## Глава 9. Многомерная локальная условная оптимизация

### 9.1. Методы последовательной безусловной оптимизации

Рассмотрим следующую многомерную [задачу локальной условной оптимизации](javascript:termInfo(%22задачу%20локальной%20условной%20оптимизации%22)): найти минимум [критерия оптимальности](javascript:termInfo(%22критерия%20оптимальности%22)) Φ(X), определенного во множестве D евклидова пространства ,

|  |  |
| --- | --- |
|  | (1) |

где [множество допустимых значений](javascript:termInfo(%22множество%20допустимых%20значений%22))

|  |  |
| --- | --- |
| (2) |  |

Основная идея [методов последовательной безусловной оптимизации](javascript:termInfo(%22методы%20последовательной%20безусловной%20оптимизации%22)) состоит в преобразовании [задачи условной оптимизации](javascript:termInfo(%22задачи%20условной%20оптимизации%22)) (1), (2) к последовательности [задач безусловной оптимизации](javascript:termInfo(%22задач%20безусловной%20оптимизации%22))

|  |  |
| --- | --- |
| ­ | (3) |

Где функции, которые возрастают вблизи границ [области допустимых значений](javascript:termInfo(%22области%20допустимых%20значений%22)) D и тем быстрее, чем больше значение параметра . В качестве приближенного решения задачи (1), (2) принимается решение вспомогательной задачи (3) при достаточно большом .

Поясним идею [методов последовательной безусловной оптимизации](javascript:termInfo(%22методов%20последовательной%20безусловной%20оптимизации%22)) примером.

Пример 1

Пусть

|  |  |
| --- | --- |
|  | (4) |

и имеется одно ограничение типа равенств с [ограничивающей функцией](javascript:termInfo(%22ограничивающей%20функцией%22))

|  |  |
| --- | --- |
|  | (5) |

Положим

|  |  |
| --- | --- |
|  | (6) |

где — вещественная константа. На рисунках рис. 1, рис. 2, рис. 3 приведены линии уровня функции при соответственно.

Линии уровня на рисунках рис. 1, рис. 2, рис. 3 получены с помощью следующей MATLAB-программы:

;

;

;

;

;

;

;

;

Рисунки показывают, что при увеличении параметра минимум функции приближается к решению задачи (3), (4), (5), (6)

Среди [методов последовательной безусловной оптимизации](javascript:termInfo(%22методов%20последовательной%20безусловной%20оптимизации%22)) выделяют [метод штрафных функций](javascript:termInfo(%22метод%20штрафных%20функций%22)) и [метод барьерных функций](javascript:termInfo(%22метод%20барьерных%20функций%22)).

В [методе штрафных функций](javascript:termInfo(%22Метод%20штрафных%20функций%22)) функцию , которая в этом случае называется [штрафной функцией](javascript:termInfo(%22штрафной%20функцией%22)), подбирают таким образом, чтобы при больших функция мало отличалась от функции при и быстро возрастала при удалении точки от границы [области допустимых значений](javascript:termInfo(%22области%20допустимых%20значений%22)) D. В методе штрафных функций точка X в процессе поиска может выходить за границы области D (см. рис. 4). Т.е. метод штрафных функций относится к классу [методов внешней точки](javascript:termInfo(%22методов%20внешней%20точки%22)). Рассмотренный выше прим. 1 также иллюстрирует метод штрафных функций.

|  |
| --- |
| ?n=1 |

Рис. 1. К прим. 1. Точка минимума функции при α = 0 имеет координаты (3, 2). Решением задачи (3), (4), (5), (6) является точка с координатами (2.5, 1.5).

|  |
| --- |
| ?n=2 |

Рис. 2. К прим. 1. Точка минимума функции при α = 1 имеет координаты (2.666…, 1.666…). Решением задачи (3), (4), (5), (6) является точка Х∗ с координатами (2.5, 1.5).

|  |
| --- |
| ?n=3 |

Рис. 3. К прим. 1. Точка минимума функции Qα(Х) при α = 2 имеет координаты (2.6, 1.6). Решением задачи (3), (4), (5), (6) является точка Х∗ с координатами (2.5, 1.5).

|  |
| --- |
| ?n=4 |

Рис. 4. К методу штрафных функций (n = 1) Интервал [a, b] — область допустимых значений D; γ > β > α.

В [методе барьерных функций](javascript:termInfo(%22Метод%20барьерных%20функций%22)) функцию , которая в этом случае называется [барьерной функцией](javascript:termInfo(%22барьерной%20функцией%22)), подбирают таким образом, чтобы при больших функция мало отличалась от функции при и быстро возрастала при приближении точки к границе [области допустимых значений](javascript:termInfo(%22области%20допустимых%20значений%22)) D. В методе барьерных функций точка X в процессе поиска не может выходить за границы области D(см. рис. 5). Это означает, что метод барьерных функций относится к классу [методов внутренней точки](javascript:termInfo(%22методов%20внутренней%20точки%22)).

|  |
| --- |
| ?n=5 |

Рис. 5. К методу барьерных функций (n = 1) Интервал [a, b] — область допустимых значений D; γ > β > α.

В вычислительной практике преимущественно используется [метод штрафных функций](javascript:termInfo(%22метод%20штрафных%20функций%22)). Поэтому в дальнейшем ограничимся именно им.

