Лекция 5

**ПРЯМЫЕ МЕТОДЫ БЕЗУСЛОВНОЙ МИНИМИЗАЦИИ ФНП**

**1. Общие принципы *n*-мерной минимизации**

Для числового решения задач безусловной минимизации , разработано много алгоритмов, использующих *итерационные процедуры* вида:

, (1)

позволяющие при определенных условиях построить последовательность  такую, что  (2)

***Определение 1.*** Последовательность , удовлетворяющая требованию (2), называется ***минимизирующей*** для функции .

Говорят, что минимизирующая последовательность сходится к мно­жеству , если

, (3)

где  – расстояние от точки *x* до множества *U*.

Если множество *U* состоит из единственной точки , то для сходящейся к  минимизирующей последовательности выполняется .

***Определение 2.*** Последовательность  сходится к точке  ***линейно*** (со скоростью геометрической прогрессии), если существует такое число , что выполняется неравенство  или .

***Определение 3.*** Сходимость называют ***сверхлинейной*** (т.е. более быстрой, чем определяемая любой геометрической прогрессией), если

 при .

***Определение 4.*** Скорость сходимости называется ***квадратичная***, если справедлива оценка  или , где .

Для многих алгоритмов скорость сходимости последовательности  характеризуется и другими неравенствами, .

**Выделяют следующие условия остановки** (**критерий окончания**) **итерационного процесса**:

;

; (4)

,

где заданная точность (погрешность).

Вычислительные алгоритмы простейших процедур (1), как правило, основанные на рекуррентных формулах вида:

 (5)

где – направление поиска точки  из точки , а число – величина шага, которая выбирается так, чтобы выполнялось условие .

**Данные алгоритмы различаются способами построения вектора  и выбора шага .**

***Определение 5.*** Будем говорить, что в итерационном процессе (1) производится **исчерпывающий спуск**, если величина шага  находится из решения одномерной задачи минимизации

, . (6)

**Теорема 1** (*Свойство исчерпывающего спуска*). Для дифференцируемой в *Еn* функции  в итерационном процессе (5) с выбором шага  в соответствии с (6) для всех  выполняется условие

. (7)

*Доказательство*. Запишем необходимое условие минимума функции одной переменной  из (20), используя правило дифференцирования сложной функции: . Так как , то  и .

Геометрическая иллюстрацию соотношения (7) в пространстве Е2:

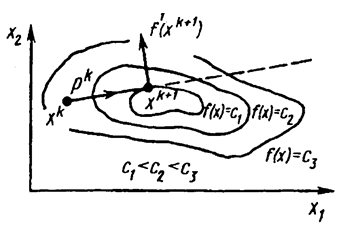


Рисунок 1 – Ортогональность направления  градиенту    
при исчерпывающем спуске

При перемещении из точки  вдоль прямой, задаваемой вектором  в направлении убывания функции, происходит пересечение линий уровня функции  до тех пор, пока либо не будет достигнута стационарная точка , либо прямая не коснется в точке  некоторой линии уровня функции . Равенство (7) означает *ортогональность направления*  градиенту  при исчерпывающем спуске.

Свойство (7) позволяет в явном виде найти величину  для квадратичной функции.

**Теорема 2** (*Свойство исчерпывающего спуска*). Для квадратичной функции  величина  исчерпывающего спуска в итерационном процессе (5) будет равна

. (8)

*Доказательство*. Итерационный процесс (5) имеет вид . Умножив это равенство слева на матрицу *А* квадратичной функции  и прибавив к обеим частям вектор *b*, получим: . Учитывая, что градиент квадратичной функции равен , имеем:

. (\*).

Подставляя (\*) в равенство (7), получаем формулу:

;

;

.

***Определение 6.*** Направление вектора  называется ***направлением убывания*** функции  в точке , если при всех достаточно малых положительных  выполняется неравенство .

В итерационном процессе (5) используются, как правило, направления убывания. Сформулируем признак направления убывания.

