

Федеральное агентство по образованию
Нижегородский государственный университет им. Н.И. Лобачевского

Национальный проект «Образование»
Инновационная образовательная программа ННГУ. Образовательно-научный центр
«Информационно-телекоммуникационные системы: физические основы и
математическое обеспечение»

С. Ю. Городецкий

Лабораторный практикум по методам локальной оптимизации в программной системе LocOpt

*Учебно-методические материалы по программе повышения
квалификации «Информационные технологии и компьютерное
моделирование в прикладной математике»*

Нижний Новгород
2007

*Учебно-методические материалы подготовлены в рамках
инновационной образовательной программы ННГУ: Образовательно-
научный центр «Информационно-телекоммуникационные
системы: физические основы и математическое обеспечение»*

Городецкий С.Ю. Лабораторный практикум по методам локальной оптимизации в программной системе LocOpt. Учебно-методический материал по программе повышения квалификации «Новые подходы в исследованиях и разработках информационно-телекоммуникационных систем и технологий». Нижний Новгород, 2007, 121 с.

В современных условиях изучение такого важного раздела прикладной математики, как вычислительные методы локальной оптимизации, не может быть признано полным, если в соответствующие учебные курсы не включены специальные компьютерные практикумы, позволяющие исследовать поведение упомянутых выше методов на различных примерах конкретных задач. Программная система LocOpt является специализированной программной лабораторией, ориентированной на проведение подобных исследований. LocOpt предоставляет удобные средства для постановки задач оптимизации с требуемыми свойствами, а также обладает богатыми возможностями при визуализации и анализе структуры решаемых задач. Кроме того, эта программная система включает разнообразные формы наблюдения за работой вычислительных методов. Система LocOpt включает широкий набор методов безусловной скальной оптимизации и общие методы учета функциональных ограничений. LocOpt создана сотрудниками лаборатории «Динамика и оптимизация» кафедры ТУиДМ ВМК ННГУ.

Представленный ниже учебно-методический материал включает описание этой программной лаборатории, а также описание включенных в LocOpt методов безусловной локальной оптимизации и общих методов учета ограничений.

© С.Ю. Городецкий

ОГЛАВЛЕНИЕ

1. ОБЩЕЕ ПРЕДСТАВЛЕНИЕ О ПРОГРАММНОЙ ЛАБОРАТОРИИ LocOpt.	5
1.1. Назначение программной лаборатории и вид главного окна LocOpt.....	5
1.2. Методы безусловной локальной оптимизации и общие методы учета ограничений, включенные в систему LocOpt.....	7
2. МЕТОДЫ БЕЗУСЛОВНОЙ ЛОКАЛЬНОЙ ОПТИМИЗАЦИИ.....	8
2.1. Общая структура методов локального поиска, принцип локального спуска	8
2.2. Априорные предположения о задаче и их роль в интерпретации результатов испытаний	10
2.3. Классификация траекторных методов локального поиска	11
2.4. Стратегии выбора шагового множителя.....	11
2.5. Градиентные методы и метод Ньютона, оценки скорости сходимости	20
2.6. Эффективные методы второго порядка	28
2.6.1. Расширение области сходимости метода Ньютона за счет выбора шагового множителя.....	29
2.6.2. Стратегии модификации матриц Гессе при нарушении их положительной определенности.....	31
2.7. Методы первого порядка, явно изменяющие метрику пространства.....	38
2.7.1. Квазиньютоновские методы	39
2.7.2. Модифицированные квазиньютоновские методы	47
2.7.3. Методы растяжения пространства (R-алгоритмы Н.З. Шора)	48
2.8. Методы сопряженных направлений.....	51
2.8.1. Сопряженные направления и их свойства	51
2.8.2. Метод сопряженных градиентов Флетчера–Ривса	53
2.9. Некоторые методы прямого поиска	57
2.9.1. Метод Нелдера–Мида	57
2.9.2. Метод Хука–Дживса	59
3. МЕТОДЫ УЧЕТА ОГРАНИЧЕНИЙ В ЛОКАЛЬНОЙ ОПТИМИЗАЦИИ... 62	62
3.1. Особенности применения методов локального поиска при двусторонних ограничениях на переменные.....	62
3.1.1. Особенности учета двусторонних ограничений на переменные в методах гладкой оптимизации	62
3.1.2. Учет двусторонних ограничений в методах прямого поиска.....	66
3.2. Методы учета ограничений общего вида.....	67
3.2.1. Метод внешнего штрафа.....	67
3.2.1.1. Общее описание и некоторые свойства	68
3.2.1.2. Условия сходимости метода штрафов и адаптивная настройка коэффициента штрафа.....	70
3.2.1.3. Структура задач со штрафом и характер поведения оценок.....	73
3.2.1.4. Оценки скорости сходимости метода внешнего штрафа	75
3.2.1.5. Недостаточность локальных методов при использовании метода штрафов	76
3.2.1.6. Метод штрафов и оценка множителей Лагранжа.....	77
3.2.2. Метод модифицированных функций Лагранжа	78
3.2.2.1. Преобразование к форме задачи с ограничениями-равенствами.....	78
3.2.2.2. Построение метода модифицированных функций Лагранжа для задач с ограничениями-равенствами.....	79

3.2.2.3. Метод модифицированной функции Лагранжа в задачах с ограничениями-неравенствами	82
--	----

4. ОБЩИЕ ПРИНЦИПЫ ПОСТРОЕНИЯ И ПРАВИЛА ИСПОЛЬЗОВАНИЯ

СИСТЕМЫ LocOpt.....	86
4.1. Наборы функций и списки задач.....	86
4.1.1. Наборы функций	86
4.1.2. Списки задач	87
4.2. Подготовка задач для исследования	88
4.3. Проведение экспериментов при исследовании задач оптимизации	90
4.3.1. Выбор метода локальной оптимизации	91
4.3.2. Выбор способа учета ограничений.....	92
4.3.3. Изменение положения начальной точки.....	94
4.3.4. Автоматический выбор начальной точки локального поиска.....	95
4.3.5. Выполнение расчета	95
4.3.6. Редактирование исследуемой задачи	95
4.3.7. Навигация по окнам экспериментов	96
4.4. Отбор результатов при подготовке отчета по проведенному исследованию..	96
4.4.1. Отбор результатов из архива расчетов и их копирование.....	97
4.4.2. Копирование изображений	98
4.5. Основные инструментальные средства среды LocOpt.....	99
4.5.1. Окно стандартного набора задач.....	99
4.5.2. Конструктор функций и задач.....	100
4.5.3. Окно просмотра линий равного уровня функций.....	100
4.5.4. Окно просмотра линий равного уровня и структуры задачи	101
4.5.5. Окно эксперимента	102
4.5.6. Окно просмотра поверхности.....	103
4.5.7. Окно для исследования одномерных сечений	104
4.5.8. Окно настройки параметров расчета.....	105
4.5.9. Окно навигации по окнам экспериментов	105
4.6. Описание интерфейса главного окна среды LocOpt.....	105
4.7. Использование конструктора функций и задач	107
4.8. Интерфейс окна эксперимента	111
4.9. Средства составления отчетов.....	116
4.10. Создание и подключение DLL пользователя.....	116
4.10.1. Пример заголовочного файла DLL и текста cpp-файла реализации DLL.	117
ЛИТЕРАТУРА	124
Основная литература	124
Дополнительная литература.....	124

1. ОБЩЕЕ ПРЕДСТАВЛЕНИЕ О ПРОГРАММНОЙ ЛАБОРАТОРИИ LocOpt

1.1. Назначение программной лаборатории и вид главного окна LocOpt

Программная лаборатория LocOpt является учебно-исследовательской программной системой, позволяющей всесторонне и в максимально наглядной форме *изучать свойства методов локальной оптимизации и учета ограничений* в задачах поиска оптимальных решений.

В LocOpt можно решать задачи следующего вида:

$$f(y) \rightarrow \min, y \in Y, \quad (1.1)$$

$$Y = \{y \in D: g_j(y) \leq g_j^+, j=1, \dots, m\}, \quad (1.2)$$

$$D = \{y=(y_1, \dots, y_N): a_i \leq y_i \leq b_i, i=1, \dots, N\}, \quad (1.3)$$

с числом переменных до десяти и практически неограниченным числом функциональных ограничений.

В программную лабораторию включены инструментальные средства, позволяющие выполнять исследования по широкому спектру направлений:

- изучать структуру функций и задач из предоставляемых системой стандартных наборов (исследование выполняется с помощью средств построения линий равного уровня и поверхностей);
- конструировать новые наборы функций и задач с заданными свойствами (исследование выполняется с помощью специального инструмента — конструктора задач, а также окон оперативной визуализации линий равного уровня целевых функций и структур допустимых множеств);
- исследовать особенности функций вспомогательных задач, возникающих в процессе работы методов учета ограничений: в методе внешнего штрафа и методе модифицированных функций Лагранжа;
- изучать сравнительное поведение нескольких групп методов локальной оптимизации в задачах без функциональных ограничений;
- исследовать работу этих методов в сочетании с методами внешнего штрафа и методом модифицированных функций Лагранжа для задач с функциональными ограничениями;
- определять решения конкретных оптимизационных задач;
- формировать результаты выполненных расчетов в виде, удобном для составления отчетов.

Потребность в изучении методов локальной оптимизации обусловлена тем, что эти методы являются полезным, а часто и необходимым инструментом поиска оптимальных решений. Можно указать несколько таких ситуаций:

- известна приближенная оценка глобально-оптимального решения, найденная с недостаточной точностью, в этом случае для решения задачи достаточно с высокой точностью найти локально-оптимальное решение, соответствующее имеющейся оценке;
- необходимо предварительное исследование структуры решаемой задачи, например, можно установить многоэкстремальный характер задачи оптимального выбора, если методы локальной оптимизации, запущенные из нескольких начальных точек получили существенно разные решения;
- в задачах высокой размерности, когда регулярные методы поиска глобального решения не могут быть применены из-за чрезвычайно больших вычислительных затрат на покрытие области точками испытаний, практически единственным средством решения таких задач остаются методы локальной оптимизации, совмещенные, возможно, с процедурами предварительного отбора начальных точек.

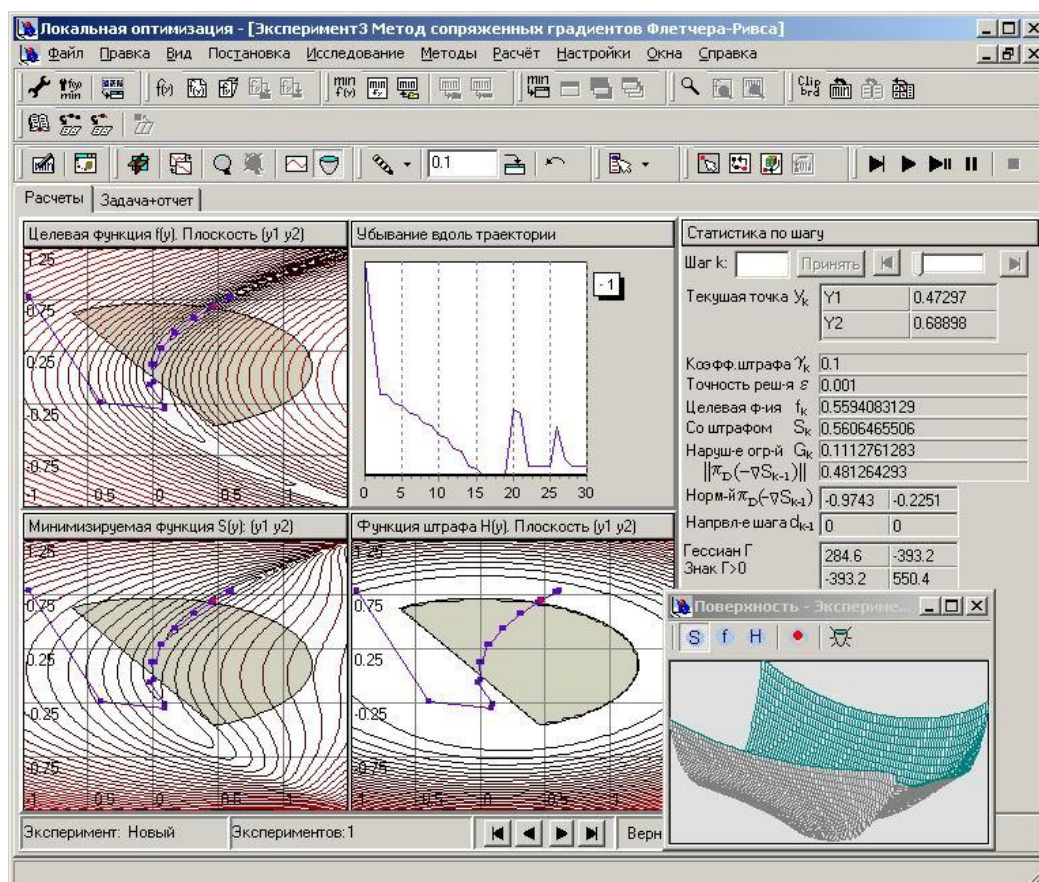


Рис. 1.1. Вид главного окна приложения LocOpt с развернутым окном эксперимента

Программная лаборатория реализована в формате приложения, поддерживающего многодокументный интерфейс: в области главного окна можно открыть несколько дополнительных окон для проведения экспериментов. Вид главного окна работающего приложения LocOpt v.2 на одном из этапов работы с развернутым в нем окном эксперимента показан на рис.1.1.

Правила пользования системой LocOpt, а также методические рекомендации и примеры ее использования приведены ниже в главе 4.

1.2. Методы безусловной локальной оптимизации и общие методы учета ограничений, включенные в систему LocOpt

В этом разделе приведен перечень методов, включенных в версии 2 и 3 системы LocOpt. Более детальное описание алгоритмов и краткие элементы теории их построения приведены в главах 2 и 3.

Методы безусловной локальной оптимизации включают:

- метод Ньютона;
- метод Ньютона–Рафсона (метод Ньютона с регулировкой шага);
- метод Ньютона–Рафсона с модификацией матрицы Гессе;
- квазиньютонов DFP–метод (переменной метрики);
- DFP–метод с модификацией оценочной матрицы;
- метод растяжения пространства Шора Н.З.;
- метод наискорейшего градиентного спуска;
- метод сопряженных градиентов Флетчера–Ривса;
- метод прямого поиска Нелдера–Мида;
- метод прямого поиска Хука–Дживса.

Эти методы включают в свой состав учет двухсторонних ограничений на переменные, которые всегда присутствуют в постановках задач оптимизации в системе LocOpt.

В качестве метода учета функциональных ограничений общего вида в системе LocOpt версии 2 используется метод внешнего степенного штрафа, а в версию 3 дополнительно добавлен метод модифицированных функций Лагранжа.

2. МЕТОДЫ БЕЗУСЛОВНОЙ ЛОКАЛЬНОЙ ОПТИМИЗАЦИИ

В этой главе будет рассмотрено несколько групп методов поиска локально-оптимальных решений в задачах общего вида без ограничений:

$$f(y) \rightarrow \min, y \in R^N. \quad (2.1)$$

Указание на общий вид задачи означает, что относительно функции $f(y)$ не выдвигается каких-либо существенных предположений кроме достаточной гладкости. Поэтому ниже в последующих разделах не приводятся специальные методы локальной оптимизации, применимые при существенных дополнительных предположениях о минимизируемой функции, а рассматриваются лишь методы общего вида.

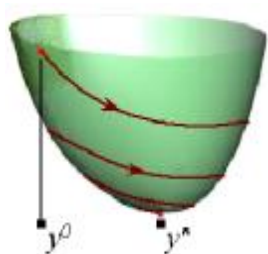
Кроме того, следует отметить, что в программной лаборатории LocOpt задачи локальной оптимизации в форме (2.1) никогда не ставятся. В этой системе всегда принимается, что область поиска является многомерным параллелепипедом D , т.е. задача принимает форму

$$f(y) \rightarrow \min, y \in D = \{y \in R^N : a \leq y \leq b\} \quad (2.2)$$

Поэтому в некоторых моментах приведенного в разделе 2.1 описания методов безусловной локальной оптимизации будет учитываться наличие ограничений вида $y \in D$. В основном, такой учет будет сделан в алгоритмах выбора шагового множителя. В остальных частях общего описания методов наличие области D пока учитываться не будет.

Специальные модификации методов локальной оптимизации для учета двухсторонних ограничений на переменные описаны в п. 2.2.2 раздела 2.2. Методы учета общих функциональных ограничений приведены в п. 2.2.1.

2.1. Общая структура методов локального поиска, принцип локального спуска



К настоящему времени разработано достаточно много методов локальной оптимизации для решения задач общего вида. Большинство из них используют принцип локального спуска, когда метод последовательно на каждом шаге переходит к точкам с существенно меньшими значениями целевой функции. Почти все эти методы могут быть представлены в виде итерационного соотношения

$$y^{k+1} = y^k + x^k d^k, \quad (2.3)$$

где y^k — точки *основных испытаний*, состоящих в вычислении $I^k = I(y^k)$ — набора тех или иных *локальных характеристик* целевой функции в точке y^k , d^k — направления смещения из точек y^k , вычисляемые по результатам основных испытаний, а x^k — коэф-

фициенты, определяющие величины смещений вдоль выбранных направлений (т.е. шаговые множители). Методы такого типа можно назвать траекторными.

В набор вычисляемых для функции локальных характеристик $I^k = I(y^k)$ могут входить: значение функции $f^k = f(y^k)$, вектор градиента $\nabla f^k = \nabla f(y^k)$, матрица вторых производных (гессиан) $\Gamma_k = \nabla^2 f(y^k)^1$. Какой именно набор характеристик измеряется — зависит как от свойств решаемой задачи, так и от выбранного метода оптимизации.

Для определения величин смещений x^k вдоль направлений d^k методы могут выполнять вспомогательные (*рабочие*) шаги. Это приводит к дополнительным измерениям локальных характеристик целевой функции вдоль направления d^k (рис. 2.1).

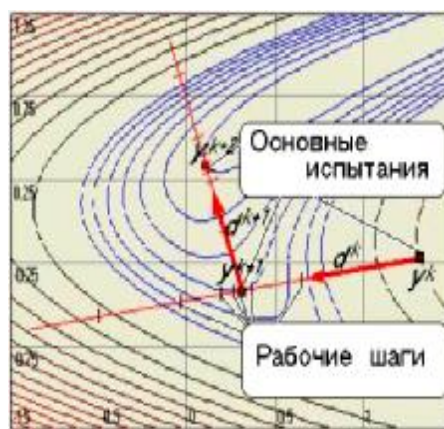


Рис. 2.1. Общая структура организации поиска локального минимума

Переходы от точек y^k к точкам y^{k+1} выполняются таким образом, чтобы обеспечить существенное убывание значений функции $f^k = f(y^k)$ в результате шага, достаточное для сходимости метода. Простое выполнение условий строгого убывания функции, когда для всякого k выполняется $f^{k+1} < f^k$, очевидно, не может гарантировать сходимости к решению задачи.

В качестве одного из условий останова вычислений в методах локальной оптимизации, применяемых в задачах без ограничений (2.1) при непрерывной дифференцируемости целевых функций, часто используется достаточная малость нормы градиента

$$\|\nabla f(y^k)\| \leq \epsilon. \quad (2.4)$$

Нужно отметить, что выполнение условия (2.4), в общем случае, конечно же не гарантирует близость точки y^k к решению задачи по координатам. Чтобы

¹ Обозначения Γ_k или Γ_k^f для значений матриц Гессе $\nabla^2 f(y^k)$ в дальнейшем будем использовать в целях сокращения записи.

такая связь стала возможной, необходимо наложить на $f(y)$ достаточно жесткие ограничения. Однако следует иметь в виду, что даже при существенных предположениях о виде функции, как правило, остаются неизвестными константы, входящие в описание ее свойств. Это усложняет или же делает невозможным использование оценок близости к решению, даже если они существуют.

Для методов, не использующих вычисление градиента (например, для методов прямого поиска Хука–Дживса или Нелдера–Мида), основное правило останова строится по другим, по сравнению с (2.4), правилам, своим для каждого из методов.

В общем случае, кроме основного могут использоваться другие эвристические правила, определяющие достаточную малость изменения за фиксированное число шагов $s \geq 1$ либо координат точек на траектории метода

$$\|y^{k+s} - y^k\| \leq \epsilon,$$

либо значений целевой функции вдоль этой траектории

$$\|f^{k+s} - f^k\| \leq \epsilon.$$

Однако в системе LocOpt два последние правила останова не используются.

В критерий останова всегда включаются также ограничения на объем вычислений, например, на число итераций, выполненных методом, или на число измерений функции.

2.2. Априорные предположения о задаче и их роль в интерпретации результатов испытаний

Выбор направлений d^k при выполнении итераций методов локального поиска в большинстве методов происходит на основе результатов основных испытаний I^k . Для того, чтобы было возможно определить d^k , необходимо использовать имеющуюся априорную информацию, т.е. принятые предположения о свойствах решаемой задачи. Очевидно, что при отсутствии таких предположений, обоснованный выбор точек очередных испытаний был бы невозможен.

Совокупность предположений относительно свойств решаемой задачи назовем *моделью задачи*. В большинстве случаев принятая модель может быть описана в терминах принадлежности целевой функции $f(y)$ некоторому классу функций: $f \in F$. Выбор модели существенно влияет на интерпретацию результатов проведенных испытаний.

В задачах локальной оптимизации общего вида априорная информация о функциях бывает достаточно скудной. Например, если о функции f известно только то, что она непрерывно дифференцируема, т.е. принадлежит классу $F = C^1$, а испытание в точке y^k состоит в вычислении значения функции f^k и градиента ∇f^k , то, пожалуй, единственной

разумной стратегией в определении d^k , использующей имеющуюся информацию, следует признать выбор *антиградиента* $d^k = -\nabla f^k$ — направления скорейшего локального убывания функции. Если бы о целевой функции было известно больше, например, что $f \in C^2$ (т.е. f локально близка к квадратичной функции), то при той же измеряемой информации можно было бы указать значительно лучшие стратегии выбора направлений поиска, чем антиградиент, используя квадратичную модель функции и результаты предыдущих измерений градиента.

2.3. Классификация траекторных методов локального поиска

Проведем классификацию в зависимости от той локальной информации, которую метод получает при выполнении основных испытаний.

Если метод использует результаты испытаний, включающие вычисление производных функции до k -го порядка, то его относят к *методам k -го порядка*.

Обычно выделяют методы *второго порядка* (используют вычисления функции, ее градиента и матрицы Гессе), *первого порядка* (используют вычисления функции и ее градиента), а также *нулевого порядка* (используют только вычисления функции).

Если метод нулевого порядка не использует предположений о гладкости функции, то его называют *методом прямого поиска*.

Методы прямого поиска основаны на эвристических правилах определения направлений убывания минимизируемой функции, и их структура может отличаться от описанной в разделе 2.1. Почти все остальные методы соответствуют структуре (2.3), и, следовательно, требуют для своей реализации разработки специальных вычислительных процедур, позволяющих определять в (2.3) коэффициенты одномерных смещений (*шаговых множителей*) x^k вдоль выбираемых направлений d^k .

2.4. Стратегии выбора шагового множителя

Итерационная форма методов локального поиска (2.3) требует многократного применения процедур выбора одномерных смещений x^k вдоль направлений d^k . В процессе работы метода эти процедуры могут выполняться сотни и тысячи раз. Это предъявляет повышенные требования к эффективности таких процедур. Остановимся на принципах и алгоритмах определения смещений.

Величина коэффициента x^k , определяющего длину шага вдоль направления d^k , может выбираться различными способами.

Постоянный шаговый множитель

В простейшем случае используются постоянные шаговые множители x^k . Например, в

методе Ньютона (см. ниже) всегда $x^k \equiv 1$. Сходимость градиентного метода с $d^k = -\nabla f^k$ при определенных свойствах минимизируемой функции также можно обеспечить, выбирая x^k равным постоянному $a > 0$, значение которого должно быть согласовано со свойствами класса функций.

Алгоритмический выбор шагового множителя

В более общем случае величина шагового множителя x^k выбирается алгоритмически, исходя из некоторых заданных критериев. Они подбираются так, чтобы удовлетворяющий им шаговый множитель не мог стать слишком большим, но, с другой стороны, не был бы и слишком малым. Весьма распространенными при построении методов являются следующие: критерий близости к минимуму по направлению и его модификации, критерий существенности убывания функции, а также требование по степени уменьшения первоначального интервала возможных значений x^k .

В основе первого критерия лежит требование, чтобы в точке $y^k + x^k d^k$ величина скорости изменения функции f в направлении d^k была в заданное число раз меньше скорости ее изменения в точке y^k . Это требование формализуется следующим образом. Задается малый положительный коэффициент h , и величина x^k определяется условием

$$x^k \in \Pi_1(h), \quad 0 \leq h < 1, \quad (2.5)$$

$$\Pi_1(h) = \{x \geq 0 : |(\nabla f(y^k + x \cdot d^k), d^k)| \leq h \cdot (-\nabla f(y^k), d^k)\}. \quad (2.6)$$

Напомним, что используемые в формулах (2.6), (2.8) скалярные произведения градиента функции $f(y)$ на вектор направления d определяют производные функции f в точке y в направлении d , а именно

$$\nabla f(y)/\nabla d = (\tilde{\nabla} f(y), d).$$

На рис. 2.2 условию (2.5), (2.6) соответствует область с надписью «Выполнен критерий 1».

Первый критерий (2.5)–(2.6) может быть заменен на близкий к нему модифицированный критерий (2.5'), (2.6') следующего вида

$$x^k \in \Pi'_1(h), \quad 0 < h < 1, \quad (2.5 \text{ б})$$

$$\Pi'_1(h) = \{x > 0 : (\nabla f(y^k + x d^k), d^k) \geq h(\nabla f(y^k), d^k)\}. \quad (2.6 \text{ б})$$

На рис. 7.7 множеству $\Pi'_1(h)$ соответствует совокупность отрезков. Критерий (2.5)–(2.6) и модификация (2.5')–(2.6') не позволяют шаговому множителю стать слишком малым.

В основе второго критерия (существенности убывания функции) лежит требование:

$$x^k \in \Pi_2(\mu), \quad 0 < \mu < 1, \quad (2.7)$$

$$\Pi_2(\mu) = \{x \geq 0: f(y^k + x \cdot d^k) \leq f(y^k) + \mu \cdot x \cdot (\nabla f(y^k), d^k)\}. \quad (2.8)$$

Этот критерий выполняет роль сдерживающего фактора, препятствующего неограниченному увеличению x^k (см. рис. 2.2).

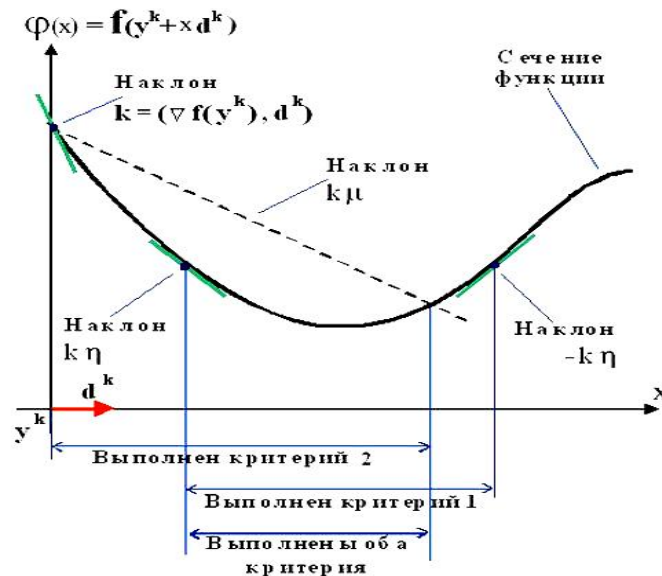


Рис. 2.2. Критерии выбора коэффициента одномерного шага

Если первый и второй критерии используются совместно, то окончательное условие выбора x^k состоит в одновременном выполнении требований (2.5)–(2.6), (2.7)–(2.8). При этом для их непротиворечивости на значения параметров m и h накладывается дополнительное требование вида $m < h$, т.е.

$$x^k \hat{I} P = P_1(h) \cap P_2(m), \quad 0 < m < h < 1. \quad (2.9)$$

На рис. 2.2 дана иллюстрация выбора x^k на основе совокупности этих двух критериев.

Замечание. В системе LocOpt во всех методах, требующих настройки шагового множителя, используется способ выбора (2.9). Однако, с учетом имеющихся двухсторонних ограничений на переменные вида $y \in D$ (см. (2.1)), само условие (2.9) может никогда не выполняться, поэтому в алгоритмы определения шагового множителя, применяемые в системе LocOpt дополнительно добавлено требование остановки по условию сжатия более чем в s раз ($0 < s \ll 1$) исходного интервала поиска по переменной x .

Можно также использовать модифицированное условие, соответствующее совместному применению (2.5')–(2.6') (модифицированный критерий 1) и (2.7)–(2.8) (критерий 2)

$$x^k \in \Pi' = \Pi'_1(h) \cap \Pi_2(m), \quad 0 < m < h < 1, \quad (2.9 \phi)$$

которое, по имени автора, называют правилом Вулфа (Wolfe. P). Вид множества Π' показан на рис. 2.3.

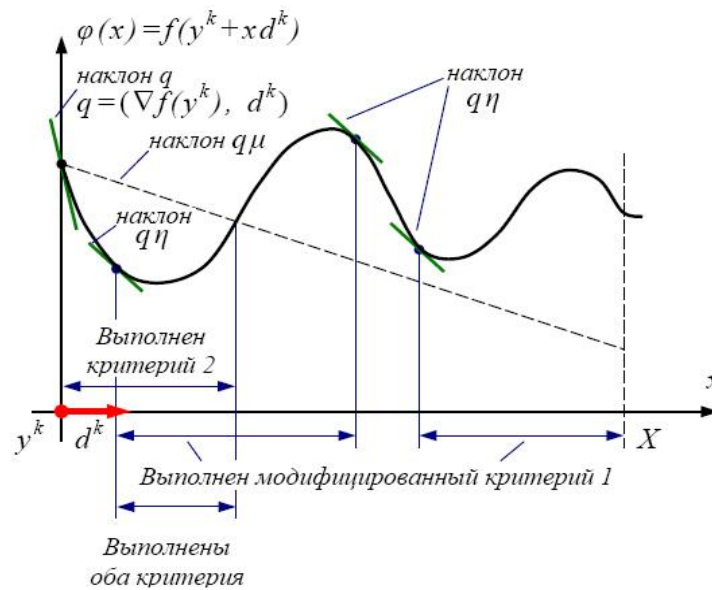


Рис. 2.3. Критерии выбора коэффициента x в правиле Вулфа

Использование одномерной оптимизации

Если при выборе коэффициента одномерного шага используется условие (2.9) с $h \ll 1$, то говорят, что величина шага (шаговый множитель) выбирается из условия «аккуратного» одномерного поиска. При исследовании сходимости методов считается, что такой выбор соответствует выбору x^k из условия достижения минимума функции одномерного сечения

$$j(x) = f(y^k + x d^k). \quad (2.10)$$

Многие методы локальной оптимизации, например, методы сопряженных направлений, при своей вычислительной реализации используют «аккуратный» одномерный поиск. Если условие малости $h \ll 1$ не выполняется, то значение x^k , удовлетворяющее (2.9), также может быть найдено с использованием одномерной оптимизации при соответствующем правиле останова.

Как организовать выбор x^k алгоритмически? Процесс выбора разбивается на два этапа. На первом этапе определяется промежуток $[0, X]$, на котором следует искать значение x^k . В задачах без явных ограничений на переменные этот промежуток должен иметь пересечение с искомым множеством P из (2.9). Если же в задаче есть ограничения $y \in D$, то такого пересечения может не существовать, и тогда значение X определяется из условия попадания на границу множества D из (2.2).

Ниже будем предполагать, что если множество D неограничено ($D = R^N$), то дифференцируемая функция $f(y)$ должна быть ограничена снизу. Это обеспечит конечность приведенного ниже алгоритма.

АЛГОРИТМ ОПРЕДЕЛЕНИЯ ПРОМЕЖУТКА $[0, X]$.

ШАГ 0. При решении задачи без ограничений (2.1) выбирается начальное значение $X^0 = \mathbb{Y}$. При решении задачи с ограничениями (2.2), включающими принадлежность точки y параллелепипеду D , по текущей точке y^k и направлению d^k начальное значение X^0 определяется как наименьшее значение x , при котором точка $y^k + x d^k$ попадает на границу этого параллелепипеда. На остальных шагах алгоритма применительно к задаче без ограничений (2.1) формально применяется $D = R^N$.

ШАГ 1. Выбирается малое $d > 0$ и $x = 0$.

ШАГ 2. Полагается $x = x + d$

ШАГ 3. Если точка $y^k + x d^k \notin D$, то окончательно принимается $X = X^0$ и процесс останавливается. Если $y^k + x d^k \in D$ и выполняется одно из условий: значение $f(y^k + x d^k)$ больше предыдущего или $p = (\tilde{N}f(y^k + x d^k), d^k) > 0$, т.е. обнаруживается значение x , при котором функция f в одномерном сечении возрастает, то полагается $X = x$ и процесс останавливается. Иначе, если $x \notin \Pi_2(m)$, процесс также останавливается с выбором $X = x$. В остальных случаях удваивается величина шага $d := 2d$ и происходит возврат на шаг 2.

Заметим, что в процессе работы алгоритма для каждого полученного x , не выводящего из D , целесообразно проверять условие $x \in \Pi$ из (2.9). Если оно выполнено, следует прервать выполнение алгоритма и окончательно выбрать $x^k = x$, отказавшись от дальнейших вычислений. Если же выполнение данного алгоритма не привело к выбору x^k , то необходимо перейти ко второму этапу. На этом этапе для выбора x^k на промежутке $[0, X]$ выполняется процедура поиска минимума функции одного переменного $j(x)$ из (2.10), являющейся одномерным сечением функции $f(y)$ в направлении d^k . Поиск продолжается до момента первого попадания значения x в множество P или же до того момента, когда будет достигнут заданный коэффициент сжатия σ ($0 < \sigma < 1$) для текущего интервала, содержащего решение, по отношению к длине исходного интервала $[0, X]$.

Дополнительное условие останова (по коэффициенту сжатия интервала) необходимо для задач с нарушением гладкости, а также для задач с ограничениями вида $y \in D$, в

которых минимум может достигаться на границе интервала (т.е. при $x = X$) без выполнения условия $x \in \Pi$.

Заметим, что при поиске минимума функция $j(x)$ обычно считается *униmodalной*, что позволяет применить известный *алгоритм золотого сечения*, близкий к ϵ -оптимальному методу Фибоначчи. Однако во многих задачах локальной оптимизации функции $j(x)$ являются достаточно гладкими, что позволяет применять к поиску минимума алгоритмы, основанные на построении квадратичных аппроксимаций $j(x)$ по результатам ее измерений. Такие методы называют *квазиньютоновскими*. Известно, что при определенных условиях они способны обеспечить скорость сходимости более высокого порядка, чем метод золотого сечения. Одним из таких условий является достаточная близость начальной точки к минимуму функции, что априори гарантировать невозможно. По этой причине использование квазиньютоновских шагов должно выполняться со специальным контролем и в сочетании с другими надежными методами.

Поэтому для осуществления одномерного поиска гладкой *униmodalной* функции наиболее подходящими являются *регуляризованные алгоритмы*, представляющие комбинацию метода золотого сечения с квазиньютоновым алгоритмом [3].

РЕГУЛЯРИЗОВАННАЯ ПРОЦЕДУРА ОДНОМЕРНОГО ПОИСКА состоит в следующем.

ШАГ 0. Полагаем $A=0$, $B=X$, $\tau = (-1+5^{1/2})/2$, $0 < \delta < \tau$.

ШАГ 1. Выполняем три вычисления по методу золотого сечения:

ШАГ 1.1. Вычисляем $x_1 = B - (B-A)t$, $x_2 = A + t(B-A)$ и $j_1 = j(x_1)$, $j_2 = j(x_2)$.

ШАГ 1.2. Если $j_1 \leq j_2$, то полагаем $B = x_2$ и $x_3 = B - (B-A)x$, $j_3 = j(x_3)$. При $j_1 \leq j_3$ полагаем $A = x_3$, $x = x_1$, иначе $B = x_1$, $x = x_3$. Если $j_1 > j_2$, то полагаем $A = x_1$ и $x_3 = A + (B-A)t$, $j_3 = j(x_3)$. При $j_2 > j_3$, полагаем $A = x_2$, $x = x_3$, иначе $B = x_3$, $x = x_2$.

В результате выполнения шага 1 получаем три точки с вычисленными значениями функции, а также интервал $[A, B]$ с расположенной внутри него точкой x , соответствующей лучшему вычисленному значению функции.

ШАГ 2. Определяем u — точку измерения функции по квазиньютоновскому правилу. А именно, u определяется как точка минимума квадратичной аппроксимации функции $j(x)$, построенной по значениям x_1, j_1 ; x_2, j_2 ; x_3, j_3 :

$$u = (-j_1(x_3^2 - x_2^2) + j_2(x_3^2 - x_1^2) - j_3(x_2^2 - x_1^2)) / (2(j_1(x_3 - x_2) - j_2(x_3 - x_1) + j_3(x_2 - x_1))).$$

ШАГ 3. Определяем v — точку измерения функции по правилу золотого сечения:

$$n = \begin{cases} A + (x - A)t, & x > (A + B)/2, \\ B - (B - x)t, & x < (A + B)/2. \end{cases}$$

ШАГ 4. Выбираем точку w для очередного вычисления функции: Если $u \in [\min(n;x); \max(n;x)]$; т.е. если точка квазиньютоновского шага u незначительно уклоняется от середины отрезка $[A, B]$, то в качестве точки нового измерения выбирается квазиньютонова точка $w = u$, однако, чтобы предотвратить ее слишком близкое размещение к точке прежнего измерения x , ее положение корректируется

$$w = \begin{cases} u + d \cdot \text{sign}(u - x), & |u - x| < d, \\ u, & |u - x| > d. \end{cases}$$

Если же u не принадлежит указанному интервалу, используем точку золотого сечения, полагаем $w = n$.

ШАГ 5. Вычисляем $j_w = j(w)$. Проверяем, принадлежит ли w множеству Π из (2.9). Если она принадлежит, или же достигнут предельный уровень сжатия интервала, т.е. $|B-A| < \sigma X$, то переходим на шаг 7, если нет, переходим на шаг 6.

ШАГ 6. Воссоздаем ситуацию, соответствующую шагу 2. А именно, через y обозначим левую из точек w, x , (а через j_y —соответствующее ей значение функции), через z — правую из точек w, x а через j_z — соответствующее значение функции). Если $j_y \leq j_z$, то полагаем $B=z, x=y$. Если $j_y > j_z$, то полагаем $A=y, x=z$. Выделяем из точек x_1, x_2, x_3, w три точки с наименьшими значениями функции и обозначаем их через x_1, x_2, x_3 , а соответствующие им значения функции — через j_1, j_2, j_3 . Переходим на шаг 2.

ШАГ 7. Выполняем завершающие операции: вычисляем $j(x)$ на концах интервала $[A, B]$, в качестве x^k выбираем ту из трех точек w, A, B , где достигается меньшее значение функции j , и останавливаем поиск.