[Штрафная функция](javascript:termInfo(%22Штрафная%20функция%22)) в общем случае имеет вид

|  |  |
| --- | --- |
|  | (7) |

где — двумерный вектор параметров штрафной функции; — весовые коэффициенты, могущие изменяться в процессе итераций, , —функционалы над функциями , , соответственно.

Функционалы , в формуле (7) должны удовлетворять очевидным требованиям:

при ,

при ;

?k=10

?k=10

В качестве функционалов , можно взять расстояния в какой-либо метрике от точки до соответствующей границы множества . Однако, вычисление этих расстояний, а значит и значений [штрафной функции](javascript:termInfo(%22штрафной%20функции%22)), может быть затруднительным. Поэтому обычно применяют штрафные функции более удобного вида.

Так в качестве функционалов обычно используют функционалы

,

в качестве функционалов — функционалы

,

где .

В качестве критерия окончания итераций в [методе последовательной безусловной оптимизации](javascript:termInfo(%22методе%20последовательной%20безусловной%20оптимизации%22)) можно использовать неравенство

|  |  |
| --- | --- |
| , | (8) |

где ?k=10— четное число итераций, — требуемая точность решения по .

Недостатком [метода последовательной безусловной оптимизации](javascript:termInfo(%22метода%20последовательной%20безусловной%20оптимизации%22)) является значительное усложнение структуры минимизируемой функции (см. рис. 1) — плата за исключение ограничений.

Схема метода штрафных функций.

Задаем начальную точку и полагаем счетчик числа итераций .

Исходя из точки , одним из методов локальной безусловной оптимизации решаем задачу — находим точку .

Проверяем условие окончания поиска ?k=10. Если условие окончания поиска выполнено, то полагаем и завершаем итерации. Иначе — по некоторому правилу увеличиваем значения параметров , , полагаем

и переходим к п.3?k=10

Примечание 1

В зависимости от метода локальной безусловной оптимизации, который используется для решения задач (3), [метод последовательной безусловной оптимизации](javascript:termInfo(%22метод%20последовательной%20безусловной%20оптимизации%22)) может быть детерминированным и случайным, нулевого, первого или второго порядка.

### 9.2. Метод скользящего допуска

Рассмотрим следующую многомерную [задачу условной локальной оптимизации](javascript:termInfo(%22задачу%20условной%20локальной%20оптимизации%22)): найти минимум [критерия оптимальности](javascript:termInfo(%22критерия%20оптимальности%22)) , определенного во множестве ?k=10евклидова пространства ?k=10,

|  |  |
| --- | --- |
| , | (1) |

где [множество допустимых значений](javascript:termInfo(%22множество%20допустимых%20значений%22))

|  |  |
| --- | --- |
| . | (2) |

Основы метода скользящего допуска.

[Метод скользящего допуска](javascript:termInfo(%22Метод%20скользящего%20допуска%22)) существенно использует множество

|  |  |
| --- | --- |
| . | (3) |

где неотрицательный скаляр — [критерий скользящего допуска](javascript:termInfo(%22Критерий%20скользящего%20допуска%22)), — неотрицательно определенный функционал над множеством всех [ограничивающих функций](javascript:termInfo(%22ограничивающих%20функций%22)) , .

При этом функционал должен быть сконструирован таким образом, чтобы при и значение возрастало по мере удаления точки от границы [области допустимых значений](javascript:termInfo(%22области%20допустимых%20значений%22)) . [Критерий скользящего допуска](javascript:termInfo(%22Критерий%20скользящего%20допуска%22)) определяет требуемую точность выполнения ограничений, которые формируют область допустимых значений, и конструируется таким образом, чтобы обеспечить его уменьшение с ростом количества итераций ?k=10.

Точка называется [допустимой точкой](javascript:termInfo(%22Допустимая%20точка%22)), если , [почти допустимой точкой](javascript:termInfo(%22Почти%20допустимая%20точка%22)) — если , [недопустимой точкой](javascript:termInfo(%22Недопустимая%20точка%22)) — если . Поскольку величина с ростом номера итерации уменьшается, отклонение от границы области , при котором точка считается допустимой, сужается, так что в пределе рассматриваются только допустимые точки.

[Метод скользящего допуска](javascript:termInfo(%22Метод%20скользящего%20допуска%22)) может быть скомбинирован со многими из рассмотренных ранее многомерных методов локальной безусловной оптимизации. Будем называть метод, с которым комбинируется метод скользящего допуска, [базовым методом](javascript:termInfo(%22Базовый%20метод%22)).

Одна итерация [метода скользящего допуска](javascript:termInfo(%22метода%20скользящего%20допуска%22)) состоит из одного или двух этапов:

1) С помощью [базового метода](javascript:termInfo(%22базового%20метода%22)), исходя из сточки , выполняем итерацию по решению [задачи локальной безусловной оптимизации](javascript:termInfo(%22задачи%20локальной%20безусловной%20оптимизации%22))

|  |  |
| --- | --- |
|  | (4) |

- находим точку . Если (точка является [допустимой точкой](javascript:termInfo(%22допустимой%20точкой%22)) или [почти допустимой точкой](javascript:termInfo(%22почти%20допустимой%20точкой%22))), то полагаем и заканчиваем данную итерацию.

