Московский
Авиационный
Институт

Методы оптимизации

Поиск безусловного экстремума

Кафедра 806

Выполнили студенты группы
М8О-308Б-21:

Ползикова Алина
Белоносов Кирилл
Шевлякова София

Москва 2024

Постановка задачи

Найти безусловный экстремум:

$$f(x) = a\left(x_1 - \left[\frac{n}{2}\right]\right)^2 + b(x_2 - m)^2$$

а) с помощью необходимых и достаточных условий;

б) методом градиентного спуска с постоянным шагом;

в) методом Полака-Рибьера;

г) методом Ньютона-Рафсона;

Аналитическое решение

Найти безусловный экстремум с помощью необходимых и достаточных условий

$$f(x) = a\left(x_1 - \left[\frac{n}{2}\right]\right)^2 + b(x_2 - m)^2$$

1. Проверим необходимое условие экстремума первого порядка: $grad\, f(x) = 0$

$$\frac{df(x)}{dx_1} = 2a\left(x_1 - \left[\frac{n}{2}\right]\right)$$

$$\frac{df(x)}{dx_2} = 2b(x_2 - m)$$

$$grad\, f(x) = \begin{pmatrix} 2a\left(x_1 - \left[\frac{n}{2}\right]\right) \\ 2b(x_2 - m) \end{pmatrix}$$

	m = 8 n = 3	m = 8 n = 17	m = 8 n = 28
x*	(1, 8)	(8, 8)	(14, 8)

Аналитическое решение

1. Проверим достаточное условие экстремума первого порядка:

Матрица Гессе

$$H(x) = \begin{pmatrix} 2a & 0 \\ 0 & 2b \end{pmatrix}$$

- Если a > 0 и b > 0, то H(x) > 0 достаточное условие экстремума первого порядка выполняется – x* точка минимума f(x*) = 0
- Если a < 0 и b < 0, то H(x) < 0 достаточное условие экстремума первого порядка выполняется – x* точка максимума f(x*) = 0
- Если a < 0 и b > 0 или a > 0 и b < 0 – нет экстремума
- При остальных значениях a и b достаточное условие экстремума первого порядка не выполняется, значит надо проверить необходимое условие экстремума второго порядка

Аналитическое решение

3. Проверим необходимое условие экстремума второго порядка:

- Если $a = 0$ и $b > 0$, то $H(x) \geq 0$ – x^* может быть локальным минимумом, требуется дополнительное исследование
- Если $a = 0$ и $b < 0$, то $H(x) \leq 0$ – x^* может быть локальным максимумом, требуется дополнительное исследование
- Если $a > 0$ и $b = 0$, то $H(x) \geq 0$ – x^* может быть локальным минимумом, требуется дополнительное исследование
- Если $a < 0$ и $b = 0$, то $H(x) \leq 0$ – x^* может быть локальным максимумом, требуется дополнительное исследование
- Если $a = 0$ и $b = 0$, то $H(x) = 0$ – требуется дополнительное исследование

Метод градиентного спуска с постоянным шагом

Алгоритм

Шаг 1. Задать x^0, $0<\varepsilon<1$, $\varepsilon_1>0$, $\varepsilon_2>0$, M – предельное число итераций.

Найти градиент функции в произвольной точке $\nabla f(x)=\left(\dfrac{\partial f(x)}{\partial x_1},\dots,\dfrac{\partial f(x)}{\partial x_n}\right)^T$.

Шаг 2. Положить $k=0$.

Шаг 3. Вычислить $\nabla f\left(x^k\right)$.

Шаг 4. Проверить выполнение критерия окончания $\left\|\nabla f\left(x^k\right)\right\|<\varepsilon_1$:

а) если критерий выполнен, расчет закончен, $x^*=x^k$;

б) если критерий не выполнен, то перейти к шагу 5.

