Глава 4. Математическое обеспечение САПР – решение задачи синтеза, методы оптимизации.

4.1. Введение в проблему параметрического синтеза.

4.1.1. Общие сведения, постановка задачи.

Цель всех методов – определить оптимальное решение, то есть решение, которое по тем или иным признакам предпочтительнее перед другими. Параметры, совокупность которых образуют решения, называют элементами решения.

В общем виде это классическая математическая задача нахождения минимума функции или функционала принадлежит к классу так называемых классических математических вариационных задач, давно разрабатываемых математиками. Числовая функция, определённая на множестве функций, называется функционалом. Что бы найти экстремум функции, необходимо продифференцировать модель по всем переменным, приравнять каждое уравнение к нулю и решить полученную систему. Однако использование классических методов вариационного исчисления в ряде случаев на практике затрудненно, а иногда и невозможно по ряду причин:

- Когда параметров много, задача решения системы напрямую может оказаться непростой.
- Когда на элементы решения наложены ограничения. Оптимум (наименьшее значение целевой функции) может находиться на границе области поиска, то есть целевая функция может вообще не иметь экстремума . При этом возникает многомерная вариационная задача при ограничениях, трудно реализуемая традиционными методами.
- В некоторых задачах функция $f(\alpha,x)$ вообще не имеет производных, например для целочисленного значения аргумента.

Все эти причины делают задачу поиска отнюдь не тривиальной, и классические методы вариационного исчисления оказываются малоэффективными. Это и привело к появлению альтернативных, чисто машинных (вычислительных) методов математического программирования. Методов,

основанных на использовании модели, как средства анализа и специальных процедур выбора параметров, как средства автоматизированного поиска решения (самый примитивный способ - сканирование). Это не строгие, а так называемые эвристические методы, основанные на здравом смысле, интуиции и аналогиях.

В математическом программировании сложились понятия следующих разделов:

Линейное программирование – целевая функция линейна, а множество, на котором ищется экстремум целевой функции, задаётся системой линейных равенств и неравенств. В связи с широким использованием линейного программирования сложились целые классы задач, структура которых позволяет создать специальные методы их решения, в отличие от методов решения задач общего характера. Линейное программирование используется для решения различных военных, промышленных, экономических и организационных задач. Например, такими задачами являются:

- транспортная задача – план перевозки однородного груза, чтобы стоимость перевозок была минимальной.
- задача оптимального планирования производства – сколько единиц каждого вида продукции должно произвести предприятие, чтобы прибыль была максимальной.
- задача о размещении – согласование размещения пунктов потребления продукции с транспортной задачей и задачей оптимального планирования производства.
- задача о посевной площади – требуется определить, какую площадь на каждом участке нужно отвести под ту или иную культуру, чтобы прибыль была максимальна, а урожай по каждому виду культуры был не меньше, чем установлено по плану.

Нелинейное программирование – целевая функция и ограничения нелинейны. Нелинейное программирование принято подразделять следующим образом:

- **Выпуклое программирование** – выпукла целевая функция и выпукло множество, на котором решается задача минимизации.
- **Квадратичное программирование** – целевая функция квадратична, а ограничения задаются в виде линейных равенств и неравенств (особый класс задач выпуклого программирования, часто встречающийся на практике).
- **Многоэкстремальные задачи** – различного вида специализированные задачи. Например, задачи минимизации на выпуклом множестве вогнутых функций.

Целочисленное программирование - особый вид математического программирования, когда на переменные накладываются условия целочисленности.

Динамическое программирование – решение многоэтапных задач, когда неоптимальность на каждом этапе в отдельности может привести к оптимальности в целом.

Можно выделить два типа задач оптимизации - задачи безусловной оптимизации (без ограничений) и задачи условной оптимизации, когда на параметры наложены ограничения в виде различных типов равенств и неравенств. Ограничения объективно появляются при проектировании технических объектов и вытекают из конкретной физической и технологической реализуемости внутренних параметров элементов, ограниченности ресурсов и т. п. При постановке задачи оптимизации учет ограничений иногда бывает принципиально необходим. Так, если целевая функция имеет вид $f(x)=a+bx$ и не наложены ограничения на параметр x, то задача поиска экстремального значения $f(x)$ становится некорректной. Ограничения суживают область поиска оптимального решения, и искомый экстремум становится условным. Различают прямые и функциональные ограничения.

Прямые ограничения имеют вид

$$x_{hi} \le x_i \le x_{bi} \quad \text{при} \quad i \in [1 \div n] \qquad (4.3)$$

где x_{hi}, x_{bi} - минимально и максимально допустимые

значения *i*-го управляемого параметра; *n* — размерность пространства варьируемых параметров. Например, для многих объектов параметры элементов не могут быть отрицательными: $x_{hi} > 0$ (геометрические размеры, электрические сопротивления, массы и т. п.).

