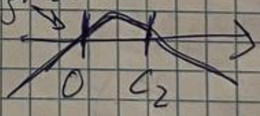
при любом $y, \Delta y$ мы можем подобрать Δx такие, что знак будет разный причем он будет сколь угодно малым

$\Delta x^3 (y + \Delta y)^5 (4 - \Delta x - 7y - 7\Delta y) \ge 0$
≤ 0

оба реализуются при любом $y, \Delta y$

(ур. вида $\Delta x^3 \cdot C_1 \cdot (C_2 - \Delta x) \ge 0$
≤ 0

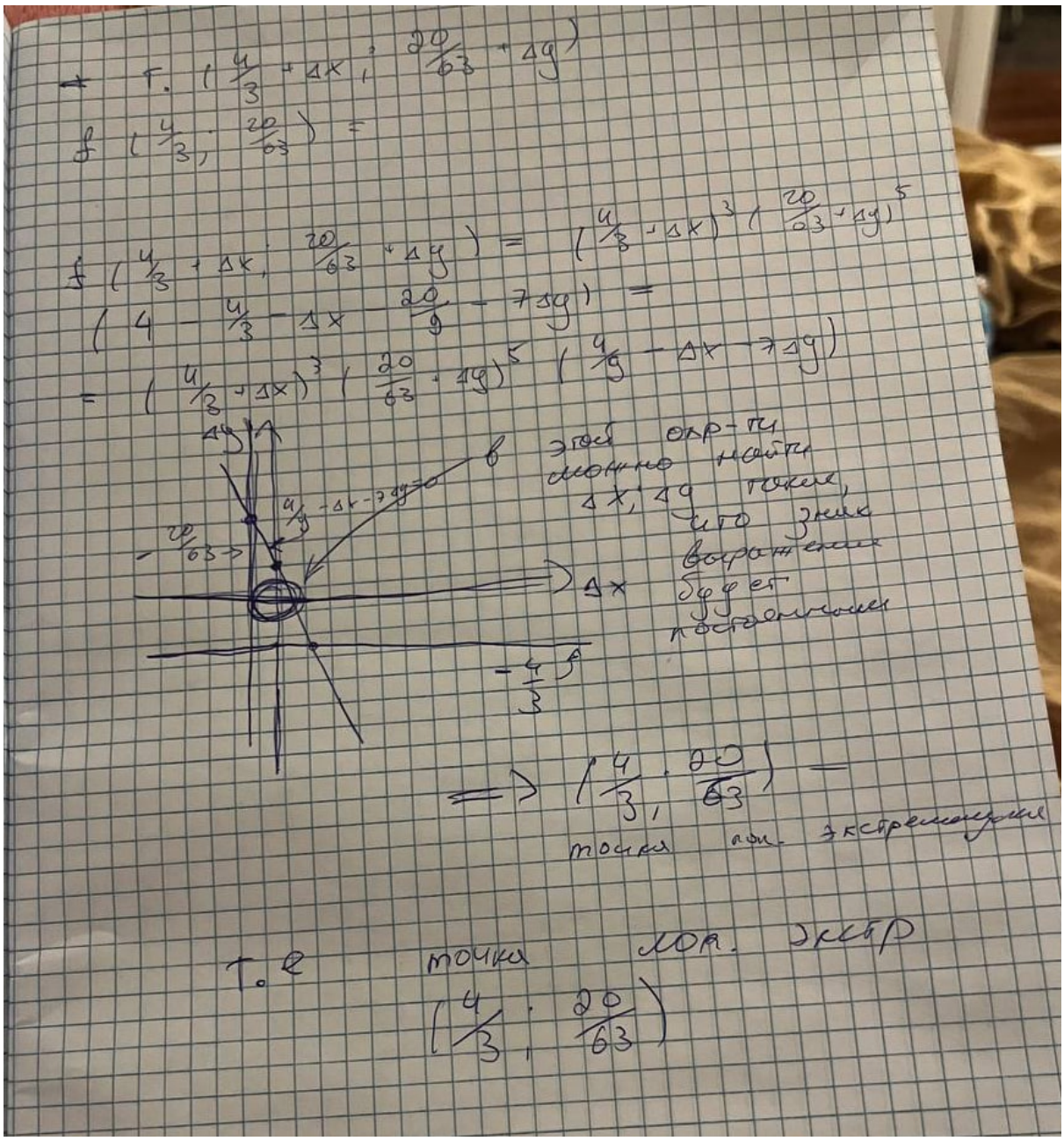
разные знаки близко к 0



2) Точки $(x, 0)$

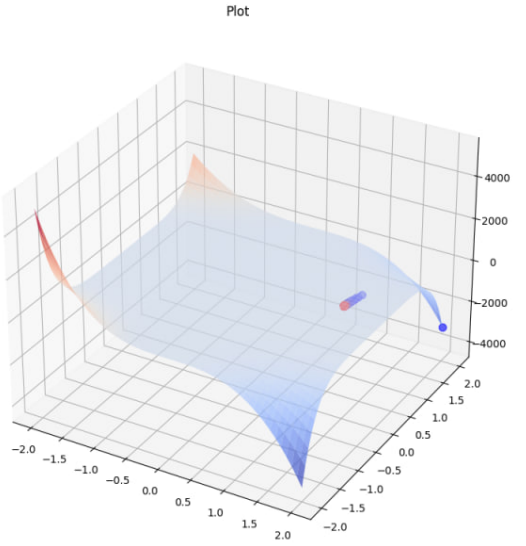
абсолютно аналогично, это стационарные т., но экстр.

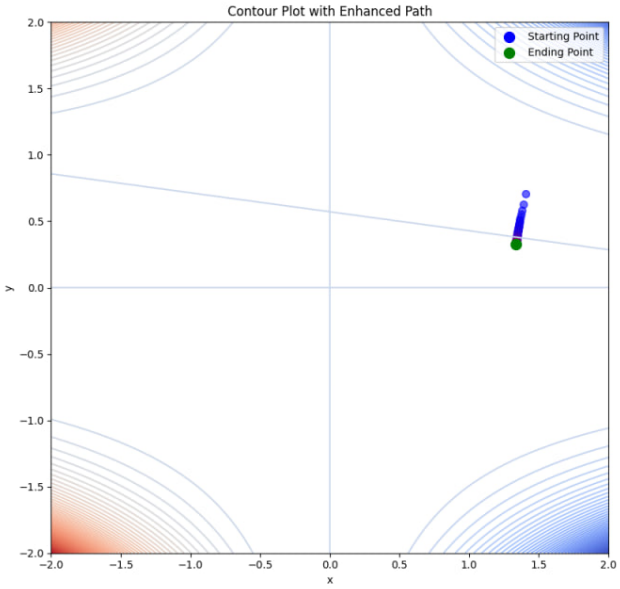
3) т $\left(\frac{4}{3}; \frac{20}{63}\right)$


т. $\left(\frac{4}{3}+\Delta x;\ \frac{20}{63}+\Delta y\right)$
$\left(\frac{4}{3};\ \frac{20}{63}\right)$ =
$f\left(\frac{4}{3}+\Delta x;\ \frac{20}{63}+\Delta y\right) = \left(\frac{4}{3}+\Delta x\right)^3 \left(\frac{20}{63}+\Delta y\right)^5$
$\left(4 - \frac{4}{3} - \Delta x \quad \frac{20}{9} - \frac{20}{63} - \Delta y\right) =$