При практических вычислениях регуляризованный алгоритм обеспечивает автоматический переход от стратегии золотого сечения к квазиньютоновой стратегии при попадании процесса поиска в достаточно малую окрестность локального минимума функции $j(x)$. Если функция сечения трижды непрерывно дифференцируема и в локальном минимуме функции $x^* \quad j''(x^*) > 0$, квазиньютонова стратегия обеспечивает сверхлинейную сходимость порядка $p = 1,3$ (доказательство можно найти в [7]).

Конечность приведенного алгоритма определения x^k на отрезке $[A, B] = [0, X]$ всегда гарантируется тем, что в самом плохом случае метод обеспечивает сжатие интервала на шаге с коэффициентом не хуже, чем $1 - d/(B - A) < 1$. Конечно, эта оценка очень пессимистична. В действительности метод сходится гораздо быстрее (может сходиться сверхлинейно). Однако здесь важна лишь констатация конечности процедуры выбора x^k .

Ниже описаны два других способа выбора шагового множителя, приводимые для большей полноты изложения.

Замечание. Два описанных ниже правила более экономичны по вычислительным затратам, чем уже рассмотренный метод одномерной минимизации. Однако, эти правила менее универсальны, т.к. не могут использоваться в сочетании с любыми правилами выбора направлений локального спуска. Заметим, что в версиях 2 и 3 среды LocOpt они не реализованы, в LocOpt всегда используется правило одномерной минимизации.

Правила выбора шагового множителя без одномерной оптимизации

Существуют методы локальной оптимизации, в которых определение минимума в сечениях (2.10) на шаге не является обязательным, (например, для некоторых вариантов квазиньютоновских методов). При этом можно использовать правило Вулфа (2.6'), (2.8). Приведенный ниже алгоритм учитывает также возможное наличие двухсторонних ограничений на переменные $y \in D$. В этом случае может оказаться, что $\Pi' \mathbf{I} D = \emptyset$. При этом алгоритм остановится после сжатия исходного интервала поиска $[0, X]$ в s раз.

АЛГОРИТМ ВЫБОРА x^k ПО ПРАВИЛУ ВУЛФА

ШАГ 0. Задаем $0 < m < h < 1$, $0 < s \ll 1$, $\Theta_1 > 1$, $0 < \Theta_2 < 1$ и начальное $\tilde{x} > 0$. При $D = R^N$ принимаем $x^- = x^+ = 0$, полагаем $X = \tilde{x}$. При заданном D , определяющем параллелепипедные ограничения, полагаем $x^- = 0$, $x^+ = X$, где X соответствует наименьшему $x \geq 0$, приводящему точку $y^k + x d^k$ на границу D .

Выбираем начальное $x = \min\{\tilde{x}; X\}$.

ШАГ 1. Если $x^+ > 0$ и $x^+ - x^- < \sigma \cdot X$, выполняем останов. Если при этом $x^+ = X$ и $f(X) < f(x)$, то принимаем $x = X$. Окончательно полагаем $x^k = x$. Если же условие останова не выполнилось, переходим на шаг 2.

ШАГ 2. Если x принадлежит областям (2.6') и (2.8), то полагаем $x^k = x$ и выполняем останов, иначе — на шаг 3.

ШАГ 3. Если x не принадлежит области (2.8), полагаем $x^+ = x$, иначе, если x не принадлежит области (2.6'), полагаем $x^- = x$.

ШАГ 4. Если $x^+ = 0$, то выбираем $x > x^-$, например, по правилу $x = x^- \cdot \Theta_1$ и переходим на шаг 2, иначе — на шаг 5.

ШАГ 5. Выбираем новое $x \in (x^-; x^+)$, например, по правилу $x \equiv x^- \cdot \Theta_2 + x^+ \cdot (1 - \Theta_2)$ и переходим на шаг 1.

Дополнительно заметим, что в силу использования в приведенных выше алгоритмах проверки (2.5)–(2.6) или (2.5 ♢)–(2.6 ♢), выбор x^k в них сопряжен с многократным вычислением градиента функции $f(y)$. Ниже будут рассмотрены методы, лишенные этого недостатка.

Правила выбора шагового множителя без многократного вычисления производной

Если вычисления градиента функции дорогостоящи, выгодно использовать более простые и потенциально более экономичные алгоритмы выбора шагового множителя, в которых градиент измеряется только единожды. Здесь можно указать на правила Голдстейна и Армихо (см., например, [10]).

В правиле Голдстейна критерий из (2.5)–(2.6) заменяется на требование

$$x^k \in \bar{\Pi}_1(1-m), 0 < m < 1/2, \\ \bar{\Pi}_1(1-m) = \{x \geq 0: f(y^k + xd^k) \geq f(y^k) + x \cdot (1-m)(\nabla f(y^k), d^k)\},$$

а вместо (2.7) используется

$$x^k \in \Pi_2(m), 0 < m < 1/2.$$

Таким образом, шаговый множитель выбирается из условия

$$x^k \in \bar{\Pi}_1(1-m) \cap \Pi_2(m),$$

проверка которого не требует дополнительных вычислений градиента функции f .

Несколько подробнее остановимся на правиле Армихо, так же, как правило Голдстейна, использующем лишь однократное вычисление градиента функции при выборе x^k . Рассматриваемое правило имеет очень простую вычислительную реализацию.

ВЫЧИСЛИТЕЛЬНАЯ РЕАЛИЗАЦИЯ ПРАВИЛА АРМИХО заключается в определении x^k как наибольшего из значений $x_s = h^s \cdot X$ для $s = 0, 1, 2, \dots$, при котором выполняется условие

$$x \in \Pi_2(m), 0 < m < 1/2.$$

Здесь $X > 0$ играет роль первого пробного значения x и, естественно, его выбор оказывает заметное влияние на реальную «экономичность» алгоритма выбора шагового множителя. Заметим, что вне зависимости от выбора X алгоритм выбора x^k всегда будет конечен, поскольку (см., например, [10]) множество $\Pi_2(m)$ обязательно содержит в себе отрезок $[0, d]$ при некотором $d > 0$ и при некотором конечном s одна из точек $h^s \cdot X$ в него обязательно попадет.

Выбор значения X может быть основан на следующем подходе. При использовании правила Армихо в методах Ньютона–Рафсона (см. п. 2.6.1), очевидно, следует выбрать $X = 1$, поскольку вектор направления d^k нужным образом масштабирован.

В других методах можно использовать специальные приемы выбора X , например, следуя [10], можно ввести некоторое вспомогательное $\tilde{x} > 0$ и, используя на луче $\{y^k + xd^k : x > 0\}$ значения $j(0) = f(y^k)$, $j'(0) = (\nabla f(y^k), d^k)$, $j(\tilde{x}) = f(y^k + \tilde{x}d^k)$, определить точку X^* минимума построенной квадратичной интерполяции, выбрав затем $X = X^*$.

2.5. Градиентные методы и метод Ньютона, оценки скорости сходимости

Градиентные методы предельно просты. Их можно описать общим итерационным соотношением (2.3), имеющим вид $y^{k+1} = y^k + x^k d^k$, где направление смещения из точки y^k совпадает с направлением *антиградиента* $d^k = -\nabla f^k$.

В этом направлении дифференцируемая функция $f(y)$ локально (в достаточно малой окрестности точки y^k) убывает быстрее всего, т.к. ее производная $\nabla f(y^k)/\|v^k\|$ в точке y^k , вычисленная в некотором направлении v^k ($\|v^k\|=1$), может быть представлена в виде скалярного произведения $\nabla f(y^k)/\|v^k\| = (\nabla f(y^k), v^k)$, которое достигает своего минимума именно в направлении антиградиента.

Заметим, что в общем случае направление антиградиента в точке y^k , очевидно, не совпадает с направлением на локальный минимум (рис. 2.4). Более того, даже если такое совпадение обнаруживается, то это свойство, вообще говоря, не будет инвариантно по отношению к растяжениям пространства переменных.

Существует несколько модификаций градиентного метода, различающихся правилом выбора величины смещения x^k в направлении антиградиента. Наиболее простой способ состоит в выборе постоянного $x^k = a > 0$, но при этом возникает проблема подбора его величины для обеспечения сходимости. Кроме того, определение шагового множителя возможно по одной из схем, рассмотренных в разделе 2.4, например, на основе правила (2.9), или правила Вулфа (2.9') или же по правилу Армихо. Если в (2.9) считать $h \rightarrow 0$, то приходим к вычислительно точно не реализуемому, но удобному для теоретического анализа варианту градиентного метода, так называемому методу *наискорейшего градиентного поиска*.

Приведем правило выполнения шага в этом методе в том случае, когда в задаче оптимизации отсутствуют функциональные ограничения из (1.2), то есть допустимое множество $Y=D$. Применительно к этой задаче в методе наискорейшего градиентного поиска величина шагового множителя x^k определяется из условия (2.11).

$$f(y^k + x^k d^k) = \min \{ f(y^k + xd^k) : x \geq 0, y^k + xd^k \in Y=D \}. \quad (2.11)$$

Таким образом, x — величина смещения вдоль d^k , определяется минимумом функции f на пересечении множества Y и луча $y^k + x d^k$, где $x \geq 0$. Как уже отмечалось ранее, вычислительной реализацией правила (2.11) следует считать метод «аккуратного» одномерного поиска (см. раздел 2.4), основанный на правиле (2.9) при $0 < h < 1$.

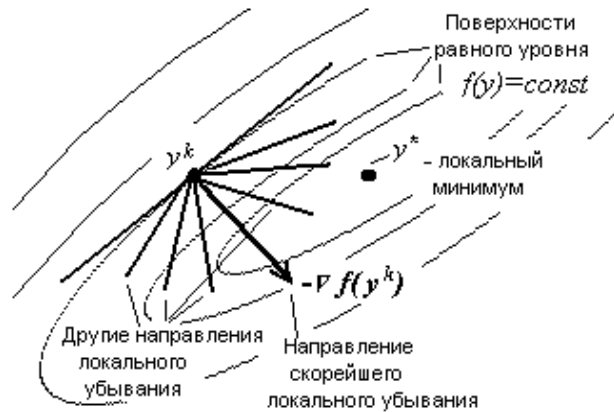


Рис. 2.4. Отличие направления антиградиента от направления на локальный минимум

Еще один взгляд на градиентные методы: они основаны на локальной линейной модели поведения функции $f(y)$ в окрестности точки y^k последнего испытания (рис. 2.5). Именно для линейной модели направление антиградиента является наилучшим с точки зрения задачи поиска минимума (для квадратичной модели это уже не так). Заметим, что методом используется не только локальная линейная модель. Выбор величины смещения по правилу (2.11) неявно предполагает нелинейность функции, особенно в задаче без ограничений.

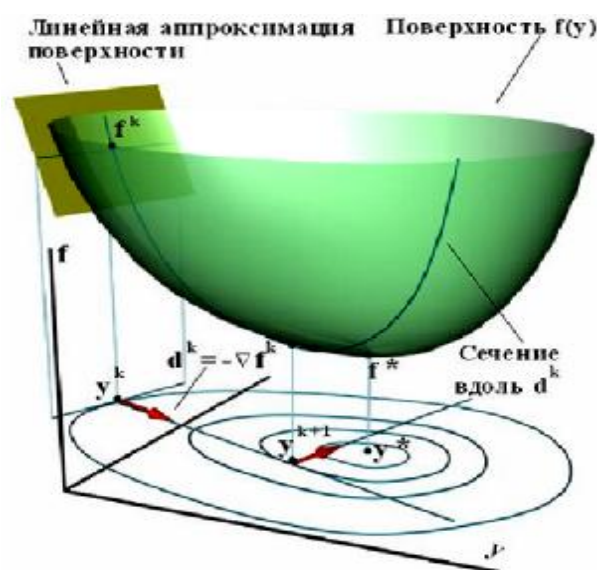


Рис. 2.5. Наискорейший градиентный поиск

Следует заметить, что направления смещения на двух последовательных шагах d^k и d^{k+1} взаимно ортогональны, если решение вспомогательной задачи (2.11) достигается во внутренней точке области поиска. Это следует из условий экстремума для (2.11) в форме теоремы Лагранжа.

Рассмотрим процедуры градиентного поиска для задач безусловной оптимизации (2.1), когда $D = R^N$. Их модификации для случая допустимого множества вида $Y = D$ из (2.2) приведена в п.3.1.1.

Сходимость процедур градиентного поиска может быть доказана при достаточно слабых предположениях о функции, минимум которой ищется.

Теорема. Пусть в задаче (2.1) функция $f(y)$ непрерывно дифференцируема, ограничена снизу и ее градиент удовлетворяет условию Липшица с некоторой константой L , т.е. $\forall y', y'' \|\nabla f(y') - \nabla f(y'')\| \leq L \cdot \|y' - y''\|$. Тогда метод градиентного поиска для любой начальной точки y^0 строит последовательность y^k такую, что $\|\nabla f(y^k)\| \rightarrow 0$ для $k \rightarrow \infty$ при следующих вариантах выбора шагового множителя:

- a) при $x^k = a$, $0 < a < 1/L$;
- b) при использовании правила (2.11) — наискорейший градиентный поиск;
- c) при использовании правила (2.9);
- d) при использовании правила Вулфа (2.9);
- e) при использовании правила Армихо.

При определенных дополнительных предположения относительно функции f из утверждения теоремы будет следовать сходимость y^k к точке минимума y^* .

Сам факт сходимости еще не говорит об эффективности метода. Ряд аналитических оценок показывают, что эффективность метода наискорейшего градиентного поиска в общем случае достаточно низка. Рассмотрим его работу на квадратичных функциях вида

$$f(y) = (y^T \Gamma y) / 2 + c^T y, \quad \Gamma^T = \Gamma \quad (2.12)$$

с положительно определенной матрицей Γ (т.е. строго выпуклых).

Известно, что на таких функциях метод наискорейшего градиентного поиска не обладает, в общем случае, конечной сходимостью (т.е. не определяет точку минимума за конечное число итераций), а порождает последовательность точек, сходящуюся к точке минимума y^* со скоростью геометрической прогрессии, знаменатель которой может быть близок к единице.

Теорема. Для квадратичной функции (2.12) с симметричной положительно определенной матрицей метод наискорейшего градиентного поиска сходится из любой

начальной точки y^0 со скоростью геометрической прогрессии со знаменателем, не превосходящим значения q . При этом справедливы следующие оценки:

$$Sa = a(y^0), T > 0 : 0 \leq a \leq q = (I_{\min}/I_{\max} - 1)^2 / (I_{\min}/I_{\max} + 1)^2,$$

$$f(y^k) - f(y^*) \leq a^k (f(y^0) - f(y^*)),$$

$$\|y^k - y^*\| \leq T a^{k/2} \|y^0 - y^*\|,$$

где I_{\min} и I_{\max} — минимальное и максимальное собственные числа матрицы вторых производных $\nabla^2 f(y) = G$.

Отметим, что отношение Канторовича q , определяющее оценку скорости сходимости, непосредственно зависит от степени обусловленности $\lambda_{\max} / \lambda_{\min}$ матрицы Гессе $\nabla^2 f(y^*)$.

Замечание. Утверждения теоремы об оценке скорости сходимости метода наискорейшего градиентного спуска асимптотически остаются, в определенном смысле, справедливы и для неквадратичного случая. Например, можно доказать, что если f дважды непрерывно дифференцируема, существует минимум y^* , Гессиан $\nabla^2 f(y^*)$ положительно определен и $\lambda_{\min}, \lambda_{\max}$ — крайние значения его собственных чисел, то верхний предел отношения $(f^{k+1} - f(y^*)) / (f^k - f(y^*))$ при $k \rightarrow \infty$ (если $\forall k f^k \neq f(y^*)$) не превосходит приведенного выше значения q .

Вернемся к анализу квадратичного случая. Из оценок теоремы о скорости сходимости следует, что конечная сходимость из любой начальной точки y^0 возможна только при $q=0$ ($I_{\min}=I_{\max}$), когда поверхности равного уровня функции f являются сферами.

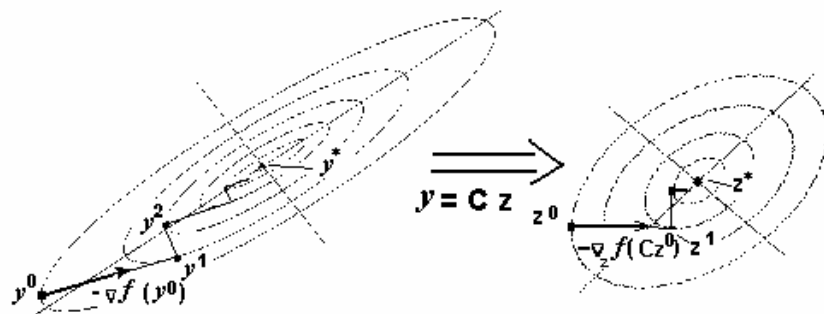


Рис. 2.6. Влияние масштабирования на поведение и скорость сходимости наискорейшего градиентного поиска

Если же поверхности равного уровня сильно вытянуты (рис. 2.6), что соответствует $I_{\min} \ll I_{\max}$, то значение q в оценке скорости сходимости будет близко к единице, и скорость сходимости к решению может оказаться чрезвычайно низкой, за исключением точек y^0 , лежащих на главных осях эллипсоидов $f(y) = \text{const}$ (в этих точках $a(y^0) = 0$).

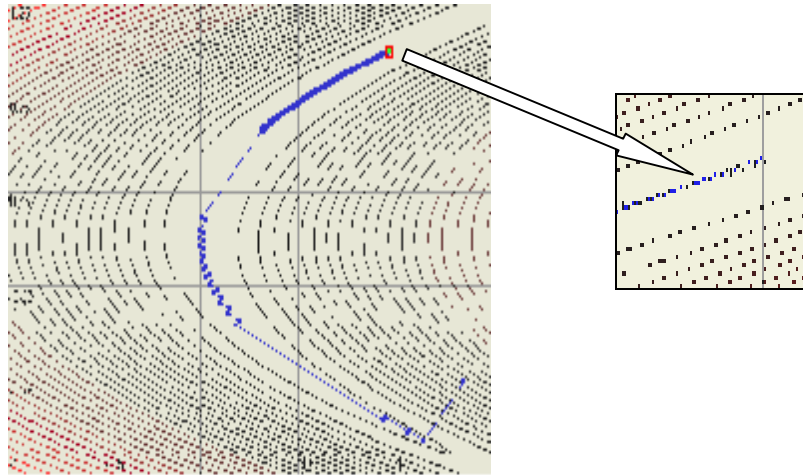


Рис. 2.7. Траектория наискорейшего градиентного поиска на функции Розенброка (2000 итераций)

На рис. 2.7 приведен построенный в ЛосОрт пример, показывающий медленную сходимость метода наискорейшего градиентного поиска для функции овражной структуры.

Таким образом, высокая скорость сходимости градиентного метода может быть обеспечена только за счет предварительного масштабирования задачи, т.е. выполнения такой линейной замены переменных $y = Cz$, которая приводила бы (в новых переменных) к выполнению условия $I_{\min} \approx I_{\max}$.

Направление антиградиента не инвариантно по отношению к линейным заменам переменных. Если выполнен переход к переменным z : $y = Cz$ (где C — матрица преобразования), то градиент функции в новых переменных z , очевидно, изменится и может быть вычислен как

$$\tilde{N}_z f(Cz) = C^T \tilde{N} f(y). \quad (2.13)$$

Если перевести антиградиентное направление, вычисленное в пространстве z , в направление в пространстве старых переменных y , то получим скорректированное направление поиска

$$\bar{d}^k = C(-\tilde{N}_z f(Cz^k)) = CC^T(-\tilde{N} f(y^k)). \quad (2.14)$$

Потребуем, чтобы \bar{d}^k было направлением на локальный минимум квадратичной функции (2.12). Возникающая матрица CC^T для коррекции направления легко вычисляется для строго выпуклых квадратичных функций вида (2.12). Действительно, пусть в точке y^k для $f(y)$ измерено значение $f^k = f(y^k)$, градиент $\nabla f^k = \nabla f(y^k)$ и матрица вторых производных $\Gamma_k = \nabla^2 f(y^k) = \Gamma$. В силу равенства нулю всех производных выше второго порядка $f(y)$ из (2.12) совпадает со своей квадратичной аппроксимацией $P^k(y)$, построенной по измерениям f^k , ∇f^k , Γ_k , выполненным в точке y^k

$$P^k(y) = (y - y^k)^T \Gamma_k (y - y^k) / 2 + (\tilde{N}f^k, (y - y^k)) + f^k. \quad (2.15)$$

Условие, определяющее y^* — точку минимума для $P^k(y)$, примет вид

$$\tilde{N}P^k(y^*) = \Gamma_k (y^* - y^k) + \tilde{N}f^k = 0, \quad (2.21)$$

откуда $y^* - y^k = (\Gamma_k)^{-1}(-\nabla f(y^k))$. Сравнивая полученное направление с (2.14), видим, что в качестве матрицы преобразования в (2.14) можно использовать $CC^T = (\Gamma_k)^{-1} = (\nabla^2 f(y^k))^{-1}$.

Если бы $f(y)$ была произвольной дважды непрерывно дифференцируемой функцией, то в (2.15) квадратичная аппроксимация $P^k(y)$ уже не совпадала бы с исходной функцией, а условие (2.21) определяло бы лишь стационарную точку для этой аппроксимации. Если именно в этой точке проводить очередное измерение локальных характеристик функции f (значения, градиента и матрицы вторых производных), приняв ее за y^{k+1} , получим *метод Ньютона*

$$\nabla^2 f(y^k)(y^{k+1} - y^k) = -\nabla f(y^k),$$

который можно формально записать в виде явного итерационного соотношения

$$y^{k+1} = y^k + (\nabla^2 f(y^k))^{-1}(-\tilde{N}f(y^k)) \quad (2.22)$$

(направление шага $d^k = (\nabla^2 f(y^k))^{-1}(-\tilde{N}f(y^k))$, коэффициент длины шага $\alpha^k = 1$). Полезно обратить внимание на то, что этот метод, выполняя на каждом шаге преобразование пространства переменных. Построенное им направление d^k соответствует антиградиентному направлению функции f , если его вычислить в преобразованном пространстве.

На метод Ньютона можно посмотреть с другой точки зрения. А именно, правило итерации (2.22) основано на использовании квадратичной модели поведения функции $f(y)$, стационарная точка которой ищется. Использование квадратичной модели приводит к отказу от использования направления антиградиента функции, вычисленного в пространстве исходных переменных, и применению вместо него скорректированного антиградиентного направления, приводящего в результате шага в стационарную точку текущей квадратичной аппроксимации функции.

Геометрическая интерпретация правила выбора направления поиска в методе Ньютона для выпуклой квадратичной функции приведена на рис. 2.8.

Что можно сказать о сходимости метода Ньютона? Ответ дает следующая теорема.

Теорема. Для дважды непрерывно дифференцируемых функций (т.е. для $f \in C^2(D)$) с невырожденной матрицей $\nabla^2 f(y^*)$ существует такая ε -окрестность стационарной точки y^* функции $f(y)$, что для любой начальной точки y^0 из этой окрестности метод Ньютона будет

сходиться сверхлинейно, а при выполнении в этой окрестности для матриц Гессе условия Липшица, — квадратично.

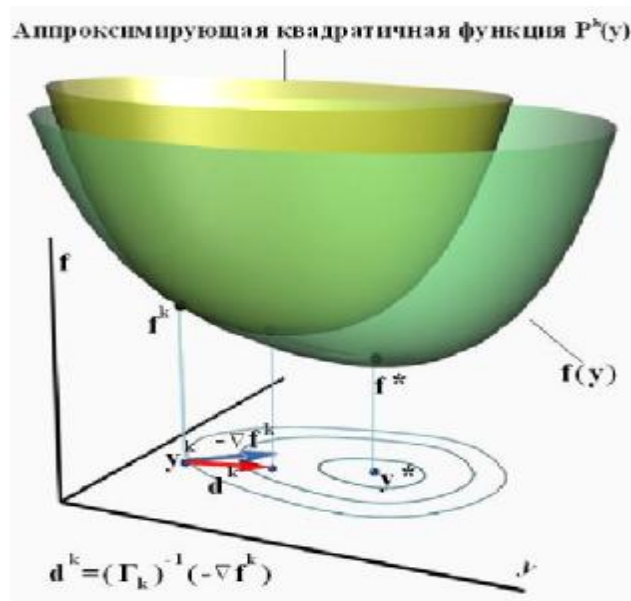


Рис. 2.8. Выбор направления в методе Ньютона по квадратичной аппроксимации

Определение. *Линейной* называют сходимость, при которой некоторая мера Δ^k близости к решению убывает по закону геометрической прогрессии. Таким образом Δ^k и Δ^{k+1} являются бесконечно малыми одного порядка при $k \rightarrow \infty$.

Линейная сходимость по значению функции характерна для методов градиентного поиска.

Определение. *Сверхлинейной* называют сходимость более быструю, чем у любой геометрической прогрессии. При сверхлинейной сходимости Δ^{k+1} имеет более высокий порядок малости по отношению к Δ^k при $k \rightarrow \infty$.

Таким образом, можно сказать, что метод *сходится к y^* сверхлинейно* в пространстве исходных переменных, если при достаточной близости начальной точки y^0 к y^* $\exists k > 0$ и последовательность чисел a_{k+1}, \dots, a_{k+m} из интервала $[0,1)$, стремящаяся к 0 при $m \rightarrow \infty$, что $\forall m > 0$ будет выполнено неравенство

$$\|y^{k+m} - y^*\| \leq a_{k+1} \dots a_{k+m} \|y^k - y^*\|.$$

Определение. *Квадратичной* называют сходимость, при которой некоторая мера погрешности на следующем шаге Δ^{k+1} является величиной не менее чем второго порядка малости по отношению к Δ^k , т.е. $\exists T \geq 0$, что $\Delta^{k+1} \leq T \cdot (\Delta^k)^2$, начиная с некоторого k .

Таким образом, метод *сходится к y^* квадратично* по исходным переменным, если $\exists \epsilon > 0$ и $T > 0$, что $\forall y^0$ из окрестности $\|y^0 - y^*\| < \epsilon$:

$$\|y^{k+1} - y^*\| \leq T \|y^k - y^*\|^2.$$

Таким образом, из последней теоремы следует, что при достаточно общих условиях метод Ньютона обладает тем, чего лишены градиентные методы — высокой скоростью сходимости. Принцип квадратичной аппроксимации является фундаментальной идеей, лежащей в основе многих эффективных методов. Однако сходимость гарантируется только в некоторой (обычно заранее не известной!) окрестности решения. Вне этой окрестности метод Ньютона может вообще расходиться. Кроме того, итерация (2.22) требует вычисления вторых производных и решения линейных систем $\Gamma_k d^k = -\nabla f^k$ относительно d^k ($\Gamma_k = \nabla^2 f(y^k)$). Существенно также то, что в прикладных задачах достаточно часто встречаются ситуации, когда в точках последовательности y^k матрица $\nabla^2 f(y^k)$ оказывается отрицательно определенной, знаконеопределенной или вырожденной. В последнем случае итерация (2.22) неприменима. Если же $\nabla^2 f(y^k)$ не вырождена, но не знакоположительна, то, как следует из приведенной выше теоремы, метод Ньютона может сходиться к стационарной точке функции f , не являющейся точкой минимума, а представляющей собой точку максимума или седловую точку. При этом направление шага может не являться направлением строгого локального убывания, поскольку знак произведения

$$(d^k, \nabla f^k) = -(\nabla f^k)^T \nabla^2 f(y^k) \nabla f^k$$

не определен. В этом случае метод Ньютона не является методом локального спуска.

Пример ситуации со сложным и несколько неожиданным поведением метода Ньютона в задаче с двумя переменными, вызванным изменением знакоопределенности матриц вторых производных вдоль траектории поиска, показан на рис. 2.9. Поясним этот рисунок. Характер зависимости функции от параметров представлен линиями равного уровня и соответствует функции

$$f(y_1, y_2) = 20(\cos(3 \cdot y_1) - y_2)^2 + (y_2 - 4y_1)^2.$$

Область локального минимума, отмеченного жирной точкой, охвачена замкнутыми линиями уровня. Начальная точка поиска $y^0 = (-1; 1,5)$. В этой точке матрица Гессе Γ_0 оказывается знаконеопределенной и метод переходит в стационарную (седловую) точку первой квадратичной аппроксимации, попадая на «дно оврага» минимизируемой функции. На следующих трех шагах, приводящих к спуску вдоль «оврага», матрицы Гессе Γ_k ($k = 1, 2, 3$) оказываются положительно определенными. На четвертом шаге точка y^k

($k = 4$) попадает в область со знакопеременными матрицами Гессе. В результате метод переходит в точку y^k ($k = 5$), являющуюся седловой для соответствующей квадратичной аппроксимации, возвращаясь вверх по «дну оврага» и вновь попадая в окрестность точки y^k ($k = 3$). Матрица Гессе опять оказывается положительно определена. Далее траектория совершает несколько колебаний так, что точки y^k последовательно оказываются в окрестностях точек y^k ($k = 5$) и y^k ($k = 3$).

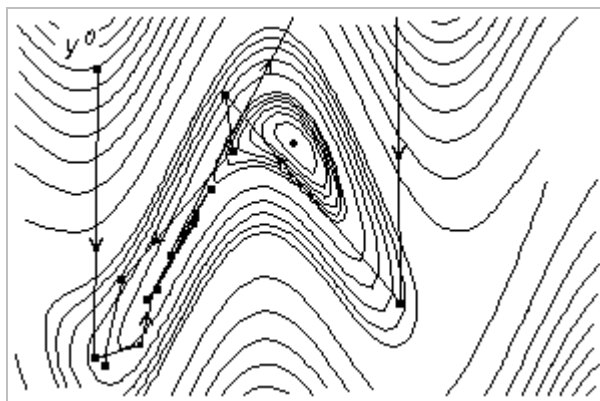


Рис. 2.9. Неожиданное поведение метода Ньютона

Приведенные на рис. 2.7, 2.9 простые примеры наглядно показывают, что для эффективного решения прикладных задач, характеризующихся плохим масштабированием, возможным вырождением и знаконеопределенностью матриц вторых производных, классические методы поиска локального экстремума в своем исходном виде являются малопригодными. Необходимы методы, сочетающие сходимость из любой начальной точки с высокой скоростью сходимости вблизи решения и сохраняющие свои свойства в ситуациях, характерных для прикладных задач.

2.6. Эффективные методы второго порядка

В этом разделе описаны методы безусловной локальной оптимизации, которые вычисляют в точке поиска y^k значения $f(y^k)$, $\nabla f(y^k)$, $\nabla^2 f(y^k)$, т. е. явно используют значения матриц вторых производных. Эти методы являются улучшенными модификациями метода Ньютона, расширяющими область его сходимости и включающими методы борьбы с нарушением знакоположительности матриц Гессе.

В своей исходной форме метод Ньютона с точки зрения задачи на безусловный локальный минимум обладает тремя существенными недостатками: возможной расходимостью для начальных точек, взятых вне некоторой окрестности стационарных точек задачи, неприменимостью при вырождении матрицы вторых производных минимизируемой функции и возможной сходимостью к точкам максимумов или иных

стационарных точек со знаконеопределенными матрицами Гессе. Эти недостатки могут быть преодолены за счет модификаций метода Ньютона.

Первая модификация связана с расширением области сходимости (глобализацией сходимости) за счет изменения правила выбора шага. Нужного эффекта можно добиться двумя способами: либо за счет локализации смещения точки y^{k+1} по отношению к y^k , либо за счет использования для выбора длины шага процедур его регулировки с использованием одной из стратегий одномерного поиска, рассмотренных в разделе 2.4. Например, можно использовать одномерную оптимизацию вдоль выбранного направления с использованием алгоритма «аккуратного» одномерного поиска или более экономичное по количеству измерений правило Армихо, выбрав первое пробное значение $x = a = 1$.

Локализация смещения в методе Ньютона связана с выбором точки y^{k+1} из условия решения задачи на минимум квадратичной аппроксимации $P^k(y)$ из (2.15) при дополнительном условии $\|y - y^k\| \leq d_k$, т.е.

$$y^{k+1} = \arg \min \{P^k(y) : \|y - y^k\| \leq d_k\}.$$

При этом должна быть задана некоторая стратегия выбора d_k . В настоящем учебно-методическом материале этот метод не рассматривается, в систему ЛосОрт он также не включен.

Локализацию смещения можно реализовать несколько иначе, с использованием штрафной добавки $(g_k / 2) \|y - y^k\|^2$ к квадратичной аппроксимации $P^k(y)$. Такой подход приведет к методу Левенберга-Марквардта [7]

$$y^{k+1} = y^k + (\nabla^2 f(y^k) + g_k E)^{-1} (-\nabla f(y^k)),$$

где E — единичная матрица. Направление шага в этом методе является смесью Ньютонова направления и направления антиградиента. Указанный метод в систему ЛосОрт также не включен.

Детальнее разберем подход, связанный с выбором коэффициента шага с помощью одномерного поиска.

2.6.1. Расширение области сходимости метода Ньютона за счет выбора шагового множителя

В методе Ньютона коэффициент длины шага $x^k \equiv 1$. В модифицированном методе (методе с регулировкой шага) x^k будем выбирать по алгоритму «аккуратного» одномерного поиска, описанному в пункте 2.4. Это приведет к сходимости из любой

начальной точки для достаточно широкого класса функций и сохранению высокой (обычно сверхлинейной) скорости сходимости в окрестности решения. Метод Ньютона с регулировкой шага на основе одномерной оптимизации называют *методом Ньютона–Рафсона*.

Рассмотрим свойства данного метода. Выберем специальный класс тестовых функций F , имеющих единственный минимум, часто используемый для изучения свойств методов локального поиска. В качестве F возьмем класс $F_{m,M}$ дважды непрерывно дифференцируемых функций, обладающих тем свойством, что $\exists m > 0$ и $M < \infty$, $m < M$, что $\forall y \in R^N, z \in R^N$,

$$m\|z\|^2 \leq z^T \nabla^2 f(y) z \leq M\|z\|^2. \quad (2.23)$$

Такие функции будут сильно выпуклы.

Можно показать, что условие (2.23) равносильно тому, что все собственные числа $I_1(y), \dots, I_N(y)$ матриц $\nabla^2 f(y)$ лежат между m и M . Для этого достаточно заменить матрицу Гессе ее разложением через ортогональную R и диагональную матрицы (с собственными числами на диагонали), а затем перейти к новым переменным $w = R^T z$.

Замечание. Условие (2.23) гарантирует положительную определенность матрицы $\nabla^2 f(y)$, достаточную для сходимости метода Ньютона к минимуму $f(y)$ из любой начальной точки, выбранной в достаточной близости от него. Однако из произвольно выбранной точки y^0 метод Ньютона для функции $f(y)$ из $F_{m,M}$ может не сходиться.

Таким образом, для метода Ньютона выбранный тестовый класс, с точки зрения сходимости из любой начальной точки, хорошим не является. Можно доказать, что метод Ньютона–Рафсона обладает на этом классе сходимостью из любой точки, это означает, что данный метод является улучшенной модификацией метода Ньютона.

Свойство. Любая функция $f(y)$ из класса $F_{m,M}$ имеет единственный минимум.

Это непосредственно следует из сильной выпуклости функции.

Свойство. Для функций $f(y)$ из класса $F_{m,M}$ существует взаимосвязь между ошибкой по координате и ошибкой по значению функции, выражаемая соотношением

$$0,5 m \|y - y^*\|^2 \leq f(y) - f(y^*) \leq 0,5 M \|y - y^*\|^2.$$

Справедливость сразу вытекает из (2.23) и разложения

$$f(y) - f(y^*) = (y - y^*)^T \nabla^2 f(q) (y - y^*)/2.$$

Свойство. Для функций $f(y)$ из класса $F_{m, M}$ существует взаимосвязь между ошибкой по значению функции и нормой градиента

$$m(1+m/M)(f(y) - f(y^*)) \leq \|\tilde{N}f(y)\|^2.$$

Поведение метода Ньютона с регулировкой шага на классе $F_{m, M}$ определяется следующей теоремой. Ее доказательство опирается на приведенные выше свойства.

Теорема. Метод Ньютона с регулировкой шага на функциях $f \in F_{m, M}$ для любой начальной точки y^0 порождает последовательность y_k , сходящуюся к точке минимума y^* со сверхлинейной скоростью.

Замечание. Если дополнительно потребовать от функции $f \in F_{m, M}$ существования непрерывных третьих производных или липшидовости вторых производных в окрестности y^* , можно доказать, что метод Ньютона–Рафсона будет сходиться квадратично.

Таким образом, модификация Ньютона–Рафсона расширяет область сходимости метода Ньютона.

2.6.2. Стратегии модификации матриц Гессе при нарушении их положительной определенности

Вторая модификация метода Ньютона связана с преодолением случаев отсутствия положительной определенности матрицы вторых производных. Следует обратить внимание на то, что при нарушении положительной определенности $\Gamma_k = \nabla^2 f(y^k)$ (а значит и $(\Gamma_k)^{-1}$, если она существует) направление смещения $d^k = (\Gamma_k)^{-1}(-\nabla f(y^k))$ может не быть направлением убывания. Действительно, производная функции f в точке y^k по направлению d^k оценивается следующим образом:

$$\langle \nabla f(y^k), d^k \rangle = (\tilde{N}f(y^k), d^k) = -(\tilde{N}f(y^k), (\Gamma_k)^{-1} \tilde{N}f(y^k)).$$

В рассматриваемом случае знак этого произведения не определен и может оказаться положительным. В этом случае метод Ньютона–Рафсона применить нельзя, т.к. при его использовании смещение вдоль d^k будет нулевым.

ПРОСТОЙ АЛГОРИТМ. Простейший выход из этого положения может состоять в проверке знака скалярного произведения $(\nabla f(y^k), d^k)$ на каждом шаге. Если окажется, что $(\nabla f(y^k), d^k) < 0$, то выполняется обычный шаг по методу Ньютона–Рафсона, если же $(\nabla f(y^k), d^k) \geq 0$, то проводится замена направления поиска на антиградиентное направление $d^k = -\nabla f(y^k)$ и выполняется шаг по методу наискорейшего градиентного спуска.

Замечание. Рассмотренная простая стратегия часто не оправдывает себя, поскольку при наличии «овражности» в области знаконеопределенности матриц Гессе простой алгоритм вырождается в метод наискорейшего градиентного поиска, очень плохо работающий в «оврагах».

Таким образом, необходима более гибкая модификация метода. Необходимо как-то использовать матрицу Γ_k , хотя непосредственно ее применить нельзя. Основная идея, на основе которой выполняется модификация матриц, состоит в том, чтобы заменить Γ_k на достаточно близкую к ней (в смысле некоторой нормы) положительно определенную матрицу $\bar{\Gamma}_k$ и затем использовать ее в итерационном соотношении метода Ньютона с регуляризацией шага

$$y^{k+l} = y^k + x^k \cdot d^k; \quad d^k = (\bar{\Gamma}_k)^{-1}(-\tilde{N}f(y^k)) \quad (2.24)$$

Заметим, что вместо приведенного в (2.24) правила выбора шагового множителя, соответствующего методу Ньютона–Рафсона, можно использовать правило Армихо с параметром $a = 1$.