Функциональные ограничения, как правило, представляют собой условия работоспособности выходных параметров, не вошедших в целевую функцию. Функциональные ограничения могут быть:

1) типа *равенств*

$$\varphi(x) = 0 \qquad (4.4)$$

2) типа *неравенств*

$$\phi(x) > 0, \qquad (4.5)$$

В подавляющем большинстве случаев задачи параметрической оптимизации технических объектов сводятся к задачам условной оптимизации. Решение задач можно выполнить с помощью одного из двух подходов. В первом подходе учитывается, что большинство развитых методов оптимизации ориентировано на поиск безусловного экстремума, поэтому их применение к решению задачи условной оптимизации требует, чтобы эта задача была предварительно сведена к задаче безусловной оптимизации. Во втором подходе используются методы, специально разработанные для решения задач нелинейного программирования с ограничениями.

Методы решения весьма разнообразны по стратегии поиска минимума целевой функции, и в этом смысле можно выделить методы общего характера и методы адаптированные к конкретным задачам. Большое количество работ посвящено исследованию методов оптимизации [1 - 12] – вопросам сходимости, быстродействия и устойчивости. Значительный класс задач связан с применением стохастических методов поиска, а также с применением генетических алгоритмов для решения технических задач. Особый класс представляет решение задач, в которых целевая функция имеет

разрывы. В дальнейшем изложении материала на основе литературных данных приводим краткий обзор основных методов оптимизации, применяемых в науке и технике, уделяя основное внимание нелинейному программированию.

4.1.2. Выбор целевой функции.

Существует целый ряд методов, позволяющих для специального класса задач, получить решение с определенной степенью точности. Однако, по-видимому, невозможно создать единый метод, позволяющий получить точное решение общей задачи нелинейного программирования за конечное число шагов.

Одна из основных проблем постановки экстремальных задач заключается в формулировке целевой функции. Сложность выбора целевой функции состоит в том, что любой технический объект первоначально имеет векторный характер критериев оптимальности (многокритериальность). Причем улучшение одного из выходных параметров, как правило, приводит к ухудшению другого, так как все выходные параметры являются функциями одних и тех же варьируемых параметров и не могут изменяться независимо друг от друга. Такие выходные параметры называют конфликтными параметрами. Сведение многокритериальной задачи к однокритериальной называют сверткой векторного *критерия.* Задача поиска его экстремума сводится к задаче математического программирования.

В зависимости от того, каким образом выбираются и объединяются выходные параметры в скалярной функции качества, различают частные, аддитивные, мультипликативные, минимаксные, статистические критерии и т.д. Объединение нескольких выходных параметров, имеющих в общем случае различную физическую размерность, в одной скалярной целевой функции требует предварительного нормирования этих параметров. Наиболее распространены из них следующие критерии.

Взвешенный аддитивный критерий (4.6) применяют, когда можно выделить две группы выходных параметров. Полагаем, что все $y_j(x)$ безразмерны. Тогда для случая минимизации целевой функции свертка векторного критерия будет иметь вид

$$f(x)=\sum_{j=1}^{q} a_j y_j^{-}(x)+\sum_{j=q+1}^{m} a_j y_j^{+}(x) \qquad (4.6)$$

где $a_j > 0$ – весовой коэффициент, определяющий степень важности j – го параметра (обычно a_j выбираются проектировщиком и в процессе остаются постоянными).

В первую группу входят выходные параметры, значения которых в процессе оптимизации нужно увеличивать $y_j^{+}(x)$. (Например, производительность, помехоустойчивость, вероятность безотказной работы и т. п.). Во вторую - выходные параметры, значения которых следует уменьшать $y_j^{-}(x)$. (Например, расход топлива, длительность переходного процесса, перерегулирование, смещение и пр.).

Мультипликативный критерий может применяться в тех случаях, когда отсутствуют условия работоспособности типа равенств и выходные параметры не могут принимать нулевые значения. Тогда минимизируемая мультипликативная целевая функция имеет вид

$$f(x)=\prod_{j=1}^{q} y_j^{-}(x) / \prod_{j=q+1}^{m} y_j^{+}(x) \qquad (4.7)$$

Большую ценность для разработчика представляют результаты оптимизации по статистическим критериям, в которых целевой функцией является вероятность выполнения всех заданных условий работоспособности. Однако использование этих критериев в процессе оптимизации связано с

проведением многократного статистического анализа методом Монте-Карло, что, для достаточно сложных моделей анализа, требует использования компьютеров очень высокой производительности.

