Санкт-Петербургский государственный университет
Факультет прикладной математики - процессов управления
Кафедра математической теории игр и статистических решений

Численные методы: минимизация квадратичной функции

Сардарян Армен, группа 301
5 семестр, 2019 год

1 Постановка задачи

Пусть X – евклидово n-мерное пространство, обозначаемое далее $\mathbb{R}^n$, $Q = \mathbb{R}^n$.
В пространстве $\mathbb{R}^n$ рассмотрим квадратичную функцию:

$$x = \frac{1}{2}x^T Ax + x^T b, \tag{1}$$

где A – положительно определенная матрица, т.е. $A = A^T$ и имеют место неравенства:

$$m\|x\|^2 \leq (x, Ax) \leq M\|x\|^2, m \geq 0 \tag{2}$$

Для такой функции существует единственная точка минимума $\overline{x}$, удовлетворяющая СЛАУ $Ax + b = 0$.
Задание для самостоятельного выполнения с использованием компьютера:
Для функции

$$f(x, y, z) = 2x^2 + (3 + 0.1N)y^2 + (4 + 0.1N)z^2 + xy - yz + xz + x - 2y + 3z + N \tag{3}$$

найти точку минимума с точностью $\varepsilon = 10^{-6}$. Здесь N – номер студента по списку группы. $N = 18$. Задание выполнить с использованием МНГС и МНПС.

2 Методы наискорейшего спуска

Задача: имея точку $x_k \in \mathbb{R}^n$ построить точку $x_{k+1} \in \mathbb{R}^n$ такую, чтобы выполнялось соотношение:

$$f(x_{k+1}) < f(x_k) \tag{4}$$

Будем искать точку x_{k+1} в следующем виде:

$$x_{k+1} = x_k + \mu_k q, \tag{5}$$

$$\mu_k = -\frac{q^T(Ax_k + b)}{q^T Aq}, \tag{6}$$

где q - заданный вектор из $\mathbb{R}^n$, называемый направлением спуска, а μ_k – искомый параметр, называемый шагом метода в направлении спуска. Продолжая указанные построения, получим последовательность x_k, которую естественно назвать последовательностью убывания для функции f.

2.1 Метод наискорейшего градиентного спуска

Если в формулах (5) считать, что $q = Ax_k + b$, то соответствующий метод построения последовательности x_k называют *градиентным методом.* Если к тому же шаг метода μ_k выбирается по формуле (6), то такой метод называют (одношаговым) *методом наискорейшего градиентного спуска* (МНГС). В этом случае формула (6) принимает вид:

$$\mu_k = -\frac{\|Ax_k + b\|^2}{(Ax_k + b)^T A(Ax_k + b)} \tag{7}$$

2.1.1 Реализация МНГС на языке Python

```
import numpy as np

eps = 1e-6

A = np.array([[4, 1, 1],
              [1, 9.6, -1],
              [1, -1, 11.6]])

b = np.array([[1.], [-2.], [3.]])

x0 = np.array([[0.], [0.], [0.]])

def f(x):
    return (np.dot(np.dot(x.transpose(), A), x))[0][0] / 2 + np.dot(x.transpose(),
    b)[0][0] + 18

def mngs(A, b, eps):
    k = 0
    x = b
    q = np.dot(A, x) + b
    while np.linalg.norm(q) > eps:
        k += 1
        m = -np.dot(q.T, q) / np.dot(np.dot(q.T, A), q)
        x = x + m * q
        q = np.dot(A, x) + b
    return x, k

print('Метод наискорейшего градиентного спуска')
x_ming, k = mngs(A, b, eps)
print('Минимум функции: ', f(x_ming))
print('Достигается в точке: ', x_ming.T)
print('Кол-во итераций: ', k)
```

2.1.2 Результат работы программы

Минимум функции: 17.336074980872247
Достигается в точке: $[-0.24808712,\ 0.21136189,\ -0.21901304]$
Кол-во итераций: 23

2.2 Метод наискорейшего покоординатного спуска

В случае выбора направлений спуска q в формуле (5) на каждом шаге в виде $q = e^i = (\underbrace{0, \cdots, 0, 1}_{i}, 0, \cdots, 0)^T$, где $e^i - i$-ый орт пространства $\mathbb{R}^n$, метод носит название *метода покоординатного спуска*. При выборе шага метода μ_k по формуле (6) его называют *методом наискорейшего покоординатного спуска*(МНПС). В этом случае формула (6) принимает вид:

$$\mu_k = -\frac{e^i(Ax_k + b)}{e^i A e^i} \tag{8}$$

2.2.1 Реализация МНПС на языке Python

```
import numpy as np

eps = 1e-6

A = np.array([[4, 1, 1],
              [1, 9.6, -1],
              [1, -1, 11.6]])

b = np.array([[1.], [-2.], [3.]])

x0 = np.array([[0.], [0.], [0.]])

def f(x):
    return (np.dot(np.dot(x.transpose(), A), x))[0][0] / 2 + np.dot(x.transpose(),
    b)[0][0] + 18

def mnps(A, b, eps):
    k = 0
    x = b.copy()
    n = A.shape[0]
    q = np.dot(A, x) + b
    while np.linalg.norm(q) > eps:
        i = k % n
        k += 1
        m = -q[i][0] / A[i][i]
        x[i][0] = x[i][0] + m
        q = np.dot(A, x) + b
    return x, k

print('Метод наискорейшего покоординатного спуска')
x_minp, k = mnps(A, b, eps)
print('Минимум функции: ', f(x_minp))
print('Достигается в точке: ', x_minp.T)
print('Кол-во итераций: ', k)
```

2.2.2 Результат работы программы

Метод наискорейшего покоординатного спуска
Минимум функции: 17.33607498087227
Достигается в точке: $[-0.24808725,\ 0.21136199,\ -0.21901297]$
Кол-во итераций: 22