Метод градиентного спуска с постоянным шагом

Шаг 5. Проверить выполнение неравенства $k \geq M$:

а) если неравенство выполнено, то расчет окончен: $x^* = x^k$;

б) если нет, то перейти к шагу 6.

Шаг 6. Задать величину шага t_k.

Шаг 7. Вычислить $x^{k+1} = x^k - t_k \nabla f\left(x^k\right)$.

Шаг 8. Проверить выполнение условия

$$f\left(x^{k+1}\right) - f\left(x^k\right) < 0 \quad \text{(или } f\left(x^{k+1}\right) - f\left(x^k\right) < -\varepsilon \left\| \nabla f\left(x^k\right) \right\|^2 \text{)}:$$

а) если условие выполнено, то перейти к шагу 9;

б) если условие не выполнено, положить $t_k = \dfrac{t_k}{2}$ и перейти к шагу 7.

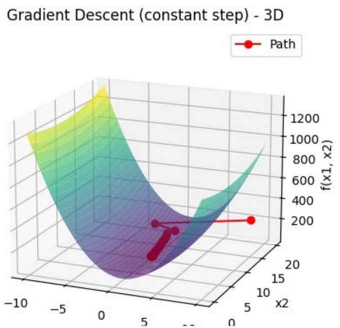
Шаг 9. Проверить выполнение условий

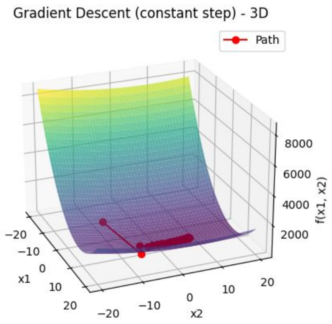
$$\left\| x^{k+1} - x^k \right\| < \varepsilon_2, \quad \left| f\left(x^{k+1}\right) - f\left(x^k\right) \right| < \varepsilon_2:$$

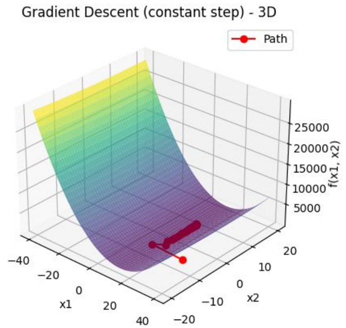
а) если оба условия выполнены при текущем значении k и $k = k-1$, то расчет окончен, $x^* = x^{k+1}$;

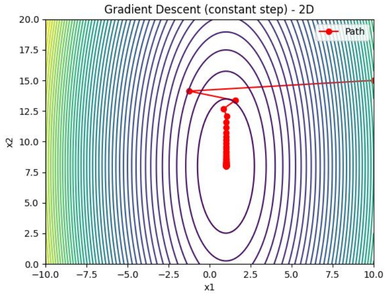
б) если хотя бы одно из условий не выполнено, положить $k = k+1$ и перейти к шагу 3.

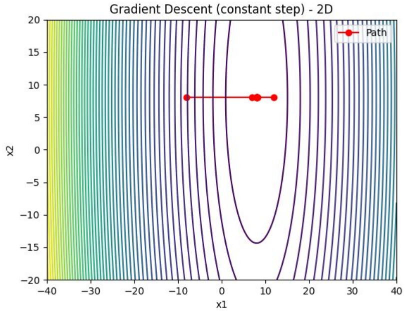
Метод градиентного спуска с постоянным шагом

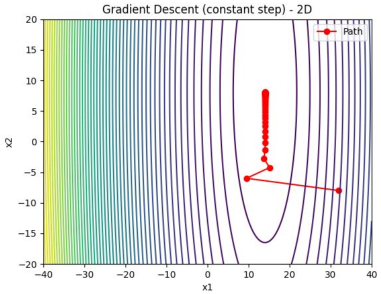












$x_0 = (10, 15)$
eps = 1e5
iter = 87
$x^* = (1, 8.00006307)$

$x_0 = (-8, 8)$
eps = 1e5
iter = 12
$x^* = (7.99999905, 8)$

$x_0 = (32, -8)$
eps = 1e5
iter = 93
$x^* = (14, 7.99993531)$

Метод Полака-Рибьера

Входные параметры: $\mathbf{x}$ – начальная точка поиска, $f(\mathbf{x})$ – процедура вычисления функции, ε – допустимая погрешность.