Какие вычислительные сложности возникают при разработке алгоритма расчёта?

Во-первых – размерность задачи. Можно считать, что задача с числом переменных менее 15 является малой задачей. При числе переменных в диапазоне от 15 до 50 – средняя задача, и при числе переменных более 50 – большая задача. Эта градация весьма условна, т.к. даже при числе переменных равным двум, найти решение может оказаться достаточно сложно.

Во-вторых – вычисление производных целевой функции (в тех алгоритмах, где они используются). Во многих задачах целевая функция формируется на основе численного моделирования на ЭВМ. Например, при расчете волноводного группирователя линейного ускорителя электронов (ЛУЭ) целевая функция базируется на расчётных параметрах сгустков электронов, которые определяются из решения (в самой простой модели сгустков) системы обыкновенных дифференциальных уравнений.

В-третьих – решение многих нелинейных задач, как правило, довольно «дорогостоящая» (большое время счёта) процедура из-за многократного вычисления целевой функции или из-за того, что сама задача заключается в решении многих подзадач, связанных между собой. При расчёте линейного ускорителя электронов, например, это связано с необходимостью выбора изменения амплитуды высокочастотного ускоряющего поля и фазовой скорости волны по длине ускоряющей структуры. Выбор этих характеристик базируется на табличных экспериментальных данных.

В-четвёртых - возможное плохое масштабирование задач, в которых диапазоны изменений отдельных переменных отличаются друг от друга на несколько порядков. Например, одна переменная изменяется в диапазоне от 1 до 10, вторая - в пределах от 10^6 до 10^7. Масштабирование - пре-

образование переменных с помощью соответствующих коэффициентов к такому виду, когда значение переменных оказываются сопоставимы по своей величине, а значение целевой функции измеряется в безразмерных величинах близких к единице.

Решение любой задачи состоит из нескольких этапов и начинается с корректной постановки инженерной задачи. Необходимо чётко представлять преимущества, недостатки и специфические особенности различных методов оптимизации. Каждой задаче должен соответствовать адекватный метод решения.

Выбор целевой функции непосредственно связан и вытекает из постановки решаемой задачи, начиная с самого первого её этапа - построения упрощённой *модели* решаемой задачи, отражающей основные действующие факторы, и установление закономерностей, которыми определяется поведение рассматриваемой системы. В этом вопросе нужно исходить из принципа перехода от простого к сложному. Чтобы сузить область поисков минимума, необходимо построить ряд дополняющих и уточняющих моделей, т.е. каждая последующая модель отражает все свойства предшествующей модели и отличается от неё включением в анализ более тонких и сложных параметров реальной системы. Это позволяет значительно сократить область первоначального поиска минимума. Вообще говоря, любая модель является упрощенным образом реальной системы, поэтому необходимо оценить факторы, которые можно не учитывать как слабо влияющие на работу системы в целом. Построение модели начинается с *определения границ* изучаемой системы. Границы служат средством выделения системы из окружающей среды. Между системой и внешней средой всегда существуют взаимосвязи. Определение границ является первым приближением описания реальной системы. Взаимосвязь между системой и внешней средой, таким образом, фиксируется на определенном уровне. Конкретный выбор границ осуществляется, в самом общем случае, методом последовательных приближений на основе анализа численных экспериментов. В процессе

расчётов может оказаться, что границы выбраны слишком жёсткими. В этом случае нужно произвести коррекцию первоначального варианта. Таким образом, оптимизационное исследование приобретает итерационный характер.

Следующим этапом является математическая запись выбранной модели - установление соотношений между параметрами модели. Например, в приложении к физическим задачам математическими моделями часто являются системы дифференциальных и линейных алгебраических уравнений.

На основе предшествующих этапов производится построение *целевой функции*, числовое значение которой (минимум или максимум) соответствует оптимальному варианту решения задачи. Это основной показатель оптимизационного исследования позволяющий выбрать компромиссный вариант из возможных решений задачи. К любой системе могут предъявляться взаимоисключающие требования. Целевая функция состоит, как правило, из нескольких параметров, значения которых должны стремиться к достижению требуемых значений (параметров цели). Степень приближения полученного решения к параметрам цели зависит от формы построения целевой функции. Численное значение целевой функции является единственным критерием, который может использоваться при определении оптимального решения задачи. Возможны различные варианты формирования целевой функции – во-первых, все параметры входят в построение целевой функции, во-вторых, определяется степень важности каждого параметра, и часть из них выводится в раздел ограничений. Например, один из параметров системы выбирается как основной и по нему ведётся оптимизация, а остальные рассматриваются как ограничения. Другой путь – формирование обобщенной целевой функции, содержащей несколько параметров цели, которая учитывает конкурирующие требования при их достижении. При этом необходимо продумать механизм их совместимости. Это напрямую связано с выбором независимых переменных, которые позволяют адекватно описать поведение выбранной системы. Переменные могут быть двух типов - переменные, значения которых могут из-