Переход от Γ_k к положительно определенной матрице $\bar{\Gamma}_k$ обычно выполняется с помощью факторизации Γ_k , т.е. разложения ее в произведение матриц определенного вида.

Наиболее естественным представляется использование *спектрального разложения*. Определим для Γ_k набор собственных чисел I_1, \dots, I_N и систему ортонормированных собственных векторов u^1, \dots, u^N . Тогда возможно следующее представление

$$G_k = I_1 u^1 (u^1)^T + \dots + I_N u^N (u^N)^T = ULU^T,$$

где матрица U составлена из вектор-столбцов u^1, \dots, u^N , L — диагональная матрица с числами I_1, \dots, I_N по диагонали. Такое представление матрицы называется *спектральным разложением*. Если положительная определенность Γ_k нарушена, то существует $I_i \leq 0$.

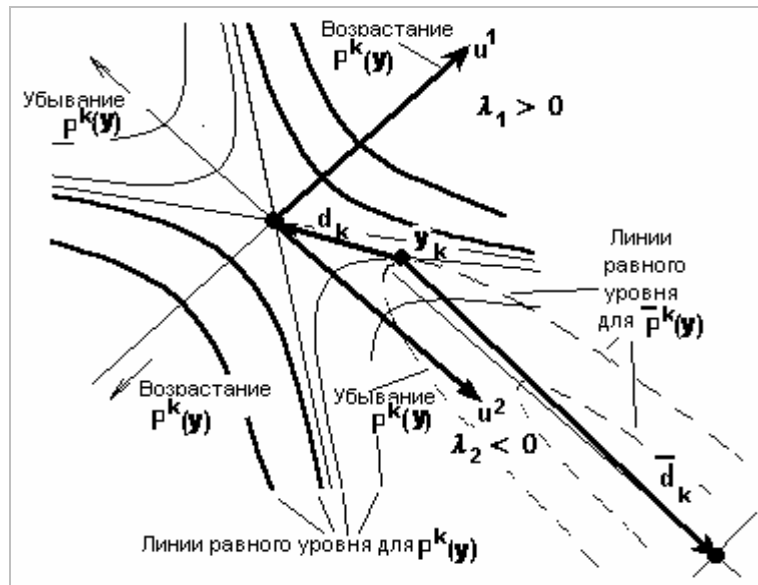


Рис. 2.10. Изменение линий равного уровня квадратичной функции при замене матрицы на положительно определенную

Матрица $\bar{\Gamma}_k$ строится так, что у нее сохраняются все собственные векторы u_1, \dots, u_N , а собственные числа заменяются на новые \bar{l}_i так, что

$$\bar{l}_i = \begin{cases} l_i, & l_i > d, \\ d, & l_i \leq d, \end{cases} \quad (2.25)$$

где d — малое положительное число. После этого полагается

$$\bar{G}_k = U \bar{L} U^T, \quad (2.26)$$

где в диагональной матрице \bar{L} на диагонали используются числа $\bar{l}_1, \dots, \bar{l}_N$.

При этом подпространство локальной положительной кривизны функции $f(y)$ сохраняется, а подпространство отрицательной кривизны функции заменяется подпространством малой положительной кривизны. Если построить квадратичную аппроксимацию функции $f(y)$ по результатам ее испытания в точке y^k и затем заменить в ней матрицу вторых производных Γ_k на модифицированную матрицу (2.25)–(2.26), то произойдут качественные изменения в структуре аппроксимации.

На рис. 2.10, на примере пространства двух переменных, показаны изменения в линиях равного уровня квадратичной аппроксимации $P^k(y)$ после замещения знаконеопределенной матрицы Γ_k положительно определенной $\bar{\Gamma}_k$. На этом рисунке видно, как направление d^k метода Ньютона, приводящее в стационарную точку поверхности $P^k(y)$, заменяется новым направлением \bar{d}^k , приводящим в минимум измененной аппроксимации $\bar{P}^k(y)$.

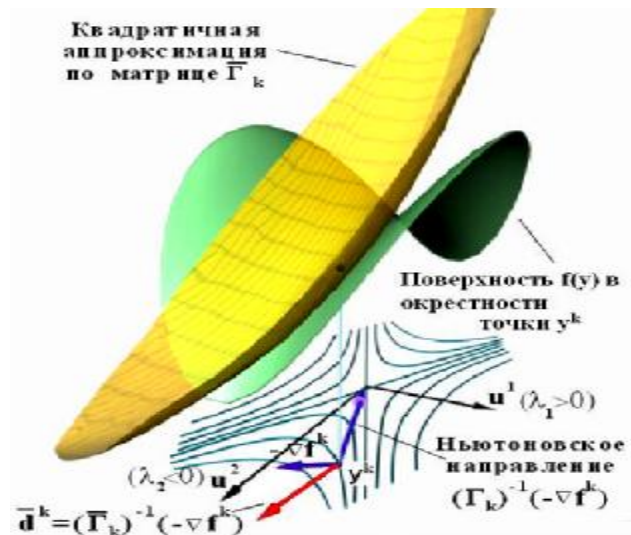


Рис. 2.11. Изменение вида аппроксимирующей поверхности в случае знаконеопределенной матрицы Гессе

Для создания наглядных представлений об изменении характера аппроксимирующей поверхности при замене Γ_k на положительно определенную $\bar{\Gamma}_k$, полезно обратиться к иллюстрациям, представленным на рис. 2.11–2.12.

Поведение функции $f(x)$, представленное на рис. 2.11, соответствует линиям равного уровня, показанным на рис. 2.10. Следует обратить внимание на то, что кривизна модифицированной аппроксимирующей поверхности $\bar{P}^k(y)$, построенной по измененной матрице Γ_k , рассматриваемая в направлении u^1 положительной локальной кривизны поверхности $f(y)$, совпадает с локальной кривизной самой $f(y)$ в этом направлении. В то же время, в направлении u^2 отрицательная кривизна заменена малой положительной.

Несколько иное соответствие между первоначальной и модифицированной аппроксимациями возникает в том случае, когда матрица Γ_k отрицательно определена. В этом случае все собственные направления матрицы Γ_k трансформируются в направления малой положительной кривизны (рис. 2.12). Это приводит к замене направления Ньютона d^k на новое — \bar{d}^k , ориентированное на точку минимума измененной аппроксимации.

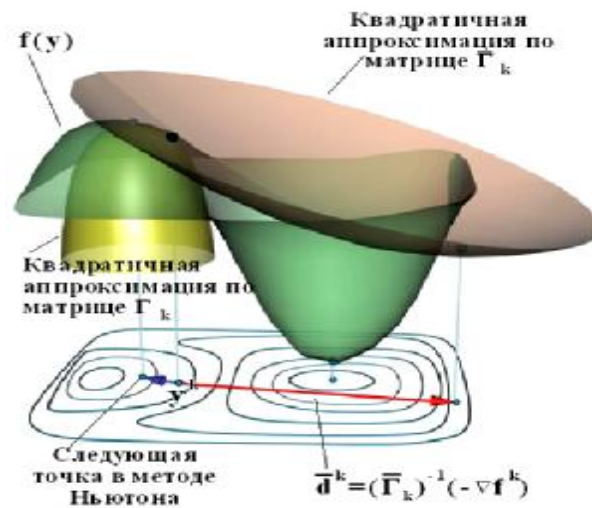


Рис. 2.12. Влияние модификации матрицы при ее начальной отрицательной определенности

Таким образом, метод коррекции матрицы на основе спектрального разложения весьма нагляден и прост для понимания. Однако этот подход имеет один существенный недостаток — большие затраты по вычислениям, связанные с поиском собственных векторов и чисел симметричной матрицы. В общем случае вычислительный процесс построения спектрального разложения для симметричной матрицы является бесконечно-шаговым и требует для выполнения одной итерации объема вычислений порядка $O(N^3)$.

При разработке вычислительных методов оптимизации для построения положительно определенных матриц Γ_k по исходным матрицам Γ_k вместо спектрального разложения часто используется модифицированное разложение Холецкого [3], вычислительная реализация которого проще и требует лишь конечного числа операций.

В теории матриц известно, что для любой симметричной положительно определенной матрицы Γ существует нижняя треугольная матрица L с единичной диагональю и диагональная матрица D с положительной диагональю, что справедливо *разложение Холецкого* $\Gamma = L^T D L$, т.е.

$$\begin{pmatrix} \gamma_{11} & \gamma_{12} & \dots & \gamma_{1N} \\ \gamma_{12} & \gamma_{22} & \dots & \gamma_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ \gamma_{1N} & \gamma_{2N} & \dots & \gamma_{NN} \end{pmatrix} = \begin{pmatrix} 1 & 0 & \dots & 0 \\ l_{12} & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ l_{1N} & l_{2N} & \dots & 1 \end{pmatrix} \begin{pmatrix} d_{11} & 0 & \dots & 0 \\ 0 & d_{22} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & d_{NN} \end{pmatrix} \begin{pmatrix} 1 & l_{12} & \dots & l_{1N} \\ 0 & 1 & \dots & l_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{pmatrix}, \quad (2.27)$$

где $d_{11}, \dots, d_{NN} > 0$.

Отсюда вытекает, что

$$g_{11} = d_{11}, \quad g_{1j} = d_{11} l_{1j}, \quad (j > 1), \quad (2.28)$$

$$g_{ii} = d_{ii} + (l_{1i})^2 d_{11} + (l_{2i})^2 d_{22} + \dots + (l_{(i-1)i})^2 d_{(i-1)(i-1)}, \quad (i = 2, \dots, N), \quad (2.29)$$

$$g_{ij} = (l_{1i})(l_{1j}) d_{11} + (l_{2i})(l_{2j}) d_{22} + \dots + (l_{(i-1)i})(l_{(i-1)j}) d_{(i-1)(i-1)} + (l_{ij}) d_{ii}, \quad (2.30)$$

для $i < j \leq N$.

Из (2.28)–(2.30) легко получить формулы для определения коэффициентов разложения Холесского. Расчет их значений производится строка за строкой для матриц D и L . Порядок вычисления этих коэффициентов удобно пояснить следующей диаграммой:

$$d_{11} \text{ } P \text{ } l_{12}, l_{13}, l_{14}, \dots; d_{22} \text{ } P \text{ } l_{23}, l_{24}, \dots; d_{33} \text{ } P \text{ } l_{34}, l_{35}, \dots$$

Получим

$$d_{11} = \gamma_{11}, \quad l_{1j} = (1/d_{11})\gamma_{1j}, \quad (j = 2, \mathbf{K}, N), \quad (2.31)$$

$$d_{ii} = g_{ii} - \sum_{s=1}^{i-1} (l_{si})^2 d_{ss}, \quad (i > 1),$$

$$l_{ij} = \left(g_{ij} - \sum_{s=1}^{i-1} (l_{si})(l_{sj})d_{ss} \right) / d_{ii}, \quad (i > 1, j = i+1, \mathbf{K}, N). \quad (2.32)$$

Построим теперь *модифицированное разложение Холесского* для произвольной (не обязательно положительно определенной) симметричной матрицы Γ . В процессе разложения будем производить коррекцию получаемых элементов d_{ij} так, чтобы модифицированные элементы \bar{d}_{ij} удовлетворяли условию

$$\bar{d}_{ij} \geq d > 0. \quad (2.33)$$

Это обеспечит положительность с «запасом» элементов модифицированной диагональной матрицы \bar{D} . Отметим, что условие (2.33) не может являться единственным условием модификации. Действительно, близость к нулю некоторых модифицированных элементов \bar{d}_{ij} при их дальнейшем использовании в (2.32) может привести к лавинообразному росту элементов l_{ij} при вычислениях. При обычном разложении Холесского это невозможно, т.к. из (2.29) вытекает, что

$$(l_{si})^2 d_{ss} \leq g_{ii} \leq \max \{ g_{ii} : i = 1, \dots, N \}, \quad (s=1, \dots, i).$$

Обозначим через $g^* = \max \{ g_{ii} : i = 1, \dots, N \}$ и введем $b^2 \geq g^*$. Наложим требование, чтобы для модифицированных элементов разложения выполнялось

$$d_{ss} \leq b^2; \quad ((l_{si})^2 \bar{d}_{ss}) \leq b^2 \quad (s = 1, \dots, N, i > s). \quad (2.34)$$

Выполним необходимую модификацию за счет изменения только диагональных элементов матрицы Γ . Обозначим через Δ_i добавки к элементам g_{ii} . Тогда, согласно (2.33)–(2.34), должно выполняться:

$$\begin{aligned} \bar{d}_{ii} &= g_{ii} + \Delta_{ii} - \sum_{s=1}^{i-1} (\bar{l}_{si})^2 \bar{d}_{ss} \geq d > 0, \\ \max_{j=i+1, \mathbf{K}, N} (\bar{l}_{ij})^2 \bar{d}_{ii} &= \\ &= \left(\left(\max_{j=i+1, \mathbf{K}, N} \left(g_{ij} - \sum_{s=1}^{i-1} (\bar{l}_{si})(\bar{l}_{sj})\bar{d}_{ss} \right)^2 \right) / \left(g_{ii} + \Delta_{ii} - \sum_{s=1}^{i-1} (\bar{l}_{si})^2 \bar{d}_{ss} \right) \right) \leq b^2. \end{aligned}$$

Если обозначить

$$c_i^2 = \max_{j=i+1, \mathbf{K}, N} \left(g_{ij} - \sum_{s=1}^{i-1} (\bar{l}_{si})(\bar{l}_{sj})\bar{d}_{ss} \right)^2, \quad (2.35)$$

$$\tilde{d}_{ii} = g_{ii} - \sum_{s=1}^{i-1} (\bar{l}_{si})^2 \bar{d}_{ss}, \quad (2.31)$$

то

$$\Delta_i = \max \{0; c_i^2 / b^2 - \tilde{d}_{ii}; d - \tilde{d}_{ii}\}.$$

Это соответствует выбору

$$\bar{d}_{ii} = \max \{ \tilde{d}_{ii}; c_i^2 / b^2; d \}, \quad (2.32)$$

$$\bar{l}_{ij} = \left(\gamma_{ij} - \sum_{s=1}^{i-1} (\bar{l}_{si})(\bar{l}_{sj}) \bar{d}_{ss} \right) / \bar{d}_{ii}. \quad (2.33)$$

Элементы (2.32), (2.33) определяют модифицированные матрицы \bar{L} , \bar{D} . В качестве положительно определенной матрицы-приближения для Γ используется $\bar{\Gamma} = \bar{L}^T \bar{D} \bar{L}$.

Заметим, что сумма квадратов поправок к элементам матрицы Γ равна $\Delta_1^2 + \dots + \Delta_N^2$. Для уменьшения этой величины в [3] рекомендуется выбирать

$$b^2 = \max \{ g^*; x / (N^2 - 1)^{1/2}; e_M \}, \quad (2.34)$$

где e_M — наименьшее положительное вещественное число в машинной арифметике,

$$g^* = \max \{ g_{ii} : i=1, \mathbf{K}, N \}, \quad x = \max \{ |g_{ij}| : 1 \leq i \leq N, 1 \leq j \leq N, i \neq j \}. \quad (2.35)$$

Приведем пошаговое описание метода Ньютона с регулировкой шага и модификацией матрицы вторых производных на положительную определенность с использованием модифицированного преобразования Холесского.

АЛГОРИТМ МОДИФИЦИРОВАННОГО МЕТОДА НЬЮТОНА.

ШАГ 0. Задаются начальная точка y^0 , параметры выбора коэффициента одномерного шага $0 < m < h < 1$, $0 < \sigma < 1$; параметр останова e и параметр модификации $d > 0$. Полагается $k = 0$.

ШАГ 1. Вычисляются $f^k = f(y^k)$, $\tilde{N}f^k = \tilde{N}f(y^k)$, $G_k = \nabla^2 f(y^k)$.

ШАГ 2. Вычисляется b^2 по формулам (2.34), (2.35). Строятся матрицы модифицированного разложения Холесского \bar{L}_k , \bar{D}_k для матрицы G_k .

ШАГ 3. Определяется модифицированное направление ньютоновского шага $\bar{d}^k = (G_k)^{-1}(-\nabla f^k)$ путем последовательного решения двух систем линейных уравнений с треугольными матрицами

$$\begin{aligned} (\bar{L}_k)^T v^k &= -\tilde{N}f^k, \\ (\bar{D}_k \bar{L}_k^T) \bar{d}^k &= v^k. \end{aligned}$$

ШАГ 4. Вычисляется x^k по алгоритму выбора коэффициента одномерного шага $x^k \in \Pi$ из (2.9) или правилу Армихо. Полагается $y^{k+1} = y^k + x^k \bar{d}^k$

ШАГ 5. Вычисляется $f^{k+1} = f(y^{k+1})$, $\nabla f^{k+1} = \nabla f(y^{k+1})$, $\Gamma^{k+1} = \nabla^2 f(y^{k+1})$, полагается $k := k+1$.

ШАГ 6. Если $\|\nabla f^k\| \leq \epsilon$, то производится останов. Точка y^k выдается в качестве оценки решения. Если же $\|\nabla f^k\| \geq \epsilon$, то осуществляется переход на шаг 2.

Известно, что выполнение модифицированного разложения Холецкого требует около $(1/6)N^3$ операций, а последующее определение \bar{d}^k требует числа операций порядка N^2 , поскольку связано с решением линейных систем с треугольными матрицами.

Замечание. Модифицированный метод Ньютона сохраняет поисковые возможности на функциях с областями «плохого» поведения (вырожденность, знаконеопределенность матриц G_k), поскольку новое направление поиска \bar{d}^k , в силу гарантированной положительной определенности матриц \bar{G}_k , обязательно является направлением строгого локального убывания и соответствует направлению антиградиента в некоторой новой метрике пространства. Кроме того, метод обладает сверхлинейной скоростью сходимости в окрестности решения, если функция в этой окрестности дважды непрерывно дифференцируема и принадлежит классу $\Phi_{m,m}$ из (2.23), а параметр d модификации матриц достаточно мал.

Справедливость последнего утверждения следует из того, что при сделанных предположениях в указанной окрестности матрицы Γ_k будут положительно определены «с запасом» и поэтому модификация матриц производиться не будет, т.е. $\bar{\Gamma}_k = \Gamma_k$, а, следовательно, метод будет точно совпадать с методом Ньютона с регуляризацией шага.

2.7. Методы первого порядка, явно изменяющие метрику пространства

В этом разделе будет продолжено изучение методов, основанных на квадратичной модели поведения минимизируемой функции. В отличие от предыдущего раздела, рассматривается несколько групп методов, которые хотя и используют предположение о достаточной гладкости функции, но не измеряют матриц вторых производных. Методы первой группы (квазиньютоновские методы) строят матричные оценки, заменяющие матрицы Гессе в вычислительных процедурах. При этом сходимость используемых матричных оценок к $\nabla^2 f(y^*)$ в общем случае не обязательна. Методы второй группы — методы растяжения также основаны на построении вспомогательных матриц на основе некоторых эвристических принципов. Обе группы методов будут рассмотрены в данном разделе.

Методы третьей группы явно никаких матриц могут не строить, хотя неявно метрику пространства изменяют (методы сопряженных направлений). Они рассматриваются в разделе 2.8.

2.7.1. Квазиньютоновские методы

Методы этого класса относятся к методам первого порядка, но опираются на фундаментальную идею использования квадратичной модели функции. Они используют результаты испытаний, состоящих в вычислении $f(y^k)$, $\nabla f(y^k)$. Предполагается, что функция f обладает свойствами, соответствующими квадратичной модели. Следовательно, у функции f существует симметричная матрица вторых производных, недоступная непосредственному измерению.

Казалось бы, в этих условиях самым естественным являлось конечно-разностное оценивание гессиана в каждой точке y^k по измерениям градиента на множестве узлов, размещенных с некоторым шагом h в окрестности данной точки. Получив оценку, можно применить модифицированный метод Ньютона с регулировкой шага. Однако данный подход требует слишком большого объема вычислений и, кроме того, может быть связан со значительными погрешностям оценивания при компьютерных вычислениях при малых h за счет особенностей конечноразрядной арифметики. Это приводит к тому, что данный подход обычно не применяется. Оказывается, оценки гессиана можно строить без дополнительных вычислений градиента функции $f(y)$. Именно такой подход используется в квазиньютоновских методах.

Идея, положенная в основу *квазиньютоновских методов*, состоит в том, чтобы по результатам измерения градиентов функции f в точках y^k траектории поиска попытаться построить симметричную матрицу G_k , являющуюся некоторой оценкой «кривизны» поверхности $f(y)$ на траектории поиска, т.е. некоторым образом оценивающую $\nabla^2 f$, или построить симметричную матрицу H_k , являющуюся оценкой обратной матрицы $(\nabla^2 f)^{-1}$. После выполнения k -го испытания, G_k рассматривается как заменитель матрицы вторых производных $\nabla^2 f(y^k)$, а H_k — как заменитель $(\nabla^2 f(y^k))^{-1}$. Точка очередного измерения будет выбираться по правилу

$$y^{k+1} = y^k + x^k d^k, \quad (2.36)$$

$$d^k = (G_k)^{-1}(-\tilde{\nabla} f^k) \text{ (или } d^k = (H_k)(-\tilde{\nabla} f^k) \text{),} \quad (2.37)$$

где x^k определяется одним из методов, рассмотренных в разделе 2.4. В зависимости от способа построения оценочных матриц и информации о свойствах функции f используется либо правило «аккуратного» одномерного поиска

$$x^k \hat{I} P = P_1(h) \hat{I} \ddot{u} P_2(m), 0 < m < h < 1,$$

либо правило Вулфа

$$x^k \hat{I} P^c = P_1^c(h) \hat{I} \ddot{u} P_2(m), 0 < m < h < 1,$$

либо правило Армихо.

Методы вида (2.36), (2.37) выбирают направления перемещения, совпадающие с антиградиентными направлениями функции f , вычисленными в некотором пространстве с преобразованной метрикой. Фактически, на каждом шаге выполняется изменение метрики пространства переменных (поворот осей и перемасштабирование) за счет домножения градиента на соответствующую матрицу. Поэтому второе название этих методов — *методы переменной метрики*. Обычно матричные оценки строят так, чтобы при определенных условиях для квадратичных функций не более чем через N шагов можно было ожидать совпадения матрицы G_k с матрицей $\nabla^2 f(y^*)$, а H_k — с $(\nabla^2 f(y^*))^{-1}$. В общем случае такое совпадение не является обязательным, достаточно выполнения более слабого условия, обеспечивающего высокий порядок скорости сходимости.

Кратко обсудим некоторые идеи, лежащие в основе построения оценок G_k и H_k . Более подробно мотивация способов построения G_k и H_k приведена, например, в [3], см. также [10, 14, 16] и библиографию в [5]. Рассмотрим случай, когда $f(y) = (y^T \Gamma y)/2 + (c, y) + b$ — квадратичная функция с симметричной положительно определенной матрицей Γ .

Пусть

$$\begin{aligned} D^k &= y^{k+1} - y^k, \\ z^k &= \tilde{N} f^{k+1} - \tilde{N} f^k, \end{aligned} \quad (2.38)$$

где $\nabla f^k = \nabla f(y^k)$, $\nabla f^{k+1} = \nabla f(y^{k+1})$. Тогда, очевидно, будут выполняться равенства

$$z^k = \tilde{N} f(y^k + D^k) - \tilde{N} f(y^k) = G D^k, (G^{-1}) z^k = D^k.$$

Потребуем, чтобы таким же условиям удовлетворяли, соответственно, оценка G_{k+1} матрицы Γ и оценка H_{k+1} матрицы Γ^{-1} , построенные по $(k+1)$ -му измерению градиента. А именно, пусть выполняется требование

$$z^k = G_{k+1} D^k \quad (H_{k+1} z^k = D^k). \quad (2.39)$$

Условия (2.39) называются *квазиньютоновскими условиями*.

Наложим дополнительные требования на матрицы оценок. Поскольку сама матрица Γ симметрична, потребуем выполнения свойства симметрии от матрицы G_{k+1} , (H_{k+1}) , положив

$$G_{k+1} = (G_{k+1})^T \quad (\text{или } H_{k+1} = (H_{k+1})^T). \quad (2.40)$$

Будем определять ее новое значение путем коррекции предыдущей матрицы

$$G_{k+1} = G_k + U_k \quad (\text{или } H_{k+1} = H_k + U_k),$$

где поправки U^k строятся в виде матриц ранга 1 или 2 и находятся из условий (2.39), (2.40).

Естественно, эти условия определяют поправки неединственным образом. Рассмотрим возможные приемы их построения на примере оценок G_k . Оценки вида H_k строятся аналогично.

Простейший способ определения поправочной матрицы ранга 1, предложенный Бройденом, состоит в том, чтобы составить ее из вектор-столбцов невязок вида $z^k - G_k D^k$, помноженных на специально подобранные числа. Нетрудно проверить, что при произвольных v^k , для которых $(v^k)^T \Delta^k \neq 0$, нужная оценка определяется следующей формулой:

$$G_{k+1} = G_k + (z^k - G_k D^k)(v^k)^T / ((v^k)^T D^k). \quad (2.41)$$

Это соотношение при произвольном n^k не сохраняет симметрию матрицы G_{k+1} , однако если положить в ней $v^k = (z^k - G_k D^k)$, то сохранение симметрии будет обеспечено. Таким образом, получим *формулу Бройдена (B-формулу)*:

$$G_{k+1} = G_k + (z^k - G_k D^k)(z^k - G_k D^k)^T / ((z^k - G_k D^k)^T D^k). \quad (2.42)$$

Существуют и другие способы оценивания. Например, можно несимметричную поправку в формуле (2.41) заменить похожей симметричной. Непосредственной проверкой можно убедиться (полезно в качестве упражнения выполнить эту проверку), что для любого вектора v^k такого, что $(v^k)^T \Delta^k \neq 0$, соотношение

$$G_{k+1} = G_k + ((z^k - G_k D^k)(v^k)^T + v^k(z^k - G_k D^k)^T) / ((v^k)^T D^k) - (v^k)(v^k)^T (z^k - G_k D^k)^T D^k / ((v^k)^T D^k)^2 \quad (2.43)$$

дает оценочную матрицу, удовлетворяющую (2.39), (2.40). Содержательное описание метода получения этой формулы приведено в [3]. В общем случае поправочная матрица в (2.43) имеет ранг 2. Из этого соотношения при различном выборе v^k можно получить несколько конкретных формул.

Положив в (2.41) $v^k = z^k$, после ряда преобразований получим соотношение, представимое в следующем виде:

$$G_{k+1} = G_k - G_k D^k (D^k)^T G_k / ((D^k)^T G_k D^k) + z^k (z^k)^T / ((z^k)^T D^k) + a_k \times (D^k)^T G_k D^k w^k (w^k)^T, \quad (2.44)$$

где $a_k = 1$, а

$$w^k = z^k / ((z^k)^T D^k) - G_k D^k / ((D^k)^T G_k D^k). \quad (2.45)$$

Поскольку в (2.44), (2.45) $w^k (w^k)^T$ — симметричная матрица и, как можно показать, $(w^k)^T \Delta^k = 0$ (проверьте это в качестве упражнения), то, в силу (2.39), последнее слагаемое в (2.44) не влияет на выполнение квазиньютоновского условия. Поэтому в (2.44) можно

использовать любое неотрицательное значение коэффициента a_k , обычно выбираемое так, чтобы $\alpha_k \in [0, 1]$. При $a_k \equiv 0$ из (2.44) получаем формулу Дэвидона–Флетчера–Пауэлла (DFP-формула), а при $a_k \equiv 1$ (2.44), (2.45) определяет формулу Бroyдена–Флетчера–Гольдфарба–Шэнно (BFGH-формула).

В приведенных итерационных соотношениях начальное значение выбирается так, чтобы G_0 была симметрична и положительно определена, обычно полагают $G_0 = E$ (единичная матрица).

Аналогичные приведенным выше формулы для оценивания обратной матрицы Гессе имеют следующий вид:

$$\begin{aligned} B \text{ — формула для } H_k \\ H_{k+1} = H_k + (D^k - H_k z^k) (D^k - H_k z^k)^T / ((D^k - H_k z^k)^T z^k); \end{aligned} \quad (2.42\phi)$$

$$\begin{aligned} DFP \text{ — формула для } H_k \\ H_{k+1} = H_k - H_k z^k (z^k)^T H_k / ((z^k)^T H_k z^k) + D^k (D^k)^T / ((D^k)^T z^k); \end{aligned} \quad (2.44\phi)$$

$$\begin{aligned} BFGH \text{ — формула для } H_k \\ H_{k+1} = H_k - H_k z^k (z^k)^T H_k / ((z^k)^T H_k z^k) + \Delta^k (\Delta^k)^T / ((\Delta^k)^T \Delta^k) + \\ + (z^k)^T H_k \Delta^k w^k (w^k)^T, \end{aligned} \quad (2.45')$$

где

$$w^k = \Delta^k / ((\Delta^k)^T z^k) - H_k z^k / ((z^k)^T H_k z^k).$$

Основная особенность DFP и BFGH формул заключается в сохранении положительной определенности оценочных матриц при весьма слабых предположениях об условиях выбора шагового множителя x^k в (2.36). Положительная определенность чрезвычайно важна, поскольку только в этом случае итерация (2.36) является методом локального спуска.

Приведем соответствующую теорему [10], [14] или [5].

Теорема. Если $G_k (H_k)$ — симметричная положительно определенная матрица, выбор точки y^{k+1} происходит согласно (2.36), (2.37), а шаговый множитель x^k удовлетворяет условиям (2.5с)–(2.6с), т.е. выполнено неравенство

$$(\nabla f(y^{k+1}), d^k) > (\nabla f(y^k), d^k),$$

то формула (2.44), (2.45) (или аналогичная ей для H_{k+1}) при $0 \leq a_k \leq 1$ порождает положительно определенную симметричную матрицу $G_{k+1} (H_{k+1})$.

Таким образом, при использовании DFP и BFGH-формул для обеспечения положительной определенности матричных оценок при выборе шагового множителя необходимо использовать методы, основанные либо на аккуратном одномерном поиске,

либо на правиле Вулфа, которое является более экономичным. При этом в методе Вулфа целесообразно выбирать начальное приближение $x^k = 1$. В весьма общих предположениях можно показать, что при достаточной близости y^k к решению значение $x^k = 1$ сразу будет удовлетворять правилу Вулфа, что существенно сократит объем вычислений. Аналогичные изменения следует сделать и в методе аккуратного одномерного поиска.

Напомним, что оба указанные правила выбора x^k требуют, вообще говоря, многократного вычисления градиента функции. Однако, использование более «дешевого» в этом смысле правила Армихо уже не гарантирует сохранения положительной определенности матричных оценок.

Замечание. Как уже отмечалось ранее, в системе LocOpt при выборе шаговых множителей используется только правило аккуратного одномерного поиска.

Следующая теорема устанавливает конечную сходимость квазиньютоновских методов для строго выпуклых квадратичных функций $f(y)$.

Теорема. Пусть для квадратичной функции $f(y)$, $y \in R^N$ с положительно определенной матрицей вторых производных Γ применяется метод (2.36)–(2.37) при выборе x^k из условия достижения минимума по направлению, а матричные оценки для G_k (H_k) строятся с использованием DFP или BFGH-формул. Пусть также ни одна из точек y^0, y^1, \dots, y^{N-1} не совпадает с точкой y^* минимума функции f . Тогда направления d^0, d^1, \dots, d^{N-1} нетривиальны, удовлетворяют условию $(d^i, \Gamma d^j) = 0$ ($0 \leq i < j \leq N-1$) (а, следовательно, эти направления Γ -сопряжены), все оценочные матрицы симметричны и положительно определены. Кроме того, $G_N = \Gamma$ ($H_N = \Gamma^{-1}$).

Заметим, что в силу утверждения теоремы точка y^N будет получена по правилу метода Ньютона и, следовательно, совпадет с y^* .

ДОКАЗАТЕЛЬСТВО теоремы можно найти, например, в [1], [14] применительно к DBF-формуле для H_k , а также в [10], применительно к DBF или BFGH-формулам для H_k .

Построенные алгоритмы могут быть применены для достаточно произвольных гладких функций f , не являющихся квадратичными.

Замечание. Потребуем, чтобы f была дважды непрерывно дифференцируемой в R^N , и чтобы в ее стационарной точке y^* выполнялось достаточное условие минимума второго порядка. Если функция f не является квадратичной, то матричные оценки G_k (H_k) уже

не обязаны сходиться к $\nabla^2 f(y^*)$. Вместо этого должно иметь место предельное соотношение вида $(H_k - (\nabla^2 f(y^*))^{-1})\nabla f(y^k) = o(\|\nabla f(y^k)\|)$ или его аналог для G_k (см., например, [5]).

Поскольку в качестве начального значения G_0 (H_0) годится любая симметричная матрица, то в силу приведенной выше теоремы, при соблюдении указанных в ней условий выбора шагового множителя, можно не выполнять переустановку значений оценочных матриц в начальное значение, соответствующее единичной матрице. Однако на практике периодические рестарты с восстановлением этих матриц в начальное значение E все же целесообразны, поскольку приводят к встраиванию в процедуру квазиньютоновского локального спуска последовательности чисто градиентных шагов, что приводит к глобализации сходимости метода. Например, в системе LocOpt рестарты всегда выполняются на шагах, кратных N .

Следует также иметь в виду, что, согласно имеющейся вычислительной практике, более предпочтительной считается BFGH-формула, поскольку она обеспечивает лучшую локальную квадратичную аппроксимацию функции по сравнению с DFP, что уменьшает затраты на одномерный поиск для выбора значения шагового множителя, а также снижает требования к точности этого выбора.

Еще одно замечание необходимо сделать относительно положительной определенности оценочных матриц, гарантированной теоремой. Можно построить примеры [3], показывающие, что при выполнении вычислений в конечно-разрядной арифметике может нарушаться положительная определенность матриц, вычисляемых по DFP или BFGH-формулам, за счет возникающего отбрасывания относительно малых величин.

В результате процесс вычислений может все же привести к тому, что матрицы G_k окажутся вырожденными или знаконеопределенными. При этом направление шага d^k в (2.37) перестанет быть направлением убывания функции и величина смещения x^k в (2.36) может оказаться равной нулю. Простейший способ коррекции в этом случае состоит в замене направления, построенного по правилу (2.37), на обычное антиградиентное направление (хотя эта стратегия не является лучшей).

Возможность указанных выше ситуаций приводит в неквадратичном случае к необходимости дополнительного контроля за поведением оценочных матриц и к большей безопасности при использовании оценки прямой матрицы, а не обратной, хотя это увеличивает вычислительные затраты на определение d^k . Это не является безусловно обязательной, но может быть признано разумной стратегией.

Принцип организации поиска в квазиньютоновских методах проиллюстрирован на рис. 2.13. На нем пунктиром показан возможный вид линий равного уровня для квадратичных аппроксимаций функции $f(y)$, построенных по матричным оценкам G_k . Поскольку $G_0 = E$, то первая из этих линий уровня, построенная для точки y^0 , является сферой, а на остальных шагах сферы преобразуются в эллипсоиды. Выбираемые далее методом направления поиска d^k проходят через центры этих эллипсоидов, являющиеся стационарными точками построенных квадратичных аппроксимаций. Эти направления являются антиградиентными направлениями в пространствах с новыми метриками, связанными с матрицами G_k . Они могут существенно отличаться от направлений антиградиента в исходном пространстве.

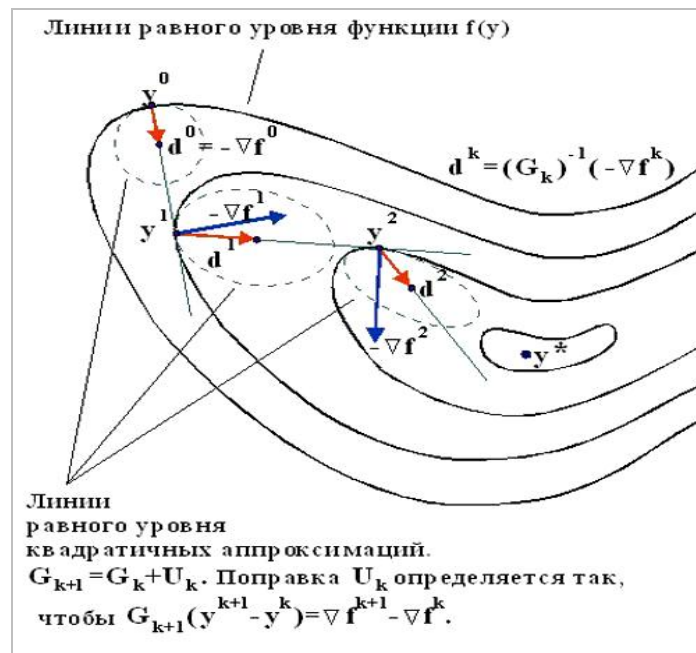


Рис. 2.13. Направления поиска в квазиньютоновских методах

Замечание. При вычислительной реализации алгоритма через прямые оценочные матрицы G_k желателен контроль за сохранением их положительной определенности, а также требуется экономичное вычисление направления поиска согласно (2.37), т.е. определение $d^k = (G_k)^{-1}(-\tilde{N}f^k)$. Эти операции следует проводить совместно. Для этого удобно использовать разложение Холецкого для матриц G_k . Если при этом для диагональных элементов матрицы D_k нарушится условие положительности $d_{ii} > 0$, то в качестве направления поиска можно выбрать обычное антиградиентное направление, или же модифицировать разложение Холецкого с коррекцией малых положительных, а также отрицательных элементов d_{ii} . Построенное разложение $G_k = L_k^T D_k L_k$ следует использовать для определения вектора d^k из решения двух линейных систем с треугольными

матрицами, имеющими положительные элементы на диагоналях: $(L_k)^T v = -\tilde{N} f^k$ и $(D_k L_k) d^k = v$.

МОДЕЛЬНАЯ СХЕМА АЛГОРИТМА на примере использования оценок G_k с рестартами через N шагов и одномерного поиска.

ШАГ 0. Определяем $\epsilon > 0$ — параметр останова, m , h и σ — параметры одномерного поиска ($0 < m < h \ll 1$, $0 < \sigma \ll 1$). Задаем точку начала поиска y^0 .

ШАГ 1. Полагаем $G_0 = E$ и вычисляем $f^0 = f(y^0)$, $\nabla f^0 = \nabla f(y^0)$, $k = 0$.

ШАГ 2. Выполняем преобразование Холесского для матрицы G_k . Если преобразование выполнить не удалось, полагаем $d^k = (-\nabla f^k)$ и переходим на шаг 4. В противном случае получаем элементы разложения $G_k = L_k^T D_k L_k$.

ШАГ 3. Определяем направление поиска $d^k = (G_k)^{-1}(-\nabla f^k)$ путем решения двух систем с треугольными матрицами

$$(L_k)^T v = -\tilde{N} f^k, \quad (D_k L_k) d^k = v. \quad (2.46)$$

ШАГ 4. Определяем x^k с помощью алгоритма “аккуратного” одномерного поиска или правила Вулфа, вычисляем

$$\begin{aligned} y^{k+1} &= y^k + x^k d^k, \\ f^{k+1} &= f(y^{k+1}), \quad \tilde{N} f^{k+1} = \tilde{N} f(y^{k+1}), \\ D^k &= y^{k+1} - y^k, \quad z^k = \tilde{N} f^{k+1} - \tilde{N} f^k. \end{aligned}$$

ШАГ 5. Если $k=N$, проверяем критерий останова: при $\|\nabla f^{k+1}\| \leq \epsilon$ останавливаем поиск и принимаем y^{k+1} в качестве решения; при $\|\nabla f^{k+1}\| > \epsilon$ полагаем $y^0 = y^{k+1}$ и переходим к шагу 1. Если $k \neq N$, то полагаем $k = k+1$ и переходим к шагу 6.