Выходной параметр $\mathbf{x}$ – конечная точка поиска.

1. Вычислить $\mathbf{g_x} = \nabla f(\mathbf{x})$ и положить $\mathbf{d} = -\mathbf{g_x}$.

2. Вычислить $r = \arg\min\limits_{\lambda} f(\mathbf{x} + \lambda \cdot \mathbf{d})$, $\mathbf{s} = r \cdot \mathbf{d}$.

3. Положить $\mathbf{x} = \mathbf{x} + \mathbf{s}$, $\mathbf{g_y} = \mathbf{g_x}$.

4. Вычислить $\mathbf{g_x} = \nabla f(\mathbf{x})$, $\beta = \mathbf{g_x}^T \cdot (\mathbf{g_x} - \mathbf{g_y}) \big/ (\mathbf{g_y}^T \cdot \mathbf{g_y})$.

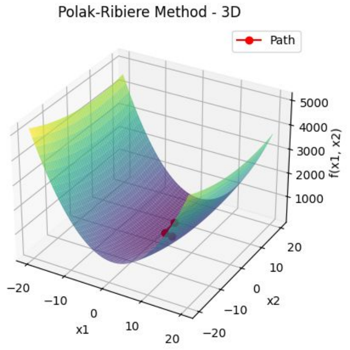
5. Положить $\mathbf{d} = -\mathbf{g_x} + \beta \cdot \mathbf{d}$.

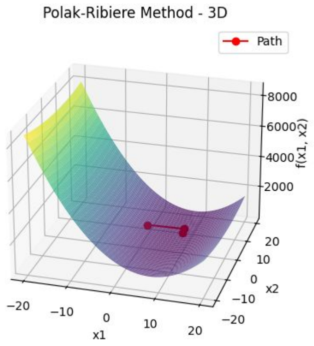
6. Если $\|\mathbf{s}\| > \varepsilon$, то перейти к шагу 2.

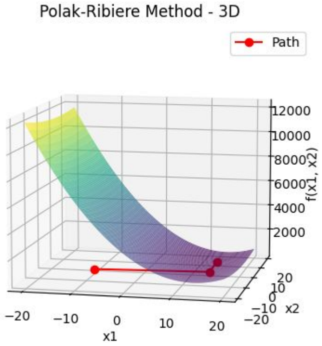
7. Остановиться.

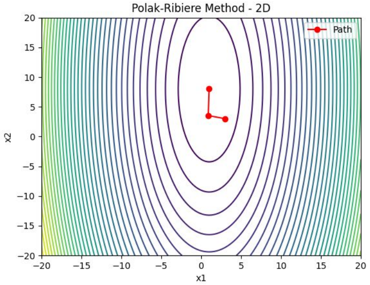
Этот алгоритм отличается от алгоритма метода Флетчера – Ривса только вычислением параметра β на шаге 4.