меняться в достаточно широком диапазоне, и переменные, значения которых фиксированы и определяются внешними факторами. Часть переменных может иметь вероятностный характер. Оценивая степень важности каждого параметра системы, необходимо выбирать независимые переменные таким образом, чтобы незначительные их них не перегружали задачу. При выборе целевой функции и независимых переменных строится модель, описывающая связь между переменными задачи и влияние независимых переменных на степень достижения конечной цели.

Часто целевая функция имеет экономическую составляющую. Например, требование высокого качества произведенного товара (максимальность) всегда конкурирует с минимизацией материальных расходов и энергопотребления. В задачах создания уникальных физических установок или устройств достижение необходимых технических параметров может являться значительно более весомым фактором, чем материальные и экономические затраты.

Обычно в физических задачах приходится сравнивать различные по своей природе параметры. Например, внешние габариты устройства или установки и её энергетические характеристики. Возможный путь решения – обезразмеривание параметров системы за счёт введения нормирующих коэффициентов и весовых коэффициентов.

Дальнейшим этапом является *численный эксперимент* и оценка результатов расчётов, которая устанавливает практическую добротность, как выбранной модели, так и целевой функции. Последующая стратегия поиска заключается либо в принятии выбранной модели и целевой функции, либо в их уточнении, которое связано с усложнением модели и целевой функции за счёт ранее не учитываемых свойств рассматриваемой системы. Несмотря на универсальность методов оптимизации, их успешное применение к решению конкретной задачи зависит от глубины понимания физической сути задачи и чёткого представления об области применения того или иного алгоритма поиска оптимума.

4.1.2. Задачи нелинейного программирования.

Задачам нелинейного программирования в настоящее время уделяется значительное внимание. Предложен ряд методов, позволяющих с определенной точностью получать решения для специального класса задач. Надежные и одновременно экономичные методы поиска глобального экстремума в настоящее время неизвестны. Практически большинство используемых методов является методами локального поиска (выпуклого программирования).

Задачи поиска оптимальных параметров различных устройств и систем связаны с анализом функций, определённых на множествах конечномерного евклидова пространства. В дальнейшем рассмотрении материала основное внимание будет уделено нелинейному программированию и, особенно, вопросам квадратичного программирования, так как многие задачи могут быть сведены к квадратичному программированию. Определим основные понятия выпуклого анализа, которые используются в дальнейшем.

Наборы $x^T = (x_1, x_2, ...x_n)$ называются *точками (векторами)*, а числа $x_1, x_2, ...x_n$ - их *координатами.*

Совокупность всех наборов $x^T = (x_1, x_2, ...x_n)$ n вещественных чисел $x_1, x_2, ...x_n$ называют *евклидовым пространством размерности n,* если выполняются следующие условия.

Пусть $x \in E_n$, $y \in E_n$ и α – вещественное число. Тогда

$x + y = (x_1 + y_1, x_2 + y_2, ...x_n + y_n)$ (сложение)

$\alpha x = (\alpha x_1, \alpha x_2, ...\alpha x_n)$ (умножение на число)

$\langle x, y \rangle = \sum_{i=1}^{n} x_i * y_i$ (скалярное произведение)

Множество X n-мерного евклидового пространства E_n, называется *выпуклым*, если с любыми двумя точками

$x \in X$ и $y \in X$ ему принадлежит и соединяющий их отрезок $[x, y]$.

Выпуклость множества X означает, что из $x, y \in X$ следует $z = \alpha x + (1-\alpha) y \in X$ для всех $0 \le \alpha \le 1$.

В E_2, например, выпуклы отрезок, полупрямая, прямая, круг, треугольник, полуплоскость и вся плоскость.

Функция $f(x)$ определённая на выпуклом множестве X, называется *выпуклой*, когда для любых $x, y \in X$ и всех $\alpha \in [0,1]$ выполняется неравенство

$$f(\alpha x + (1-\alpha) y) \le f(\alpha x) + (1-\alpha) f(y) \qquad (4.8)$$

Очевидно, что каждая точка любой хорды графика функции $f(x)$ лежит либо над графиком, либо принадлежит ему (Рис 4.1.).

Если для любого $\alpha \in (0,1)$ неравенство (4.8) строгое, то функция $f(x)$ - *строго выпуклая*. Примером выпуклой функции является квадратичная функция.