2) Если (точка является недопустимой), то отыскиваем точку , лежащую ближе к границе области . Для этого с помощью того же [базового метода](javascript:termInfo(%22базового%20метода%22)), исходя из точки , решаем [задачу локальной безусловной оптимизации](javascript:termInfo(%22задачу%20локальной%20безусловной%20оптимизации%22))

|  |  |
| --- | --- |
|  | (5) |

с условием окончания итераций

|  |  |
| --- | --- |
|  | (6) |

и заканчиваем данную итерацию.

Достоинством [метода скользящего допуска](javascript:termInfo(%22метода%20скользящего%20допуска%22)) является то, что степень нарушения ограничений по мере приближения к минимуму минимизируемой функции постепенно уменьшается. Т.е. на первых итерациях ограничения могут удовлетворяться приближенно, а высокая точность удовлетворения ограничений необходима лишь в окрестности решения. Это обстоятельство позволяет сократить полный объем вычислений по сравнению с другими методами.

Одна из сложностей применения [метода скользящего допуска](javascript:termInfo(%22метода%20скользящего%20допуска%22)) — возможные осцилляция решения относительно границы области (см. ниже).

Комбинация метода скользящего допуска с методом Нелдера-Мида.

При комбинации [метода скользящего допуска](javascript:termInfo(%22метода%20скользящего%20допуска%22)) с [методом Нелдера-Мида](javascript:termInfo(%22методом%20Нелдера-Мида%22)) можно предложить разные виды [критерия скользящего допуска](javascript:termInfo(%22критерия%20скользящего%20допуска%22)). Чаще всего в качестве этого критерия используют следующую функцию координат вершин деформируемого многогранника :

|  |  |
| --- | --- |
|  | (7) |

Здесь — вектор координат центра тяжести многогранника , так что величина

есть среднее расстояние вершин многогранника от его центра тяжести.

Из (7) следует, что [критерий скользящего допуска](javascript:termInfo(%22критерий%20скользящего%20допуска%22)) ?k=10является положительно определенной функцией координат вершин многогранника . С другой стороны, поскольку размер многогранника при приближении к точке минимума уменьшается (в пределе до нуля), то справедливо предельное соотношение

Для задачи (1), (2) в качестве функционала обычно используют функционал

|  |  |
| --- | --- |
|  | (8) |

где

Из (8) следует, что функционал обладает следующим свойством

Из (8) вытекает также, что если значение функционала мало, то точка находится недалеко от границы области .

Примечание 1

Поскольку [метод Нелдера-Мида](javascript:termInfo(%22метод%20Нелдера-Мида%22)) является детерминированным методом нулевого порядка, комбинация [метода скользящего допуска](javascript:termInfo(%22метода%20скользящего%20допуска%22)) с методом Нелдера-Мида также представляет собой детерминированный метод нулевого порядка.

Упрощенная схема комбинации метода скользящего допуска и метода Нелдера-Мида.

Симплекс с вершинами обозначим.

Задаем начальный симплекс и полагаем счетчик числа итераций .

С помощью [метода Нелдера-Мида](javascript:termInfo(%22метода%20Нелдера-Мида%22)), исходя из симплекса , выполняем одну итерацию по решению [задачи локальной безусловной оптимизации](javascript:termInfo(%22задачи%20локальной%20безусловной%20оптимизации%22)) (4) — находим симплекс с вершинами .

Вычисляем значения функционала во всех вершинах симплекса и значение [критерия скользящего допуска](javascript:termInfo(%22критерия%20скользящего%20допуска%22)) . Находим вершину симплекса , в которой значение функционала максимально, т.е. вершину, которая расположена дальше всех от границы области . Обозначим эту вершину .

Если (точка является [допустимой точкой](javascript:termInfo(%22допустимой%20точкой%22)) или [почти допустимой точкой](javascript:termInfo(%22почти%20допустимой%20точкой%22))), то проверяем условие окончания поиска (см. схему [метода Нелдера-Мида](javascript:termInfo(%22метода%20Нелдера-Мида%22))). Если это условие выполнено, то завершаем итерации. Если условие окончания поиска не выполнено, то формируем симплекс с вершинами , полагаем и переходим к п.2.

Если (точка является [недопустимой точкой](javascript:termInfo(%22недопустимой%20точкой%22))), то с помощью [метода Нелдера-Мида](javascript:termInfo(%22метода%20Нелдера-Мида%22)), исходя из точки , решаем [задачу локальной безусловной оптимизации](javascript:termInfo(%22задачу%20локальной%20безусловной%20оптимизации%22)) (5) с критерием окончания итераций (6) — находим точку . Формируем новый симплекс с вершинами полагаем и переходим к п.2

Ослабление осцилляций решения

Как отмечалось выше, одной из сложностей применения [метода скользящего допуска](javascript:termInfo(%22метода%20скользящего%20допуска%22)) являются возможные осцилляция решения относительно границы области D. Поясним суть этого явления на примере.