ШАГ 6. Производим вычисление матрицы G_{k+1} по, DFP -формуле (2.44) с $a_k = 0$ или по $BFGH$ -формуле (2.44), (2.45) с $a_k = 1$. Переходим на шаг 2.

На рис. 2.14 приведен построенный в ЛосОпт пример поведения траектории квазиньютоновского метода с настройкой шагового множителя. Полезно сопоставить этот результат с приведенным на рис. 2.7 поведением в тех же условиях метода наискорейшего градиентного поиска.

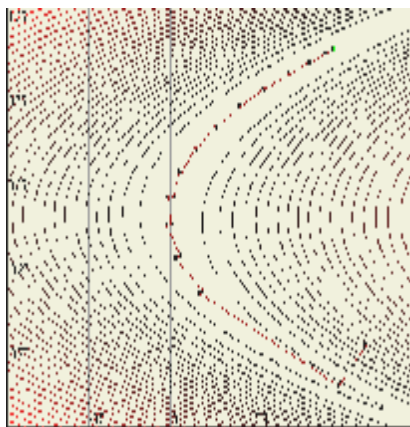


Рис. 2.14. Траектория квазиньютоновского метода на функции Розенброка (30 итераций)

2.7.2. Модифицированные квазиньютоновские методы

Материал излагается на примере методов, использующих заменители G_k прямой матрицы Гессе. В этих методах в случаях возможного нарушения за счет погрешностей вычислений положительной определенности оценочных матриц G_k вместо использования антиградиентного направления выполняется замена матрицы G_k на близкую к ней положительно определенную матрицу \bar{G}_k , построенную с использованием модифицированного преобразования Холесского (или другого подобного преобразования). Рис. 2.15 иллюстрирует изменения вида квадратичной аппроксимации функции $f(y)$ в результате выполнения указанного преобразования для случаев различной знакоопределенности исходной матрицы G_k .

ОПИСАНИЕ АЛГОРИТМА (версия на основе оценок G_k с рестартами).

ШАГ 0. Определяем $\epsilon > 0$ — параметр останова, d — параметр модификации матрицы в модифицированном преобразовании Холесского ($d > 0$), m , h и σ — параметры одномерного поиска ($0 < m < h < 1$, $0 < \sigma < 1$). Задаем точку начала поиска y^0 .

ШАГ 1. Полагаем $G_0 = E$ и вычисляем $f^0 = f(y^0)$, $\nabla f^0 = \nabla f(y^0)$, $k = 0$.

ШАГ 2. Выполняем модифицированное преобразование Холесского для матрицы G_k , получаем $G_k \Rightarrow \tilde{G}_k = L_k^T D_k L_k$ и выполняем замену G_k на \tilde{G}_k

ШАГ 3. Определяем направление поиска $d^k = (\tilde{G}_k)^{-1}(-\nabla f^k)$ путем решения двух систем с треугольными матрицами

$$L_k^T v = -\tilde{N} f^k, \quad (D_k L_k) d^k = v.$$

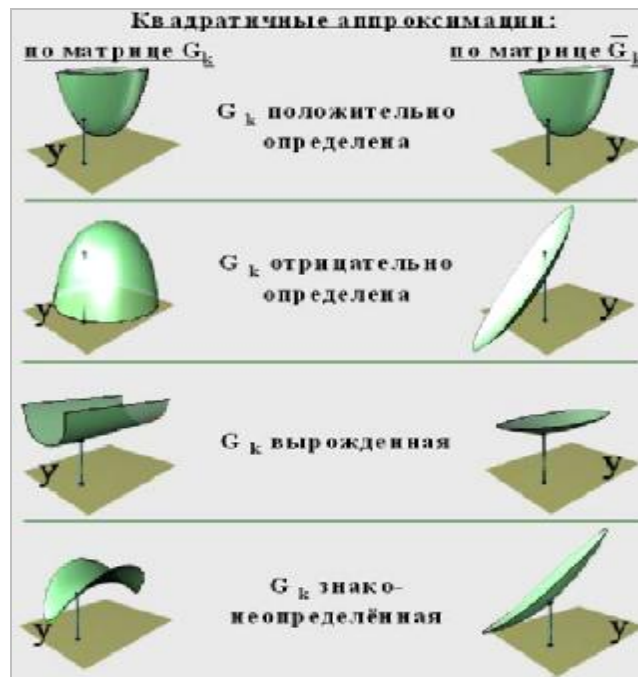


Рис. 2.15. Влияние модификации матрицы G_k на вид аппроксимирующей поверхности

ШАГ 4. Определяем $x^k \in \Pi$ с помощью алгоритма “аккуратного” одномерного поиска или правила Вулфа $x^k \in \Pi'$ с использованием начального приближения $x^k = \tilde{x} = 1$, вычисляем

$$\begin{aligned} y^{k+1} &= y^k + x^k d^k, \\ f^{k+1} &= f(y^{k+1}), \tilde{N}f^{k+1} = \tilde{N}f(y^{k+1}), \\ D_k &= y^{k+1} - y^k, z^k = \tilde{N}f^{k+1} - \tilde{N}f^k. \end{aligned}$$

ШАГ 5. Если $k=N$, Проверяем критерий останова: при $\|\nabla f^{k+1}\| \leq \varepsilon$ останавливаем поиск и принимаем y^{k+1} в качестве решения; при $\|\nabla f^{k+1}\| > \varepsilon$ продолжаем поиск. Если $k=N$, полагаем $y^0 = y^{k+1}$ и переходим к шагу 1. Если $k \neq N$, то полагаем $k = k+1$ и переходим к шагу 6.

ШАГ 6. Производим вычисление матрицы G_{k+1} по, *DFP*-формуле (2.44) с $a_k = 0$ или по *BFGH*-формуле (2.44), (2.45) с $a_k = 1$. Переходим на шаг 2.

Практический опыт показывает, что для широкого класса задач описанные алгоритмы весьма экономичны по числу шагов.

2.7.3. Методы растяжения пространства (R-алгоритмы Н.З. Шора)

Автором этой группы методов является украинский математик Н.З. Шор. Предложенные им методы основаны на эвристических принципах подбора матриц преобразования пространства [19]. Ниже методы растяжения кратко рассматриваются в

одном из своих видов для задачи безусловной оптимизации в предположении непрерывной дифференцируемости минимизируемой функции.

Как уже отмечалось, преобразования пространства в алгоритмах Н.З. Шора сводятся к последовательным растяжениям в специально подбираемых направлениях. Эти методы называют *R-алгоритмами*. Они по структуре близки к квазиньютоновским методам переменной метрики, но основаны не на оценивании (в обобщенном смысле) матриц вторых производных, а на построении некоторой матрицы преобразования пространства B_k . В алгоритме она описывает переход от новых координат z к исходным: $y = B_k z$. Матрица B_k строится как произведение матриц преобразования $R_b(x^e)$, выполняющих растяжение или сжатие пространства z в b раз в направлениях x^e ($e = 1, 2, \dots, k$), $\|x^e\| = 1$.

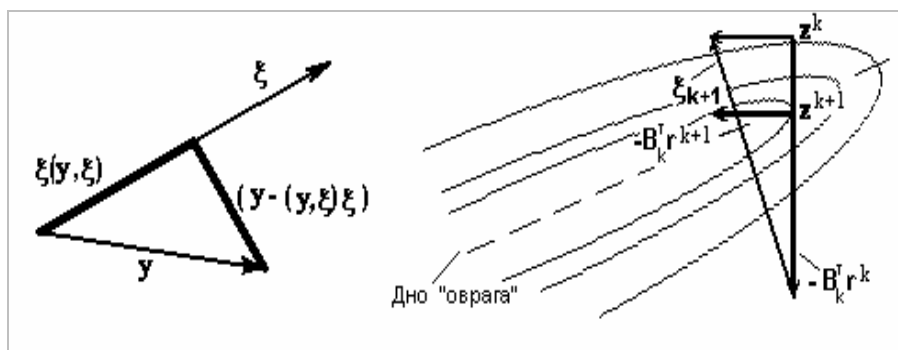


Рис. 2.16. Стратегия построения матрицы растяжения пространства

Нетрудно увидеть (рис. 2.16), что

$$R_b(x) y = (y - (y, x) x) + b \times (y, x) x = (E + (b - 1) x x^T) y. \quad (2.47)$$

Следовательно, $R_b(x) = E + (b - 1) x x^T$.

Пусть $r^k = \nabla f(y^k)$ — градиент функции в исходном пространстве, а \bar{r}^k — это значение градиента, подсчитанного в соответствующей точке z в новом пространстве переменных. Тогда

$$\bar{r}^k = \tilde{N}_z f(B_k z^k) = B_k^T r^k.$$

Для отыскания минимума функции $f(y)$ будем использовать схему метода наискорейшего градиентного поиска, но так, чтобы на каждом шаге k градиент вычислялся в новом пространстве, связанном с матрицей преобразования B_k (рис. 2.17).

В этом пространстве будем в качестве очередного направления растяжения выбирать вектор $x^{k+1} = -B_k^T (r^k - r^{k-1})$, определяющий разность двух последовательных измерений вектора градиента в пространстве, связанном с B_k . Этот вектор будет близок к нормали для многообразия, на котором лежит дно оврага минимизируемой функции (см. рис. 2.14), если рассматривать эти объекты в пространстве новых переменных z .

В найденном направлении x^{k+1} будем осуществлять дополнительное растяжение пространства в фиксированное число раз (с коэффициентом $a \gg 2$ или $a \gg 3$). При возврате к исходным координатам этой операции будет соответствовать сжатие в направлении x^{k+1} с коэффициентом $b = 1/a$. Следовательно, $B_{k+1} = B_k R_{1/a}(x^{k+1})$.

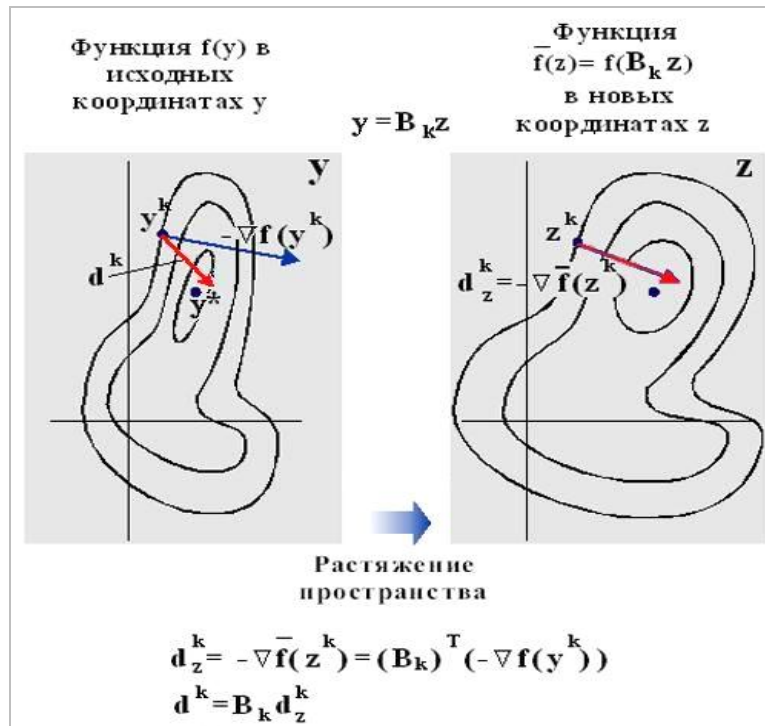


Рис. 2.17. Выбор направления в методе растяжения пространства

Мы приходим к следующему АЛГОРИТМУ МЕТОДА РАСТЯЖЕНИЯ.

ШАГ 0. Задаются $\epsilon > 0$ — параметр критерия останова, $0 < m < h \ll 1$, $0 < \sigma \ll 1$ — параметры алгоритма выбора коэффициента одномерного шага, y^0 — начальная точка поиска, a — коэффициент растяжения пространства.

ШАГ 1. Вычисляются $f^0 = f(y^0)$, $r^0 = \nabla f(y^0)$, полагается $B_0 = E$, $k = 0$.

ШАГ 2. Вычисляется величина коэффициента одномерного шага x^k методом «аккуратного» одномерного поиска. Определяются

$$\begin{aligned} y^{k+1} &= y^k + x^k B_k (B_k)^T (-r^k), \\ f^{k+1} &= f(y^{k+1}), r^{k+1} = \tilde{N}f(y^{k+1}). \end{aligned} \quad (2.48)$$

ШАГ 3. Если $\|r^{k+1}\| < \epsilon$, то выполняется останов метода, иначе переходим к шагу 4.

ШАГ 4. Выбирается направление дополнительного растяжения

$x^{k+1} = (B_k)^T (r^{k+1} - r^k)$ и выполняется его нормировка $x^{k+1} := x^{k+1} / \|x^{k+1}\|$.

ШАГ 5. Пересчитывается матрица преобразования с учетом растяжения пространства в a раз вдоль x^{k+1} :

$$B_{k+1} = B_k R_{1/a}(x^{k+1}). \quad (2.49)$$

ШАГ 6. Если хотя бы один из элементов b_{ij} матрицы B_{k+1} превысит по модулю некоторое заранее установленное пороговое значение, то все элементы этой матрицы делятся на модуль элемента b_{ij} . Изменяется $k = k + 1$ и выполняется переход к шагу 2.

О сходимости и оценках скорости сходимости этого метода для задач достаточно общего вида известно немного. Н.З. Шор исследовал сходимость для специального класса *приближенно-однородных* функций, удовлетворяющих условию

$$m \cdot (f(y) - f(y^*)) \leq (\nabla f(y), y - y^*) \leq M(f(y) - f(y^*))$$

при некоторых положительных $m < M \quad \forall y \in R^N$. Для некоторых версий метода доказана линейная сходимость по значениям функции.

Из результатов А.С. Немировского – Д.Б. Юдина [15] следует, что для произвольных выпуклых $f(y)$ метод не может сходиться по значению функции (в гарантированном смысле) быстрее, чем геометрическая прогрессия со знаменателем вида $q = (1 - 1/(C \cdot N))$. Метод интересен возможностью его использования в недифференцируемом случае.

2.8. Методы сопряженных направлений

2.8.1. Сопряженные направления и их свойства

Построение методов *сопряженных направлений* основано на квадратичной модели поведения минимизируемой функции. Предположим вначале, что $f(y)$ — квадратичная функция (2.11) с положительно определенной матрицей.

Определение. Система линейно-независимых векторов p^0, p^1, \dots, p^{N-1} для симметричной матрицы Γ называется Γ -сопряженной, если

$$i = 0, \dots, N-1; j = 0, \dots, N-1; i \neq j: (p^i, \Gamma p^j) = 0. \quad (2.50)$$

Определение. Пусть M – линейное многообразие, Γ – симметричная матрица, $x \neq 0$ и $x \notin M$ и

$$z \in M: (x, \Gamma z) = 0, \quad (2.51)$$

тогда вектор x называется Γ -сопряженным с многообразием M .

Лемма. Если p^0, p^1, \dots, p^{N-1} — все отличны от нуля, Γ — не только симметрична, но еще и положительно определена, тогда из (2.50) следует линейная независимость векторов p^0, p^1, \dots, p^{N-1} .

В условиях леммы сопряженность означает ортогональность в смысле некоторого нового скалярного произведения.

Для построения методов, использующих сопряженные направления, чрезвычайно важным является свойство, определяемое следующей леммой.

Лемма. Пусть $f(y)$ — квадратичная функция вида (2.12) с симметричной положительно определенной матрицей Γ , а M и L — линейные многообразия, причем

$M \subset L$, тогда, если z — точка минимума $f(y)$ на M , а u — точка минимума $f(y)$ на L , то вектор $(u - z)$ будет Γ -сопряжен с многообразием M , включенным в L .

Это утверждение иллюстрируется рис. 2.18.

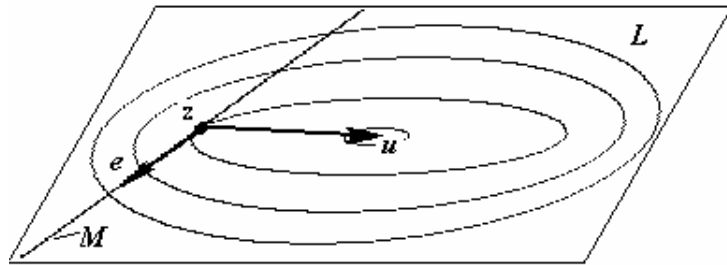


Рис. 2.18. Иллюстрация к лемме

Построим теперь вычислительные процедуры поиска минимума квадратичной функции $f(y)$, использующие Γ -сопряженные направления.

Определение. Поисковые процедуры вида (2.52)–(2.53) называются методами сопряженных направлений:

$$y^{k+1} = y^k + x^k p^k, \quad (2.52)$$

$$f(y^k + x^k p^k) = \min\{f(y^k + x p^k): -\mathbb{Y} < x < +\mathbb{Y}\}. \quad (2.53)$$

Применение сопряженных направлений при построении методов оптимизации связано с замечательным свойством этих направлений приводить в минимум строго выпуклой квадратичной функции не более чем за N шагов.

Теорема. Пусть $f(y)$ — квадратичная функция вида (2.11) с симметричной положительно определенной матрицей Γ , а p^0, p^1, \dots, p^{N-1} — система Γ -сопряженных векторов. Тогда для любой начальной точки y^0 процедура поиска вида (2.52)–(2.53) приводит в минимум квадратичной функции с симметричной положительно определенной матрицей Γ не более, чем за N шагов, т.е. $y^N = y^*, f(y^N) = f(y^*)$.

Утверждение теоремы иллюстрирует рис. 2.19.

Отметим, что в силу приведенной теоремы рассмотренные в п. 2.7.1 квазиньютоновские методы для квадратичной функции $f(y)$ являются, по существу, методами сопряженных направлений. Разберем еще один возможный способ построения метода сопряженных направлений, не требующий построения матричных оценок, используемых квазиньютоновскими методами.

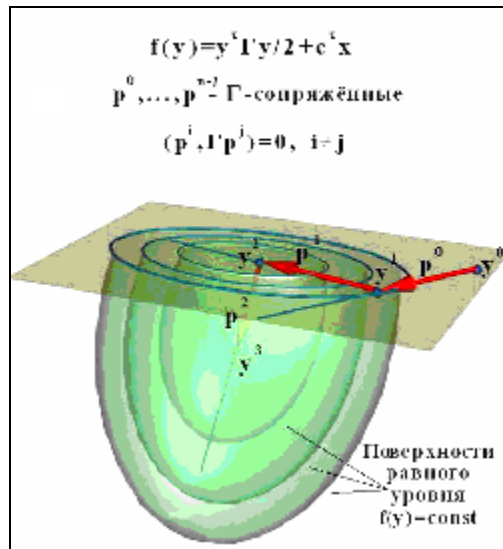


Рис. 2.19. Иллюстрация свойства методов сопряженных направлений из теоремы

2.8.2. Метод сопряженных градиентов Флетчера–Ривса

Этот метод — наиболее важный в классе методов сопряженных направлений. Он использует испытания первого порядка, когда у функции f в точке y^k определяются значения $f(y^k)$ и $\nabla f(y^k)$. Для построения метода сопряженных направлений необходимо по результатам испытаний получить систему Γ -сопряженных векторов p^0, \dots, p^{N-1} при условии, что сама матрица Γ является неизвестной.

Выведем необходимые соотношения для одного из возможных методов такого типа — метода сопряженных градиентов Флетчера–Ривса (1964 год) [9]. Выберем

$$p^0 = -\tilde{N}f^0, \quad \tilde{N}f^0 = \tilde{N}f(y^0). \quad (2.54)$$

Пусть векторы p^0, \dots, p^{k-1} построены. Положим

$$p^k = -\tilde{N}f^k + b_{k-1}p^{k-1}, \quad \tilde{N}f^k = \tilde{N}f(y^k), \quad (2.55)$$

где y^k определяется условиями (2.52), (2.53). Подберем b_{k-1} из условия

$$(p^k, \Gamma p^{k-1}) = 0.$$

Получим

$$b_{k-1} = ((\tilde{N}f^k)^T \Gamma p^{k-1}) / ((p^{k-1})^T \Gamma p^{k-1}). \quad (2.56)$$

Значение x^k , удовлетворяющее (2.53) для функции f , можно получить из условия экстремума $(\nabla f(y^k + x^k p^k), p^k) = 0$, если его переписать в виде

$$((\tilde{N}f^k)^T p^k) + ((p^k)^T \Gamma p^k) x^k = 0.$$

Отсюда можно показать, что x^k будет иметь вид

$$x^k = -((p^k)^T \tilde{N}f^k) / ((p^k)^T \Gamma p^k) = -((\tilde{N}f^k)^T \tilde{N}f^k) / ((\tilde{N}f^k)^T \Gamma p^k). \quad (2.57)$$

Для этого в числителе и знаменателе первой дроби необходимо выразить p^k из (2.55) и воспользоваться тем, что $((p^{k-1})^T \nabla f^k) = 0$ по теореме Лагранжа, и $(p^{k-1}, \Gamma p^k) = 0$ по построению.

Кроме того, умножая (2.52) на Γ , получим дополнительное соотношение

$$\tilde{N}f^{k+1} = \tilde{N}f^k + x^k Gp^k. \quad (2.58)$$

Лемма. Последовательность векторов градиентов $\nabla f^0, \nabla f^1, \dots, \nabla f^{N-1}$ образует взаимно ортогональную систему, а направления p^0, p^1, \dots, p^{N-1} Γ -сопряжены.

ДОКАЗАТЕЛЬСТВО. Пользуясь соотношением (2.55), (2.58), (2.57), лемму можно доказать методом математической индукции.

Действительно, по построению, p^1 сопряжен с p^0 . Кроме того, $p^0 = -\nabla f^0$, а по теореме Лагранжа ∇f^1 ортогонально p^0 , следовательно, ∇f^1 ортогонально ∇f^0 . Таким образом, для двух векторов лемма верна.

Предположим, что при $k < (N-1)$ векторы в системе p^0, p^1, \dots, p^k взаимно сопряжены, а векторы $\nabla f^0, \nabla f^1, \dots, \nabla f^k$ — взаимно ортогональны. Покажем, что эти свойства сохраняются у данных систем векторов при включении в них p^{k+1} и ∇f^{k+1} .

Рассмотрим значения $i < k$, тогда

$$((\tilde{N}f^{k+1})^T \tilde{N}f^i) = ((\tilde{N}f^k + x^k Gp^k)^T \tilde{N}f^i) = x^k (Gp^k)^T (-p^i + b_{i-1} p^{i-1}) = 0.$$

Равенство нулю получается за счет сопряженности p^k с векторами p^i и p^{i-1} .

Рассмотрим теперь $i=k$. Аналогично предыдущему

$$((\tilde{N}f^{k+1})^T \tilde{N}f^k) = ((\tilde{N}f^k + x^k Gp^k)^T \tilde{N}f^k) = 0.$$

Равенство нулю здесь можно получить, используя выражение из (2.57) для величины x^k .

Осталось доказать сопряженность системы векторов p^i для $i=1, \dots, k+1$. Сопряженность двух последних векторов следует из способа их построения. Осталось рассмотреть $i < k$.

$$\begin{aligned} ((p^{k+1})^T Gp^i) &= (-\tilde{N}f^{k+1} + b_k p^k)^T Gp^i = (-\tilde{N}f^{k+1})^T Gp^i = \\ &= (-\tilde{N}f^{k+1})^T (\tilde{N}f^{i+1} - \tilde{N}f^i) / x^i = 0. \end{aligned}$$

Последнее равенство нулю вытекает из уже доказанной ортогональности градиентов.

Метод сопряженных направлений для функции $f(y)$, являющейся положительно определенной квадратичной формой, в целом, построен. Однако в формулу (2.56) для вычисления коэффициента b^{k-1} вошла неизвестная матрица Γ . Но это не является существенным, поскольку формула (2.56) может быть переписана в другом виде. Чтобы показать этот факт, выразим Γp^{k-1} в числителе (2.56) из (2.58), а p^{k-1} в знаменателе (2.56) из (2.55). Тогда

$$\begin{aligned} b_{k-1} &= (\tilde{N}f^k, (\tilde{N}f^k - \tilde{N}f^{k-1})) / (x^{k-1} (-\tilde{N}f^{k-1} + b_{k-2} p^{k-2})^T Gp^{k-1}) = \\ &= (\tilde{N}f^k, \tilde{N}f^k) / (-\tilde{N}f^{k-1})^T x^{k-1} Gp^{k-1}. \end{aligned}$$

Выражая $x^{k-1} \Gamma p^{k-1}$ из (2.58) и пользуясь ортогональностью ∇f^k и ∇f^{k-1} , окончательно получим

$$b_{k-1} = \|\tilde{N}f^k\|^2 / \|\tilde{N}f^{k-1}\|^2. \quad (2.59)$$

Если не преобразовывать числитель в предшествующем соотношении, получим другую форму записи коэффициента

$$b_{k-1} = (\nabla f^k, \nabla f^k - \nabla f^{k-1}) / \|\nabla f^{k-1}\|^2, \quad (2.59\phi)$$

которая в рассматриваемом квадратичном случае эквивалентна (2.59).

Замечание. Построенный метод находит минимум любой квадратичной функции с положительно определенной матрицей Гессе не более, чем за N шагов. Отметим, что для вычисления x^k должен быть использован «аккуратный» одномерный поиск (т.е. параметр η одномерного поиска должен быть выбран близким к нулю).

Применение метода сопряженных градиентов к функции $f(y)$ общего вида, естественно, не может обеспечить конечность процедуры поиска минимума. В неквадратичном случае генерируемые направления p^k уже не являются Γ -сопряженными для какой-либо матрицы Γ . В своей «традиционной» форме после выполнения серии из N шагов метод, как правило, повторно запускается из последней найденной точки, что приводит к встраиванию в процедуру поиска градиентных шагов. Это положительно влияет на область сходимости метода. Кроме того, две приведенные в (2.59), (2.59') формулы для определения b_{k-1} перестают быть эквивалентными. Вычислительная практика показывает, что предпочтительнее использовать формулу (2.59').

Соответствующий алгоритм, рассчитанный на функции достаточно общего вида, может быть записан следующим образом.

АЛГОРИТМ МЕТОДА СОПРЯЖЕННЫХ ГРАДИЕНТОВ ФЛЕТЧЕРА–РИВСА

ШАГ 0. Задаются $\epsilon > 0$ — параметр останова, $0 < m < h < 1$, $0 < \sigma < 1$ — параметры одномерного поиска, y^0 — начальная точка.

ШАГ 1. Вычисляются $f^0 = f(y^0)$, $\nabla f^0 = \nabla f(y^0)$, $p^0 = -\nabla f^0$, $k = 0$.

ШАГ 2. Если $(\nabla f^k, p^k) \geq 0$, то направление p^k не является направлением локального убывания функции, поэтому заменяем $p^0 = -\nabla f^k$, $y^0 = y^k$ и полагаем $k = 0$. Иначе направление p^k сохраняем. Переходим на шаг 3.

ШАГ 3. Вычисляется величина коэффициента одномерного шага x^k методом «аккуратного» одномерного поиска. Определяются

$$y^{k+1} = y^k + x^k p^k, \\ f^{k+1} = f(x^{k+1}), \tilde{N}f^{k+1} = \tilde{N}f(x^{k+1}).$$

Полагается $k = k + 1$.

ШАГ 4. Проверяется критерий останова: при $\|\nabla f^k\| \leq \epsilon$ поиск прекращается и y^k выдается как оценка решения; при $\|\nabla f^k\| > \epsilon$ переходим к шагу 5.

ШАГ 5. Если $k = N$, полагается $y^0 = y^N$ и происходит возврат на шаг 1.

Если $k < N$, то переходим на шаг 6.

ШАГ 6. Вычисляем b^{k-1} по формуле (2.59) или (2.59) и p^k по формуле (2.55).
Переходим на шаг 2.

Что известно о сходимости и скорости сходимости построенного метода? Для достаточно гладких задач она — весьма высокая. Вопросы сходимости и скорости сходимости исследуются, например, в [6, 11, 16]. Можно показать [16], что для функций из класса $F_{m,M}$, описанного в (2.23), метод Флетчера–Ривса сходится со сверхлинейной скоростью.

Замечание. Метод более чувствителен к «аккуратности» одномерного поиска по сравнению с квазиньютоновскими методами. Отклонение x^k от минимума вдоль выбранного направления может приводить к тому, что построенное направление p^{k+1} может не являться направлением локального убывания функции. В этом случае p^k заменяется на антиградиентное направление. Учет этой ситуации происходит на шаге 2 описания алгоритма.

Свойство строгого локального убывания функции f в точке y^k в направлении p^{k+1} определяется отрицательностью соответствующего скалярного произведения, которое в силу (2.55) представимо в виде

$$(\nabla f^{k+1}, p^{k+1}) = -(\nabla f^{k+1}, \nabla f^{k+1}) + b_k (\nabla f^{k+1}, p^k).$$

Если x^k соответствует минимуму вдоль направления p^k , то $(\nabla f^{k+1}, p^k) = 0$ и знак всего выражения — отрицательный. Ошибки в определении минимума по направлению или же использование правила Вулфа могут нарушать это свойство. Возникающие при этом частые замены p^k на антиградиент могут существенно снизить эффективность метода.

Квазиньютоновские методы и метод сопряженных градиентов являются наиболее часто используемыми при оптимизации достаточно гладких функций. При этом квазиньютоновские методы затрачивают дополнительные операции на вычисление и использование матричных оценок, зато могут применять более экономичное правило Вулфа для определения шагового множителя. В то же время метод сопряженных градиентов не тратит вычислительные ресурсы на построение матричных оценок, но требует дополнительных вычислений градиента для реализации «аккуратного» одномерного поиска. Поэтому можно отдать предпочтение методу сопряженных градиентов, если вычисления $\nabla f(y)$ относительно «дешевы», а размерность N велика. В

обратной ситуации более экономичными могут оказаться квазиньютоновские методы (но только при хорошо продуманной вычислительной реализации).

2.9. Некоторые методы прямого поиска

В отличие от рассмотренных ранее, методы прямого поиска не используют каких-либо предположений о гладкости минимизируемой функции. Функция может не только не иметь производных, но может содержать разрывы. При поиске минимума эти методы измеряют только значения функции. Поскольку гладкости нет, то при выборе направлений смещения методы не могут использовать аппроксимаций функции по результатам ее измерения. Правила размещения измерений в них основываются на некоторых эвристических логических схемах.

Наиболее популярными в практике расчетов являются следующие методы прямого поиска: Хука–Дживса [17] (более популярен), метод деформируемого многогранника Нелдера–Мида [18] и его модификация — комплексный метод Бокса. Нужно заметить, что последний метод применим только к выпуклым функциям, поэтому здесь он не рассматривается. Ниже будут описаны первые два метода.

Замечание. Несмотря на кажущуюся простоту и теоретическую необоснованность методов прямого поиска, они хорошо зарекомендовали себя в реальных расчетах, не требующих высокой точности, и могут быть успешно использованы на первых этапах поиска решения.

Это можно объяснить следующим образом. Методы гладкой оптимизации требуют дополнительных ресурсов для вычисления градиентов, а также для реализации одномерного поиска, поэтому итерация этих методов может быть значительно «дороже» выполнения шага в методах прямого поиска.

Следует заметить, что хотя методы прямого поиска не рассчитаны на гладкость функций, их нельзя относить к методам недифференцируемой оптимизации. Методы прямого поиска, например, не могут, в общем случае, решать плохо обусловленные задачи, недифференцируемые именно вдоль «дна оврага», по которому должна проходить траектория поиска. Несмотря на это, методы прямого поиска полезно включать как разумное дополнение в пакеты программ для оптимизационных расчетов.

2.9.1. Метод Нелдера–Мида

В методе Нелдера–Мида вокруг начальной точки поиска в пространстве переменных размещается начальный симплекс — конфигурация из $(N+1)$ -й точки (в пространстве R^2 они образуют вершины треугольника, а в R^3 — вершины пирамиды). Затем происходит

перемещение симплекса путем отражения вершины с наибольшим значением функции относительно центра тяжести противолежащего основания симплекса. При этом используются специальные операции, связанные с растяжением симплекса в направлении убывания функции и операции сжатия при неудачных пробных перемещениях. Дадим формальное описание алгоритма.

АЛГОРИТМ МЕТОДА НЕЛДЕРА–МИДА

ШАГ 0. Задаем векторы h^1, h^2, \dots, h^{N+1} , определяющие положение вершин стандартного симплекса с центром в начале координат, и числа S_1, S_2, \dots, S_{N+1} , определяющие размеры начального симплекса; $e_y > 0$, $e_f > 0$ — параметры останова; a, b, c, d — параметры отражения, растяжения, сжатия к основанию, сжатия к лучшей вершине ($a > 0$, $b > 1$, $0 < c < 1$, $0 < d < 1$). Задаем также начальную точку y^0 .

ШАГ 1. Формируем начальный симплекс с координатами вершин y^1, \dots, y^{N+1} ,

$$y^j = y^0 + S_j h^j; (j = 1, \dots, N+1).$$

Вычисляем $f^j = f(y^j)$. (При этом в y^0 вычисление не выполняется).

ШАГ 2. Определяем номера худшей и лучшей вершины

$$f^h = \max \{f_j : j=1, \dots, N+1\}; f^e = \min \{f_j : j=1, \dots, N+1\}.$$

ШАГ 3. Определяем центр тяжести основания, противолежащего худшей вершине

$$\bar{y} = \frac{1}{N} \left(\sum_{j=1, j \neq h}^{N+1} y^j \right).$$

ШАГ 4. Проверяем критерий останова. Вычисляем

$$\bar{y} = \frac{1}{N+1} \left(\sum_{j=1}^{N+1} y^j \right), \quad \bar{f} = \frac{1}{N+1} \left(\sum_{j=1}^{N+1} f^j \right), \quad \delta_y = \frac{1}{N+1} \left(\sum_{j=1}^{N+1} (y^j - \bar{y})^2 \right)^{1/2},$$

$$\delta_f = \frac{1}{N+1} \left(\sum_{j=1}^{N+1} (f^j - \bar{f})^2 \right)^{1/2}.$$

Если $d_y < e_y$ и $d_f < e_f$, то выполняем останов, выдаем оценку решения y^e, f^e . Если условия останова не выполнены, переходим на шаг 5.

ШАГ 5. Выполняем отражение с коэффициентом $a > 0$

$$y^* = \bar{y} + a(\bar{y} - y^h)$$

и вычисляем $f^* = f(y^*)$.

ШАГ 6. Если $f^* > f^e$, то переходим на шаг 7. Если $f^* \leq f^e$, то выполняем растяжение

$$y^{**} = \bar{y} + b(y^* - \bar{y}), \quad b > 1, \quad f^{**} = f(y^{**});$$

при $f^{**} \leq f^*$ заменяем $y^h := y^{**}$, $f^h := f^{**}$ и переходим на шаг 2;

при $f^{**} > f^*$ заменяем $y^h := y^*$, $f^h := f^*$ и переходим на шаг 2.

ШАГ 7. Если $\forall j=1, \dots, N+1$, но $j \neq e$, выполняется $f^e < f^* < f^j$, то заменяем $y^h := y^*$, $f^h := f^*$ и переходим на шаг 2, иначе — на шаг 8.

ШАГ 8. Если $f^* < f^h$, то выполняем сжатие к основанию. Для этого вычисляем

$$y^{\wedge} = \bar{y} + c(y^h - \bar{y}), \quad f^{\wedge} = f(y^{\wedge}), \quad 0 < c < 1,$$

заменяем $y^h := y^{\wedge}$, $f^h := f^{\wedge}$ и переходим на шаг 2.

Если $f^* \geq f^h$, то выполняем сжатие к лучшей вершине:

$$y^j := y^e + d(y^j - y^e), \quad 0 < d < 1, \quad f^j := f(y^j) \quad (j = 1, \dots, N+1), \quad j \neq e.$$

Переходим на шаг 2.

Преобразования симплекса в пространстве R^2 при операциях отражения, растяжения и сжатия показаны на рис. 2.17.

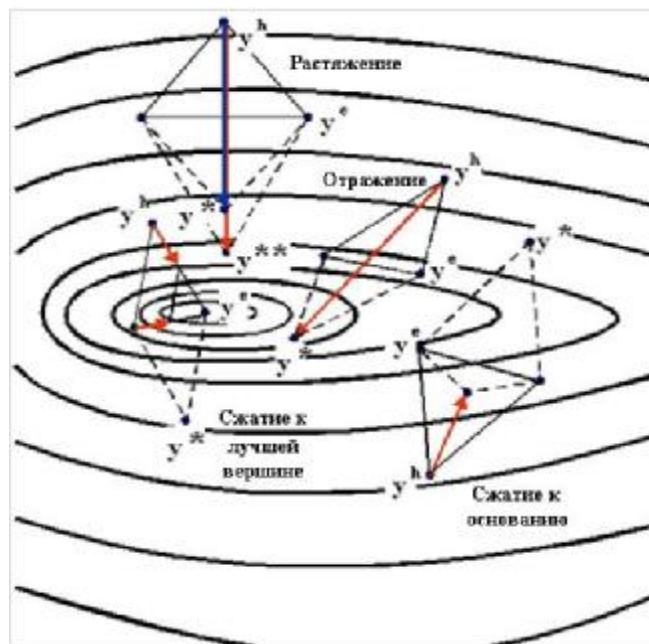


Рис. 2.20. Случаи использования различных типовых операций с симплексом в методе Нелдера–Мида

Авторы метода рекомендовали следующие значения параметров $a = 1$; $b = 1,5$; $c = 0,5$; $d = 0,5$ (кстати, метод чувствителен к их изменениям).

Метод Нелдера–Мида имеет тот недостаток, что для сильно овражных функций может происходить *вырождение симплекса*, особенно при числе переменных $N > 2$.

Термин «вырождение» означает, что все точки симплекса с некоторого шага размещаются в многообразии размерности меньшей, чем N , или же попадают в малую его окрестность, величина которой много меньше расстояния между точками симплекса.

2.9.2. Метод Хука–Дживса

В этом разделе приводится краткое описание метода Хука–Дживса, который был специально разработан именно для задач с оврагами [17]. В этом методе поиск минимума на каждом шаге происходит в результате смещения вдоль некоторого направления-образца (*шаг по образцу*), которое строится, а затем корректируется в результате

специальных пробных покоординатных перемещений, называемых построением конфигурации.

Построение конфигурации из точки z осуществляет отображение z в точку $\bar{y} = F(z)$, где F — оператор построения конфигурации. Он устроен так, что направление $(\bar{y} - z)$ является направлением убывания функции f в окрестности z . Для описания оператора F введем следующие обозначения: e^i — i -й координатный орт, h — параметр, определяющий величину координатного перемещения. Тогда переход от z к \bar{y} осуществляется согласно следующему алгоритму.