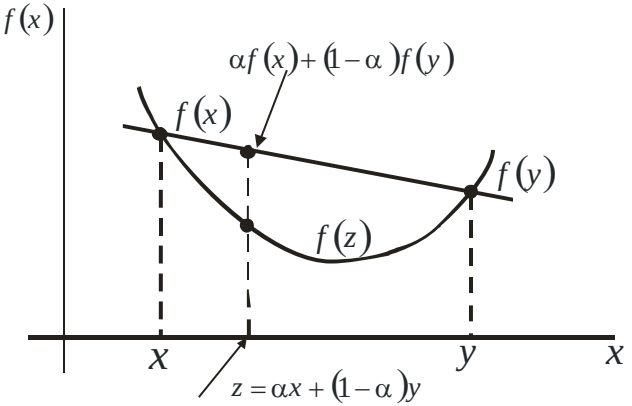


Рис.4.1. Выпуклая функция.

Вогнутой функцией называется такая функция $f(x)$, для которой $(-f(x))$ выпукла. Следовательно, если

$$f(\alpha x + (1-\alpha) y) \ge f(\alpha x) + (1-\alpha) f(y) \qquad (4.9)$$

для любых $x, y \in X$ и всех $\alpha \in [0,1]$, то $f(x)$ - вогнутая функция.

Основная задача математического программирования заключается в следующем.

Дано множество $X = \{x: \varphi_i(x) \geq 0, \ i = \overline{1,m}\}$, где $\varphi_i(x), \ i = \overline{1,m}$ - заданные скалярные функции.

Пусть скалярная функция $f(x)$ определена на множестве X.

Минимизация функции $f(x)$ на множестве X называется основной задачей математического программирования. Существуют различные формы записи этой задачи:

$$f(x) \to min, \ x \in X, \qquad (4.10)$$

или эквивалентная вторая форма записи

$$min\{f(x): x \in X\} \ \text{ и } \ \min_{x \in X} f(x). \qquad (4.11)$$

Эти формы записи означают, что ставится задача:

1. Либо найти точку минимума функции $x^* \in X$

$$f(x^*) = \min_{x \in X} f(x)$$

x^* - является точкой минимума функции $f(x)$ на множестве X, если $f(x^*) \leq f(x)$ для всех $x \in X$. Таких точек может быть несколько и даже бесконечное множество (в зависимости от свойств множества X и функции). Множество всех минимумов $f(x)$ на X обозначим как X^*.

2. Либо, если не существует такая точка x^*, то найти

$$f^* = \inf_{x \in X} f(x)$$

Это означает, что $f(x)$ не ограничена снизу на множестве X и в качестве нижней грани принимается значение $(-\infty)$.

3. Либо убедиться, что $X = \emptyset$ (множество пусто).

Пример 1. Пусть функция $f(x) = sin^2(\pi / x)$ определена на множестве $X = \{x: 1 \leq x \leq 2\}$. Минимальное зна-

чение равно нулю, множество X^* состоит из единственной точки $x^* = 1$.

Пример 2. Пусть функция $f(x) = \ln(x)$ определена на множестве $X = \{x : 0 < x \le 1\}$. В этом случае $X^* = \emptyset$, так как во всех точках из X функция принимает конечные значения, а для последовательности $x_k = 1/k \quad (k = 1, 2, ..)$ имеет $\lim_{k\to\infty}(x_k) = -\infty$.

Пример 3. Функция $f(x) = |x| + |x-1| - 1$ определенная на множестве $X = \{x : |x| \le 1\}$ принимает своё наименьшее значение, равное нулю, во всех точках отрезка $X^* = \{x : 0 \le x \le 1\}$.