Пример 1

Рассмотрим двумерную [задачу условной оптимизации](javascript:termInfo(%22задачу%20условной%20оптимизации%22)) (1), когда [критерий оптимальности](javascript:termInfo(%22критерий%20оптимальности%22)) равен

и [множество допустимых значений](javascript:termInfo(%22множество%20допустимых%20значений%22)) ?k=10определяется ограничениями

|  |  |
| --- | --- |
|  | (9) |

Положим, что на ?k=10-ой итерации координаты вершин текущего симплекса равны ,. Тогда после одной итерации по решению [задачи локальной безусловной оптимизации](javascript:termInfo(%22задачи%20локальной%20безусловной%20оптимизации%22)) (4) [методом Нелдера-Мида](javascript:termInfo(%22методом%20Нелдера-Мида%22)) получим симплекс с вершинами (см. рис. 1).

|  |
| --- |
| ?n=1 |

Рис. 1. К прим. 1. После успешного отражения вершины выполнено успешное растяжение симплекса.

Линии уровня функции на рис. 1 получены с помощью следующей MATLAB-программы:

x = 0 : 0.15 : 5;

y = x;

[X, Y] = meshgrid(x);

Z = -X - Y;

V = [-1, -2, -3, -4, -5, -6, -7, -8, -9, -10];

[C, h] = contour(X, Y, Z, V);

clabel(C, h);

Положим далее, что точка , расположенная далее всех от границы области D, является [недопустимой точкой](javascript:termInfo(%22недопустимой%20точкой%22)), т.е. . Тогда при решении помощью [метода Нелдера-Мида](javascript:termInfo(%22метода%20Нелдера-Мида%22)) [задачи локальной безусловной оптимизации](javascript:termInfo(%22задачи%20локальной%20безусловной%20оптимизации%22)) (5) возможна ситуация, приведенная на рис. 2.

|  |
| --- |
| ?n=2 |

Рис. 2. К прим. 1. После успешного отражения вершины X03 выполнено растяжение симплекса и отражение вершины X02.

Из (8), (9) следует, что если и точка X лежит в первой четверти системы координат , то . На рис. 2 показаны линии уровня функции именно для этого случая. Линии уровня получены с помощью следующей MATLAB-программы:

x = 0 : 0.01 : 5;

y = x;

[X, Y] = meshgrid(x);

Z = + - 9.;

Рассмотренный пример иллюстрирует тот факт, что поскольку вершина симплекса расположена далеко от границы области D, то после операций отражения и растяжения точка может оказаться глубоко в недопустимой области. В результате в процессе минимизации функционала может получиться точка , которая снова оказывается далеко от границы области . И т.д.

Эффект, рассмотренный в прим. 1, и называется осцилляцией решения относительно границы области .

Для ослабления влияния осцилляций в простейшем случае можно вместо точки использовать точку — середину отрезка .

Чаще с этой целью используют квадратичную интерполяцию функции на отрезке по трем точкам , где — также середина отрезка — см. рис. 3. Обозначим эту интерполирующую функцию (см. параграф 4.7). Вместо точки в этом случае можно использовать один из нулей функции либо его приближенное значение, найденное, например, методом касательных.

|  |
| --- |
| ?n=3 |

Рис. 3. Использование квадратичной интерполяции функции на отрезке по трем точкам для ослабления осцилляций. Случай, когда точка принадлежит области допустимых значений .

### 9.3. Модифицированный метод комплексов

Рассмотрим многомерную [задачу локальной условной оптимизации](javascript:termInfo(%22задачу%20локальной%20условной%20оптимизации%22))

|  |  |
| --- | --- |
|  | (1) |

где [множество допустимых значений](javascript:termInfo(%22множество%20допустимых%20значений%22)) определяется только ограничениями типа неравенств и представляет собой гиперпараллелепипед, т.е.

|  |  |
| --- | --- |
|  | (2) |

Здесь — нижняя и верхняя границы [области допустимых значений](javascript:termInfo(%22области%20допустимых%20значений%22)) по -му измерению (см. рис. 1).

|  |
| --- |
| ?n=1 |

Рис. 1. Область допустимых значений в виде гиперпараллелепипеда;

[Метод комплексов](javascript:termInfo(%22Метод%20комплексов%22)) в многомерной [задаче безусловной оптимизации](javascript:termInfo(%22задаче%20безусловной%20оптимизации%22)) рассмотрен в параграфе 8.2. В данном параграфе рассматривается модификация этого метода для решения многомерной [задачи условной оптимизации](javascript:termInfo(%22задачи%20условной%20оптимизации%22)) - [модифицированный метод комплексов](javascript:termInfo(%22модифицированный%20метод%20комплексов%22)).

Основные операции метода комплексов.

Напомним, что [комплексом](javascript:termInfo(%22комплексом%22)) называется многогранник с вершинами (не обязательно выпуклый). Рекомендуется использовать комплекс с вершинами. Так же, как при решении [задачи безусловной оптимизации](javascript:termInfo(%22задачи%20безусловной%20оптимизации%22)), при решении задачи (1) [методом комплексов](javascript:termInfo(%22методом%20комплексов%22)) используются следующие операции:

генерация случайного [комплекса](javascript:termInfo(%22комплекса%22));

[отражение вершины комплекса с растяжением](javascript:termInfo(%22отражение%20вершины%20комплекса%20с%20растяжением%22));

[сжатие комплекса](javascript:termInfo(%22сжатие%20комплекса%22)).