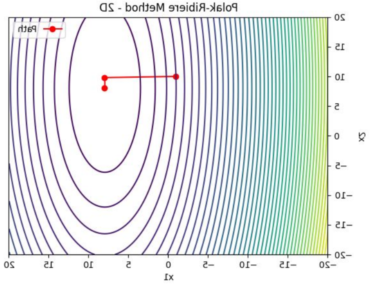
Метод Полака-Рибьера

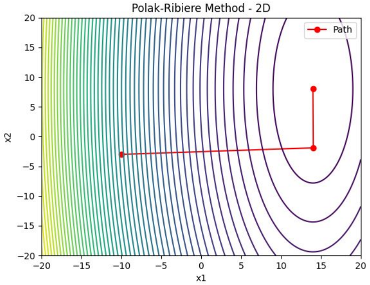












$x_0 = (3, 3)$
eps = 1e5
iter = 3
$x^* = (1.0, 8.0)$

$x_0 = (-1, 10)$
eps = 1e5
iter = 3
$x^* = (8.0, 8.0)$

$x_0 = (-10, -3)$
eps = 1e5
iter = 3
$x^* = (14.0, 8.0)$

Метод Ньютона-Рафсона

Алгоритм

Шаг 1. Задать $x^0, \varepsilon_1 > 0, \varepsilon_2 > 0$, M – предельное число итераций. Найти градиент $\nabla f(x)$ и матрицу Гессе $H(x)$.

Шаг 2. Положить $k = 0$.

Шаг 3. Вычислить $\nabla f\left(x^k\right)$.

Шаг 4. Проверить выполнение условия $\left\|\nabla f\left(x^k\right)\right\| \leq \varepsilon_1$:

а) если неравенство выполнено, то расчет закончен и $x^* = x^k$;

б) если нет, перейти к шагу 5.

Шаг 5. Проверить выполнение условия $k \geq M$:

а) если неравенство выполнено, расчет окончен и $x^* = x^k$;

б) если нет, перейти к шагу 6.

Шаг 6. Вычислить элементы матрицы $H\left(x^k\right)$.

Метод Ньютона-Рафсона

Шаг 7. Найти обратную матрицу $H^{-1}\left(x^k\right)$.

Шаг 8. Проверить выполнение условия $H^{-1}\left(x^k\right) > 0$:

а) если условие выполняется, то найти $d^k = -H^{-1}\left(x^k\right)\nabla f\left(x^k\right)$;

б) если нет, то положить $d^k = -\nabla f\left(x^k\right)$.

Шаг 9. Определить $x^{k+1} = x^k + t_k d^k$.

Шаг 10. Найти шаг t_k^* из условия $\varphi(t_k) = f\left(x^k + t_k d^k\right) \to \min\limits_{t_k}$.

Шаг 11. Вычислить $x^{k+1} = x^k + t_k^* d^k$.

Шаг 12. Проверить выполнение неравенств

$$\left\| x^{k+1} - x^k \right\| < \varepsilon_2, \qquad \left| f\left(x^{k+1}\right) - f\left(x^k\right) \right| < \varepsilon_2:$$

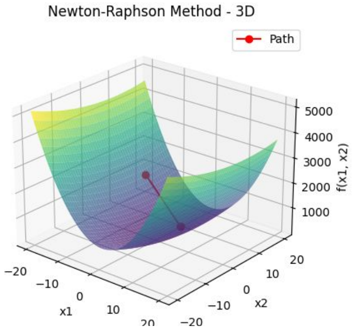
а) если оба условия выполнены при текущем значении k и $k = k-1$, то расчет окончен и $x^* = x^{k+1}$;

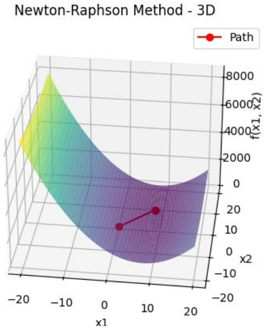
б) в противном случае положить $k = k+1$ и перейти к шагу 3.