АЛГОРИТМ ПОСТРОЕНИЯ КОНФИГУРАЦИИ $\bar{y} = F(z)$:

ШАГ 0. Полагаем $\bar{y} = z$.

ШАГ 1. Для i от 1 до N выполняем действия:

если $f(\bar{y} + h e^i) < f(\bar{y})$, то полагаем $\bar{y} := \bar{y} + h e^i$,

иначе, если $f(\bar{y} - h e^i) < f(\bar{y})$, то $\bar{y} := \bar{y} - h e^i$,

в противном случае — значение \bar{y} не меняется.

На рис. 7.23 показаны примеры построения конфигураций для нескольких случаев положения точки z . Пунктирными линиями отмечены пробные перемещения, не приведшие к уменьшению значения функции.

АЛГОРИТМ МЕТОДА ХУКА–ДЖИВСА

ШАГ 0. Задаются начальная точка y^0 , параметр останова $\epsilon > 0$, параметр построения конфигурации $h \gg \epsilon$, а также параметр увеличения шага $a = 2$.

ШАГ 1. Полагаем $z^1 = y^0$, $k = 0$.

ШАГ 2. Строим конфигурацию $y^{k+1} = F(z^{k+1})$.

ШАГ 3. Если $f(y^{k+1}) < f(y^k)$, то $k := k+1$ и переходим на шаг 4, иначе, если $h \leq \epsilon$, выполняем ОСТАНОВ поиска, если $h > \epsilon$, то дальнейшие действия зависят от того, как была построена точка y^{k+1} : строилась ли конфигурация с использованием шага по образцу (в этом случае $k > 0$) или она строилась от точки y^0 (в этом случае $k = 0$). Если окажется, что $k=0$, то сокращаем h вдвое ($h := h/2$) и переходим на шаг 1, если же $k > 0$, то полагаем $y^0 = y^k$, $k=0$ и также переходим на шаг 1.

ШАГ 4. Выполняем шаг по образцу $z^{k+1} = y^k + a(y^k - y^{k-1})$ и переходим на шаг 2.

Можно следующим образом пояснить смысл действий, выполняемых на шагах 2,3,4. Шаг 4 введен для того, чтобы метод обладал способностью быстрого увеличения величины смещения на одной итерации в том случае, когда точка y^k находится достаточно далеко от решения. При этом, прежде чем сделать z^{k+1} текущей точкой итерации, из точки

z^{k+1} выполняется построение конфигурации. За счет этого, в случае получения точки с $f(y^{k+1}) < f(y^k)$, следующий шаг по образцу в общем случае будет выполняться в измененном, по отношению к предыдущему, направлении, что позволяет методу адаптироваться к изменению рельефа функции.

Наконец, при получении значения $f(y^{k+1}) \geq f(y^k)$ на шаге 3 совершается попытка запустить метод сначала из точки $y^0 = y^k$ — лучшей найденной точки. При этом, если такой попытки еще не было, то параметр h не изменяется, а если она уже была, h предварительно сокращается вдвое.

Некоторые из возможных ситуаций показаны на рис. 2.21.



Рис. 2.21. Перемещения точки в методе Хука–Дживса из двух начальных положений y^0

3. МЕТОДЫ УЧЕТА ОГРАНИЧЕНИЙ В ЛОКАЛЬНОЙ ОПТИМИЗАЦИИ

В этой главе приведены методы учета ограничений двух типов, включенные в систему LocOpt. Первую группу составляют двухсторонние ограничения на переменные вида $y \in D$. Функции задачи могут быть неопределенны вне области D . Учет таких ограничений приводит к необходимости специальной модификации рассмотренных во второй главе методов локального поиска. В систему LocOpt включены именно модифицированные методы. Вторая группа ограничений — ограничения общего вида. Для их учета в LocOpt версии 2 используется метод внешнего степенного штрафа, а в систему LocOpt версии 3 дополнительно включен метод модифицированных функций Лагранжа.

3.1. Особенности применения методов локального поиска при двусторонних ограничениях на переменные

Двусторонние ограничения на переменные являются частным случаем линейных неравенств, и их можно было бы учесть на основе одной из общих методик для ограничений такого типа. Однако их вид настолько прост, с одной стороны, а, с другой стороны, специфичен, что наиболее правильным решением является отдельное описание способа работы с ними. Наличие таких ограничений неизбежно приводит к пересмотру ранее построенных в разделах 2.5–2.9 алгоритмов и созданию их модификаций для задач с двусторонними ограничениями. Особенностью этих модифицированных методов является то, что их траектории никогда не покидают области поиска D из (2.2).

В зависимости от типа метода его модификация выполняется по-разному. Ниже будет приведен один из возможных подходов, максимально использующий ранее рассмотренные методы безусловной локальной оптимизации. В его рамках общие принципы построения модифицированных методов можно предложить для методов гладкой оптимизации, а для методов прямого поиска приходится использовать в каждом случае свои уникальные подходы.

3.1.1. Особенности учета двусторонних ограничений на переменные в методах гладкой оптимизации

Характерными чертами многих методов гладкой оптимизации в R^N являются:

- выполнение рестартов из последней достигнутой точки через определенное число шагов, которое может выбираться зависимым от размерности пространства поиска

(например, это может относиться к методам квазиньютоновского типа и обычно относится к методу сопряженных градиентов — см. материал разделов 2.7–2.8);

- выполнение одномерного поиска в выбранном направлении (в большинстве методов) либо перемещение в выбранном направлении с заранее заданным коэффициентом величины шага (метод Ньютона).

Рассмотрим теперь задачу с двусторонними ограничениями на переменные

$$f(y) \rightarrow \min, \quad y \in D, \quad D = \{y \in R^N : a \leq y \leq b\}. \quad (3.1)$$

Наличие таких ограничений будет оказывать влияние на реализацию каждого из этих двух процессов (возможные рестарты и одномерный поиск). А именно, процессы одномерных перемещений могут выводить на фрагменты границы множества D (описанные ранее алгоритмы одномерного поиска учитывали такую возможность). После выхода на границу поиск должен продолжаться на линейном координатном многообразии меньшей размерности. Одномерные перемещения в этом многообразии могут выводить процесс поиска на ограничения по другим переменным, что будет приводить к дальнейшему понижению размерности многообразия поиска. Кроме того, поиск на возникающих многообразиях, кроме контроля пересечения границ допустимого множества, будет иметь также ту особенность, что методы, периодически выполняющие рестарты с частотой, связанной с размерностью пространства, должны будут производить их чаще, чем при поиске во всем пространстве. Еще одним важным моментом является правило возврата процесса поиска с текущего многообразия на многообразие (или в пространство) более высокой размерности в том случае, когда это приводит к уменьшению значения функции. Решение об увеличении размерности подпространств поиска принимается на основе проверки выполнения условий Куна–Таккера в полученной оптимальной точке на текущем пересечении граничных координатных многообразий. Эта проверка приобретает особо простой вид благодаря специальной структуре допустимого множества D .

Дадим описание правил выполнения следующих операций:

- учет выходов на новые фрагменты границы при одномерных перемещениях;
- организация поиска на многообразиях размерности $n < N$;
- определение моментов возврата с многообразий текущей размерности n в многообразия большей размерности.

Для описания текущего многообразия поиска введем два множества J_a и J_b . В последующем они будут содержать наборы номеров переменных, по которым текущая

точка поиска выведена на нижние или верхние граничные значения. Если хотя бы одно из этих множеств не пусто, то их совокупность идентифицирует линейное многообразие размерности $n < N$, в котором происходит поиск. Это многообразие соответствует фиксации части компонент y_i вектора y на граничных значениях. Перед началом поиска, в предположении, что начальная точка y^0 выбрана внутри множества D из (3.1), множества J_a и J_b должны быть пусты.

Пусть в результате очередного шага, выполненного в направлении d^{k-1} , процесс поиска вышел на границу множества D в точке y^k . Пусть этот участок границы имеет размерность $n < N$. Для текущей точки y^k и направления d^{k-1} скорректируем множества J_a и J_b , включив в них номера переменных, по которым процесс поиска вышел, соответственно, на верхние или нижние границы их изменения. Формально определим правило коррекции следующим образом.

$$J_a := J_a \dot{\cup} \{i \in \{1, \dots, N\} : y_i^k = a_i; d_i^{k-1} < 0\}, \quad (3.2)$$

$$J_b := J_b \dot{\cup} \{i \in \{1, \dots, N\} : y_i^k = b_i; d_i^{k-1} > 0\}. \quad (3.3)$$

Первое изменение множеств J_a и J_b соответствует переходу процесса поиска из пространства размерности N на линейное многообразие меньшей размерности за счет фиксации дополнительных компонент y_i вектора y . Введем базис в пространстве свободных (нефиксированных) координат. Для этого из набора единичных координатных ортов e^1, \dots, e^N выделим те векторы, e^j для которых $j \notin (J_a \cup J_b)$. Составим из этих векторов матрицу W_k , используя их как вектор-столбцы

$$W_k = (e^{j_1}, \dots, e^{j_n}).$$

Введем вектор переменных y_w для возникшего линейного подпространства $R_w = R^n$. Переход процесса поиска на линейное многообразие из исходного пространства R^N равносителен замене переменных $y = y^k + W_k y_w$ с переходом к поиску в пространстве переменных y_w .

Вычисляя в старых переменных значения $\nabla f^k = \nabla f(y^k)$ и $\Gamma_k = \nabla^2 f(y^k)$, легко пересчитать их в соответствующие значения в новых переменных, используя известные соотношения

$$\tilde{N}_{y_w} f^k = (W_k)^T \tilde{N} f^k; \quad G_{y_w k} = (W_k)^T G_k W_k.$$

Нужно обратить внимание на то, что в пространстве новых переменных методы гладкой оптимизации должны быть запущены заново из точки $y_w^0 = 0$, соответствующей текущей точке поиска y^k . Заметим также, что если используемый метод включает рестарты, то при поиске минимума по переменным y_w эти рестарты необходимо выполнять через число шагов, согласованное с размерностью n многообразия поиска.

Например, для квазиньютоновских методов и метода сопряженных градиентов рестарты следует выполнять через n шагов.

Рассмотрим процесс поиска на многообразии. Выбор очередного направления поиска в переменных y_w происходит по обычным правилам, характерным для выбранного метода. Однако при реализации этих правил необходимо все векторы и матрицы использовать для размерности n нового пространства, т.е., в частности, вместо значений ∇f^k и Γ_k необходимо использовать $\tilde{N}_{y_w} f^k$ и $\Gamma_{y_w k}$.

Работа методов в пространстве переменных y_w имеет дополнительную специфику, связанную с тем, что в действительности метод решает исходную N -мерную задачу с двусторонними ограничениями, наличие которых влияет на правила выполнения шага методом. Допустим, метод находится в точке y^k и, согласно правилам применяемого алгоритма, в пространстве переменных y_w выбрано направление поиска d_w^k . Тогда (так же, как при учете произвольных линейных ограничений) выполняется пересчет направления d_w^k в направление d^k в исходном пространстве переменных:

$$d^k = W_k d_w^k.$$

После выбора направления поиска происходит перемещение в этом направлении. Фактически, смещение выполняется в многообразии, соответствующем множествам J_a и J_b . В зависимости от типа метода это перемещение выполняется либо с фиксированным коэффициентом одномерного шага $x = \text{const}$, либо за счет поиска минимума вдоль выбранного направления. В обоих случаях учитываются ограничения на свободные переменные.

Процедура одномерного поиска, приведенная в разделе 2.4, учитывает их автоматически. Если при ее выполнении точка $y^{k+1} = y^k + x^k d^k$, переместившись вдоль многообразия, выходит на границу области поиска по новым переменным, то их номера следует добавить в множества J_a или J_b , соответственно, дополнительно применив правила их коррекции (3.2), (3.3) для $k=k+1$.

Если же перемещение точки должно быть выполнено с фиксированным коэффициентом длины шага (как это происходит, например, в методе Ньютона), то, в случае выхода точки за пределы изменения переменных, коэффициент x длины шага уменьшается таким образом, чтобы точка $y^{k+1} = y^k + x^k d^k$ оказалась на границе множества D . Далее выполняется описанная выше коррекция множеств J_a и J_b .

Осталось рассмотреть случай, когда одномерное перемещение в пространстве переменных y_w не приводит к выходу точки y^{k+1} на новые фрагменты границы. В этом случае необходимо проверить условия возврата к многообразию или пространству более

высокой размерности, который может быть выполнен за счет исключения из множеств J_a или J_b номеров части переменных. Правила исключения для методов первого порядка следующие:

$$J_a := J_a \setminus \{i \in \hat{I} \mid J_a: \nabla f(y^k) / \nabla y_i < 0\}, \quad (3.4)$$

$$J_b := J_b \setminus \{i \in \hat{I} \mid J_b: \nabla f(y^k) / \nabla y_i > 0\}. \quad (3.5)$$

Условия исключения переменной в (3.4), (3.5) определяются тем, что в текущей точке поиска становится положительной проекция антиградиента функции f на внутреннюю (по отношению к области D) нормаль к гиперплоскости $y_i = a_i$ или $y_i = b_i$. При выполнении этого условия существует направление смещения с этой гиперплоскости внутрь области поиска, при котором функция f будет строго локально убывать. Таким образом, если в результате коррекции (3.4), (3.5) хотя бы одно из множеств J_a или J_b изменится, необходимо перейти к поиску в многообразии более высокой размерности, выполнив в нем рестарт метода из последней точки предшествующего поиска, аналогично тому, как это было описано выше. Поскольку все рассмотренные методы первого порядка при рестартах на первом шаге используют антиградиентное направление, то правила коррекции (3.4), (3.5) обеспечат для них строгое убывание функции на очередном шаге.

Для методов второго порядка следует вместо (3.4), (3.5) использовать другое правило коррекции, поскольку направление их шага при рестарте выбирается по общему для всех шагов правилу и не совпадает с направлением антиградиента. А именно, для метода Ньютона и Ньютона–Рафсона направление смещения в исходном пространстве определяется как $d^k = (\Gamma_k)^{-1}(-\nabla f^k)$, а для метода с модификацией матрицы Гессе — соотношением $d^k = (\bar{\Gamma}_k)^{-1}(-\nabla f^k)$, где $\bar{\Gamma}_k$ — модифицированная Γ_k .

Следовательно, правила (3.4), (3.5) заменяются для этих методов на

$$J_a := J_a \setminus \{i \in \hat{I} \mid J_a: d_i^k < 0\}, \quad (3.6)$$

$$J_b := J_b \setminus \{i \in \hat{I} \mid J_b: d_i^k > 0\}, \quad (3.7)$$

где d_i^k — i -я компонента вектора направления d^k , при указанном выше способе его выбора. В остальном для этих методов все действия выполняются аналогично предыдущему.

3.1.2. Учет двусторонних ограничений в методах прямого поиска

В методах прямого поиска способ учета двусторонних ограничений уникален для каждого из этих методов. Более того, некоторые из них не имеют общепринятых модификаций для задач с двусторонними ограничениями. Типичным примером является метод Нелдера–Мида. Некоторые же методы, например метод Хука–Дживса, напротив, легко обобщаются на такие задачи.

Модифицированный метод Хука–Дживса

Рассмотрим принципы модификации для метода Хука–Дживса. В нем выполняются действия только двух типов: шаг по образцу и построение конфигурации.

С учетом ограничений, шаг по образцу выполняется таким образом, что при выходе рабочей точки за границы множества D величина последнего смещения корректируется так, чтобы точка оказалась на границе D . При построении конфигурации в обычном методе координатные перемещения выполняются с шагом h . В модифицированном методе величины координатных перемещений не превосходят h , а в случае, если эти перемещения выводят из множества D , заменяются на перемещения до его границ. Кроме того, при наличии параллелепипедных ограничений у $\hat{I} D$ целесообразно встроить в метод преобразование множества D к единичному гиперкубу $0 \leq y \leq 1$. Фактически, это приведет к тому, что величина вспомогательного шага, ранее равная h , станет зависеть от номера координатного направления и вычисляться как доля h от длины соответствующего ребра параллелепипеда D .

Модифицированный метод Нелдера–Мида

Рассмотрим метод Нелдера–Мида. На первый взгляд, правила метода допускают аналогичный способ модификации. Однако при этом границы области поиска будут трансформировать правила отражения и растяжения симплекса, что потребует введения дополнительных правил. Плохой их выбор может приводить к быстрому вырождению симплекса в окрестности границ области.

Возможные варианты модификаций этого метода рассмотрены, например, в книге Ф. Гилла и У. Мюррея «Численные методы условной оптимизации» (1977, п. 7.13). Однако они представляются не вполне удачными. Поэтому в LocOpt включена другая версия метода для случая параллелепипедных ограничений, предложенная В. Стариковым [4]. В ней модифицированы правила отражения и растяжения, а также учтены возможности перехода процесса поиска на грани, с уменьшением размерности симплекса, и его возврата в пространство исходной размерности.

3.2. Методы учета ограничений общего вида

В этом разделе будут рассмотрены такие общие методы учета ограничений, как метод внешнего степенного штрафа и метод модифицированных функций Лагранжа.

3.2.1. Метод внешнего штрафа

Метод штрафов рассмотрим на примере задач с ограничениями-неравенствами (1.1)–(1.3), поскольку исследование задач именно такого типа поддерживает система LocOpt.

3.2.1.1. Общее описание и некоторые свойства

Семейство методов штрафных функций представлено в этом пункте методами внешнего штрафа. Информацию по методу внутреннего штрафа (метод барьеров) можно найти, например, в книге М. Базара, К. Шетти [1], раздел 9.3. Методы внешнего штрафа просты в реализации, обладают сходимостью при весьма слабых условиях и могут использоваться в учебных целях и при практических расчетах, не требующих высокой точности и экономичности. Их следует признать наиболее наглядными из общих методов учета ограничений. Методы штрафов играют особую роль в теории условной оптимизации, поскольку сама идея введения «штрафа» является весьма плодотворной и позволяет получить весьма эффективные методы. Штрафы по нарушению ограничений используются, например, при построении модифицированных функций Лагранжа, а дополнительные штрафы, построенные по невязке в условиях стационарности, применяются в методах точных гладких штрафных функций.

Введем понятие функции штрафа применительно к методу внешнего штрафа для задачи (1.1)–(1.3).

Определение. *Определенную на E функцию $H(y)$, удовлетворяющую условию*

$$H(y) = \begin{cases} 0, & y \in Y, \\ > 0, & y \in E \setminus Y, \end{cases} \quad (3.8)$$

где Y – допустимая область задачи, называют функцией (внешнего) штрафа.

Поскольку геометрическая структура множества Y не известна и лишь косвенно определяется через функции ограничений в (3.2), то штраф $H(y)$ можно определить в виде функции, зависящей от значений $g(y)$, т.е. $H(y) = H(g(y))$. Существуют различные способы задания этой функции. Наиболее часто используется *степенная функция штрафа* вида

$$H(y) = \sum_{j=1}^m \left(c_j \max \{ 0; g_j(y) - g_j^+ \} \right)^r, \quad (3.9)$$

где $c_1, \dots, c_m > 0$, $r > 0$.

Если считать, что функции g уже приведены к единому масштабу общей физической размерности за счет ранее выполненной нормировки, то коэффициенты c_i можно опустить. Функция $H(y)$ трактуется как «штраф», который накладывается на целевую функцию за нарушение ограничений.

Целевая функция задачи со штрафом строится в виде

$$Sg(y) = f(y) + g H(y) \quad (3.10)$$

с $g > 0$. Параметр g называют *коэффициентом штрафа*. Метод заключается в решении последовательности задач со штрафом вида (3.3) при $b = g_k$ для возрастающей (обычно — неограниченно возрастающей) последовательности чисел $g = g_k$

$$S_g(y) \rightarrow \min, y \in E. \quad (3.11)$$

Укажем на некоторые очевидные свойства метода штрафа.

Свойства. *А. При $r \leq 1$ функция $H(g)$ из (3.9) не является гладкой по переменным g .*

В. При $r > 1$ $H(g)$ становится непрерывно дифференцируемой, а при $r > 2$ — дважды непрерывно дифференцируемой по g .

С. При больших значениях g функции задачи со штрафом $S_g(y)$ становятся сильно овражными.

Под *овражностью* здесь понимается существование многообразий в пространстве переменных y , вдоль которых функция $S_g(y)$ изменяется много медленнее, чем при смещении вдоль направлений, локально ортогональных к ним.

Замечание 1. Относительно свойств А и В следует иметь в виду, что большинство методов локальной оптимизации, которые могли бы использоваться в сочетании с методом штрафов, являются весьма чувствительными к степени гладкости и нарушению гладкости. Использование негладких штрафов существенно сужает набор возможных методов локальной оптимизации.

Замечание 2. Если функция штрафа (3.9) является гладкой, то нельзя гарантировать, что решение задачи со штрафом (3.10), (3.11) при каком-либо конечном γ будет совпадать с решением исходной задачи (3.1)–(3.2). Поэтому для гладкой функции штрафа приходится рассматривать бесконечно возрастающую последовательность значений коэффициента штрафа.

На рис. 3.1 показано поведение функций задачи со штрафом в случае одного переменного и одного ограничения $g(y) \leq 0$ при показателе степени в штрафе $r=2$. Можно видеть изменение функции штрафа при увеличении коэффициента g , а также вид функции штрафной задачи $S_g(y)$ при одном из значений коэффициента штрафа.

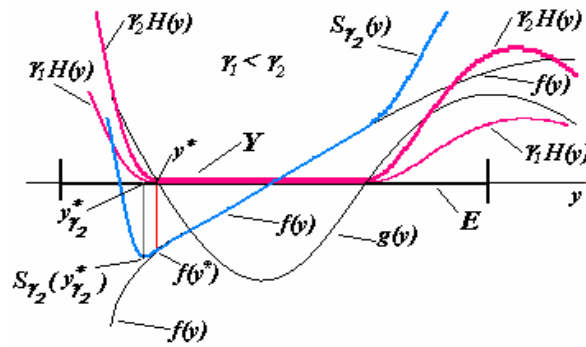


Рис. 3.1. Пример поведения функции штрафной задачи

Следует обратить внимание на то, что в задаче, представленной на рис. 3.1, за счет непрерывной дифференцируемости функции штрафа $H(y)$ ее производная $H'(y^*)$ равна нулю, и поэтому ни при каком конечном значении g точка минимума функции $S_g(y)$ не будет совпадать с решением исходной задачи (на рисунке это косвенно проявляется в том, что точка $y_g^* \neq y^*$ при любых значениях $g = g_2$).

При достаточно общих условиях можно обосновать сходимость процедуры метода штрафов при $g \rightarrow \infty$.

3.2.1.2. Условия сходимости метода штрафов и адаптивная настройка коэффициента штрафа

Изучая вопросы сходимости метода штрафа, следует учитывать, что решение задач (3.11) при $g = g_k$ всегда происходит с некоторой вычислительной погрешностью $e = e_k$. Предположим, что приближенные решения $y_g^*(e)$ удовлетворяют условию

$$S_g(y_g^*(e)) \leq S_g(y_g^*) + e. \quad (3.12)$$

Теорема (о достаточных условиях сходимости метода штрафов). Пусть выполнены следующие условия:

1. Функции $f(y)$, $g(y)$, $h(y)$ и $H(y)$ непрерывны на E .
2. Существуют решения $y^* \in U^*$ исходной задачи, а при достаточно больших $g > 0$ — решения y_g^* вспомогательных задач со штрафом.
3. Существует компакт $K \subseteq E$, что для достаточно больших g и малых e точки $y_g^*(e)$, определяющие приближенные решения задач со штрафом (согласно (3.12)), содержатся в K .
4. Используются значения $g = g_k \rightarrow \infty$, $e = e_k \rightarrow 0$ при $k \rightarrow \infty$, $g_k > 0$, $e_k \geq 0$.

Тогда предельные точки y^* и последовательности $y_{g_k}^*(e)$ являются решениями исходной задачи (3.1), (3.2) и существуют пределы

$$\lim_{k \rightarrow \infty} f(y_{g_k}^*(e_k)) = f(y^*), \quad (3.13)$$

$$\lim_{k \rightarrow \infty} r(y_{g_k}^*(e_k), Y^*) = 0, \quad (3.14)$$

где $r(\cdot)$ определяет расстояние от точки до множества в терминах используемой метрики пространства.

Можно сформулировать и доказать теоремы о локальной сходимости метода штрафов при использовании локальных методов поиска решений вспомогательных задач.

Для вычислительной реализации метода штрафов необходимо выбрать закон изменения g_k и e_k . При этом можно использовать конкретные числовые последовательности. Удобно использовать также адаптивные алгоритмы настройки коэффициента штрафа и точности решения штрафных задач. Эвристические алгоритмы адаптации способны формировать последовательности g_k , e_k в зависимости от специфики решаемой задачи. В частности, для негладкого штрафа такие алгоритмы могут автоматически подобрать коэффициент штрафа g , соответствующий точной штрафной задаче. Опишем один из возможных алгоритмов. Он основан на оценках скорости убывания невязки по ограничениям на каждом шаге. Именно этот алгоритм использован в системе LocOpt.

Невязкой в точке y назовем величину $G(y)$, показывающую степень нарушения ограничений в этой точке. Определим невязку с учетом нормировочных коэффициентов $c_j > 0$, использованных в функции штрафа

$$G(y) = \max \{ \max \{ c_j (g_j(y) - g_j^+): j=1, \dots, m \}; 0 \}, \quad (3.15)$$

ОПИСАНИЕ АДАПТИВНОГО АЛГОРИТМА НАСТРОЙКИ КОЭФФИЦИЕНТА ШТРАФА.

ШАГ 0. Задаются: $e_0 > 0$ — начальная точность решения штрафных задач, $e > 0$ — требуемая точность их решения, $d > 0$ — требуемая точность по ограничениям, $0 < a < 1$ — ожидаемый коэффициент убывания невязки по ограничениям на шаге, $b > 1$ — коэффициент увеличения штрафа, $b_1 > 1$ — дополнительный коэффициент увеличения штрафа, $0 < n < 1$ — коэффициент повышения точности решения штрафной задачи, $g = g_0$ — начальное значение коэффициента штрафа.

ШАГ 1. Решаем задачу со штрафом (3.11), (3.10) с точностью e_0 , получаем оценку решения $y_{g_0}^*(e_0)$. Вычисляем начальную невязку полученного решения по ограничениям

$G_0 = G(y_{g_0}^*(e_0))$. Полагаем $k = 0$ — номер выполненной итерации. Строим увеличенное значение коэффициента штрафа $g_{k+1} = b g_k$ и изменяем значение точности $e_{k+1} = n e_k$.

ШАГ 2. Проверяем критерий останова: если $G_k < d$ и точность решения задачи $e_k \leq e$, то выполняем останов процесса решения. Если $G_k < d$, но точность решения задачи $e_k > e$, то полагаем $e_{k+1} = e$ и переходим на шаг 3, если же $G_k \geq d$ — сразу переходим на шаг 3.

ШАГ 3. Решаем задачу со штрафом (3.11), (3.10) при $g_k = g_{k+1}$ с точностью e_{k+1} , получаем оценку решения $y_{g_{k+1}}^*(e_{k+1})$. Вычисляем невязку полученного решения по ограничениям $G_{k+1} = G(y_{g_{k+1}}^*(e_{k+1}))$.

ШАГ 4. Если $G_{k+1} = 0$, сохраняем $g_{k+2} = g_{k+1}$. Если $0 < G_{k+1} < a G_k$, то полагаем $g_{k+2} = b g_{k+1}$, а при $G_{k+1} \geq a G_k$: $g_{k+2} = b_1 b g_{k+1}$. Повышаем точность $e_{k+2} = n e_{k+1}$, Полагаем $k := k + 1$. Возвращаемся на шаг 2.

Описанный алгоритм формально обеспечивает выполнение требований теоремы сходимости, однако необходимо иметь в виду, что решение задачи с заданной точностью не гарантируется методами локального уточнения решений, которые обычно используются на шаге 3. Кроме того, при повышении требований к точности в малой окрестности решения начинают сказываться ошибки конечноразрядной арифметики. Поэтому при практических расчетах не следует уменьшать e_k более некоторого порогового значения, тем более, что метод штрафов может сходиться достаточно медленно и его вряд ли стоит использовать для окончательного получения решения с высокой точностью.

При проведении вычислений необходимо также учитывать возможные грубые ошибки в работе численных методов, в результате которых вместо определения глобального минимума задач со штрафом происходит определение их локального минимума, не являющегося глобальным. В частности, текущая оценка решения может оказаться в окрестности локального минимума $S_g(y)$, порожденного локальным минимумом функции штрафа $H(y)$. В этом случае при увеличении коэффициента штрафа g_k невязка по ограничениям в новой штрафной задаче не будет уменьшена. При этом следует прервать вычисления и попытаться подобрать другой вычислительный метод для поиска решения штрафных задач.

3.2.1.3. Структура задач со штрафом и характер поведения оценок

Для того, чтобы можно было прогнозировать характер поведения вычислительных методов при решении вспомогательных задач со штрафом (3.11), необходимо изучить характерные особенности, имеющиеся в структуре $S_{g_k}(y)$ — функций штрафных задач. В начале данного пункта проведем неформальное обсуждение возникающих вычислительных особенностей. Они могут быть обусловлены несколькими причинами.

Первая связана с использованием больших значений коэффициента штрафа. Это приводит к тому, что в задачах, имеющих решение на границе допустимого множества Y , будут возникать функции $S_{g_k}(y)$ с сильно овражной структурой (плохо обусловленные).

Причина заключается в том, что минимизируемая функция $f(y)$ исходной задачи, в общем случае, изменяется вдоль границы допустимого множества относительно медленно, а функция штрафа постоянна: $H(y)=0$. Если же точка начинает удаляться от границы в область недопустимости, то, за счет больших значений коэффициента штрафа, функция $S_{g_k}(y)$ будет быстро изменяться. На рис. 3.2 приведен пример, иллюстрирующий эту ситуацию. На нем показано изменение линий равного уровня функции $S_g(y)$ штрафной задачи вида (1.1)–(1.2), возникающей при поиске минимума функции $(y_1-0,1)^2+0,1(y_2-0,8)^2$ с ограничением $-9y_1^2-y_2^2 \leq -1$ при увеличении коэффициента штрафа со значения $g=1$ до значения $g=20$. В функции штрафа использован показатель степени $r=2$. Допустимое множество выделено более темным оттенком.

Вторая особенность связана с тем, что функция штрафа $H(y)$ из (3.9), аддитивно входящая в функцию $S_{g_k}(y)$, может иметь нарушение гладкости, начиная с некоторого порядка, даже при гладких функциях ограничений-неравенств. Например, у функции штрафа вида (3.9) при показателе степени $0 < r \leq 1$ на границе нарушения ограничений все частные производные, в общем случае, будут разрывны, а при $1 < r \leq 2$ первые производные станут непрерывными, но производные более высокого порядка будут терпеть разрыв. Нарушение гладкости может отрицательно сказываться на работе методов локальной оптимизации при решении задач методом штрафных функций. В то же время повышение гладкости штрафа ухудшает скорость сходимости метода штрафов за счет того, что вблизи границы Y штрафная добавка становится бесконечно малой более высокого порядка, чем расстояние до границы допустимого множества. Ситуация усложняется тем, что именно в окрестности этой границы (если решение исходной задачи (1.1)–(1.3) лежит на границе множества Y) функция $S_{g_k}(y)$ сильно «овражна» при больших

значениях g_k . Таким образом, при выборе вычислительного метода оптимизации, используемого в методе штрафов, необходимо учитывать возможное нарушение гладкости вблизи дна оврага, вдоль которого обычно выполняется поиск минимума. Заметим, что приведенные здесь рассуждения, носящие качественный характер, можно подкрепить точными оценками скорости сходимости метода штрафов, увязав их со значением показателя степени r в функции штрафа. Эти оценки приводятся ниже в соответствующей теореме.

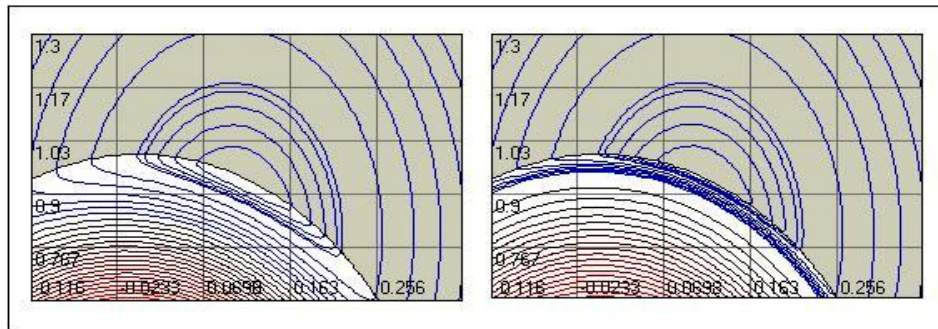


Рис. 3.2. Увеличение овражности функции задачи со штрафом $S_g(y)$ при возрастании коэффициента штрафа g с 1 до 20

Еще одна особенность функции $S_{g_k}(y)$ связана с тем, что порядок ее роста при отклонениях от границы Y может быть существенно различен, в зависимости от того, происходит отклонение внутрь множества Y или вне. Эта особенность может оказаться существенной для большой группы методов, основанных на квадратичной модели минимизируемой функции (модифицированный метод Ньютона, квазиньютоновские методы, метод сопряженных градиентов).

Замечание. Таким образом, функции штрафных задач обладают характерными особенностями, ухудшающими поисковые возможности вычислительных процедур локальной оптимизации. В разделе 3.3 будут рассмотрены методы учета ограничений, в значительной степени лишенные этих недостатков.

Свойство. Если $Y^* \subseteq \text{int } Y$ и штрафные задачи решаются точно (т.е. $e_k = 0$), то $\exists k$, что $y_{g_k}^* \in Y^*$.

Свойство. Если в методе штрафов последовательность коэффициентов штрафа образует возрастающую последовательность $g_{k+1} > g_k$, то последовательность значений $f_k = f(y_{g_k}^*)$ будет неубывающей, а последовательность $H_k = H(y_{g_k}^*)$ — невозрастающей:

$$f(y_{g_{k+1}}^*) \geq f(y_{g_k}^*), \quad H(y_{g_{k+1}}^*) \leq H(y_{g_k}^*). \quad (3.16)$$

Заметим, что в том случае, когда решение исходной задачи размещается на границе допустимого множества U и не является точкой безусловного минимума целевой функции, из теоремы следует, что решения вспомогательных штрафных задач приближаются к точке решения извне этого множества.

Рассмотрим еще один важный аспект — возможность конечной верхней оценки для коэффициента штрафа, при допущении ненулевой малой невязки в ограничениях по значению функции штрафа.

Свойство. Пусть известна хотя бы одна допустимая точка $y_U \in U$ и $\forall k: g_{k+1} \geq g_k$. Если в качестве критерия останова в методе штрафов принять выполнение условия $H(y_{g_k}^*) \leq e_H$ ($e_H > 0$), то будет существовать конечная оценка значения коэффициента штрафа g^* , такая, что при

$$g_k \geq g^* = (f(y_U) - f(y_{g_0}^*)) / e_H \quad (3.17)$$

точное решение задачи со штрафом будет удовлетворять указанному критерию останова.

Чтобы получить этот результат достаточно записать оценку, используя предыдущее свойство.

$$H(y_{g_k}^*) = \frac{(S_{\gamma_k}(y_{g_k}^*) - f(y_{g_k}^*))}{\gamma_k} \leq \frac{(f(y_U) - f(y_{g_k}^*))}{\gamma_k} \leq \frac{f(y_U) - f(y_{g_0}^*)}{\gamma_k}.$$

Потребуем, чтобы данная оценка не превосходила e_H . Это будет достигаться " $g_k \geq g^*$.

3.2.1.4. Оценки скорости сходимости метода внешнего штрафа

Сделаем дополнительные предположения о свойствах задачи.

A. Отсутствует погрешность решения задач со штрафом ($e_k \equiv 0$).

B. Используется степенной штраф вида (3.9) с $r > 0$, а невязка в ограничениях оценивается функцией $G(y)$ из (3.15).

C. Множество E — замкнуто, а f — липшицева на E с константой L в метрике $r(\cdot)$.

D. $\exists d > 0$ и $a > 0$, что $\forall y \in (O_d(U) \setminus U) \cap E$ выполняется $G(y) \geq a \chi(y, U)$.

Будем измерять ошибку оценивания решения y^* исходной задачи (1.1)–(1.3) через решение y_g^* вспомогательной задачи (3.11), используя разность значений функций исходной и вспомогательной задач. Запишем величину ошибки

$$D(g) = f(y^*) - S_g(y_g^*). \quad (3.18)$$

Заметим, что всегда $\Delta(g) \geq 0$.

Теорема (об оценке скорости сходимости). Если выполнены условия 1–4 теоремы сходимости и, кроме того, — дополнительные предположения A–D, то при точном решении вспомогательных задач со штрафом;

а) при $r \notin 1$ существует достаточно большое \bar{g} , что $\forall g \geq \bar{g}$ выполняется $D(g) = 0$;

в) при $r > 1$ и достаточно больших $g^0 \leq \Delta(g) \leq C / g^{1/(r-1)}$, где

$$C = (1 - 1/r)(L/a)^{r/(r-1)} / r^{1/(r-1)}. \quad (3.19)$$

Достаточные условия выполнения требования D приведены в форме леммы.

Лемма. Если $g_j(y)$ для $j=1, \dots, t$ выпуклы и непрерывны на выпуклом компакте E , множество $\{y \in E: g(y) \leq 0\}$ удовлетворяет условию Слейтера: $\exists \bar{y} \in E$, что $g(\bar{y}) < 0$, и, кроме того, для ограничений-равенств $\forall y \in E$ при $h(y) = 0$, выполняется: $h_s(y)$ ($s = 1, \dots, p$) — непрерывно дифференцируемы на E , $\tilde{N}h_s(y) \neq 0$, ($s = 1, \dots, p$), то $\exists d > 0$, $a > 0$, что $\forall y \in O_d(U) \setminus U \cap E$ невязка $G(y) \geq a\chi(y, Y)$.

Замечание. В условиях пункта (а) теоремы о сходимости метода штрафов решение задачи со штрафом при любом конечном $g > \bar{g}$ совпадает с решением исходной задачи. В этом случае говорят, что $gH(y)$ является точным штрафом. Данный штраф является негладким и требует специальных методов недифференцируемой оптимизации.

3.2.1.5. Недостаточность локальных методов при использовании метода штрафов

В теореме об условиях сходимости метода штрафов предполагается, что при решении каждой задачи со штрафом определяется $y_{g_k}^*(e_k)$ — оценка глобального минимума штрафной задачи. Поскольку методы многоэкстремальной оптимизации требуют значительно большего объема вычислений, чем методы локальной оптимизации, то при практических расчетах обычно прибегают к следующему приему. На первой итерации метода штрафов для вычисления $y_{g_0}^*(e_0)$ используют один из методов многоэкстремальной оптимизации, а на следующих итерациях для получения оценок $y_{g_{k+1}}^*(e_{k+1})$ ($k=1, 2, \dots$) прибегают к методам локальной оптимизации, в которых в качестве начальных точек поиска используются точки $y_{g_k}^*(e_k)$, найденные на предыдущей итерации. Таким образом, полностью отказываться от применения методов глобального поиска нельзя, они необходимы хотя бы на первой итерации метода штрафов.