Генерация случайного [комплекса](javascript:termInfo(%22комплекса%22)). Координаты вершин случайного комплекса с вершинами могут быть найдены по формуле

|  |  |
| --- | --- |
|  | (3) |

где — произвольная начальная точка, – номер вершины [комплекса](javascript:termInfo(%22комплекса%22)), — скаляр, определяющий размер комплекса, — реализация -мерного случайного вектора, — некоторая векторная норма. Обычно в качестве координат вектора используют независимые случайные величины, равномерно распределенные в интервале .

[Отражение вершины комплекса с растяжением](javascript:termInfo(%22Отражение%20вершины%20комплекса%20с%20растяжением%22)). Положим, что задан [комплекс](javascript:termInfo(%22комплекс%22)) с вершинами , и его вершину необходимо отразить через центр тяжести комплекса с растяжением. В новом комплексе все вершины, кроме -ой, совпадают с соответствующими вершинами исходного комплекса , а -я вершина находится на прямой, проходящей через центр тяжести этого комплекса и его вершину (см. рис. 2). Обозначим координаты вершин нового комплекса . Тогда имеем

|  |  |
| --- | --- |
|  | (4) |

где ?k=10— коэффициент растяжения (рекомендуемое значение — ?k=10), — вектор координат центра тяжести [комплекса](javascript:termInfo(%22комплекса%22)) ?k=10:

|  |  |
| --- | --- |
|  | (5) |
| ?n=2 | |

Рис. 2. Отражение вершины комплекса через центр его тяжести с растяжением. Пунктиром показан новый комплекс .

[Сжатие комплекса](javascript:termInfo(%22Сжатие%20комплекса%22)). Положим, что задан [комплекс](javascript:termInfo(%22комплекс%22)) с вершинами

, и его вершину необходимо переместить ближе к центру тяжести комплекса — выполнить сжатие комплекса. В новом комплексе все вершины, кроме -ой, совпадают с соответствующими вершинами исходного комплекса , а -я вершина находится на прямой, проходящей через центр тяжести этого комплекса и его вершину (см. рис. 3). Обозначим координаты вершин нового комплекса . Тогда имеем

(6)

где — коэффициент растяжения (рекомендуемое значение — 2), — вектор координат центра тяжести [комплекса](javascript:termInfo(%22комплекса%22)) (см. (5)).

|  |
| --- |
| ?n=3 |

Рис. 3. Сжатие комплекса . Пунктиром показан новый комплекс .

Упрощенная схема модифицированного метода комплексов.

Задаем начальную точку , исходя из которой должен быть построен [комплекс](javascript:termInfo(%22комплекс%22)) , величину , а также коэффициенты полагаем счетчик числа итераций .

Строим начальный [комплекс](javascript:termInfo(%22комплекс%22)) :

поочередно для по формуле (3) находим координаты вершин [комплекса](javascript:termInfo(%22комплекса%22)) ; комплекса ;

если вершина является недопустимой (выходит за границы области ), то по формуле, аналогичной формуле (6), выполняем сжатие уже построенного [комплекса](javascript:termInfo(%22комплекса%22)) с ?k=10вершинами, вдоль направления , где — центр тяжести уже найденных -ой вершин комплекса (см. рис. 4);

если после [сжатия комплекса](javascript:termInfo(%22сжатия%20комплекса%22)) вершин по-прежнему является недопустимой, повторяем описанную процедуру сжатия;

вычисляем значения функции во всех вершинах построенного [комплекса](javascript:termInfo(%22комплекса%22)) .

Находим максимальное из значений функции в вершинах [комплекса](javascript:termInfo(%22комплекса%22))

По формулам (4), (5) отражаем с растяжением вершину [комплекса](javascript:termInfo(%22комплекса%22)) — получаем вершину и новый комплекс :

если точка является не допустимой (выходит за границы области ) и , то по формуле (6), выполняем [сжатие комплекса](javascript:termInfo(%22сжатие%20комплекса%22)) вдоль направления , где — центр тяжести [комплекса](javascript:termInfo(%22комплекса%22)) , до тех пор, пока точка не станет допустимой (см. рис. 5). Переходим к п.5;

если точка является допустимой (не выходит за границы области ) и , то переходим к шагу 5;

если точка является не допустимой, но , то переходим к п. 6.

Проверяем условие окончания поиска (см. ниже). Если условие окончания поиска выполнено, то в качестве точки принимаем вершину [комплекса](javascript:termInfo(%22комплекса%22)) , к которой функция имеет наименьшее значение, вычисляем соответствующие значения и завершаем итерации. Иначе — переходим к п. 3

Если , то полагаем ; если то полагаем (см. рис. 6). Переходим к п.3?k=10

|  |
| --- |
| ?n=4 |

Рис. 4. Построение комплекса C0.

|  |
| --- |
| ?n=5 |

Рис. 5. Построение комплекса Cr+1.

|  |
| --- |
| ?n=6 |

Рис. 6. Построение комплекса Cr+1.

На рис. 4 точка оказалась за границей области . После операции [сжатия комплекса](javascript:termInfo(%22сжатия%20комплекса%22)) вершины [комплекса](javascript:termInfo(%22комплекса%22)) вдоль направления получаем вершину . Здесь ()p — центр тяжести комплекса.