**Теорема 3.** Пусть функция  дифференцируема в точке . Если вектор  удовлетворяет условию (скалярное произведение меньше нуля):

, (9)

то направление вектора  является *направлением убывания*.

*Доказательство*. Если  задает направление убывания функции , то по определению . Покажем это. Т.к.  дифференцируема, то . Тогда . При достаточно малых   задает направление убывания  в точке .

Геометрически условие (9) означает, что вектор  составляет тупой угол с градиентом .

**2. Метод циклического покоординатного спуска**

В методе используется итерационная формула вида:

, (10)

где  выбирается из условия исчерпывающего спуска:

. (11)

Этот метод заключается в последовательной минимизации целевой функции  сначала по направлению первого базисного вектора , затем второго –  и т.д., т.е. параллельно координатным осям. После окончания минимизации по направлению последнего базисного вектора  цикл повторяется.

***Алгоритм метода циклического покоординатного спуска:***

*Шаг 0.* Выбрать начальную точку , величину , критерий достижения точности ( или ). Найти , положить .

*Шаг 1.* Решить задачу одномерной минимизации (11), т.е. найти . Положить , вычислить .

*Шаг 2.* Если , то положить  и перейти к шагу 1, иначе – перейти к шагу 3.

*Шаг 3.* Проверить условие достижения точности или . Если оно выполняется, то положить  и закончить поиск. Иначе – положить ,  и к шагу 1.

***Пример 1:*** Решить задачу  методом циклического покоординатного спуска.

***Решение***: Линии уровня этой целевой функции – окружности с центром в начале координат (см. рисунок 2). Выберем произвольную начальную точку . Очевидно, два шага исчерпывающего спуска сначала по направлению , затем по  приведут в точку минимума .

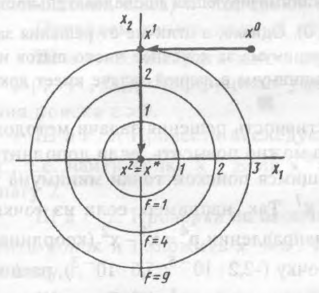


Рисунок 2 – Линии уровня функции 

***Пример 2:*** Решить задачу  методом циклического покоординатного спуска.

***Решение***: Сначала заметим, что поворот системы координат на угол  (замена переменных  и  приводит функцию к виду ). Очевидно, линии уровня этой целевой функции – эллипсы с центром в начале координат  (рисунок 3).

Применим алгоритм метода циклического покоординатного спуска.

Шаг 0. Выбираем начальную точку , критерий достижения точности , задаем . Найдем , положим .

*Итерация 1.* Шаг 1. Решим задачу одномерной минимизации (11), т.е. найдем .

Т.к. целевая функция является квадратичной, то можно определить шаг по формуле (8):

.

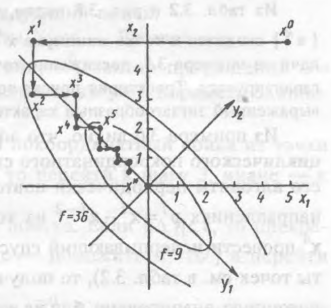


Рисунок 3 – Линии уровня функции 

Найдем градиент функции . Получаем, что . Поскольку направление убывания функции , то выражение в числителе равно .

Запишем матрицу , тогда выражение в знаменателе составит . Найдем шаг . Положим , вычислим .

Шаг 2. Так как , то полагаем  и переходим к шагу 1 новой итерации.

*Итерация 2.* Шаг 1. Найдем новый шаг . Положим , вычислим .

Шаг 2. Так как , переходим к шагу 3.

Шаг 3. Проверим условие достижения точности . Оно не выполняется. Будем проводить итерации, пока не выполниться условие окончания поиска точки минимума (см. таблицу 1).