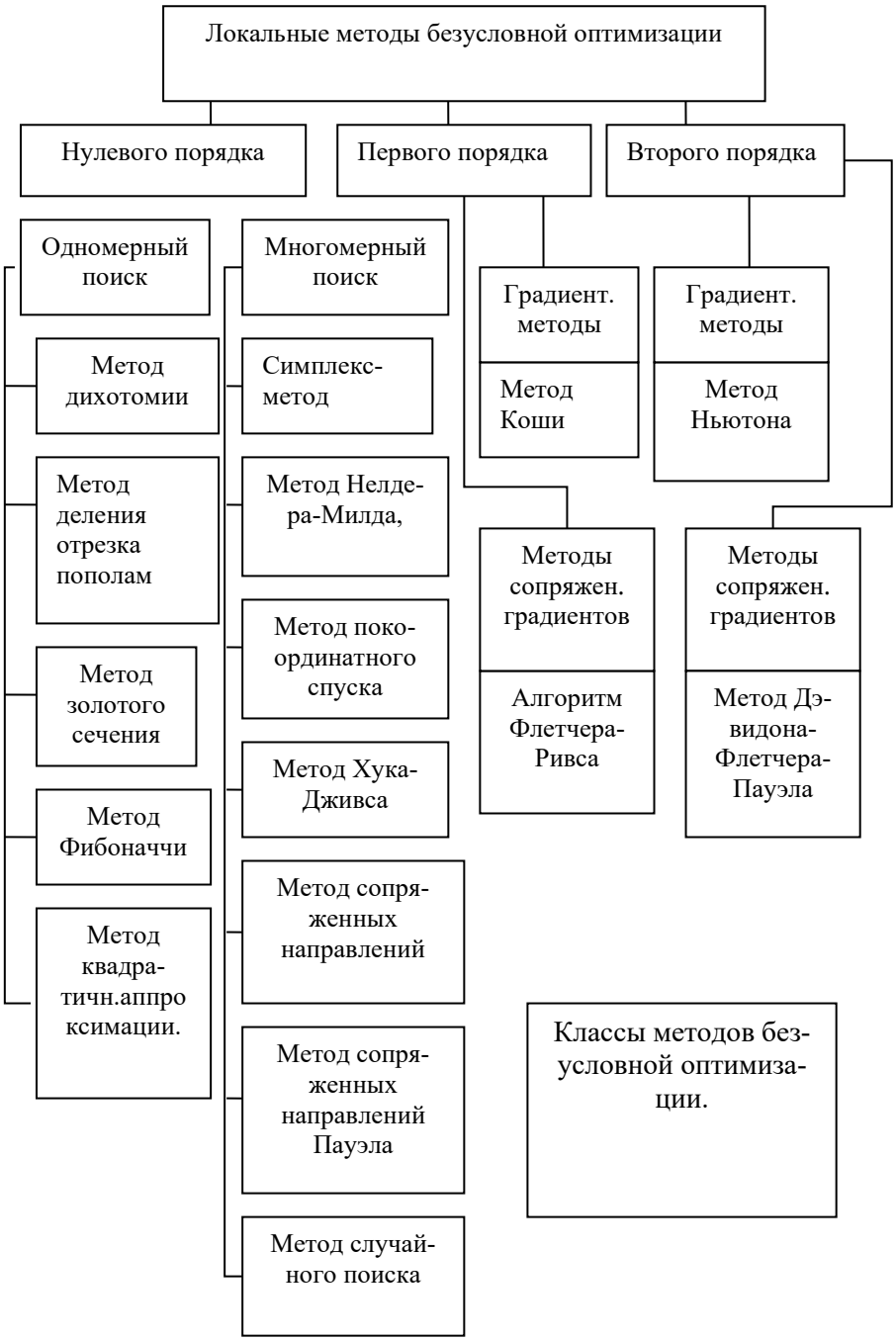
Если $X = \{x : 1 \le x \le 2\}$, то X^* содержит только одну точку $x^* = 1$. Если $X = \{x : 1 < x \le 2\}$, то $X^* = \emptyset$.

В настоящее время существует множество разнообразных алгоритмов нахождения минимумов одномерных и многомерных функций [1 - 9]. Выбор того или иного алгоритма для решения задачи является сам по себе достаточно сложной проблемой, так как необходимо адекватно сопоставить физические свойства модели с особенностями и возможностями выбранного метода поиска минимума функции. Далее будут рассмотрены некоторые наиболее характерные алгоритмы.

Задачи оптимизации, как правило, связаны с нахождением минимума многомерной целевой функции. Весь алгоритм поиска минимума в большинстве случаев можно разделить на отдельные последовательные этапы:

1. Стратегия поиска минимума многомерной целевой функции (с возможностью нахождения многомерного глобального минимума).
2. Стратегия поиска минимума одномерной целевой функции (с возможностью нахождения глобального минимума по одной переменной).

В дальнейшем изложении материала рассмотрим сначала алгоритмы поиска минимума одномерной унимодальной целевой функции, затем алгоритмы поиска минимума многомерной целевой функции и в заключение – алгоритмы поиска глобального минимума. На практике реализуются, как алгоритмы, не требующие вычисление производных минимизируемой функции, так и алгоритмы с использованием производной. Решить проблему нахождения минимума целевой функции, анализируя выражение первой производной исследуемой функции не всегда возможно. Это связано с тем, что в реальных практических задачах часто заранее неизвестно является ли рассматриваемая функция дифференцируемой. Кроме того, решение уравнения $f'(x) = 0$ может оказаться достаточно сложной задачей. Выбор того или иного алгоритма определяется общим характером решаемой задачи поиска минимума многомерной функции. Поскольку априори в общем случае неизвестно поведение функции на интервале поиска, то необходимо провести предварительное исследование на предмет определения числа возможных минимумов функций и поиска глобального минимума целевой функции.