На рис. 5 полагается, что . Точка оказалась границей области . После операции [сжатия комплекса](javascript:termInfo(%22сжатия%20комплекса%22)) вершины [комплекса](javascript:termInfo(%22комплекса%22)) вдоль направления получаем вершину .

На рис. 6 полагается, что . Точка оказалась за границей области - нарушено ограничение . Точка получена проектированием точки на прямую .

В качестве критерия окончания поиска может использоваться следующее условие: максимальная длина ребра [комплекса](javascript:termInfo(%22комплекса%22)) не превышает — требуемую точность решения по . Может использоваться также следующее аналогичное условие: максимальная разность значений функции в двух вершинах комплекса не превышает — требуемую точность решения по .

Могут использоваться также более сложные условия окончания поиска, учитывающие текущий размер [комплекса](javascript:termInfo(%22комплекса%22)) или в некотором смысле среднее значение функции в его вершинах (см. параграф 8.2).

Изложенная схема [метода комплексов](javascript:termInfo(%22метода%20комплексов%22)) приводит к "уплощению" [комплекса](javascript:termInfo(%22комплекса%22)) вблизи границы [области допустимых значений](javascript:termInfo(%22области%20допустимых%20значений%22)) , что может значительно уменьшить эффективность метода. С целью преодоления этого недостатка через фиксированное количество итераций находятся максимальная и минимальная диагонали комплекса и, если их отношение превышает заданное, то по рассмотренной схеме производится построение нового комплекса.

### 9.4. Метод линейной аппроксимации

Сделаем ряд дополнительных допущений. Пусть [множество допустимых значений](javascript:termInfo(%22множество%20допустимых%20значений%22)) ?k=10определяется только ограничениями типа неравенств и [ограничивающие функции](javascript:termInfo(%22ограничивающие%20функции%22)) являются непрерывными, дифференцируемыми и выпуклыми:

, (1)

(2)

Пусть функция также непрерывна, дифференцируема и выпукла во множестве .

Суть метода линейной аппроксимации.

[Метод линейной аппроксимации](javascript:termInfo(%22Метод%20линейной%20аппроксимации%20%22)) использует на каждой итерации линейную аппроксимацию [целевой функции](javascript:termInfo(%22целевой%20функции%22)) и [ограничивающих функций](javascript:termInfo(%22ограничивающих%20функций%22)) в окрестности текущей точки

|  |  |
| --- | --- |
|  | (3) |

|  |  |
| --- | --- |
| (4) |  |

Вместо задачи (1) на каждой итерации решается вспомогательная [задача линейного программирования](javascript:termInfo(%22задача%20линейного%20программирования%22))

|  |  |
| --- | --- |
|  | (5) |

где .

В изложенном виде метод может привести к выходу точки за пределы допустимой области (см. прим. 1).

Пример 1

Рассмотрим следующую двумерную [задачу условной оптимизации](javascript:termInfo(%22задачу%20условной%20оптимизации%22)) с тремя ограничениями типа неравенств (первое ограничение – нелинейное, второе и третье ограничения — линейные):

где

Положим, что текущая точка есть . Линеаризуем [целевую функцию](javascript:termInfo(%22целевую%20функцию%22)) и [ограничивающую функцию](javascript:termInfo(%22ограничивающую%20функцию%22)) в окрестности этой точки.

Поскольку

по формуле (3) имеем

Аналогично для [ограничивающих функции](javascript:termInfo(%22ограничивающих%20функции%22)) по формуле (4) имеем:

Пример иллюстрирует рис. 1, на котором линии уровня [целевой функции](javascript:termInfo(%22целевой%20функции%22)) получены с помощью следующей MATLAB-программы:

x = -2 : 0.1 : 6;

y = x;

[X, Y] = meshgrid(x);

Z = X.^2 + (Y - 6.).^2 - 12;

V = [-10, -5, 0, 5, 10, 20, 40, 80];

[C, h] = contour(X, Y, Z, V);

clabel(C, h);

|  |
| --- |
| ?n=1 |

Рис. 1. Точка Xr+1 лежит вне области допустимых значений D.

Примечание 1

Прямая представляет собой след от пересечения плоскости, которая является касательной к поверхности в точке , с плоскостью 0x1x2. Эта прямая не обязательно является касательной к линии — прямая может пересекать кривую , быть касательной к ней или не иметь с ней общих точек. Аналогично, линия уровня функции представляет собой след от пересечения плоскости, которая является касательной к поверхности в точке , с плоскостью .

Чтобы избежать выхода текущей точки за границы [области допустимых значений](javascript:termInfo(%22области%20допустимых%20значений%22)), следующее приближение к точке минимума функции из множества находится по формуле

|  |  |
| --- | --- |
|  | (6) |

где — решение вспомогательной [задачи линейного программирования](javascript:termInfo(%22задачи%20линейного%20программирования%22)) (5).

Величина шага в формуле (6) в разных вариантах [метода линейной аппроксимации](javascript:termInfo(%22метода%20линейной%20аппроксимации%22)) может определяться разными способами. Приведем два из множества возможных способов.