Из таблицы 1 и рисунку 3 видно, что минимизирующая последовательность  сходится к точке минимума . Однако, в отличие от решения задачи из примера 1, достижение точки минимума за конечное число шагов не гарантируется. Траектория поиска минимума в данной задаче имеет ярко выраженный зигзагообразный характер.

Таблица 1 – Итерации метода циклического покоординатного спуска

| Номер итерации | Последовательность приближений |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0 |  | 5 | 5 | 450 | – |
| 1 |  | -4 | 5 | 45 | 9,00 |
| 2 |  | -4 | 3,2 | 28,8 | 1,80 |
| 3 |  | -2,56 | 3,2 | 18,43 | 1,44 |
| 4 |  | -2,56 | 2,05 | 11,80 | 1,15 |
| 5 |  | -1,64 | 2,05 | 7,55 | 0,92 |
| 6 |  | -1,64 | 1,31 | 4,83 | 0,74 |
| 7 |  | -1,05 | 1,31 | 3,09 | 0,59 |
| 8 |  | -1,05 | 0,84 | 1,98 | 0,47 |
| 9 |  | -0,67 | 0,84 | 1,27 | 0,38 |
| 10 |  | -0,67 | 0,54 | 0,81 | 0,30 |

**Замечание.** Эффективность метода циклического покоординатного спуска существенно зависит от свойств целевой функции. Эффективность решения задачи можно повысить, если вместо покоординатных направлений , использовать направления убывания функции  и минимизировать ее по этим направлениям. Так появляется другой алгоритм.

**3. Алгоритм Хука–Дживса**

Этот алгоритм содержит две основные процедуры:

а) исследующий покоординатный поиск в окрестности данной точки, предназначенный для определения направления убывания ;

б) перемещение в направлении убывания.

*Алгоритм исследующего покоординатного поиска* из заданной точки *х* с приращениями по каждой координате .

1. Положить .

2. Сделать пробный шаг , где й базисный вектор. Если , то перейти к пункту 3, иначе – к пункту 4.

3. Сделать пробный шаг . Если , то перейти к пункту 5, иначе – к пункту 4.

4. Положить .

5. Положить . Если , то перейти к пункту 2. В противном случае исследующий поиск окончен – получена точка , для которой , если .

**Замечание.** В результате исследующего поиска может оказаться, что . Тогда исследующий поиск считается неудачным. Если при этом норма приращения  мала, т.е. , где заданная точность, то полагают . Если заданная точность не достигнута, то полагают  (постоянная >1– коэффициент уменьшения шага) и повторяют исследующий поиск.

**Полный алгоритм Хука–Дживса:**

*Шаг 0.* Выбрать начальную точку , вектор приращений  коэффициент уменьшения шага , параметр окончания поиска .

*Шаг 1.* Провести исследующий покоординатный поиск из точки , т.е. найти точку . Если , то перейти к шагу 3, иначе – к шагу 2.

*Шаг 2.* Проверка на окончание поиска. Если , то прекратить поиск и положить  (точка минимума найдена!). Иначе – положить  и перейти к шагу 1.

*Шаг 3.* Перемещение из точки  в направлении убывания : положить .

*Шаг 4.* Провести исследующий поиск в точке , т.е. найти точку . Если , то положить ,  и перейти к шагу 3. Иначе – положить  и перейти к шагу 1.

***Пример 3:*** Рассмотрим графическую иллюстрацию решения задачи  методом Хука-Дживса из начальной точки , вектор перемещений .

Целевая функция  представляет собой квадрат расстояния между точками  и . Линии уровня  это окружности с центром в точке . Поэтому значения функции в промежуточных точках алгоритма легко сравнивать, оценивая их расстояния от точки  на рисунке 4.