Локальные методы безусловной оптимизации
Нулевого порядка
Первого порядка
Второго порядка
Одномерный поиск
Многомерный поиск
Градиент. методы
Градиент. методы
Метод дихотомии
Симплекс-метод
Метод Коши
Метод Ньютона
Метод деления отрезка пополам
Метод Нелдера-Милда,
Методы сопряжен. градиентов
Методы сопряжен. градиентов
Метод золотого сечения
Метод покоординатного спуска
Алгоритм Флетчера-Ривса
Метод Дэвидона-Флетчера-Пауэла
Метод Фибоначчи
Метод Хука-Дживса
Метод квадратичн.аппроксимации.
Метод сопряженных направлений
Метод сопряженных направлений Пауэла
Метод случайного поиска
Классы методов безусловной оптимизации.

Алгоритмы решения задачи следует разделять на *принципиальный* и *реализуемый* алгоритмы. Различие между этими понятиями заключается в следующем – на каждой итерации принципиальный алгоритм может осуществлять произвольное число арифметических операций и обращений к функциям, а реализуемый алгоритм только конечное число подобных действий. При этом считается, что приемлемое приближенное целевой функции может быть вычислено за приемлемое конечное время.

Процесс конструирования метода расчёта заключается в разработке принципиального алгоритма, который затем преобразуется в реализуемый алгоритм таким образом, чтобы свести каждую его итерацию к конечному числу машинных операций.

4.2. Поиск минимума одномерных унимодальных функций.

4.2.1. Методы исключения интервалов.

Выбор метода зависит от конкретной задачи и выделить априори метод лучший по всем параметрам не представляется возможным. Различаются два подхода в стратегии поиска оптимума.

1. Определение минимума целевой функции в заданном интервале поиска, с требуемой точностью, на основе анализа её значений в пробных точках без ограничений их числа (т.е. обращений к вычислению функции).
2. Определение минимума целевой функции методом, гарантирующим наилучшую точность, при заданном заранее допустимом количестве обращений к вычислению функции.

Рассматриваемые алгоритмы поиска минимума функции одной переменной предполагают наличие только одного минимума функции (унимодальная функция) на исследуемом интервале изменений аргумента. В понятие минимизации функции следует включить не только поиск её экстремума, но и оценку минимального значения функции на краях ис-

следуемого интервала, т.е. функция может быть монотонно возрастающей или убывающей.

Рассмотрим алгоритмы нахождения локального минимума методом *сокращения интервала поиска* [1,2,3,4]. Все алгоритмы построены на анализе значений минимизируемой функции в пробных точках, расположенных внутри интервала поиска. Различие методов заключается в принципе выбора места расположения этих пробных точек.

Принцип симметричного поиска (симметричного расположения пробных точек) реализуется в задачах оптимизации несколькими способами – методом дихотомии, методом деления отрезка пополам, методом золотого сечения, а также методом Фибоначчи. Алгоритмы различаются быстродействием, которое, в основном, связано с количеством обращений к вычислению минимизируемой функции $f(x)$. В большинстве случаев этот критерий является основным, т.к. вычисление самой функции составляет, как правило, основное время счёта. При этом нужно учитывать, что процесс поиска минимума функции определяется задаваемой точностью расчётов, т.е. конечным размером интервала $[a_{кон}, b_{кон}]$

Дихотомия.

Наиболее простой алгоритм, в котором на каждой итерации интервал поиска минимума функции *сокращается вдвое*. Вычисление значения минимизируемой функции на каждой итерации производится дважды. Суть алгоритма заключается в следующем – на текущей итерации определяется $x_{ср}$ середина отрезка $[a,b]$ и вычисляются значения двух близко лежащих точек $x_1 = x_{ср} - \varepsilon$ и $x_2 = x_{ср} + \varepsilon$, где ε наперёд заданное малое значение аргумента функции (Рис.4.2). Далее сравниваются значения функции в точках x_1 и x_2.