1-й способ выбора величины шага . Величина находится как решение [задачи одномерной оптимизации](javascript:termInfo(%22задачи%20одномерной%20оптимизации%22)) функции на отрезке

(7)

2-й способ выбора величины шага . Полагаем и по формуле (6) находим вектор . Вычисляем значение [целевой функции](javascript:termInfo(%22целевой%20функции%22)) в полученной точке. Если условие

(8)

не выполнено, то уменьшаем величину шага (например, в два раза) и повторно проверяем выполнение условия (8). Дробление шага и вычисление производим до выполнения условия (8).

Схема метода линейной аппроксимации.

Рассмотрим вариант метода, в котором используется 1-й способ выбора величины шага .

Задаем начальную точку и полагаем счетчик числа итераций .

Вычисляем градиенты функций в точке .

Решаем [задачу линейного программирования](javascript:termInfo(%22задачу%20линейного%20программирования%22)) (5) – находим точку .

Решаем одномерную задачу минимизации (7) – находим величину шага и вектор .

Проверяем условие окончания поиска (см. ниже). Если условие окончания поиска выполнено, то полагаем и завершаем итерации. Иначе – полагаем и переходим к п.2?k=10

В качестве критерия окончания поиска можно использовать [стандартные условия окончания итераций](javascript:termInfo(%22стандартные%20условия%20окончания%20итераций%22))

или условии

,

где ?k=10— константа, определяющая требуемую точность решения по градиенту функции .

Отметим следующие трудности, возникающие при использовании [метода линейной аппроксимации](javascript:termInfo(%22метода%20линейной%20аппроксимации%22)):

Если функция имеет высокую степень нелинейности, то на основе решения вспомогательной задачи минимизации (5) направление поиска может быть выбрано слишком неточно (см. рис. 2), что приводит к медленной сходимости метода.

Метод требует, чтобы точка принадлежала [множеству допустимых значений](javascript:termInfo(%22множеству%20допустимых%20значений%22)) . Если это требование не выполнено, то прежде приходится использовать какой-либо метод поиска точки, принадлежащей множеству допустимых значений.

|  |
| --- |
| ?n=2 |

Рис. 2. Направление поиска (Xr+1-Xr), которое обеспечивает метод на основе линейной аппроксимации, далеко от оптимального направления (X\*-Xr).

Возможны модификации [метода линейной аппроксимации](javascript:termInfo(%22метода%20линейной%20аппроксимации%22)), при которых необходимые производные вычисляются с помощью конечных разностей.

### 9.5. Метод проекции градиента

Рассмотрим многомерную [задачу локальной условной оптимизации](javascript:termInfo(%22задачу%20локальной%20условной%20оптимизации%22))

|  |  |
| --- | --- |
| , | (1) |

где [множество допустимых значений](javascript:termInfo(%22множество%20допустимых%20значений%22)) определяется только ограничениями типа неравенств

|  |  |
| --- | --- |
|  | (2) |

и [целевая функция](javascript:termInfo(%22целевая%20функция%22)) и [ограничивающие функции](javascript:termInfo(%22ограничивающие%20функции%22)) являются непрерывными и дифференцируемыми функциями, а ограничивающие функции еще и выпуклы.

Проектирование точки на множество.

Идея [метода проекции градиента](javascript:termInfo(%22метод%20проекции%20градиента%22)) состоит в том, что если на некоторой итерации точка

|  |  |
| --- | --- |
| , | (3) |

полученная с помощью [градиентного метода наискорейшего спуска](javascript:termInfo(%22градиентного%20метода%20наискорейшего%20спуска%22)) (см. главу 7), оказывается вне [множества допустимых значений](javascript:termInfo(%22множества%20допустимых%20значений%22)) , то она возвращается на это множество. Возврат производится с помощью процедуры "[проекция точки на множество](javascript:termInfo(%22проекция%20точки%20на%20множество%22))". Напомним, что в формуле (3) — длина шага на -ой итерации в направлении ;

единичный вектор направления антиградиента функции в точке , ||\*|| — некоторая векторная норма, например, евклидова.

Определение. Проекцией точки на замкнутое множество называется ближайшая к точка множества . Т.е. точка называется проекцией точки на замкнутое множество , если

|  |  |
| --- | --- |
|  | (4) |

где — расстояние между точками в некоторой метрике, например, ?k=10

Проекцию точки на замкнутое множество будем обозначать (см. рис. 1). Очевидно, что , если .

|  |
| --- |
| ?n=1 |

Рис. 1. К определению проекции точки на множество. Прямая l является касательной к границе области D в точке .

Можно показать, что если D — замкнутое [выпуклое множество](javascript:termInfo(%22выпуклое%20множество%22)) пространства , то для любой точки существует единственная ее проекция на это множество.

Задача (4) поиска [проекции точки на множество](javascript:termInfo(%22проекции%20точки%20на%20множество%22)) также является многомерной [задачей условной оптимизации](javascript:termInfo(%22задачей%20условной%20оптимизации%22)) и ее решении может вызвать в общем случае значительные затруднения.

Задача (4) становится [задачей квадратичного программирования](javascript:termInfo(%22задачей%20квадратичного%20программирования%22)), если множество D задается лишь линейными ограничениями типа неравенств и если функция является квадратичной функцией , например, если .

Наибольший практический интерес представляет ситуация, когда множество D таково, что задача (4) может быть решена в явном виде. Приведем несколько наиболее практически важных примеров таких множеств.

Схема комбинации метода проекции градиента с методом дробления шага.