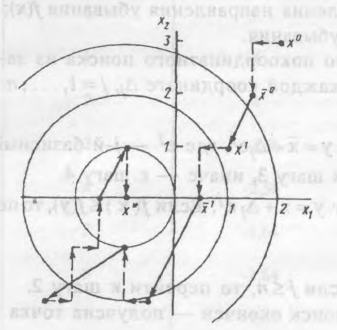


Рисунок 4 – Иллюстация к примеру 3

На рисунке 4 пунктиром выделены траектории исследующего поиска, сплошными линиями – перемещения в направлении убывания. Убедитесь в соответствии графической иллюстрации алгоритму метода Хука-Дживса.

Первая итерация метода: .

Вторая итерация: .

Третья итерация: .

Четвертая итерация: 

.

Таким образом, точка минимума найдена .

**4. Метод случайного поиска**

Во всех описанных ранее методах процедуры поиска точки минимума целевой функции детерминированы. Однако разработаны и такие методы минимизации, в которых в процедуру поиска точки минимума намеренно вводят элемент случайности.

Основой для этих методов служит итерационный процесс вида:

, (12)

где – величина шага,  – некоторая реализация *n*-мерного случайного вектора .

Будем считать, что координаты вектора  – это независимые случайные величины, равномерно распределенные на отрезке [-1; 1].

***Алгоритм метода случайного поиска:***

*Шаг 0.* Выбрать параметр точности , начальный шаг , коэффициент уменьшения шага , предельное число неудачных попыток *N*, начальную точку . Вычислить значение функции в начальной точке.

*Шаг 1.* Положить счетчик числа неудачных попыток .

*Шаг 2.* Получить реализацию случайного вектора .

*Шаг 3.* Найти пробную точку .

*Шаг 4.* Сравнить значения функции в точках  и . Если , то продолжают двигаться в направлении  –  и перейти к шагу 3, иначе – к шагу 5.

*Шаг 5.* Положить . Если , то получить новую реализацию случайного вектора  и повторить попытку движения в этом направлении – шаг 2, иначе – к шагу 6.

*Шаг 6.* Если число попыток превысило заданное предельное *N*, то проверяют условие достижения точности. Если , то поиск завершен и . Иначе уменьшают шаг  и начинают поиск заново.

Иллюстация построения последовательности (12) с помощью описанного алгоритма для функции двух переменных приведена на рисунке 5, где пунктиром показаны неудачные попытки определения новой точки из (12), не приводящие к уменьшению .

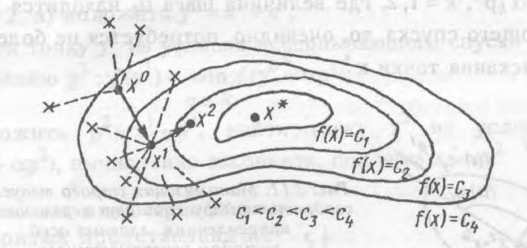


Рисунок 5 – Иллюстрация алгоритма случайного поиска в пространcтве *E2*

***Замечани***е. Обычно на практике при использовании метода случайного поиска число *N* полагают равным 3*n*, где *n* – размерность Евклидова пространства (число незаисимых переменных в целевой функции).

**Упражнения для самостоятельной работы:**

**Задание 1**. Проверить, что последовательность : является минимизирующей для функции , если:

*а*) ; *б*) .

**Решение:** Согласно (2) имеем, что.

*а*) Найдем множество  для заданной функции, используя классический метод минимизации ФНП. Вычислим первые частные производные и приравняем их к нулю: . Проверим точку (0;0) является ли она точкой минимума. Запишем гессиан , проверим его на положительность матрица Гессе положительная и точка (0;0) является точкой минимума.

Если множество *U* состоит из единственной точки , то для сходящейся к  минимизирующей последовательности выполняется . То есть при  имеем по первой координате ;  . Аналогично по второй координате: .

Далее перейдем к пределу . Получаем, что .

Значит  является минимизирующей для заданной функции .

*б*) Аналогично, : .

Имеем, что ; ; . Для  также .

Далее .

Значит  является минимизирующей для функции .

**Задание 2**. Проверить является ли минимизирующей последовательность ,  для функции .