Если начальная точка выбрана неправильно, предельная точка последовательности испытаний могут не совпадать даже с локальными решениями исходной задачи. При этом

будет наблюдаться неограниченный рост значений функции $S_{g_k}(y)$, вычисляемых в точках $y_{g_k}^*(e_k)$, при $k \rightarrow \infty$.

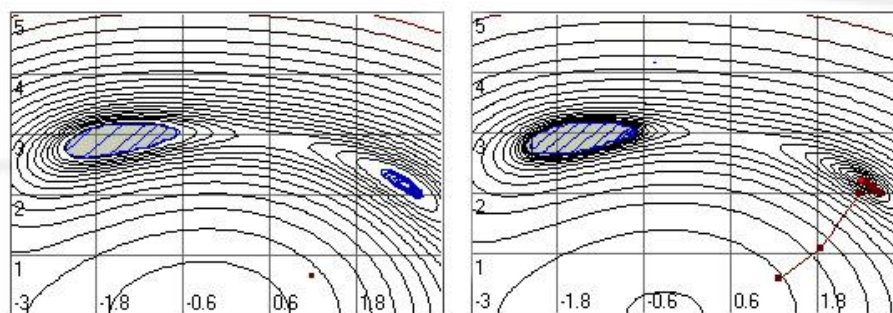


Рис. 3.3. Возможность потери решения при наличии локального минимума у функции штрафа

На рис. 3.3 приведен пример описанной выше ситуации. В этом примере решается задача поиска минимума линейной функции $-y_1 + y_2$ в прямоугольной области $-3 \leq y_1 \leq 3$, $0 \leq y_2 \leq 5$ при дополнительном ограничении $(y_1^2 + y_2^2 - 11)^2 + (y_1 + y_2 - 7)^2 + (y_1 - y_2) \leq 0,3$. Допустимая область односвязна и выделена на рисунках серым цветом.

Функция штрафа в этом примере имеет локальный минимум, расположенный в недопустимой области исходной задачи в окрестности точки $y_1 = 2,45$, $y_2 = 2,15$. При сколь угодно большем увеличении коэффициента штрафа у функции штрафной задачи сохраняется локальный минимум в окрестности этой точки. На рисунке показано поведение метода штрафов в сочетании с методом прямого поиска Хука–Дживса, запущенного из начальной точки $(1,10; 0,65)$, расположенной в области притяжения этого минимума. Видно, что правильное решение оказалось потерянным. Левый рисунок соответствует выбору начального значения коэффициента штрафа $g = 10$, а правый — значению $g = 1000$, которое было адаптивно выбрано методом на одной из итераций процесса поиска решения.

3.2.1.6. Метод штрафов и оценка множителей Лагранжа

Рассматривая метод штрафов, нельзя не остановиться еще на одном важном свойстве, связанном с возможностью оценок множителей Лагранжа для оптимальной точки через найденные методом оценки решения по переменным y . Такая особенность позволяет использовать метод штрафов для определения начального приближения как по y , так и по значениям множителей Лагранжа I для локально сходящихся исходно-двойственных

методов, выполняющих итерации в расширенном пространстве переменных y, I (примером может являться метод модифицированных функций Лагранжа

Оценки множителей Лагранжа могут быть получены при некоторых достаточно общих предположениях относительно свойств решаемой задачи.

Дополнительные предположения о задаче (1.1)–(1.3).

1. В соответствующей задаче со штрафом (3.9)–(3.11) нет ограничений вида $y \in D$ на переменные, т.е., фактически, $D = R^N$.
2. Функции исходной задачи непрерывно дифференцируемы, т.е. $f, g \in C^1(R^N)$.
3. Используется функция квадратичного степенного штрафа (3.9) с $r = 2$.
4. В допустимых точках $y \in Y$ из (1.2) выполняется достаточное условие регулярности в форме независимости градиентов

$$\nabla g_{j_1}(y), \dots, \nabla g_{j_s}(y),$$

где $\{j_1, \dots, j_s\} = J(y)$ — номера активных в точке y ограничений.

Справедлива следующая теорема.

Теорема (об оценке множителей Лагранжа). Пусть выполнены условия теоремы о сходимости метода штрафов и, кроме того, дополнительные предположения 1–4. Тогда для всякой предельной точки y_∞^* (являющейся, в силу теоремы о сходимости, глобально оптимальной) и сходящейся к ней подпоследовательности $y_{g_k}^*$ ($k = k_s, s \rightarrow \infty$) решений задач со штрафом будут существовать пределы:

$$I_j^* = \lim_{k=k_s, s \rightarrow \infty} 2 \cdot g_k \cdot \max\{0, c_j^2 (g_j(y_{g_k}^*) - g_j^+)\} \quad (j = 1, \dots, m), \quad (3.20)$$

образующие единственный набор I^* множителей Лагранжа, соответствующий решению исходной задачи $y^* = y_\infty^*$.

3.2.2. Метод модифицированных функций Лагранжа

Методы модифицированных функций Лагранжа проще построить для задач с ограничениями-равенствами. В задачах, включающих неравенства, могут использоваться различные подходы. Один из подходов состоит в преобразовании неравенств к форме ограничений-равенств за счет временного добавления вспомогательных переменных, которые затем исключаются из окончательной записи метода.

3.2.2.1. Преобразование к форме задачи с ограничениями-равенствами

В целях удобства дальнейшего изложения будем считать, что в (1.1)–(1.3) множество $D = R^N$, т.е. задача математического программирования поставлена так:

$$f(y) \text{ @ min, } y \hat{I} U = \{y \hat{I} R^N: g(y) \leq 0\}, \quad (3.21)$$

тем самым двусторонние ограничения на переменные, если они есть, переведены в разряд функциональных ограничений.

Задачу будем считать достаточно гладкой, что позволит записывать условия экстремума в дифференциальной форме. Наличие дополнительных требований вида $y \in D^{-1}R^N$ лишило бы нас возможности использовать эту форму записи в общем случае.

Избавимся от ограничений-неравенств, используя известный общий прием. Рассмотрим неравенство $g_i(y) \leq 0$. Введем дополнительные переменные z_i , чтобы $g_i(y) = -z_i^2$. Отсюда эквивалентной формой записи неравенства будет

$$g_i(y) + z_i^2 = 0.$$

Будем считать, что пространство расширено за счет добавления к y переменных z , и выполним замены

$$y := (y^T \mid \underset{z}{z}^T)^T \in R^{N+m}, \quad N := N + m.$$

Окончательно приходим к задачам, в которых присутствуют только ограничения-равенства

$$f(y) \text{ @ } \min, \quad y \in U = \{y \in R^N : h(y) = 0\}. \quad (3.22)$$

Функция Лагранжа будет иметь вид

$$L(y, m) = f(y) + (m, h(y)). \quad (3.23)$$

3.2.2.2. Построение метода модифицированных функций Лагранжа для задач с ограничениями-равенствами

Методы модифицированных функций Лагранжа можно рассматривать как достаточно удачный симбиоз метода штрафов, методов, опирающихся на поиск решений условий оптимальности (методы Лагранжа), а также двойственных методов.

Применительно к гладкой регулярной задаче с ограничениями-равенствами (3.22) пара (y^*, m^*) , соответствующая ее локальному минимуму в точке y^* , является стационарной точкой функции Лагранжа (3.23). Из теории условий оптимальности известно, что при весьма жестких требованиях к задаче (включая аффинность функции h , выпуклость f) пара (y^*, m^*) становится седловой точкой функции Лагранжа и может быть определена как решение максиминной задачи

$$L(y^*, m^*) = \max_{m \in R^p} \min_{y \in R^N} L(y, m).$$

В общем случае седловой точки y функции Лагранжа не существует, поскольку это условие не является необходимым для оптимальности, и метод, основанный на решении максимальной задачи, становится несостоятельным.

Чтобы исправить рассмотренную ситуацию, следует так изменить функцию Лагранжа, чтобы модифицированная функция сохранила все основные свойства обычной функции

Лагранжа, но для нее точка (y^*, m^*) стала хотя бы локально седловой. Поскольку на допустимом множестве, где $h(y) = 0$, функция Лагранжа совпадает с целевой функцией $f(y)$, т.е. $L(y, m) \equiv f(y)$, $y \in Y$, то на нем функция Лагранжа имеет минимум в точке решения задачи с ограничениями-равенствами (3.22). Следовательно, необходимо изменить функцию Лагранжа только на множестве, где нарушаются ограничения-равенства.

Наиболее простой вариант состоит в построении модифицированной функции за счет добавления квадратичного штрафа к обычной функции Лагранжа. Будем использовать

$$L_g(y, m) = L(y, m) + g / 2 \cdot \|h(y)\|^2. \quad (3.24)$$

ОБЩАЯ СТРУКТУРА МЕТОДА

ШАГ 0. Задаются начальные приближения y^k, m^k при $k = 0$ и $g_k > 0$.

ШАГ 1. Определяется y^{k+1} из решения задачи

$$y^{k+1} \in \text{Arg min}\{L_{g_k}(y, m^k); y \in R^N\}. \quad (3.25)$$

ШАГ 2. Корректируются значения множителей Лагранжа по некоторому правилу

$$\mu^{k+1} = P(y^{k+1}, \gamma_k, \mu^k). \quad (3.26)$$

ШАГ 3. Выбирается $g_{k+1} \geq g_k$, полагается $k := k + 1$ и происходит возврат на шаг 1.

Правило коррекции m^k должно обеспечивать сходимость оценок m^k к значению m^* , а y^k к y^* . Если в данном методе не изменять значения g , то сходимость (при определенных условиях) может быть обеспечена исключительно за счет коррекции множителей Лагранжа. Если же неограниченно увеличивать g , то метод становится похож на метод штрафных функций, в котором штраф добавлен не к функции f , а к функции Лагранжа исходной задачи. Настройка параметров m должна привести к тому, что неограниченный рост коэффициентов штрафа может стать необязательным, что улучшит обусловленность вспомогательных задач (3.26).

Вернемся к постановке (3.22), т.е. будем считать, что задача представлена в форме задачи с равенствами. Наложим на нее дополнительные ограничения, гарантирующие существование в точке y^* строгого локального минимума f на Y . Из теории условий оптимальности вытекают следующие требования.

Требования к задаче (3.22)

А. Пусть y^* — локальное решение задачи (3.22), функции f , h дважды дифференцируемы по крайней мере в некоторой окрестности точки y^* и их вторые производные непрерывны в этой точке.

В. В точке y^* выполнено достаточное условие регулярности в форме линейной независимости градиентов $\nabla h_1(y^*), \dots, \nabla h_p(y^*)$.

С. Для значения m^* множителей Лагранжа, однозначно определяемого из условия стационарности

$$\nabla_y L(y^*, m^*) = 0, \quad (3.27)$$

дополнительно выполняется условие оптимальности второго порядка:

$$d^T \nabla_{yy}^2 L(y^*, m^*) d > 0, \quad \text{что } (d, \tilde{N} h_s(y^*)) = 0 \quad (s = 1, \dots, p). \quad (3.28)$$

При этих предположениях y^* будет являться строгим локальным минимумом в задаче (3.22). Условие (3.28) можно трактовать как условие положительной определенности матрицы Гессе по переменным y для функции Лагранжа $L(y, m^*)$ на линейном многообразии касательных направлений

$$K_H(y^*) = \{d \hat{I} R^N : (d, \tilde{N} h_s(y^*)) = 0 \quad (s = 1, \dots, p)\}.$$

Заметим, что модифицированная функция Лагранжа должна иметь при $m = m^*$ в точке y^* матрицу Гессе (гессиан) по переменным y , положительно определенную на всем R^N .

Будем использовать модифицированную функцию Лагранжа $L_g(y, m)$, добавив к $L(\cdot)$ слагаемое, возрастающее с увеличением невязки по ограничениям-равенствам. Запишем ее вид еще раз

$$L_g(y, m) = f(y) + (m, h(y)) + \frac{1}{2} g \|h(y)\|^2. \quad (3.29)$$

Лемма. Для задачи (3.22), удовлетворяющей предположениям А, В и С, при достаточно большом g модифицированная, согласно (3.29), функция Лагранжа при $m = m^*$ имеет в точке y^* строгий локальный минимум по y .

Для получения конкретной формы метода необходимо построить правила коррекции оценок множителей Лагранжа. С этой целью запишем условие оптимальности для вспомогательной задачи (3.25) в точке y^{k+1}

$$\nabla f(y^{k+1}) + \nabla h(y^{k+1}) m^k + g_k \nabla h(y^{k+1}) \cdot h(y^{k+1}) = 0.$$

Если бы y^{k+1} была точкой решения исходной задачи, то соответствующие множители Лагранжа можно было бы определить из решения системы

$$-\nabla f(y^{k+1}) = \sum \tilde{m}_s \cdot \nabla h_s(y^{k+1}).$$

При сделанных в пункте (В) допущениях \tilde{m} определяются из этой системы единственным образом. Сравнивая ее с предыдущим равенством, видим, что

$$\tilde{m} = m^k + g_k h(y^{k+1}).$$

Отсюда возникает итерационное правило

$$m^{k+1} = m^k + g_k h(y^{k+1}).$$

Последовательность коэффициентов штрафа g_k может выбираться заранее, а может строиться адаптивно по некоторому эвристическому алгоритму.

Таким образом, *метод модифицированной функции Лагранжа* может быть представлен в виде следующего итерационного соотношения:

$$y_{k+1} = \arg \min \{L_{g_k}(y, m^k): y \in R^N\} \quad (3.30)$$

$$m^{k+1} = m^k + g_k h(y^{k+1}), \quad (3.31)$$

$$g_{k+1} \geq g_k.$$

В отличие от метода штрафов, последовательность g_k не обязана стремиться к бесконечности. Сходимость может быть обеспечена за счет настройки множителей m^k .

Если известно хорошее начальное приближение по y^0 , то приближение по m^0 можно получить с помощью соотношения:

$$m^0 = (\nabla h(y^0))^T \nabla h(y^0)^{-1} (\nabla h(y^0))^T \nabla f(y^0). \quad (3.32)$$

Заметим также, что хорошее приближение по y^0 , m^0 можно получить, воспользовавшись на первом этапе методом внешнего штрафа при специальном критерии останова.

Вычислительная практика показывает, что метод модифицированных функций Лагранжа часто сходится при достаточно плохих начальных приближениях и весьма умеренной начальной величине коэффициента штрафа.

3.2.2.3. Метод модифицированной функции Лагранжа в задачах с ограничениями-неравенствами

Вернемся от задачи (3.22) к задаче с неравенствами (1.1)–(1.3). Воспользуемся в явном виде рассмотренным ранее приемом сведения ограничений-неравенств к форме равенств $g_i(y) - g_i^+ + z_i^2 = 0 \ (i=1, \dots, m)$.

Построим модифицированную функцию Лагранжа в виде

$$L_g(y, z, l) = f(y) + \sum_{i=1}^m \left\{ l_i \cdot (g_i(y) - g_i^+ + z_i^2) + 0.5g(g_i(y) - g_i^+ + z_i^2)^2 \right\}. \quad (3.33)$$

Минимум по переменным (y, z) разделяется в последовательное взятие минимумов по z и по y . Найдем минимум по z в (3.33) аналитически

$$(z_i^*)^2 = \max\{0; -I_i/g - g_i(y) + g_i^+\}.$$

Подставив это выражение в (3.33), и выполнив несложные преобразования получим следующий вид модифицированной функции Лагранжа:

$$L_g(y, I, m) = f(y) + (1/(2g)) \sum_{i=1}^m ((\max\{0; g \cdot (g_i(y) - g_i^+) + I_i\})^2 - I_i^2). \quad (3.34)$$

Кроме того,

$$\partial L(y, z^*, m) / \partial I_i = g_i(y) - g_i^+ + (z_i^*)^2 = \max\{g_i(y) - g_i^+; I_i/g\}.$$

Это соотношение позволяет распространить правило коррекции оценки множителей Лагранжа, полученное выше для ограничений-равенств, на неравенства:

$$\begin{aligned} I_i^{k+1} &= I_i^k + g_k \cdot \max\{-I_i^k/g_k; g_i(y^{k+1}) - g_i^+\} = \\ &= \max\{0; I_i^k + g_k \cdot (g_i(y^{k+1}) - g_i^+)\}. \end{aligned}$$

Вычислительная схема метода приобретает следующий вид:

$$y^{k+1} = \arg \min \{L_{\gamma_k}(y, \lambda^k): y \in \mathbb{R}^N\}, \quad (3.35)$$

$$I_i^{k+1} = \max\{0; I_i^k + g_k g_i(y^{k+1})\}, \quad (3.36)$$

$$g_{k+1} \Leftarrow g_k.$$

Для построенного метода можно сформулировать теорему об условиях локальной сходимости и характере сходимости [5]. Эта теорема может быть сформулирована при следующих предположениях.

Предположения о задаче (1.1)-(1.3)

А Пусть y^* — локальное решение задачи (1.1)-(1.3) с ограничениями-неравенствами при $D = \mathbb{R}^N$, функции f, g дважды дифференцируемы по крайней мере в некоторой окрестности точки y^* и их вторые производные непрерывны в этой точке.

В В точке y^* выполнено достаточное условие регулярности в форме линейной независимости градиентов $\nabla h_1(y^*), \dots, \nabla h_p(y^*), \nabla g_j(y^*), j \in J(y^*)$.

С Для значений I^* множителей Лагранжа, однозначно определяемых из условия стационарности

$$\nabla_y L(y^*, I^*) = 0,$$

дополнительно выполняется условие оптимальности второго порядка:

$$d^T \nabla_{yy}^2 L(y^*, m^*) d > 0, \quad \text{где } d \neq 0, \text{ что } (d, \tilde{\nabla} h_s(y^*)) = 0 \ (s = 1, \dots, p), \ (d, \nabla g_j(y^*)) = 0, \ j \in J(y^*):$$

Приведем формулировку теоремы.

Теорема (о сходимости). Пусть для задачи (1.1)–(1.3) с ограничениями-неравенствами выполнены предположения **А** и **В** и **С**. Тогда $\exists \bar{\gamma} > 0, \delta > 0, T > 0$ и окрестность $C(y^*)$ точки y^* , что при $(I^k; g_k) \in \Theta(d, \bar{g})$, где

$$\Theta(d, \bar{g}) = \{(I; g): \|I - I^*\| < d \cdot g, \ g \geq \bar{g}\},$$

для метода модифицированной функции Лагранжа (3.35)–(3.36) выполнено:

1) вспомогательные задачи (3.35) имеют в окрестности $S(y^*)$ единственную стационарную точку, соответствующую y^{k+1} ;

2) правила коррекции (3.35)–(3.36) определяют оценки I^{k+1} так, что выполняются неравенства:

$$\begin{aligned}\|y^{k+1} - y^*\| &\leq T \cdot (\|I^k - I^*\|) / g^k, \\ \|I_i^{k+1} - I_i^*\| &\leq T \cdot (\|I^k - I^*\|) / g^k;\end{aligned}$$

3) найдется значение $\bar{\delta} \in (0; \delta]$, что при соблюдении условий $(I^0; g^0) \in \Theta(\bar{d}, \bar{g})$, $\bar{g} \leq g_k \leq g_{k+1} - \forall k$, на всех последующих шагах метода выполнится:

$$(I^{k+1}; g_{k+1}) \in \Theta(d, \bar{g});$$

кроме того, $I^k \rightarrow I^*$ и $y^k \rightarrow y^*$ при $k \rightarrow \infty$, причем обеспечивается линейная скорость сходимости по множителям Лагранжа, а при $g_k \rightarrow \infty$ сверхлинейная скорость сходимости.

В (3.35)–(3.36) возможна адаптивная настройка коэффициента штрафа, выполняемая в зависимости от скорости убывания (в результате выполненной итерации метода) невязки в системе условий оптимальности.

Применение метода может происходить при постоянном значении коэффициента штрафа, при заданном законе его изменения, а также с применением алгоритмов его адаптивной настройки. Здесь может быть применен подход, похожий на используемый в методе штрафов. Однако при его модификации для метода модифицированных функций Лагранжа могут возникать особенности, связанные с тем, что на промежуточных этапах поиска текущая точка может становиться допустимой, а невязка в ограничениях — обращаться в ноль. Кроме того, всегда необходимо учитывать полную невязку в условиях оптимальности, а не только невязку в ограничениях.

АЛГОРИТМ АДАПТИВНОЙ НАСТРОЙКИ КОЭФФИЦИЕНТА ШТРАФА

Введем параметр $a < 1$ — коэффициент желаемого уменьшения невязки в условиях оптимальности в результате выполнения шага методом, а также множитель $b > 1$ прироста коэффициента штрафа, например, $a = 0,25$, $b = 10$. Невязку можно принять в виде следующей функции:

$$V(y, I) = \|\nabla_y L(y, I)\| + \|\max\{0; g(y) - g^+\}\| + |(I, g(y) - g^+)|.$$

При этом возможна следующая схема адаптивной настройки параметра,

$$g_{k+1} = \begin{cases} b \cdot g_k, & V(y^{k+1}, I^{k+1}) > a \cdot V(y^k, I^k) \\ g_k, & V(y^{k+1}, I^{k+1}) \leq a \cdot V(y^k, I^k). \end{cases} \quad (3.37)$$

В качестве критерия останова можно использовать правило $V(y^k, I^k) \leq e$.

Именно такая стратегия адаптивной настройки коэффициента штрафа применяется в системе LocOpt версии 3.

Заметим, что в случае достаточно быстрого убывания невязки при адаптивной настройке коэффициент штрафа возрастать не будет. При численной реализации метода могут возникнуть специфические проблемы, связанные с ухудшением реальной точности решения вспомогательных задач (3.30) за счет ухудшения их обусловленности при достаточно больших значениях коэффициента штрафа. Как правило, начиная с некоторого g_k , невязка при вычислениях уже не убывает. Напротив, начинается ее рост, хотя и не обязательно монотонный. Это свидетельствует об использовании неоправданно завышенных значений коэффициента штрафа. Проверку такого поведения невязки можно включить в способ останова вычислительного метода.

В заключение отметим, что методы модифицированных функций Лагранжа являются весьма эффективным средством практического решения задач условной локальной оптимизации.

4. ОБЩИЕ ПРИНЦИПЫ ПОСТРОЕНИЯ И ПРАВИЛА ИСПОЛЬЗОВАНИЯ СИСТЕМЫ LocOpt

4.1. Наборы функций и списки задач

Задачи оптимального выбора, исследование которых поддерживается системой, включают в свое описание одну или несколько функций: минимизируемую (целевую) функцию, а также функции ограничений, если они присутствуют в постановке задачи. Пользователь может разрабатывать и сохранять в специальной папке свои собственные наборы функций.

Замечание. *Наборы функций* используются для того, чтобы упростить конструирование задач оптимизации с требуемыми свойствами.

Конструирование выполняется по технологии перетаскивания (Drag&Drop — перетаски и отпусти) в специальном диалоговом окне (конструктор задач) путем перетаскивания в поле целевой функции и поля функций-ограничений необходимых функций из некоторого набора. Созданные задачи объединяются в группы, называемые списками задач. Список задач можно сохранить в специальной папке, присвоив ему уникальное имя. В последующем пользователь может открыть список задач, просмотреть его и выбрать нужные задачи для исследования.

Замечание. *Списки задач* можно рассматривать как заранее подготовленные тематические коллекции, позволяющие быстро находить нужные задачи для проведения вычислительных экспериментов.

Более подробная информация содержится в пп. 4.1.1–4.1.2.

4.1.1. Наборы функций

Каждый набор функций сохраняется в виде одного двоичного файла специальной структуры с расширением “*.opt”. Все наборы функций приложение LocOpt размещает в папке Functions_Lists, расположенной внутри основной папки приложения. Вместе с системой поставляется стандартный набор функций, записанный в виде файла Standard.opt. Этот набор функций не может быть изменен, но система разрешает сохранить его под другим именем и использовать в качестве основы для создания новых наборов. На рис.4.1 приведен вид диалога по сохранению набора функций.

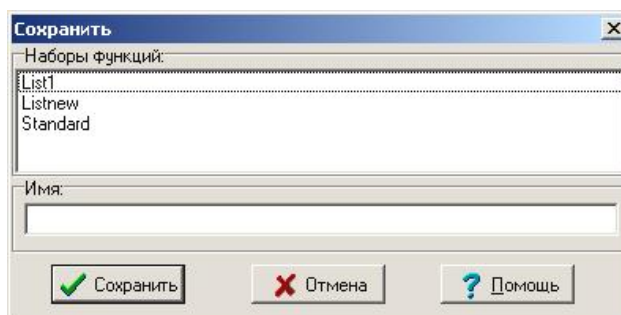


Рис. 4.1. Диалоговое окно сохранения набора функций под новым именем

Функции в наборах могут содержать параметры a_1 , a_2 , и т.д., которым придаются значения по умолчанию. С функциями связываются также области их определения. В качестве имен переменных можно использовать x_1 , x_2 , и т.д. С функциями в наборах можно выполнять различные операции: удалять из набора; добавлять в набор; изменять значения параметров и область определения; редактировать; просматривать линии уровня (рис. 4.2.)

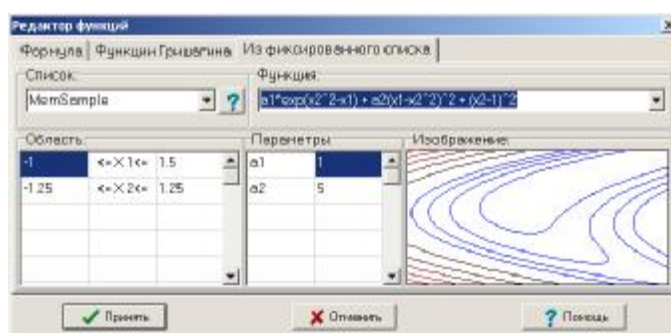


Рис. 4.2. Вид окна для определения-редактирования функции

Функции, которые можно использовать в наборах, разделяются на три типа в зависимости от способа их задания:

- функции, определяемые формульным описанием,
- набор случайно генерируемых функций (функции Гришагина),
- функции из фиксированных списков.
- функции из наборов, реализованных в форме DLL специальной структуры.

В последнем случае считается, что DLL реализует модель оптимизируемого объекта, который имеет набор параметров a_1 , a_2 , ... Эти параметры автоматически становятся параметрами всех функций-критериев, получаемых из DLL.

4.1.2. Списки задач

Списки задач размещаются в папке Tasks_Repository, расположенной в основной папке приложения LocOpt. Каждый список задач реализуется системой в виде папки, помещаемой в Tasks_Repository. Их имена могут быть любыми, кроме Standard, поскольку это имя резервируется системой для размещения стандартного набора задач.

Отдельные задачи сохраняются в списках в виде файлов со стандартными именами вида 0000000001.tsk, 0000000002.tsk, и т.д. Имена автоматически генерируются системой, причем пользователь никогда непосредственно не работает с этими файлами через их имена и может вообще не знать об их существовании.

С точки зрения пользователя существуют только именованные списки задач. Через главное меню приложения и панель инструментов можно загрузить в специальное диалоговое окно конструктора задач любой из списков задач. После этого возможна быстрая навигация по задачам набора, их удаление, просмотр, корректировка, отправка в окна экспериментов.

Вместе с системой поставляется стандартный список учебных задач, имеющий имя Standard. Система не позволяет изменять этот список как целое, однако любую задачу этого списка можно изменить и сохранить ее в другом списке, например, в новом списке. При этом используется диалоговое окно подобное тому, которое показано на рис. 4.1.

В описание задачи включаются следующие основные данные по ее постановке:


- размерность, целевая функция, двусторонние ограничения на переменные,
- функциональные ограничения вида $((+1) g_j(y)) \in g_j^+$ или $((-1) g_j(y)) \in g_j^+$,

а также дополнительные данные для методов локального поиска и метода штрафов:

- рекомендуемая точка для запуска локальных методов,
- «веса», приписанные ограничениям при использовании метода штрафов или модифицированных функций Лагранжа,
- показатель степени r ($1 \leq r \leq 5$) в функции штрафа, влияющий на порядок гладкости задачи со штрафом (для метода модифицированных функций Лагранжа этот параметр всегда полагается равным 2),
- рекомендуемое начальное значение коэффициента штрафа, при котором возникает интересная для исследования структура задачи со штрафом.

Замечание. В программной лаборатории LocOpt функции задачи со штрафом, а также модифицированные функции Лагранжа используется двойка. С одной стороны, как инструмент решения задач с ограничениями, с другой — как механизм порождения задач со сложным рельефом. В последнем случае множители Лагранжа и коэффициент штрафа фиксируются на начальных значениях.

4.2. Подготовка задач для исследования

Пользователь может подготовить необходимую для исследования задачу несколькими способами. Наиболее простой способ заключается в выборе задачи из стандартного списка. Для этого достаточно выполнить через главное меню следующие действия: «Постановка/Задача/Из стандартного набора...» или нажать кнопку  панели

инструментов. Появится диалоговая панель, содержащая кнопки с изображениями задач из стандартного списка (рис. 4.3).

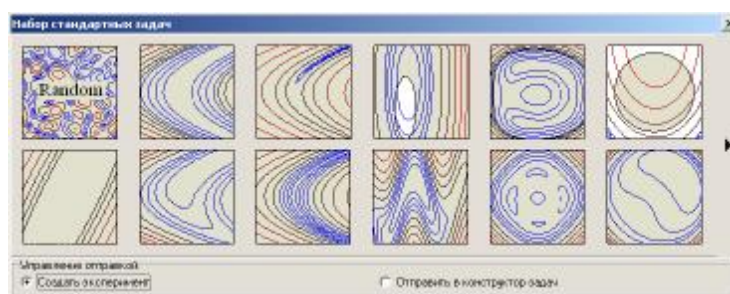





Рис. 4.3. Окно выбора задач из стандартного набора

Для выбора задачи достаточно нажать на одну из кнопок с изображениями задач. В стандартном режиме будет создано новое окно эксперимента, в которое будет отправлена выбранная задача. Если перед нажатием на кнопку с задачей заказать второй из возможных режимов (отправка в конструктор задач), то выбранная задача будет передана в конструктор, позволяющий изменить ее до того, как она будет отправлена в окно эксперимента.

Существует второй, более универсальный способ выбора задачи как из стандартного, так и из личного списка ранее подготовленных задач. Для того, чтобы им воспользоваться, достаточно нажать кнопку  — «Конструктор функций и задач». При первом нажатии в него по умолчанию будет загружен стандартный список задач и активной станет первая задача этого списка. Для загрузки другого списка задач следует выполнить через главное меню действия: «Файл/Открыть набор задач...» или нажать кнопку панели инструментов  — «Открыть набор задач». Для визуализации задач из загруженного списка следует нажать кнопку  — «Показать задачу». При этом появляется изображение текущей задачи в окне «Линии уровня задачи» (рис. 4.4).

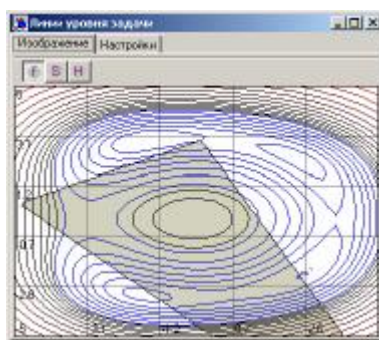






Рис. 4.4. Окно просмотра структуры текущей задачи из конструктора задач

С помощью кнопок навигации  в нижней части окна конструктора задач можно пролистывать задачи из загруженного списка. Одновременно в окне «Линии уровня задачи» будут показаны их изображения (для многомерных задач — изолинии в

одном из сечений). Выбрав нужную задачу можно отправить ее в одно из окон экспериментов, используя пункты главного меню: «Постановка/Новый эксперимент», «Постановка/Задачу — в текущее окно» или «Постановка/Задачу — во все окна» или соответствующие кнопки панели инструментов:  — новый эксперимент,  — отправить задачу в текущее окно эксперимента,  — отправить задачу во все открытые окна.

С помощью окна конструктора задач можно изменять постановки существующих в загруженном списке задач, а также создавать новые задачи. Более подробное описание процесса создания задач можно найти в разделе «Использование конструктора функций и задач».

4.3. Проведение экспериментов при исследовании задач оптимизации

После того, как задача отправлена в окно эксперимента (см. раздел «Подготовка задач для исследования»), это окно становится активным. В нем автоматически рисуются изолинии решаемой задачи и структура допустимой области (см.рис.4.5).

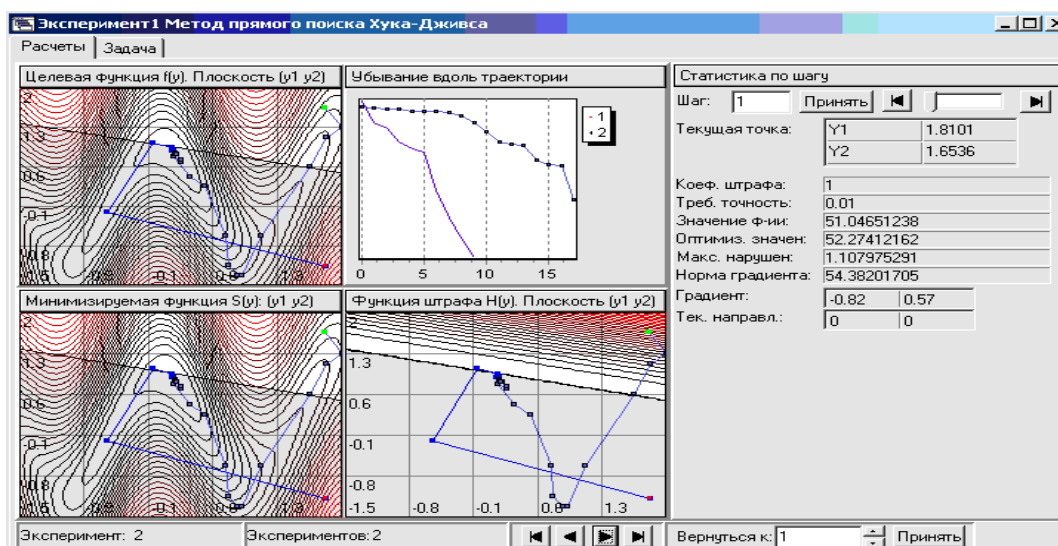


Рис. 4.5. Вид основной страницы окна эксперимента


Для удобства исследования в окне эксперимента формируется три изображения с линиями равного уровня. Вид отображаемой информации зависит от размерности задачи.

Для задач с двумя переменными показываются изображения трех типов:

- верхнее левое изображение показывает изолинии минимизируемой функции $f(y)$ и допустимую область Y , которая выделяется специальным цветом (стиль FQ);
- нижнее левое изображение показывает (в зависимости от выбранного способа учета ограничений) изолинии функции задачи со штрафом $S_g(y)$ или модифицированной функции Лагранжа $L_g(y, l)$ при установленных в данный момент значениях

коэффициента штрафа g и множителей Лагранжа I , также в этом окне отображается допустимая область (стиль S);

- правое нижнее изображение показывает изолинии функции штрафа $H(y)$ и допустимую область (стиль H).

Для задач с тремя и более переменными на этих изображениях показываются изолинии в сечениях плоскостями, параллельными координатным. По умолчанию в каждом из сечений отображаются изолинии функции задачи со штрафом и допустимая область. Секущие плоскости проводятся через установленную в задаче начальную точку локального поиска или текущую точку локального поиска. Номера переменных, по которым строятся сечения, а также тип изображения в сечении могут быть изменены с помощью специального диалогового окна, вызываемого либо через главное меню: «Вид / Окно экспериментов / Сечения плоскостями», либо с помощью кнопки  на панели инструментов. Вид открывающегося диалогового окна представлен на рисунке 4.6.

Вторая страница этого окна содержит органы управления, позволяющие изменять следующие параметры построения изображений изолиний: – количество точек разбиения по осям при построении сетки, используемой для синтеза изолиний; – способ выбора уровней при построении изолиний (логарифмическая шкала уровней или равномерная с подуровнями); – число основных линий равного уровня.

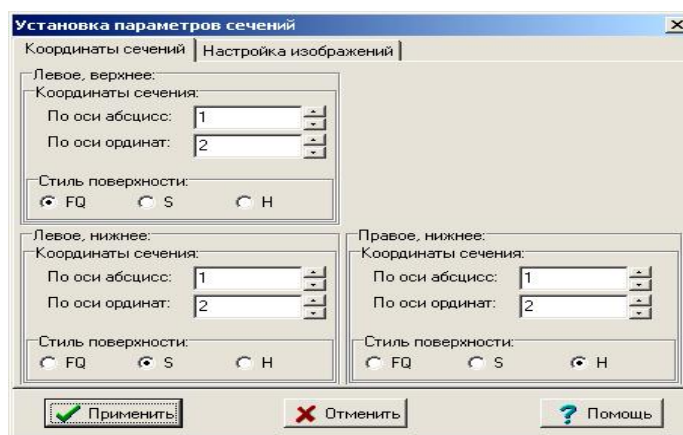



Рис. 4.6. Окно настройки параметров изображений

4.3.1. Выбор метода локальной оптимизации

Система LocOpt включает набор методов локальной оптимизации без функциональных ограничений, приведенный в п. 1.1.2. Выбор метода локальной оптимизации происходит с помощью пункта главного меню «Методы / <название метода>» или с помощью раскрывающегося списка у кнопки  на панели инструментов системы LocOpt. Теоретический материал по данным методам, иллюстрации, а также описание соответствующих им алгоритмов приведены в главе 2 и п. 3.1. Иллюстрации и

теоретический материал по методам доступны также через пункт главного меню «Справка/Методы локального поиска».

В LocOpt полностью открыт доступ к общим параметрам методов и к их индивидуальным наборам параметров. Изменение установленных общих параметров локального поиска происходит в диалоговом окне «Общие параметры расчетов», которое вызывается через главное меню: «Расчет/Параметры расчетов...». Вид этого диалога показан на рис.4.7.

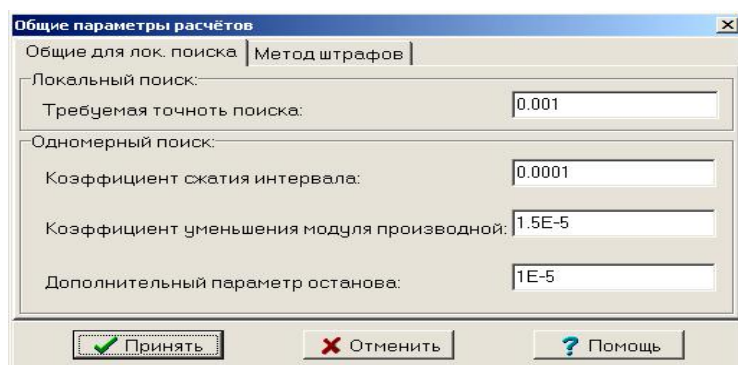


Рис. 4.7. Окно управления общими параметрами расчетов

Диалог по настройке индивидуальных параметров методов вызывается через главное меню приложения: «Методы/Параметры метода...». Перечень настраиваемых параметров зависит от выбранного метода. Вид диалога показан на рис.4.8.