$= \left(\frac{4}{3}+\Delta x\right)^3 \left(\frac{20}{63}+\Delta y\right)^5 \left(\frac{4}{9} - \Delta x - \Delta y\right)$
$\frac{4}{9} - \Delta x - \Delta y = 0$
$\frac{20}{63}$
Δx
$-\frac{4}{3}$
в этой окр-ти можно найти Δx, Δy такие, что знак выражения будет противоположным
$\Rightarrow \left(\frac{4}{3};\ \frac{20}{63}\right)$ =
точка лок. экстремума
т. е точка лок. экстр
$\left(\frac{4}{3};\ \frac{20}{63}\right)$

Часть 2. Результаты работы программы

а. Графики





б. Полученная точка и значение a_k

$(x, y) = (1.33498285, 0.3260501)$
$a_k = 10^{-4}$

в. Критерий останова

np.linalg.norm(new_point - way[-2]) – вычисляем Евклидову норму расстояния между текущей и предыдущей точкой. Если она приближается к 0, значит мы становимся все ближе к локальному минимуму. Взято $\varepsilon = 10^{-6}$ как достаточно маленькое значение, при котором программа совершает не слишком много итераций, но вычисляет достаточно точно. В случае, если бы итераций было слишком много, количество итераций было бы ограниченно сверху числом 100000.