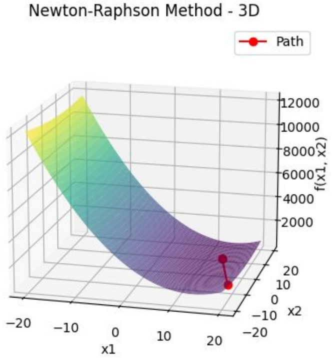
Основная функция метода Ньютона-Рафсона

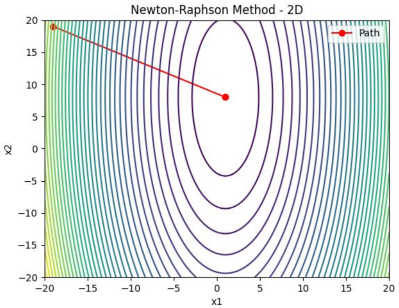
```
def newton_raphson_sympy(f, x0, tol=1e-5, max_iter=1000):
    x1, x2 = sp.symbols('x1 x2')
    grad_f = gradient_sympy(f, (x1, x2))
    hess_f = hessian_sympy(f, (x1, x2))
    x_val = np.array(x0, dtype=float)
    path = [x_val.copy()]
    for i in range(max_iter):
        grad_val = np.array(grad_f(*x_val))
        hess_val = np.array(hess_f(*x_val))
        try:
            hess_inv = np.linalg.inv(hess_val)
        except np.linalg.LinAlgError:
            print(f"Newton-Raphson: Матрица Гессе вырождена в итерации {i + 1}")
            break
        x_new = x_val - hess_inv.dot(grad_val)
        path.append(x_new.copy())
        if np.linalg.norm(x_new - x_val) < tol:
            print(f"Newton-Raphson: Минимум найден в точке {x_new} за {i + 1} итераций")
            break
        x_val = x_new
    return x_val, np.array(path)
```

Метод Ньютона-Рафсона

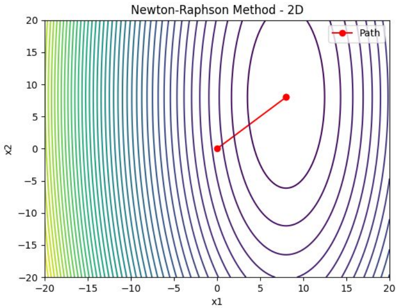




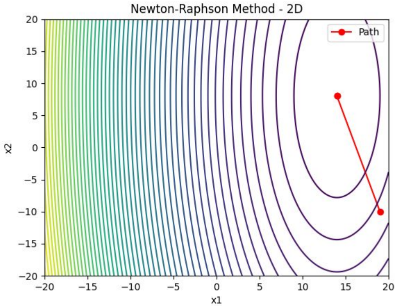




x_0 = (1, 1)
eps = 1e5
iter = 2
x^* = (1.0, 8.0)



x_0 = (0, 0)
eps = 1e5
iter = 2
x^* = (8.0, 8.0)



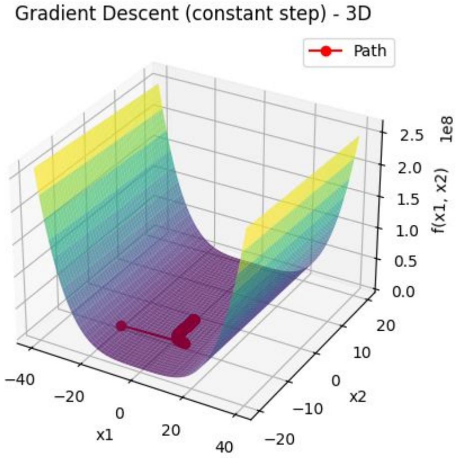
x_0 = (19, -10)
eps = 1e5
iter = 2
x^* = (14.0, 8.0)