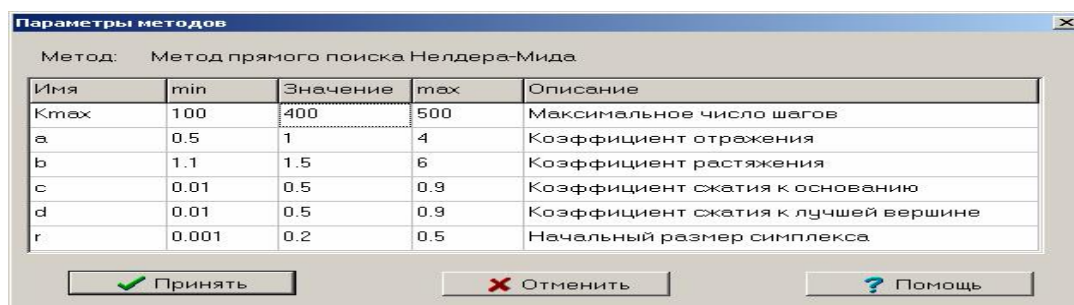



Рис. 4.8 Вид окна параметров метода на примере метода Нелдера–Мида

4.3.2. Выбор способа учета ограничений

Приложение LocOpt позволяет различным образом учитывать функциональные ограничения, имеющиеся в постановке задачи. В LocOpt v.3 имеется пять способов учета:

- отключить ограничения;
- добавить к минимизируемой функции фиксированный штраф $H(y)$;
- решать задачу методом внешнего штрафа с автоподстройкой настройкой коэффициента штрафа;
- использовать модифицированную функцию Лагранжа $L_g(y, I)$ с фиксированными значениями множителей Лагранжа и коэффициента штрафа;

- решать задачу методом внешнего штрафа с автоподстройкой настройкой коэффициента штрафа.

Нужный режим устанавливается через главное меню: «Постановка/Учет ограничений/(Отключить|Фиксированный штраф|Метод штрафов| Модифицированная функция Лагранжа| Метод модифицированной функции Лагранжа)» или с помощью кнопки  с выпадающим списком, включающим те же варианты, что и меню.

В приложении используется только один вид функции штрафа — степенной. Функция штрафа имеет параметр $r=p$, влияющий на порядок гладкости штрафа, а также набор весовых коэффициентов ограничений:




$$H(y) = \sum_{i=1}^m (\max\{c_i(g_i(y) - g_i^+); 0\})^r.$$

Проведение оптимизационных расчетов в программной лаборатории таково, что формально всегда решается задача оптимизации со штрафом

$$S_g(y) @ \min, y \in D, \quad S_g(y) = f(y) + g H(y), \\ D = \{y: a_i \leq y_i \leq b_i, i=1, \dots, N\},$$

или модифицированной функцией Лагранжа, когда вместо $S_g(y)$ используется $L_g(y, I)$.

Вне зависимости от того, есть ли в задаче ограничения, всегда минимизируется одна из этих двух функций. Если ограничений нет, эти функции совпадают с $f(y)$. Такой подход повышает гибкость в изменении постановки задачи, за счет того, что программа позволяет различным образом использовать коэффициент штрафа (как это было указано в начале данного пункта):

- при отключенных ограничениях коэффициент штрафа g полагается равным нулю и множители Лагранжа также полагаются равными нулю;
- при фиксированном штрафе или фиксированной модифицированной функции Лагранжа его значение коэффициента штрафа постоянно и выбирается из окна редактора, размещенного на панели инструментов (для подтверждения введенного значения необходимо нажать кнопку  — «Принять штраф» на панели инструментов, а для возврата к установленному по умолчанию значению нажать расположенную правее кнопку  — «Восстановить штраф»);
- при фиксированной функции Лагранжа дополнительно фиксируются коэффициенты Лагранжа. Это происходит с помощью кнопки , расположенной на панели инструментов LocOpt, задание их значений происходит в специальном всплывающем окне;
- при выбранном режиме «Метод штрафов» коэффициент штрафа подстраивается по специальному алгоритму, приведенному в пункте 3.2.1.2;

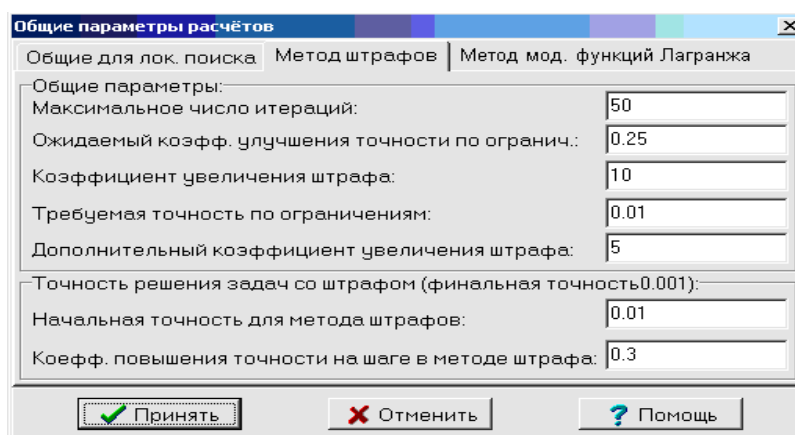




Рис. 4.9. Вторая страница окна общих параметров расчетов, содержащая параметры метода штрафов

- при выбранном режиме «Метод модифицированных функций Лагранжа» коэффициент штрафа подстраивается по специальному алгоритму, приведенному в пункте 3.2.2.3, а множители Лагранжа изменяются согласно соответствующим методу итерационным формулам.

Алгоритмы учета ограничений содержат ряд параметров, которые можно задать на второй и третьей странице диалогового окна по выбору параметров (см. рис. 4.9.)

4.3.3. Изменение положения начальной точки



Положение начальная точка запуска локальных методов отображается в виде красного маркера на картах линий равного уровня в окне эксперимента. Это положение можно изменять. Управление выбором начальной точки для запуска локального метода осуществляется с помощью пунктов главного меню «Расчет/Начальная точка», «Расчет/Восстановить начальную точку» или с помощью следующих кнопок на панели инструментов управления экспериментом:  — новая начальная точка;  — восстановить начальную точку последнего запуска локального поиска.

После нажатия кнопки выбора начальной точки необходимо с помощью щелчка левой кнопкой мыши по области карты линий уровня указать новое положение начальной точки. Для задачи с размерностью более двух при изменении начальной точки автоматически перерисуются изолинии в сечениях, поскольку сечения всегда проводятся через текущую точку. При изменении начальной точки ее координаты отображаются в правой части окна эксперимента.

Перед началом расчета следует отжать кнопку , если она осталась утопленной.




4.3.4. Автоматический выбор начальной точки локального поиска



Для возможности автоматического выбора начальной точки в программную лабораторию включен метод равномерных случайных покрытий области. В главном меню в пункте «Расчет» включены подпункты, соответствующие автоматическому выбору начальной точки с использованием случайного покрытия области, а также восстановлению прежней начальной точки, ранее выбранной по случайному покрытию.

На панели инструментов имеются кнопки, согласованные с описанными выше пунктами меню:  — автоматический выбор начальной точки за счет построения случайного покрытия области;  — возврат к прежней точке автоматического случайного выбора.

По кнопке автоматического выбора начальной точки запускается диалог, в котором необходимо указать количество точек в равномерном случайном покрытии. Автоматический выбор точки наиболее удобен при решении многомерных задач.


4.3.5. Выполнение расчета

Для начала расчета из выбранной начальной точки ранее выбранным методом достаточно нажать одну из кнопок панели инструментов:  — запуск в режиме «быстро»;  — запуск в режиме «медленно»;  — запуск в пошаговом режиме.


Расчет реализован как отдельный поток. Для приостановки расчета можно воспользоваться кнопкой , а для полной остановки — кнопкой . В каждом окне одновременно может быть запущено не более одного расчета. В разных окнах расчеты могут выполняться одновременно.

Результаты расчетов отображаются в виде траекторий и графиков убывания функций. Для разных методов траектории отображаются различными цветами. Существующие стандартные назначения цветов можно изменить в специальном диалоговом окне, которое вызывается с помощью пункта меню «Настройки». Более подробная информация приведена в пп. 4.3.8 «Окно настройки параметров расчета».

4.3.6. Редактирование исследуемой задачи

Задача, находящаяся в окне эксперимента, может быть изменена. Для изменения постановки задачи необходимо выбрать пункты меню «Постановка/Редактор задачи...» или же нажать на кнопку  панели инструментов. При этом активизируется окно конструктора задач, но в него окажется подставленной задача из текущего окна эксперимента.

В режиме редактирования можно выполнить любое изменение постановки задачи, не приводящее к изменению ее размерности. Другие задачи из ранее открытого списка задач становятся недоступными, навигация по ним в этом режиме невозможна.

После требуемого изменения задачи для передачи ее в прежнее окно эксперимента достаточно выполнить действие по отправке задачи в текущее окно эксперимента. Текущим является то окно эксперимента, которое было активно последний раз. Необходимо учитывать, что если после использования режима редактирования задачи из текущего окна эксперимента вызвать конструктор задач, то не произойдет автоматического восстановления ранее загруженного списка задач, поэтому после активизации конструктора следует выбрать из имеющейся в системе коллекции требуемый список задач и загрузить его в конструктор. Для этого достаточно воспользоваться пунктом меню «Файл/Открыть набор задач...» или нажать кнопку  на панели инструментов главного окна.

4.3.7. Навигация по окнам экспериментов

Окна экспериментов являются дочерними окнами главного окна и поэтому к ним применимы общепринятые приемы управления, доступные через пункт главного меню «Окна». Для повышения удобства навигации по экспериментам в состав программной лаборатории включено дополнительное модальное окно «Список окон экспериментов», вызываемое через пункты главного меню «Окна/Список окон экспериментов...». Вид окна показан на рис 4.10.

Для переключения в окно нужного эксперимента необходимо выделить в списке данного окна строку с нужным экспериментом и нажать кнопку «Показать».

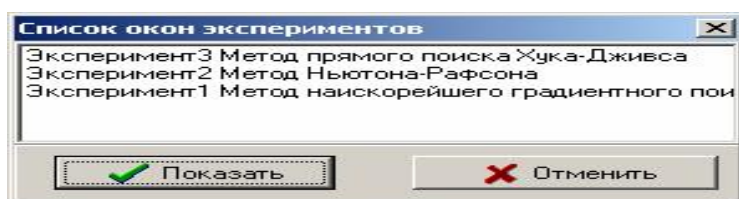


Рис. 4.10 Переключение между окнами экспериментов

4.4. Отбор результатов при подготовке отчета по проведенному исследованию

Программная лаборатория LocOpt имеет простые средства, облегчающие составление отчетов по выполненным исследованиям. Принцип их использования заключается в том, что возможно выборочное копирование промежуточных и итоговых результатов оптимизационных расчетов во встроенный текстовый редактор системы (*журнал*), где их можно подвергать необходимой предварительной ручной правке. У каждого окна

эксперимента существует свой журнал. Копирование результатов расчетов происходит из *внутреннего архива*, имеющегося у каждого окна эксперимента.

Текстовый редактор, в который может выборочно копироваться информация из архива, находится в нижней части второй страницы «Задача+отчет» окна эксперимента.

На той же странице в ее верхней части находится еще один редактор, в который автоматически помещается текст постановки задачи при ее передаче в окно эксперимента. На рис. 4.11 показана вторая страница окна эксперимента после копирования в нее информации о ходе поиска с использованием метода внешнего штрафа.

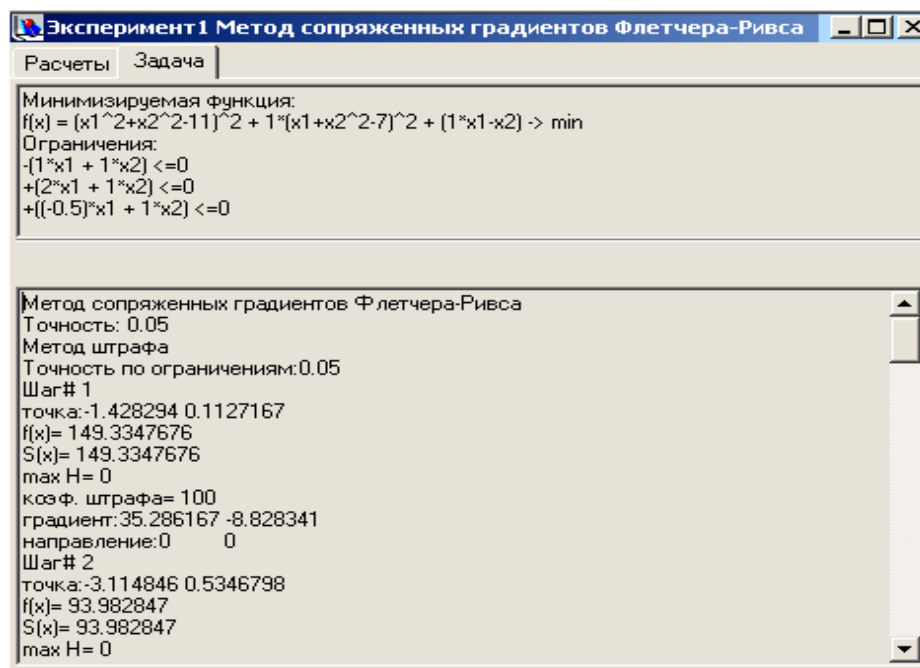




Рис. 4.11 Вторая страница окна эксперимента с окном постановки задачи и журналом расчетов

Отобранная информация может быть окончательно оформлена в редакторе MS Word. Для этого постановку задачи и нужные данные о результатах расчетов можно переслать в MS Word через буфер обмена, пользуясь специальными средствами копирования, предусмотренными в программной лаборатории. Таким же образом можно переслать изображения с линиями равного уровня, построенные графики и т.д. Для этого в приложении предусмотрена группа пунктов в выпадающем меню «Правка». Подробнее этот вопрос рассмотрен в пп. 4.4.1–4.4.2.

4.4.1. Отбор результатов из архива расчетов и их копирование

В окне эксперимента можно провести несколько оптимизационных расчетов, выбирая различные начальные точки, методы, способы учета штрафа. Все данные о выполненных

расчетах сохраняются во *внутреннем архиве* окна эксперимента. За счет этого в окне эксперимента возможен ретроспективный просмотр результатов ранее выполненных расчетов (более подробное описание приведено в разделе 4.6). Эти результаты полностью или частично могут быть скопированы из архива во встроенный редактор окна эксперимента. Для этого с помощью специальных кнопок ◀ ◁ ▷ ▶ — навигации по результатам расчетов необходимо выбрать нужный расчет, а с помощью ползунка в верхнем правом углу окна эксперимента выбрать нужный шаг в этом расчете. Затем нужно скопировать в журнал необходимую информацию по расчету с помощью пунктов главного меню: «Правка/Расчет в журнал» и «Правка/Шаг в журнал» или же кнопок на панели инструментов:  — расчет в журнал;  — шаг в журнал.

При выборе действия «Расчет в журнал» появляется диалоговое окно, в котором необходимо выбрать вид копирования: – только итоги расчета; – результаты всех шагов.

Вид окна этого диалога представлен на рис.4.12.

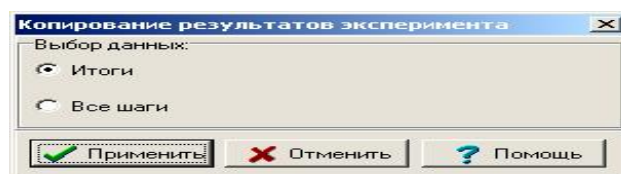





Рис. 4.12 Окно выбора данных для копирования

Для удобства копирования подготовленных результатов расчетов в буфер обмена в главном меню имеются пункты: «Правка/Копировать текст задачи» и «Правка/Копировать журнал». Аналогичные действия реализуются кнопками панели задач:  — запись постановки задачи в буфер;  — запись данных, накопленных в журнале в буфер обмена.

4.4.2. Копирование изображений

На первой основной странице окна эксперимента находится несколько изображений содержащих изолинии функций, траектории методов поиска, графики убывания функции, одномерные сечения функций вдоль направлений перемещения на шаге. Эти изображения можно поместить в буфер обмена для последующей вставки в отчет.

Для помещения их копий в буфер обмена используется следующий пункт главного меню: «Правка/Копировать эксперименты...» или кнопка  на панели задач. Выбор этого действия приводит к появлению диалогового окна в котором необходимо указать одно из четырех изображений, имеющихся в окне эксперимента. Вид этого диалогового окна приведен на рис. 4.13.

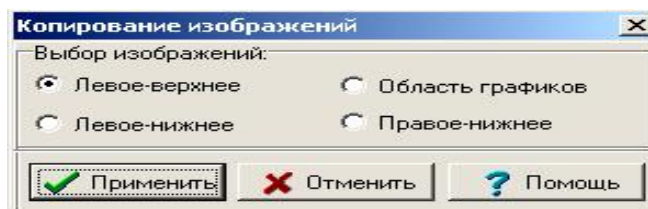



Рис. 4.13 Окно выбора копируемого изображения

4.5. Основные инструментальные средства среды LocOpt

4.5.1. Окно стандартного набора задач

Данное окно облегчает выбор задачи из стандартного набора задач. Вид окна показан на рис. 4.14. Для того чтобы вызвать окно стандартного набора задач достаточно выполнить пункт меню «Постановка/Задача/Из стандартного набора...» или нажать кнопку  на панели задач.

Окно реализовано как модальный диалог, имеющий прокручиваемую полосу кнопок с изображениями стандартных задач. В нижней части окна размещены две кнопки альтернативного выбора. Они управляют тем, куда будет передана стандартная задача, выбранная нажатием на одну из кнопок с изображениями.

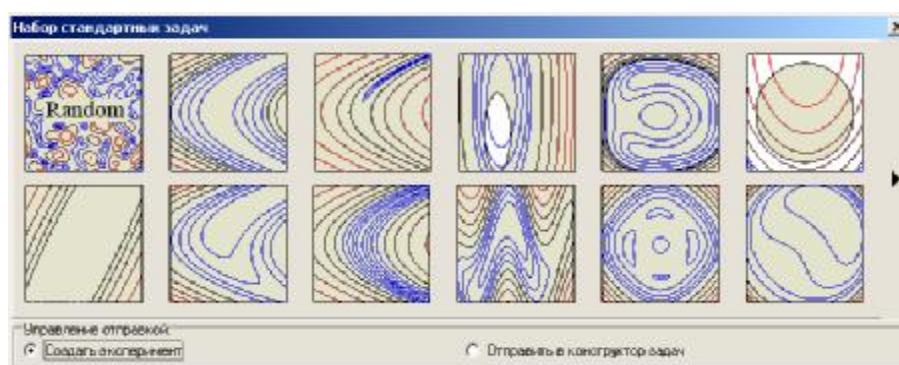


Рис. 4.14 Окно стандартного набора задач


По умолчанию активен режим «Создать эксперимент», при котором нажатие на кнопку с изображением задачи будет приводить к созданию нового окна эксперимента с отправкой в него данной задачи. Во втором режиме «Отправить в конструктор задач» произойдет перемещение выбранной задачи в окно конструктора функций и задач. При этом пользователь получит возможность внести предварительные изменения в постановку задачи и лишь потом отправить ее в окно эксперимента. Подробнее правила работы с конструктором описаны в пункте 4.5.2. и разделе 4.6.

4.5.2. Конструктор функций и задач

Это основной инструмент, позволяющий выполнить постановку новой задачи оптимизации или выбрать ее из списков имеющихся задач. Более подробная информация приведена в разделе 4.6 — «Использование конструктора функций и задач».

4.5.3. Окно просмотра линий равного уровня функций

Окно просмотра изолиний функции является плавающим окном, которое используется совместно с диалоговым окном «Конструктор функций и задач» для оперативного просмотра изолиний функции, которая выделена в списке функций, отображаемом в верхней части окна конструктора. В этом списке отображается перечень функций, содержащийся в активном наборе функций, загруженном в конструктор.

Это окно можно активизировать только в том случае, когда активен конструктор функций и задач. Для открытия окна просмотра функций нужно выполнить пункт меню «Вид/Конструктор задач/Показать функцию» или утопить кнопку  — «Показать функцию» на панели инструментов. Для того чтобы удалить данное окно, достаточно отжать указанную кнопку на панели задач. Если не закрывать данное окно, то оно будет оставаться видимым даже после переключения из конструктора задач в окно эксперимента.

Если окно «Показать функцию» включено, то при работе с конструктором задач в окне просмотра «Показать функцию» будут оперативно отслеживаться все те изменения, которые происходят с активной функцией в конструкторе функций. Например, при навигации по набору функций будет происходить синхронная перерисовка изолиний функции в окне просмотра; при любой корректировке функции (замене функции, изменении ее параметров) все изменения также будут отображаться в окне просмотра функции.

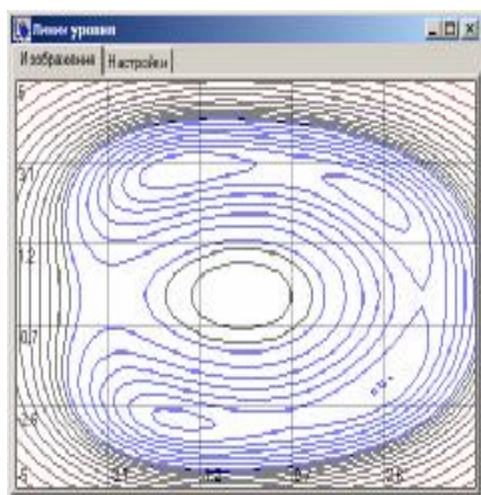


Рис. 4.15. Первая страница окна линий

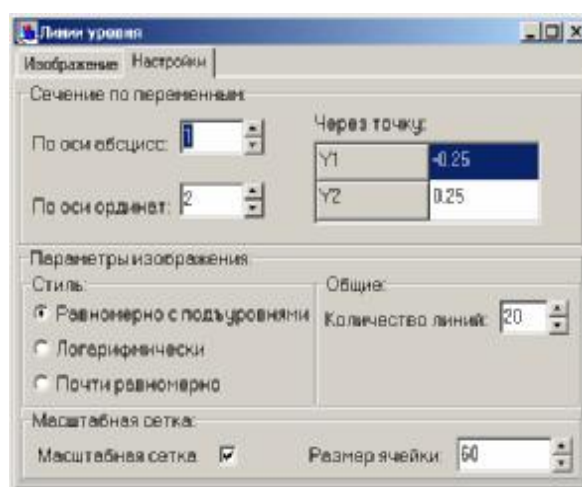


Рис. 4.16. Вторая страница окна линий

уровня функции

уровня функции


Окно просмотра функции имеет две страницы. На первой странице, представленной на рис. 4.15, строится карта линий равного уровня функции. На второй странице окна просмотра (рис. 4.16) размещены органы управления видом изображения. Они позволяют:

- изменить точность построения изолиний за счет выбора числа точек вдоль оси в прямоугольной сетке, используемой для построения изолиний;
- отключить или включить изображение масштабной сетки;
- выбрать количество основных изолиний;
- установить нужный режим выбора уровней при построении изолиний (равномерный с подуровнями, логарифмический, почти равномерный);
- для функций с числом переменных более двух – выбрать номера переменных, по которым выполняется сечение, а также координаты точки, через которую оно проводится.

Изображение изолиний из окна просмотра функции можно скопировать в буфер обмена с помощью пунктов меню «Правка/Изолинии функции – в буфер».

4.5.4. Окно просмотра линий равного уровня и структуры задачи

Окно просмотра изолиний и структуры задачи является плавающим окном, которое используется совместно с диалоговым окном «Конструктор функций и задач» для оперативного просмотра структур текущих задач, отображаемых в окне конструктора при навигации по активному списку задач.

Это окно можно активизировать только в том случае, когда активен конструктор функций и задач. Для открытия окна просмотра «Линии уровня задачи» нужно выбрать пункт меню «Вид/Конструктор задач/Показать задачу» или утопить кнопку  — «Показать задачу» на панели инструментов. Для того чтобы удалить данное окно, достаточно отжать указанную кнопку на панели задач. Если не закрывать данное окно, то оно будет оставаться видимым даже после переключения из конструктора задач в окно эксперимента.

Если окно просмотра «Линии уровня задачи» включено, то при работе с конструктором задач в окне просмотра будут оперативно отслеживаться все те изменения, которые происходят с активной задачей в конструкторе. Например, при навигации по списку задач будет происходить синхронная перерисовка структуры задачи в окне просмотра; при любой корректировке постановки задачи (изменение целевой функции и функций ограничений, добавление и удаление ограничений) все изменения также будут отображаться в этом окне.

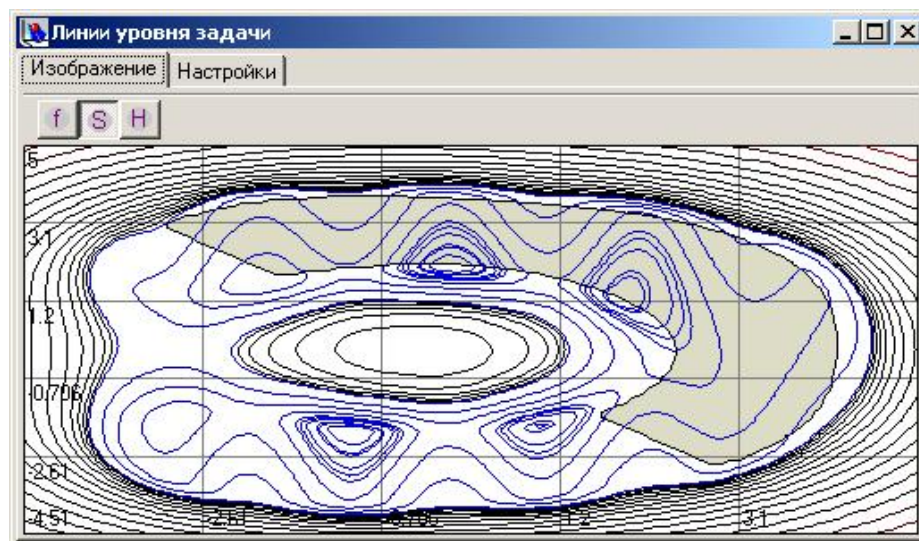


Рис. 4.17. Окно просмотра изолиний и структуры задачи

Окно просмотра структуры задачи имеет две страницы. На первой странице, представленной на рис.3.4, строится карта изолиний для функций задачи (целевой, со штрафом или функции штрафа), а также вид допустимой области. С помощью кнопок на панели инструментов этой страницы можно менять режим отображения: **f** — целевая функция и допустимая область; **S** — минимизируемая функция штрафной задачи и допустимая область; **H** — функция штрафа и допустимая область.

Вторая страница окна просмотра предназначена для настройки параметров изображения. Набор органов управления совпадает со второй страницей окна «Показать функцию» (рис.4.16). Они позволяют:

- изменить точность построения изолиний за счет выбора числа точек вдоль оси в прямоугольной сетке, используемой для построения изолиний;
- отключить или включить изображение масштабной сетки;
- выбрать количество основных изолиний;
- установить нужный режим выбора уровней при построении изолиний (равномерный с подуровнями, логарифмический, почти равномерный);
- для функций с числом переменных более двух – выбрать номера переменных, по которым выполняется сечение, а также координаты точки, через которую оно проводится.


Изображение изолиний из окна просмотра задачи можно скопировать в буфер обмена с помощью пунктов меню «Правка/Изолинии задачи – в буфер».

4.5.5. Окно эксперимента

Окно эксперимента является основным рабочим окном программной лаборатории ЛосОпт. Пример вида окна эксперимента приведен на рис. 4.5. В приложении может быть

одновременно открыто несколько окон экспериментов. В окне эксперимента выполняются все вычислительные эксперименты с конкретной задачей оптимизации, переданной в это окно эксперимента из главного окна приложения. Управление экспериментами происходит с помощью панелей управления экспериментами. В окне эксперимента обеспечивается максимально полная визуализация процесса расчетов. Возможен ретроспективный просмотр результатов ранее проведенных экспериментов и выборка из них данных для составления отчетов. Более подробное описание управления окном эксперимента приведено в разделе «Интерфейс окна эксперимента».

4.5.6. Окно просмотра поверхности

Окно поверхности является плавающим окном. Оно позволяет для двумерных задач получить объемное представление о структуре функций решаемой задачи. Данное окно доступно тогда, когда имеется открытое окно эксперимента с двумерной задачей. Для инициации окна поверхности необходимо воспользоваться пунктом меню «Исследование/Поверхность» или кнопкой  на панели задач. Вид окна показан на рис. 4.18.

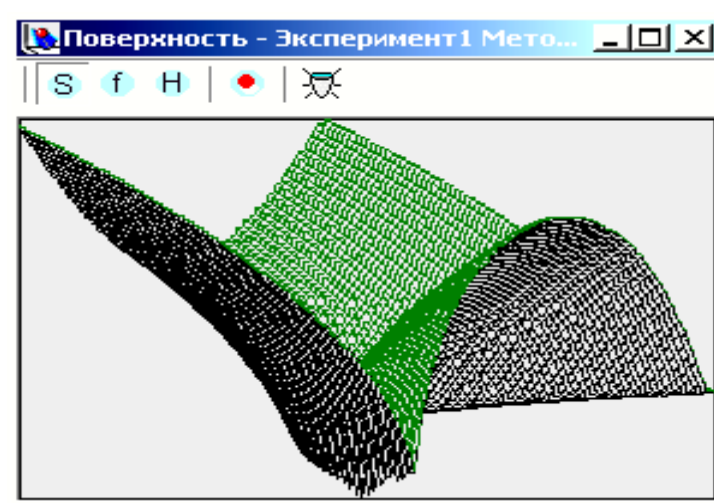







Рис. 4.18. Вид окна просмотра поверхности


Оно имеет панель инструментов, позволяющую оперативно переключать режимы отображения поверхности:  — целевая функция;  — минимизируемая функция штрафной задачи;  — функция штрафа. Кнопка  вызывает диалог по изменению параметров изображения.

При переключении между окнами экспериментов окно поверхности «плавает» сверху. Автоматического изменения вида поверхности не происходит. Номер эксперимента, к которому относится поверхность отображается в заголовочной части этого окна. Для

перерисовки поверхности для активного окна эксперимента необходимо нажать кнопку  — «Освежить».

Для копирования изображения поверхности в буфер обмена следует использовать пункты меню «Правка/Поверхность – в буфер».

4.5.7. Окно для исследования одномерных сечений

Окно для исследования одномерных сечений может использоваться только тогда, когда активно окно эксперимента. Просмотр одномерных сечений применяется для исследования кривизны и характера поведения функций задачи вдоль изучаемых направлений. Для того, чтобы войти в режим одномерных сечений необходимо выполнить пункт меню «Исследование/Одномерное сечение» или утопить кнопку  на панели инструментов.

После выполнения этого действия окно эксперимента переходит в особый режим при котором можно мышью указать вектор одномерного сечения на любой из карт изолиний. Для этого надо поместить мышь в начальную точку вектора, зажать левую кнопку мыши и переместить ее в точку конца вектора сечения. После отпускания кнопки мыши возникнет плавающее окно одномерного сечения (рис. 4.19).

В нем разными цветами показываются одномерные сечения двух функций: пометка « f » соответствует сечению целевой функции задачи $f(y)$; пометка « S » — минимизируемой функции (напомним, что роль минимизируемой функции выполняет либо функция $S_g(y)$ задачи со штрафом, либо модифицированная функция Лагранжа $L_g(y, I)$). С помощью кнопок «Добавить» и «Удалить» можно удлинять и укорачивать вектор сечения. Чтобы выбрать новое сечение окно ранее построенного сечения необходимо закрыть.



Рис. 4.19 Окно для исследования одномерных сечений функций задачи

4.5.8. Окно настройки параметров расчета

Данное окно позволяет изменять принятые по умолчанию настройки цветов, используемых для маркировки траекторий расчетов в окнах экспериментов. В окне имеется список названий методов оптимизации. Каждое название отображается своим цветом. Ниже списка находится линейка цветов (см. рис.4.20).

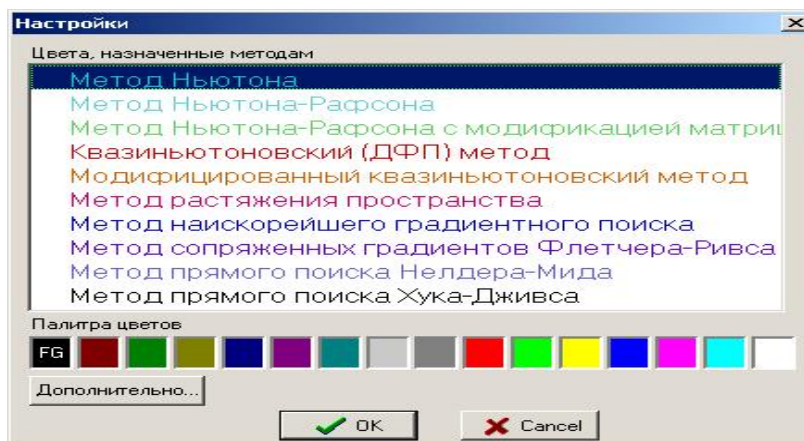


Рис. 4.20. Окно настройки параметров расчета

Если выделить название метода и щелкнуть по клетке с нужным цветом, то название метода в списке примет указанный цвет, а траектории метода станут изображаться данным цветом. В окне имеется также кнопка «Дополнительно...», которая активизирует диалог для задания общих параметров расчетов, показанный на рис. 4.7.

Для того чтобы назначенные установки по выбору цветов вступили в силу, необходимо нажать в окне кнопку «ОК».

4.5.9. Окно навигации по окнам экспериментов

Для облегчения навигации по окнам экспериментов используется специальное диалоговое окно, вызываемое через пункты главного меню «Окна/Список окон экспериментов». Вид окна показан на рис. 4.10, его использование описано в пункте 4.3.7.

4.6. Описание интерфейса главного окна среды LocOpt

Программная лаборатория реализована в формате приложения, поддерживающего многодокументный интерфейс: в области главного окна можно открыть несколько дополнительных окон для проведения экспериментов. Основное меню и панели инструментов размещены в главном окне. Общий вид главного окна работающего приложения приведен на рис. 1.1. В главном окне есть две основные инструментальные панели в верхней части окна и панель состояния в нижней его части. Присутствие этих панелей определяется с помощью пункта главного меню «Вид». На основных панелях

размещено несколько дополнительных панелей. Ниже на двух рисунках показаны главное меню и панели инструментов главного окна с пояснениями к ним.

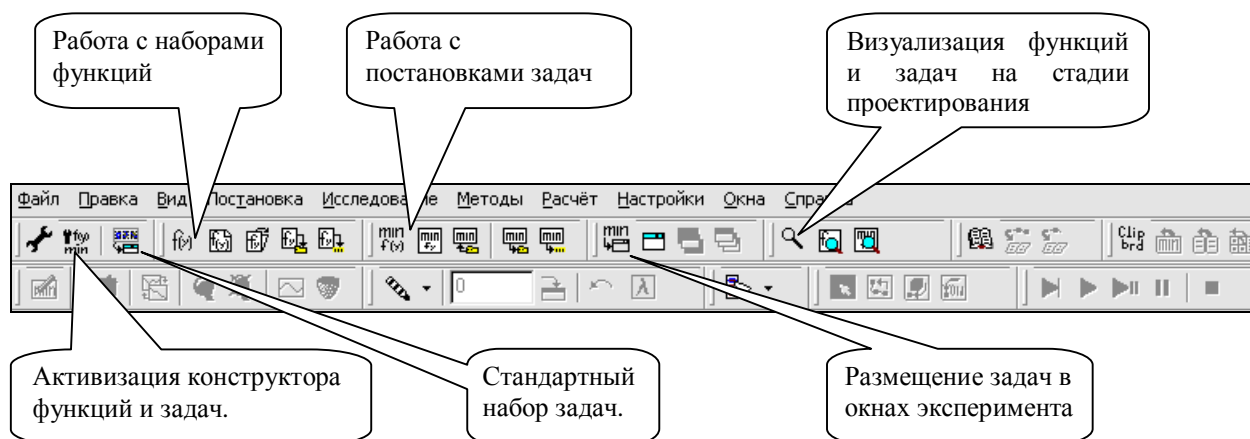


Рис. 4.21. Панели главного окна, связанные с управлением постановкой задачи

На рис.4.21 приведен вид панелей инструментов при работе с конструктором функций и задач. При этом в основном активны кнопки панели, связанные с управлением наборами функций и списками задачи, их сохранением, а также с процессом отправки задач в окна экспериментов. Все эти действия связаны с процессом постановки задачи.

Рисунок 4.22 показывает вид инструментальных панелей, который они имеют в то время, когда пользователь работает в окне эксперимента. При этом активны кнопки, связанные, в основном, с управлением экспериментом, анализом задачи и использованием результатов.

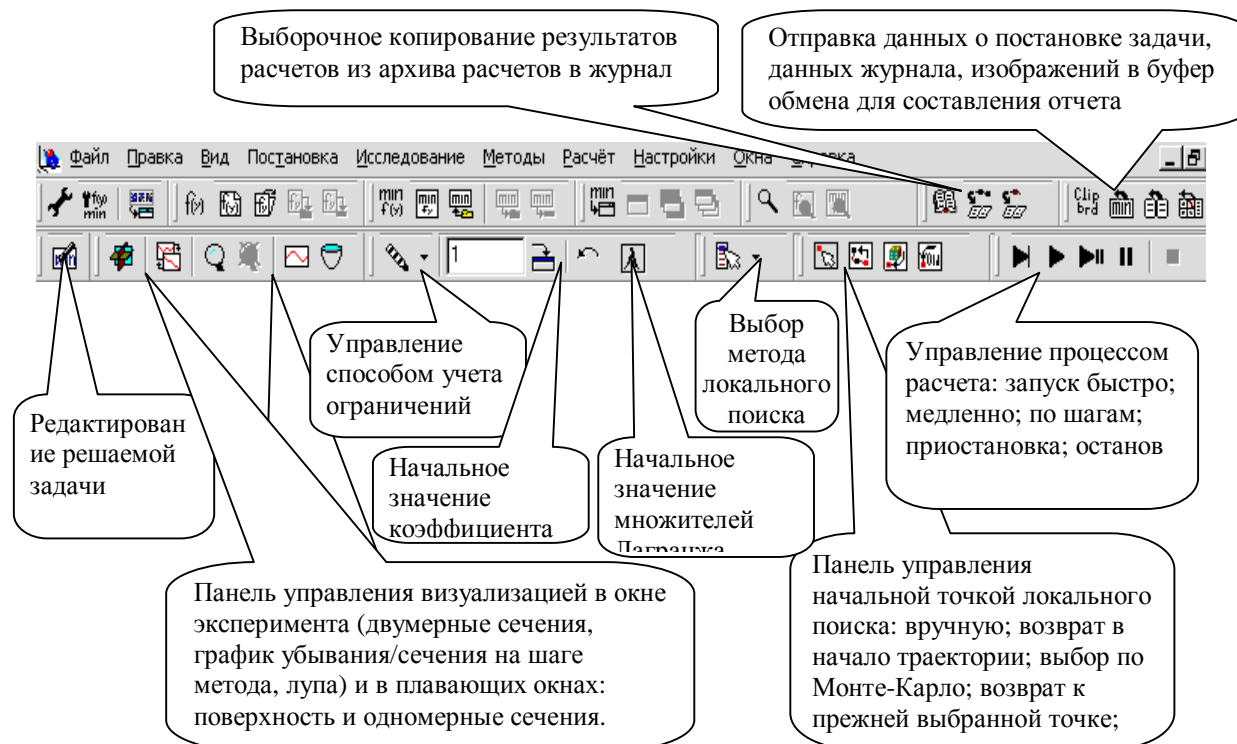


Рис. 4.22. Панели главного окна, связанные с управлением постановкой задачи

Принципы использования органов управления главного окна более подробно изложены в соответствующих разделах данного документа. Назначение большинства органов управления описано в разделе «Основные инструментальные средства».