[Метод проекции градиента](javascript:termInfo(%22Метод%20проекции%20градиента%22)) может быть скомбинирован со многими градиентными методами (см. главу 7). Рассмотрим комбинацию метода проекции градиента с градиентным методом дробления шага.

Напомним, что в [градиентном методе с дроблением шага](javascript:termInfo(%22градиентном%20методе%20с%20дроблением%20шага%22)) величина шага ?k=10находится из условия

|  |  |
| --- | --- |
| ?k=10 | (5) |

Схема метода:

Задаем начальную точку ?k=10, начальную величину шага ?k=10и коэффициент дробления шага ?k=10. Полагаем счетчик числа итераций ?k=10.

По формуле (3) вычисляем координаты точки ?k=10и проекцию ?k=10этой точки на множество D.

Вычисляем величину ?k=10— значение функции ?k=10в точке ?k=10.

Если условие дробления шага выполнено (см. параграф 7.1), то переходим к следующему пункту. Иначе – переходим к п.6.

Полагаем ?k=10и переходим к п.2.

Проверяем условие окончания поиска (см. ниже). Если условие окончания поиска выполнено, то полагаем ?k=10и завершаем итерации. Иначе – полагаем ?k=10переходим к п.2?k=10

В качестве критерия окончания поиска можно использоваться одно из [стандартных условий окончания итераций](javascript:termInfo(%22стандартных%20условий%20окончания%20итераций%22))

?k=10

?k=10

или условие ?k=10, где — константа, определяющая требуемую точность решения по градиенту функции ?k=10.

Комбинацию [метода проекции градиента](javascript:termInfo(%22метода%20проекции%20градиента%22)) и [градиентного метода с дроблением шага](javascript:termInfo(%22градиентного%20метода%20с%20дроблением%20шага%22)) иллюстрирует рис. 2, на котором показан фрагмент линий уровня [функции Химмельблау](javascript:termInfo(%22функции%20Химмельблау%22)).

|  |
| --- |
| ?n=2 |

Рис. 2. Траектория поиска минимума функции Химмельблау комбинацией метода проекции градиента и градиентного метода с дроблением шага.

Известны модификации [метода проекции градиента](javascript:termInfo(%22метода%20проекции%20градиента%22)), ориентированные на решение [задач условной оптимизации с ограничениями типа равенств](javascript:termInfo(%22задач%20условной%20оптимизации%20с%20ограничениями%20типа%20равенств%22)).

### Пример Проектирование точки на множество

Пусть [множество допустимых значений вектора варьируемых параметров](javascript:termInfo(%22множество%20допустимых%20значений%20вектора%20варьируемых%20параметров%22)) представляет собой неотрицательный октант в ?k=10. Изобразите на рисунке проекции на это множество двух точек, не принадлежащих множеству ?k=10и одной точки, принадлежащей этому множеству.

Решение

Проекция ?k=10точки ?k=10на неотрицательный октант пространства ?k=10равна?k=10, где?k=10На рис. 1 точки ?k=10лежат вне множества ?k=10а точка ?k=10- принадлежит этому множеству.

|  |
| --- |
| ?n=1 |

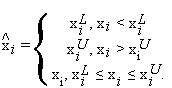
Рис. 1.

### Пример Проектирование точки на множество

Пусть [множество допустимых значений вектора варьируемых параметров](javascript:termInfo(%22множество%20допустимых%20значений%20вектора%20варьируемых%20параметров%22)) ?k=10представляет собой параллелепипед в ?k=10. Изобразите на рисунке проекции на это множество четырех точек, не принадлежащих множеству ?k=10и одной точки, принадлежащей этому множеству.

Решение

Если [множество допустимых значений вектора варьируемых параметров](javascript:termInfo(%22множество%20допустимых%20значений%20вектора%20варьируемых%20параметров%22)) ?k=10представляет собой параллелепипед в ?k=10, т.е. ?k=10, то проекция ?k=10точки ?k=10на это множество равна ?k=10, где



На рис. 1 точки X1?k=10 X2?k=10 X3?k=10 X4?k=10 лежат вне множества D, точка X5 — внутри этого множества.

|  |
| --- |
| ?n=1 |

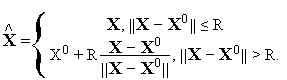
Рис. 1.

### Пример Проектирование точки на множество

Пусть [множество допустимых значений вектора варьируемых параметров](javascript:termInfo(%22множество%20допустимых%20значений%20вектора%20варьируемых%20параметров%22)) ?k=10представляет собой шар радиуса R пространства ?k=10с центром в точке ?k=10. Изобразите на рисунке проекцию на это множество точки, не принадлежащей множеству ?k=10и точки, принадлежащей этому множеству.

Решение

Если [множество допустимых значений вектора варьируемых параметров](javascript:termInfo(%22множество%20допустимых%20значений%20вектора%20варьируемых%20параметров%22)) ?k=10представляет собой шар радиуса R пространства ?k=10с центром в точке ?k=10, т.е. ?k=10, то проекция ?k=10?k=10, где



На рис. 1 точка ?k=10лежат вне множества D, точка ?k=10— внутри этого множества. Прямая ?k=10является касательной к поверхности окружности в точке PD(?k=10) = ?k=10.

|  |
| --- |
| ?n=1 |

Рис. 1.