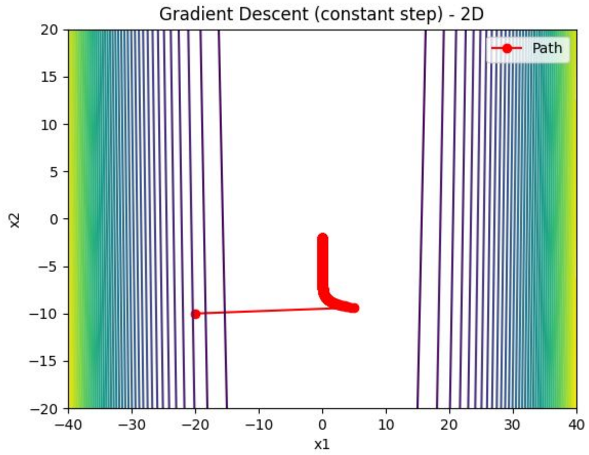
Функция Розенброка

$$f(x) = 100\left(x_2 - x_1^2\right)^2 + \left(1 - x_1\right)^2$$

Метод градиентного спуска:

x_0 = (-20, -10)
eps = 1e2
iter = 105
x* = (0.98, 0.97)

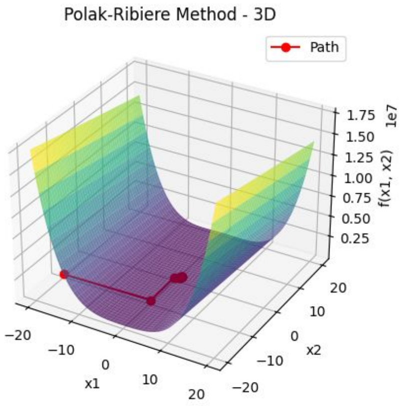


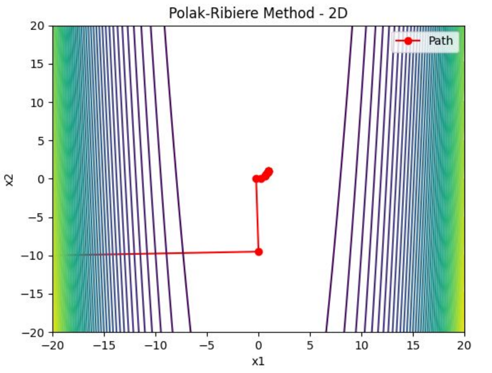


Функция Розенброка

Метод Полака-Рибьера:

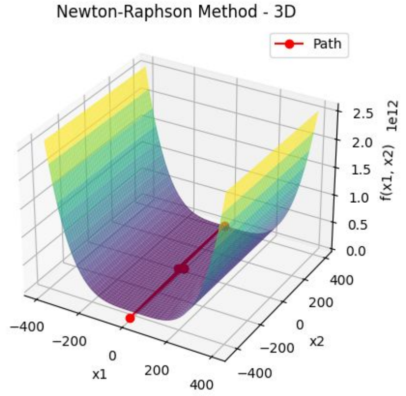
x_0 = (-20, -10)
eps = 1e5
iter = 14
x* = (0.99999988, 0.99999976)

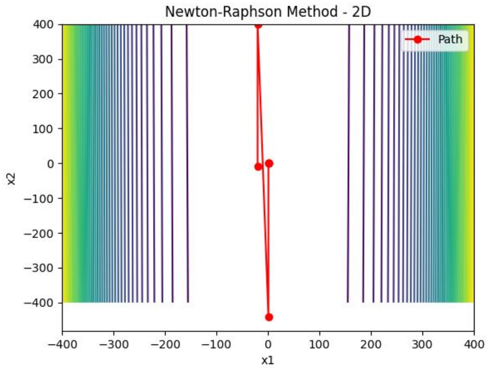




Метод Ньютона-Рафсона:

x_0 = (-20, -10)
eps = 1e5
iter = 5
x* = (1.0, 1.0)





Выводы

- На функции Розенброка метод Ньютона-Рафсона показал наибольшую точность и эффективность, нашел точку минимума за наименьшее количество итераций, так как это метод второго порядка;
- Метод градиентного спуска с постоянным шагом показал сильную зависимость от выбора начального приближения и работает дольше всех;
- Метод Полака-Рибьера показал себя как очень устойчивый метод и имеет зависимость от формы и вида функции;