г. Число итераций

17922

д. Время работы

0.1678624153137207 секунд

е. Дополнительные результаты

- Значение функции в конечной точке: 0.003355
- Точное значение: (1.3333333333333333, 0.31746031746031744)
- Значение функции в точном значении: 0.003397
- Разница между конечным и точным значениями: -0.000042

Комментарии по коду

Наша функция из условия и ее производные

```
def f(x, y):
    return x ** 3 * y ** 5 * (4 - x - 7 * y)
def grad_f(x, y):
    df_dx = x ** 2 * y ** 5 * (12 - 4 * x - 21 * y)
    df_dy = x ** 3 * y ** 4 * (20 - 5 * x - 42 * y)
    return np.array([df_dx, df_dy])
```

Вычисление градиентного спуска с константами, обоснованными выше

```
def gradient_descent(start, ak=1e-4, eps=1e-6, max_iterations=100000):
    way, grad = [np.array(start)], None
    for _ in range(max_iterations):
        grad = grad_f(*way[-1])
        new_point = way[-1] + ak * grad
        way.append(new_point)
        if np.linalg.norm(new_point - way[-2]) < eps:
            return way, "Convergence"
    return way, "Max Iterations Reached"
```

Начальную точку выбрала (2, 2) как достаточно близкую, но удобную

```
starting_point = [2, 2]
start_time = time.time()
path, break_condition = gradient_descent(starting_point)
end_time = time.time()
```

Набор штук для визуализации графика

```
path_x, path_y = zip(*path)
x_expanded, y_expanded = np.linspace(-2, 2, 600), np.linspace(-2, 2, 600)
x_expanded, y_expanded = np.meshgrid(x_expanded, y_expanded)
z_expanded = f(x_expanded, y_expanded)
def get_color_gradient(n, start_color, end_color):
    return [start_color + (end_color - start_color) * i / n for i in range(n)]

color_gradient = get_color_gradient(len(path), np.array([0, 0, 1]), np.array([1, 0, 0]))
fig = plt.figure(figsize=(16, 8))
ax1, ax2 = fig.add_subplot(121, projection='3d'), fig.add_subplot(122)
ax1.plot_surface(x_expanded, y_expanded, z_expanded, cmap=cm.coolwarm, alpha=0.7)
ax2.contour(x_expanded, y_expanded, z_expanded, 100, cmap=cm.coolwarm)

for i in range(0, len(path), 100):
    ax1.scatter(path_x[i], path_y[i], f(path_x[i], path_y[i]), color=color_gradient[i],
    alpha=0.6, s=50)
    ax2.scatter(path_x[i], path_y[i], color=color_gradient[i], alpha=0.6, s=50)
ax2.scatter([starting_point[0]], [starting_point[1]], color='blue', s=100, label='
    Starting Point')
ax2.scatter([path_x[-1]], [path_y[-1]], color='green', s=100, label='Ending Point')
ax2.set_title('Contour Plot with Enhanced Path')
ax2.set_xlabel('x')
ax2.set_ylabel('y')
ax2.legend()
plt.tight_layout()
plt.show()
```

Вычисляем значения для ответа

```
final_point = path[-1]
final_value = f(*final_point)
exact_point = (4/3, 20/63)
exact_value = f(*exact_point)
execution_time = end_time - start_time

print(f"Results:\n"
      f"- Break Condition: {break_condition}\n"
      f"- Final Point: {final_point}\n"
      f"- Function Value at Final Point: {final_value:.6f}\n"
      f"- Exact Point: {exact_point}\n"
      f"- Function Value at Exact Point: {exact_value:.6f}\n"
      f"- Difference between Final and Exact Values: {final_value - exact_value:.6f}\n"
      f"- Execution Time: {execution_time} seconds\n"
      f"- Number of Iterations: {len(path)}")
```