4.7. Использование конструктора функций и задач

Конструктор задач предназначен для выполнения постановок задач достаточно произвольного вида.

Функции, на основе которых выполняется постановка задачи, могут быть получены из трех источников:

- формульное задание,
- из набора случайно генерируемых функций Гришагина,
- из встроенного фиксированного списка функций,
- функции, определяющие критерии качества объекта, реализованного в виде DLL пользователя.

Все функции могут иметь параметры a_1 , a_2 , a_3 , и т. д. со значениями по умолчанию, а также область определения. Чтобы построить функцию, нужно выбрать (определить) ее на основе одного из источников и затем придать нужные значения ее параметрам. Можно также скорректировать область определения функции, указав новые значения для границ двусторонних ограничений на переменные, задающие область ее определения. Если в последующем данная функция будет использована в качестве функции цели при конструировании новой задачи оптимизации, то ее область определения будет подставлена в качестве значения по умолчанию вместо области изменения значений переменных задачи.

Диалоги по заданию функций имеют расположенные справа области для пробной прорисовки линий равного уровня функций. Поддерживается три режима прорисовки, органы управления которыми на рисунке скрыты под изображением изолиний. Доступ к этим органам управления предоставляется после однократного щелчка левой кнопкой мыши по области изолиний.

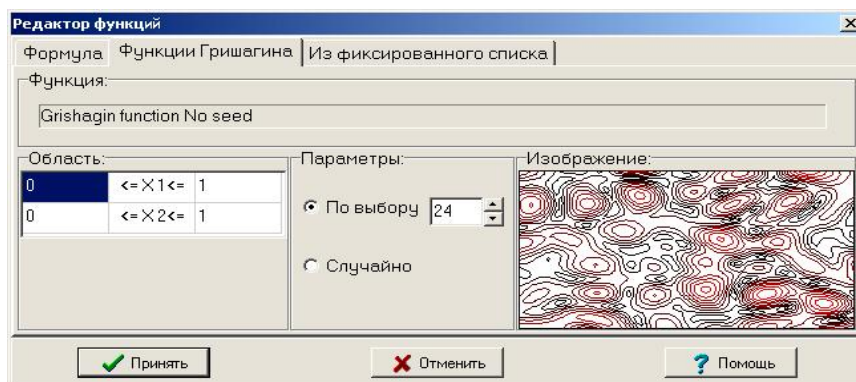


Рис. 4.23. Окно выбора случайной функции Гришагина

Вид двух закладок окон диалога показан на рис. 4.23 и рис. 4.24.

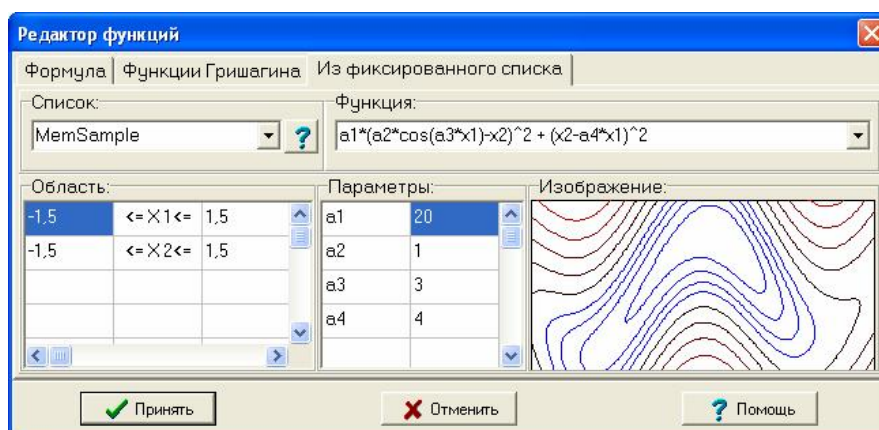


Рис. 4.24. Окно выбора функции из фиксированного списка MemSample или из DLL

Для использования функций из DLL (динамически загружаемых библиотек), необходимо предварительно выбрать одну из DLL по имени в раскрывающемся списке в блоке «Список» (рис. 4.24). После этого в раскрывающийся список «Функция» будет загружен перечень строк-описаний функций из этой DLL. Особенностью использования функций из DLL является то, что все функции из одной DLL имеют общий набор параметров. Изменение параметра одной из этих функций повлечет изменение его значения и для остальных функций. Такая реализация принята потому, что с точки зрения системы LocOpt каждая DLL описывает некоторый объект, по отношению к которому ставится задача оптимизации, а функции, хранимые в DLL, определяют связанные с этим объектом критерии качества. Таким образом, общий набор параметров набора функций фактически является набором параметров объекта, моделируемого в форме DLL.

Привила разработки DLL для среды LocOpt, а также размещение в структуре папок LocOpt файлов DLL и файлов справочной информации по DLL описаны ниже в разделе 4.10.

Вне зависимости от способа получения, созданные функции добавляются в список функций, сохраняемый конструктором функций и задач. Для того чтобы упростить в дальнейшем процесс постановки задач, созданные наборы функций можно сохранять в виде файлов в специальной папке Functions_Lists, находящейся в основной папке приложения LocOpt. Эти файлы записываются в специальном формате в виде *наборов функций*. Кроме наборов функций пользователя в приложении существует один *эталонный набор функций Standard*, изменять который нельзя.

Работа с наборами функций при их создании и использовании происходит в окне *конструктора функций и задач* (рис.4.25).

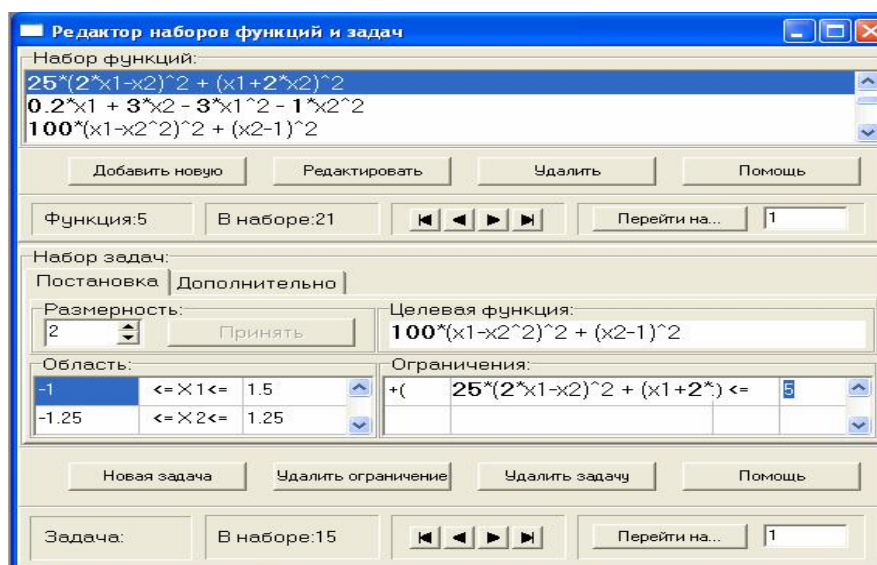


Рис. 4.25. Окно конструктора функций и задач

Окно конструктора и таблицы в нем могут изменять свои размеры. При развернутом состоянии окна можно изменять относительный размер областей в его верхней части (конструктор функций) и его нижней части (конструктор задач). Изменение размеров происходит путем захвата мышью и перемещения вверх или вниз специальной разделительной линии, видимой в средней части окна в виде горизонтальной канавки.

В верхнюю часть окна загружается (с возможностью изменения) один из ранее сохраненных наборов функций. Этот набор может быть создан заново, путем включения новых функций с помощью кнопки «Добавить новую».

При постановке новой задачи необходимо нажать кнопку «Новая задача» в нижней части окна на рис. 4.25. При построении новой задачи сначала происходит *выбор размерности*, завершаемый нажатием кнопки «Принять». После этого становятся доступными остальные поля. Задание целевой функции и функций ограничений происходит по технологии «Перетащи и отпусти» (Drag&Drop) путем перетаскивания

нужных функций в поля целевой функции и функций ограничений из списка функций, видимых в верхней части окна (в области конструктора функций). Возможны любые операции с целевой функцией и с наборами функций ограничений, размещенных в окнах создаваемой или ранее созданной задачи. В частности, можно выполнять их редактирование на месте по двойному щелчку мыши.

Замечание. Функции, имевшие параметры, отображаются в списке конструктора функций так, что вместо имен параметров подставляются их значения, выделяемые повышенной жирностью (см. рис. 4.25). Функции из DLL отображаются в специальном формате. Для них запись в строке редактора функций имеет примерно следующий вид:



Mar2000 {1.1, 2.1, 3.0, 4.1, 5.10, 6.05, 7.0, 8.0, 9.0, 10.1}

Структура строки для случая N параметров следующая:

$\langle \text{Имя_DLL} \rangle \langle \text{Имя_функции} \rangle a = \{ 1: \langle \text{значение_1} \rangle; \dots N: \langle \text{значение_N} \rangle \}$

Границы области переменных, верхние ограничения на функции ограничений и знак перед ними (плюс или минус) редактируются в ячейках таблиц «Область» и «Ограничения». Для изменения знака перед ограничением необходимо выполнить двойной щелчек мышкой по ячейке таблицы, содержащей знак функции. Значение верхней границы редактируется после двойного щелчка в области этого значения. Для того чтобы новое значение границы было принято необходимо выполнить щелчек мышкой по любой другой ячейке таблицы.

На второй закладке конструктора задач (см. рис. 4.26) задаются нормировочные коэффициенты для функций ограничений, показатель степени в штрафе при использовании метода штрафных функций и начальное значение коэффициента штрафа. Также задается координаты начальной точки запуска метода локального поиска, определяющие ее положение по умолчанию.

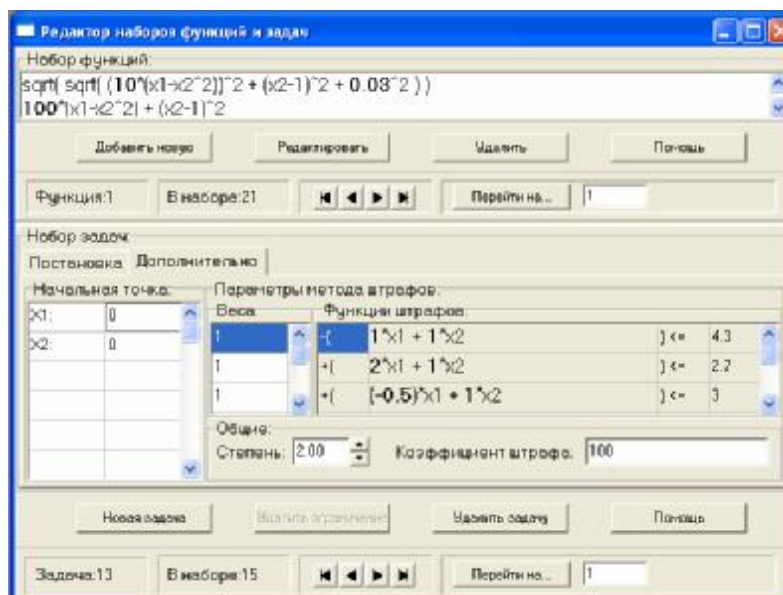


Рис. 4.26. Окно конструктора функций и задач при определении дополнительной информации о задаче

Сконструированные задачи в виде отдельных файлов сохраняются в специальных папках, размещенных в папке Tasks_Repository приложения. В ней имеется папка Standard, которая будет содержать стандартный набор задач, а также папки User1, User2, и т. д., с задачами пользователей. Задачи, записанные в одну папку, образуют *набор задач*.

Через главное меню приложения и панель инструментов можно загрузить в конструктор задач любой из наборов задач. После этого возможна быстрая навигация по задачам набора. Диалог, аналогичный диалогу по выбору папки с набором задач, показан на рис.4.1.

Для задачи, находящейся в окне конструктора, через основную панель инструментов можно заказать построение карты изолиний в отдельном плавающем окне. Для многомерных задач отображаются изолинии в заказанном координатном сечении (см. рис. 4.17).

С помощью кнопок основной панели инструментов можно отправить выбранную или построенную заново задачу в специальное окно для проведения исследований. Таких окон может быть открыто несколько. Можно заменить задачу в существующем окне, или создать для нее новое окно. Более подробно эти вопросы изложены выше в разделах 4.2 и 4.3.

4.8. Интерфейс окна эксперимента

Вид главного окна приложения с открытым окном эксперимента приведен на рис. 1.1. Начальные принципы его использования описаны в разделе 4.3 настоящего документа. Ниже будут более подробно описаны особенности и правила использования интерфейса

данного окна. Значительная часть органов управления экспериментом вынесена на панели инструментов главного окна программной лаборатории LocOpt и была рассмотрена в предыдущих разделах (см. рис. 4.23, 4.24).

Как уже отмечалось ранее в разделах 4.3, 4.4., окно эксперимента выполнено в виде блокнота с двумя страницами. Первая страница является основной при проведении вычислительных экспериментов с задачей оптимизации, помещенной в это окно. На этой странице размещены панели для графического и числового представления результатов экспериментов, а также органы управления просмотром результатов экспериментов (рис. 4.5, 4.27).

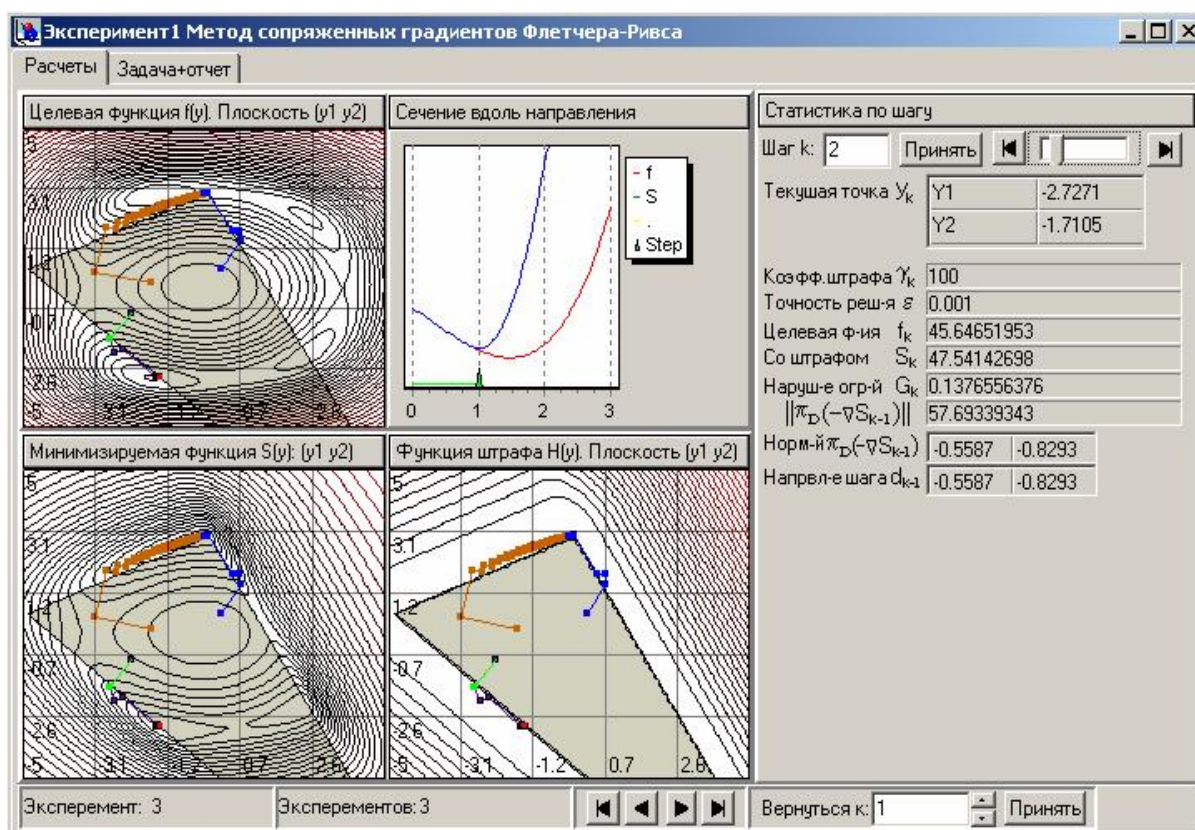


Рис. 4.27. Вид первой страницы окна эксперимента в режиме просмотра сечений на шаге поиска

Вторая страница является вспомогательной, ее вид приведен на рис. 4.11. На ней размещены окна двух текстовых редакторов. В верхнем окне по умолчанию отображается текст постановки задачи. Пользователь имеет возможность внести свои комментарии, дополнения и пояснения в этот текст, чтобы потом использовать его при формировании отчета по выполненному исследованию. Окно нижнего редактора предназначено для предварительного отбора результатов расчетов для формирования отчета. В текущую позицию редактора можно пересылать информацию о ходе и результатах


оптимизационных расчетов, ранее выполненных в окне эксперимента. Более подробно этот материал описан в пункте 4.4.1.

Рассмотрим подробнее работу с окном эксперимента в основном режиме (с использованием первой страницы этого окна). На первой странице окна эксперимента размещено четыре панели с изображениями, справа от которых находится панель числовой информации о текущем шаге поиска. Три панели с изображениями показывают изолинии функций задачи и траектории поиска в сечениях, проходящих через начальную или текущую точку поиска. Вид изолиний зависит от размерности решаемой задачи (смотри раздел 4.3).

Для задач размерности два на этих панелях размещаются изолинии следующих функций: на левой верхней панели — изолинии целевой функции и вид допустимой области; на левой нижней панели — изолинии функции задачи со штрафом и вид допустимой области; на правой нижней панели — изолинии функции штрафа.

Для задач размерности выше двух на всех трех панелях отображаются изолинии функции задачи со штрафом и вид допустимой области в различных координатных сечениях, проходящих через текущую точку поиска. Способ изменения сечений и стиля отображения изолиний описан в подразделе 4.3, вид соответствующего диалогового окна приведен на рис. 4.6.

Область правого верхнего изображения имеет тройное назначение. По умолчанию на ней отображаются графики убывания минимизируемой функции задачи на траекториях поиска (задачи со штрафом или модифицированной функции Лагранжа). Однако эту область окна можно использовать еще двумя способами: — для наблюдения за одномерными сечениями целевой и минимизируемой функций вдоль направлений поиска, выбираемых методом оптимизации; — для отображения графиков изменения множителей Лагранжа. Последний режим поддерживается только в версии 3.

Для включения указанных дополнительных режимов необходимо либо переключить пункт меню «Вид/Окно экспериментов/График убывания» в состояние «Вид/Окно экспериментов/Сечение вдоль направления» («Вид/Окно экспериментов/Множители Лагранжа»), либо нажать нужное число раз кнопку  на панели инструментов. Вид правой верхней панели в режиме просмотра сечений приведен на рис. 4.27, а в режиме графиков убывания — на рис. 4.5.

На правой информационной панели (для числовой информации) отображаются следующие значения:

- номер шага поиска;

- координаты текущей точки;
- текущее значение коэффициента штрафа;
- текущая точность поиска;
- значение целевой функции в текущей точке;
- значение невязки по ограничениям в текущей точке;
- значение минимизируемой функции (функции задачи со штрафом) в текущей точке;
- значение нормы антиградиента минимизируемой функции, вычисленного на текущем линейном многообразии поиска (с учетом двусторонних ограничений) для предыдущего шага; в качестве минимизируемой функции всегда используется задача со штрафом, при отсутствии ограничений она совпадает с целевой функцией задачи;
- значения первых компонент этого вектора после нормировки;
- значения первых компонент нормированного вектора направления выполненного шага.


Статистика по шагу		
Шаг k:	3	Принять
Текущая точка Y_k		
Y1	1.4001	
Y2	0.090842	
Кэф.штрафа γ_k		
Точность реш-я ε	0.001	
Целевая ф-ия f_k	114.1436316	
Со штрафом S_k	117.7927726	
Наруш-е огр-й G_k	0.1910272502	
$\ \pi_D(-\nabla S_{k-1})\ $	22.09411201	
Норм-й $\pi_D(-\nabla S_{k-1})$	0.07696	-0.997
Напрел-е шага d_{k-1}	0.8978	-0.4405
Проек. Гессеиан Γ_{k-1}		
	780.6	401.6
	401.6	141.3
Оценка проек. Γ_{k-1}		
	7.151	14.23
	14.23	33.46
Модиф. оцен. пр. Γ_{k-1}		
	7.151	14.23
	14.23	33.46

Рис. 4.18. Вид панели для отображения числовой информации о шаге расчета

При проведении расчетов с использованием методов второго порядка, а также квазиньютоновских методов, вычисляющих оценки матриц Гессе, в нижней части этого окна происходит отображение матрицы Гессе и других матриц, характеризующих работу метода. Для задач размерности более трех отображаются не все элементы матриц, а их угловые фрагменты размером 3x3. При выходе процесса поиска на границу изменения некоторых переменных, отображаются значения матриц меньшей размерности, вычисляемые на соответствующем линейном многообразии. Все матрицы отображаются для предыдущей точки поиска.

В версии 3 системы LocOpt в нижней части числовой информационной панели дополнительно отображаются значения множителей Лагранжа для предыдущего шага поиска.

При выполнении оптимизационных расчетов в окне эксперимента происходит накопление архива результатов расчетов. После завершения очередного расчета можно перейти в режим пошагового просмотра траектории этого расчета или любого другого ранее выполненного расчета. Для этого необходимо произвести следующие действия:

- с помощью навигационных кнопок , расположенных на панели состояния в нижней части окна эксперимента (рис. 4.27) выбрать из архива расчетов окна эксперимента результаты нужного расчета;
- с помощью ползунка или с помощью редактора номера шага и кнопки «Принять»



, расположенных в правой верхней части первой страницы окна экспериментов (рис. 4.27), следует выбрать необходимый номер шага расчета.

Для облегчения поиска нужных результатов предусмотрена специальная маркировка данных текущего просматриваемого расчета, а также шага расчета. А именно, при навигации по результатам расчетов пользователь увидит, что на картинах изолиний траектория, соответствующая текущему просматриваемому расчету, будет выделяться маркерами (точки измерений очертятся тонкими черными рамками), а в окне графиков убывания функции маркерами будет отмечаться соответствующий этому расчету график убывания функции. При изменении номера шага числовые характеристики этого шага будут отображаться на панели числовых характеристик шага (правой панели первой страницы окна эксперимента); кроме того, текущий шаг на траектории поиска будет выделяться отрезком зеленого цвета. Рис. 4.27 показывает состояние окна эксперимента после проведения трех расчетов разными методами из трех начальных точек. Окно находится в режиме просмотра четвертого шага третьего расчета. Перемещение на этом шаге выделено зеленым цветом.

При просмотре траектории поиска по шагам полезно включить режим наблюдения за сечениями функции на шаге траектории поиска. Под частью графика этого сечения зеленым цветом будет отображаться отрезок, показывающий смещение на шаге в масштабе этого изображения (см. рис.4.27).

Выход из режима просмотра происходит автоматически при запуске нового расчета.

Использование второй страницы окна эксперимента для подготовки отчета по проведенным исследованиям описано в пункте 4.4.1.

4.9. Средства составления отчетов

Программная лаборатория LocOpt имеет простые средства, облегчающие составление отчетов по выполненным исследованиям. Принцип ее использования заключается в том, что возможно выборочное копирование промежуточных и итоговых результатов оптимизационных расчетов во встроенный текстовый редактор системы (журнал), где их можно подвергать необходимой предварительной ручной правке. Копирование результатов расчетов в журнал происходит из внутреннего архива, имеющегося у каждого окна эксперимента. Способы копирования описаны в пункте 4.4.1.

Окончательное составление отчета происходит в текстовом редакторе MS Word. Программная лаборатория имеет средства управления копированием постановки задачи, журнала расчетов и построенных изображений в буфер обмена Windows из которого информацию можно вставлять в документ MS Word. Аналогичным образом можно передавать в документ формируемого отчета полученные в окнах эксперимента изображения. Более подробная информация приведена в разделе 4.4.2.

4.10. Создание и подключение DLL пользователя


Вначале опишем общие правила использования DLL пользователя в среде LocOpt.

Как уже отмечалось в разделе 4.7, если использовать DLL специального формата, разработанного специально для системы LocOpt, можно создать модель объекта, описываемого набором функций-критериев. На основе этих функций можно ставить и решать задачи оптимизации для этого объекта. В форме DLL можно создать также набор обычных функций, которые по какой-либо причине не могут быть получены с помощью формульного интерпритатора (например, если функция не представляется аналитически). В справочной системе LocOpt приведён пример DLL, обеспечивающий взаимодействие с программным комплексом LocOpt и содержащей типовой набор функций.

При написании собственной DLL необходимо учитывать, что для обеспечения ее взаимодействия со средой LocOpt эта DLL обязательно должна включать некоторый обязательный комплект функций с конкретными именами, которые должны в точности совпадать с именами в примере. Эти функции должны выполнять конкретные действия, которые описаны в пункте 4.10.1 в форме комментариев к тексту примера написания заголовочного файла DLL и примера реализации DLL.

Так как есть необходимость использовать при написании функций в DLL данных строкового типа, к создаваемому проекту необходимо добавлять библиотеки MEMMGR.LIB и BORLNDMM.DLL, входящие в поставку среды C++Builder 2006.

Компиляцию необходимо выполнить, создав проект DLL в той же версии среды разработки.

Откомпилированный файл (<имя>.dll) необходимо разместить в папке Dll_Repository, после чего среда LocOpt сможет с ним работать. Рекомендуется также создать справку по разработанной Вами DLL. Файл справки должен называться так же, как и сама DLL. Для размещения справки создайте папку с именем вашей DLL в Dll_Repository и скопируйте туда необходимые файлы справочной системы по DLL. При этом, после выбора DLL по имени в раскрывающемся списке «Список» диалогового окна рис. 4.24, можно получить доступ к имеющемуся тексту справки, нажав на кнопку  в этом окне.

4.10.1. Пример заголовочного файла DLL и текста сpp-файла реализации DLL

Вначале рассмотрим **структуру заголовочного файла**.

Начальная часть текста заголовочного файла имеет стандартный вид, приведенный ниже:

```
#ifndef _MYDLL_H
#define _MYDLL_H
#if defined(BUILD_DLL)
# define DLL_EXP __declspec(dllexport)
#else
# if defined(BUILD_APP)
# define DLL_EXP __declspec(dllimport)
# else
# define DLL_EXP
# endif
#endif

#include <vector>
#include <string>
```

Далее приводится обязательная стандартная часть текста заголовочного файла DLL с минимально необходимым набором функций, необходимых для взаимодействия с программным комплексом LocOpt. Они обязательно должны присутствовать и выполнять указанные задачи.

Методы доступа к полям:

```
extern "C" int DLL_EXP getFunctionCount();
extern "C" int DLL_EXP getFuncParamsCount(int num);
extern "C" int DLL_EXP getDimension();
extern "C" double DLL_EXP getParamValue(int num);
extern "C" std::string DLL_EXP setParamValue(int num,double value);
extern "C" double DLL_EXP getLeftAxis(int num);
extern "C" std::string DLL_EXP setLeftAxis(int num,double value);
```

```
extern "C" double DLL_EXP getRightAxis(int num);
extern "C" std::string DLL_EXP setRightAxis(int num,double value);
extern "C" std::string DLL_EXP getFuncName(int num);
extern "C" void DLL_EXP getFuncParams(int num,std::vector<int>& vec);
```

Методы для вычисления функций и её производных:

```
extern "C" bool DLL_EXP canHessian();
extern "C" bool DLL_EXP canGradient();
extern "C" double DLL_EXP getValue(std::vector<double>&,int num);
extern "C" void DLL_EXP getGradient(std::vector<double>&, int num,
                                   std::vector<double> &);
extern "C" void DLL_EXP getHessian(std::vector<double>&,int num,
                                   std::vector< std::vector<double> >&);
```

Замечание. Если в DLL не предусмотрено вычисление градиента или Гессииана, то canHessian() canGradient() должны возвращать значение false, а методы getGradient и getHessian можно опустить.

Функция для получения содержимого DLL

```
extern "C" void DLL_EXP getAll(std::vector< std::string >& );
```

Функция, инициализирующая начальные значения

```
extern "C" void DLL_EXP InitDll();
#endif
```

Более подробное описание содержится в тексте встроенной справки в пункте «Пример реализации DLL» в справочной системе LocOpt.

Ниже приведена **структура сpp-файла реализации DLL**. Стандартная начальная часть имеет вид:

```
#include <vcl.h>
#include <windows.h>
#pragma hdrstop
#pragma argsused

#define BUILD_DLL
#include "MyDll.h"

int WINAPI DllEntryPoint(HINSTANCE hinst, unsigned long reason, void* lpReserved)
{
    return 1;
}
```

Замечание. Для хранения необходимой информации об объекте можно использовать другие типы данных и имена переменных. Во избежание ошибок рекомендуется придерживаться разработанной структуры хранения, которая приведена ниже.

Список необходимых переменных для хранения информации об объекте

bool IsDllInit=false; - признак того, что DLL инициализирована и готова к работе.

bool canhessian=true; - признак наличия функции для вычисления Гесса.

bool cangradient=true; - признак наличия функции для вычисления градиента.

int dimension=2; - размерность объекта.

int functionCount=2; - число функций объекта.

std::vector <double> params(3); - вектор со значениями параметров объекта.

std::vector <double> leftaxis(2); - левые границы области.

std::vector <double> rightaxis(2); - правые границы области.

std::vector <std::string> func_names(2); - имена функций.

std::vector < int > func1_params(2); - номера параметров, используемых первой функцией.

std::vector < int > func2_params(3); - номера параметров, используемых второй функцией.

Внимание! Объявляя в DLL собственные переменные типа std::vector, не рекомендуется описывать их в заголовочном файле DLL, используя extern, так как это приводит к некорректной работе с такой переменной, например, вызов метода resize приводит к порче памяти и возникновению ошибок работы всех работающих программ. Подобный эффект возникает и при объявлении переменной данного типа, как локальной переменной функции. Поэтому все переменные данного типа рекомендуется объявлять как глобальные переменные модуля DLL.

Ниже приведены определения экспортируемых из DLL функций.

Получение числа имеющихся функций

```
int getFunctionCount() { return functionCount; }
```

Получения числа параметров, используемых функцией с номером num

```
int getFuncParamsCount(int num)
{
    switch(num)
    {
        case 0: return func1_params.size();    break;
        case 1: return func2_params.size();    break;
        default: return 0;
    }
}
```

Получение размерности объекта

```
int getDimension() { return dimension; }
```

Получение значения параметра с номером *num*

```
double getParamValue(int num)
{ if(params.size()>0)
    if(num<0) return params[0];
    else if(num>=params.size()) return params[params.size()-1];
    else return params[num];
else return 0;
}
```

Установка значения *value* параметру с номером *num*

```
std::string setParamValue(int num,double value)
{ if((num>=0)&&(num<params.size())) params[num]=value;
return std::string("NO");
}
```

Установка значения *value* левой границы области по координате *num*

```
std::string setLeftAxis(int num,double value)
{ if((num>=0)&&(num<leftaxis.size())) leftaxis[num]=value;
return std::string("NO");
}
```

Установка значения *value* правой границы области по координате *num*

```
std::string setRightAxis(int num,double value)
{ if((num>=0)&&(num<rightaxis.size())) rightaxis[num]=value;
return std::string("NO");
}
```

Замечание. Методы установки значений параметров должны возвращать строку, содержащую “NO”, если передано допустимое значение, или сообщение, которое получит пользователь, если передано недопустимое значение. Это позволяет использовать в качестве параметров номера чего-либо, задавать диапазон значений, а так же обеспечивает возможность взаимосвязи параметров между собой и с допустимой областью. Для этого надо проверить переданное значение по нужному вам критерию.

Получение значения левой границы области по координате *num*

```
double getLeftAxis(int num)
{ if(leftaxis.size()>0)
    if(num<0) return leftaxis[0];
```



```

else if(num>=leftaxis.size()) return leftaxis[leftaxis.size()-1];
    else return leftaxis[num];
else return 0;
}

```

Получение значения правой границы области по координате *num*

```

double getRightAxis(int num)
{
    if(rightaxis.size()>0)
        if (num<0) return rightaxis[0];
        else if(num>=rightaxis.size()) return rightaxis[rightaxis.size()-1];
        else return rightaxis[num];
    else return 0;
}

```

Получение имени функции с номером *num*

```

std::string getFuncName(int num)
{
    if(func_names.size()>0)
        if(num<0) return func_names[0];
        else if(num>=func_names.size()) return func_names[func_names.size()-1];
        else return func_names[num];
    else return std::string("");
}

```

Получение вектора параметров функции с номером *num*

Замечание. Параметры-векторы, передаваемые по ссылке для их заполнения требующимися данными, при работе с комплексом LocOpt будут уже иметь необходимый размер (в случае если DLL работает верно), поэтому проверка размера передаваемого параметра-вектора необязательна. Однако подобная проверка может помочь выявить ошибки при разработке DLL.

```

void getFuncParams(int num, std::vector<int>& vec)
{
    switch(num)
    {
        case 0: if(vec.size()==func1_params.size())
            for(int i=0; i<func1_params.size();i++) vec[i]=func1_params[i];
            break;
        case 1: if(vec.size()==func2_params.size())
            for(int i=0; i<func2_params.size();i++) vec[i]=func2_params[i];
            break;
    }
}

```

```

default: if(vec.size()==func1_params.size())
    for(int i=0; i<func1_params.size();i++) vec[i]=func1_params[i];
}
}

```

Получение информации о наличии методов вычисления производных

```
bool canHessian() { return canhessian; }
```

```
bool canGradient() { return cangradient; }
```

Получение значения функции с номером *num* в точке *v*

```

double getValue(std::vector<double>& v,int num)
{ if((num>=0)&&(num<functionCount)&&(v.size())>=dimension))
    switch (num)
    { case 0: return (params[0]*v[0]*v[0]+params[2]*v[1]);
      case 1: return (params[0]*v[0]*v[0]+params[1]*v[1]+params[2]*v[0]*v[1]);
      default: return 100;
    }
    else return 0;
}

```

Получение значения градиента функции с номером *num* в точке *v*

```

void getGradient(std::vector<double>& v,int num,std::vector<double> &gradient)
{ if ((num>=0) && (num<functionCount)
    && (v.size())>=dimension) && (gradient.size())>=dimension))
    switch (num)
    { case 0: gradient[0]=2*params[0]*v[0]; gradient[1]=params[2]; break;
      case 1: gradient[0]=2*params[0]*v[0]+params[2]*v[1];
              gradient[1] =params[1]+params[2]*v[0]; break;
    }
}

```

Получение значения гесссиана функции с номером *num* в точке *point*

```

void DLL_EXP getHessian(std::vector<double>& point,int num,
                        std::vector< std::vector<double> >& res)
{ if((num>=0)&&(num<functionCount)&&(point.size())>=dimension))
    switch (num)
    { case 0: res[0][0]=2*params[0]; res[0][1]=0; res[1][0]=0; res[1][1]=0; break;

```

```

        case 1: res[0][0]=2*params[0]; res[0][1]=params[2];
                res[1][0]=params[2]; res[1][1]=0; break;
    }
}

```

Получение списка имён функций имеющихся в DLL

```

void getAll(std::vector< std::string >& vector)
{
    vector.resize(func_names.size());
    for(int i=0; i<func_names.size(); i++) vector[i]=func_names[i];
}

```

Инициализация начальных значений переменных в DLL

```

void InitDll()
{
    if (IsDllInit) return;
    IsDllInit=true; params[0]=1; params[1]=5; params[2]=10;
    leftaxis[0]=-1; leftaxis[1]=-1; rightaxis[0]=1; rightaxis[1]=1;
    func_names[0]=(std::string("a1*x1^2+a2*x2"));
    func_names[1]=(std::string("a1*x1^2+a2*x2+a3*x1*x2"));
    func1_params[0]=0; func1_params[1]=2;
    func2_params[0]=0; func2_params[1]=1; func2_params[2]=2;
}

```

Заключение. В настоящий документ не включена программа выполняемого в среде LocOpt лабораторного практикума. Это связано с тем, что она приведена в приложении к изданному учебному пособию «Нелинейное программирование и многоэкстремальная оптимизация» [4]. Данный же учебно-методический материал содержит лишь подробное описание самой инструментальной среды, а также краткий обзор необходимой теории по методам локальной оптимизации. Такое распределение материала тем более оправдано, что конкретная методика проведения лабораторных работ по методам локальной оптимизации может быть существенно различной в зависимости от количества учебных часов, отводимых под лабораторный практикум, а также от содержания и объема читаемого лекционного курса. Полные возможности среды LocOpt, включая разработку собственных DLL, могут быть использованы в рамках курсов специализации.

ЛИТЕРАТУРА

Основная литература

1. Базара М., Шетти К. Нелинейное программирование. Теория и алгоритмы. – М.: Мир, 1982.
2. Васильев Ф.П. Методы оптимизации. – М.: Факториал Пресс, 2002.
3. Гилл Ф., Мюррей У., Райт М. Практическая оптимизация. – М.: Мир, 1985.
4. Городецкий С.Ю., Гришагин В.А. Нелинейное программирование и многоэкстремальная оптимизация. – Нижний Новгород: Изд-во ННГУ, 2007. 489 с.
5. Измаилов А.Ф., Солодов М.В. Численные методы оптимизации. – М.: Физматлит, 2003.
6. Карманов В.Г. Математическое программирование. – М.: Физматлит, 2000.
7. Поляк Б.Т. Введение в оптимизацию. – М.: Наука, 1983.
8. Сухарев А.Г., Тимохов А.В., Федоров В.В. Курс методов оптимизации. – М.: Наука, 1986.

Дополнительная литература

9. Аоки М. Введение в методы оптимизации. – М.: Наука, 1977.
10. Бертсекас Д. Условная оптимизация и методы множителей Лагранжа. – М.: Радио и связь, 1987.
11. Васильев Ф.П. Численные методы решения экстремальных задач. – М.: Наука, 1982.
12. Демьянов В.Ф., Васильев Л.В. Недифференцируемая оптимизация. – М.: Наука, 1981.
13. Евтушенко Ю.Г. Методы решения экстремальных задач и их применение в системах оптимизации. – М.: Наука, 1982.
14. Мину М. Математическое программирование. Теория и алгоритмы. – М.: Наука, 1990.
15. Немировский А.С., Юдин Д.Б. Сложность задач и эффективность методов оптимизации. – М.: Наука, 1979.
16. Пшеничный Б.Н., Данилин Ю.М. Численные методы в экстремальных задачах. – М.: Наука, 1975.
17. Уайлд Д.Дж. Методы поиска экстремума, 1967.
18. Химмельблау Д. Прикладное нелинейное программирование. – М.: Мир, 1975.
19. Шор Н.З. О скорости сходимости метода обобщенного градиентного спуска с растяжением пространства // Кибернетика, 1970. № 2.