Исходя из условия существования только одного минимума функции на данном интервале поиска, можно определить, какую часть интервала поиска можно исключить на следующей итерации. При выполнения условия $f(x_1) < f(x_2)$ (Рис.4.2а.) – левую, при $f(x_1) > f(x_2)$ (Рис.4.2б.) - правую. Соответствующим образом переносятся границы интервала

$a \rightarrow x_{cp}$ или $b \rightarrow x_{cp}$ для продолжения поиска на следующей итерации. Поиск завершается, когда интервал поиска сокращается до наперёд заданной малой величины δ. Возможна ситуация, когда минимум функции окажется лежащим в интервале $[x_1, x_2]$. В этом случае весь алгоритм поиска сведётся к стягиванию интервала поиска к средней точке x_{cp}. Количество обращений к вычислению целевой функции определяется как $2*N$, где N – число шагов поиска.

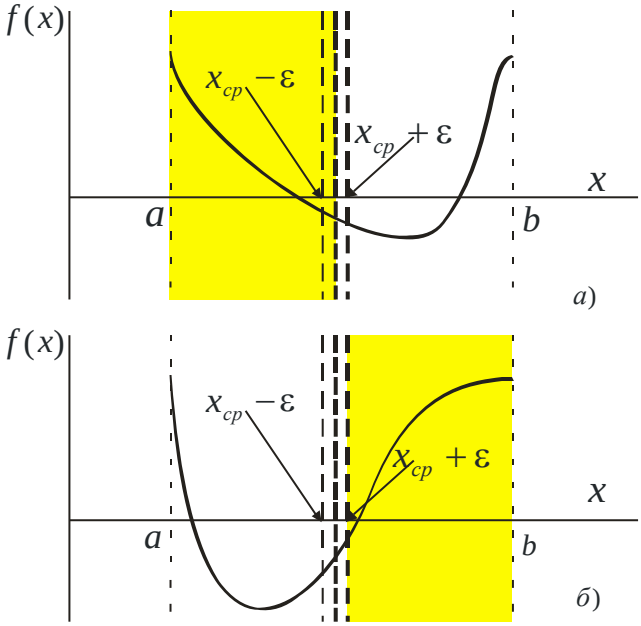


Рис.4.2. Схема сокращения интервала поиска в зависимости от расположения минимума функции.

Алгоритм поиска.

Пусть функция $f(x)$ задана на отрезке $[a,b]$ и имеет только один минимум. Параметры ε, δ – малые значения.

Шаг1: Определить середину интервала $x_{cp}=(a+b)/2$, выделить две точки на оси аргументов $x_1 = x_{cp} - \varepsilon$ и $x_2 = x_{cp} + \varepsilon$.

Шаг2: Вычислить значения функции $f(x_1)$ и $f(x_2)$.
Шаг3: Сравнить значения функции $f(x_1)$ и $f(x_2)$.
Если $f(x_1) < f(x_2)$, то присваиваем $b = x_{cp}$, иначе $a = x_{cp}$.
Шаг4: Признак завершения счета. Если $abs(b-a) < \delta$, то перейти к шагу 5, иначе перейти к шагу 1.
Шаг5: Вычислить $x_{min}=(a+b)/2$ и $f(x_{min})$

Деление отрезка пополам.

Метод, в котором на каждой итерации интервал поиска минимума функции *сокращается вдвое*. Он основан на выборе трёх пробных точек, расположенных равномерно на интервале поиска минимума целевой функции. На текущей итерации определяется x_{cp} середина отрезка $[a,b]$ и значения двух симметрично расположенных точек $x_1 = x_{cp} - abs(b-a)/4$ и $x_2 = x_{cp} + abs(b-a)/4$. Далее сравниваются значения функции в точках x_{cp}, x_1, и x_2. Можно определить, какую часть интервала поиска можно исключить на следующей итерации.

На Рис.4.3. приведены возможные положения расположения минимума целевой функции, которые могут возникнуть в процессе сокращения интервала поиска. Заштрихованные области исключаются из дальнейшего рассмотрения. При $f(x_1) > f(x_{cp})$ – правую половину (Рис.4.3а.), при $f(x_1) < f(x_{cp})$ - левую половину (Рис.4.3б.) и при $f(x_2) > f(x_{cp})$ - левую и правую четверти (Рис.4.3в.). Соответствующим образом переносятся границы интервала поиска $a \rightarrow x_{cp}$ (Рис.4.3а.) или $b \rightarrow x_{cp}$ (Рис.4.3б.) или $a \rightarrow x_1$ и $b \rightarrow x_2$ (Рис.4.3в.) и формируется очередной интервал $[a,b]$.
Поиск завершается, когда интервал поиска $[a,b]$ сокращается до заданной малой величины δ.

На первом шаге поиска производится три обращения к вычислению целевой функции в точках x_{cp}, x_1, и x_2 на последующих шагах поиска – два обращения (в точках x_1, и x_2) Общее количество обращений к вычислению целевой функции определяется как $(2*N+1)$, где N – число шагов поиска.

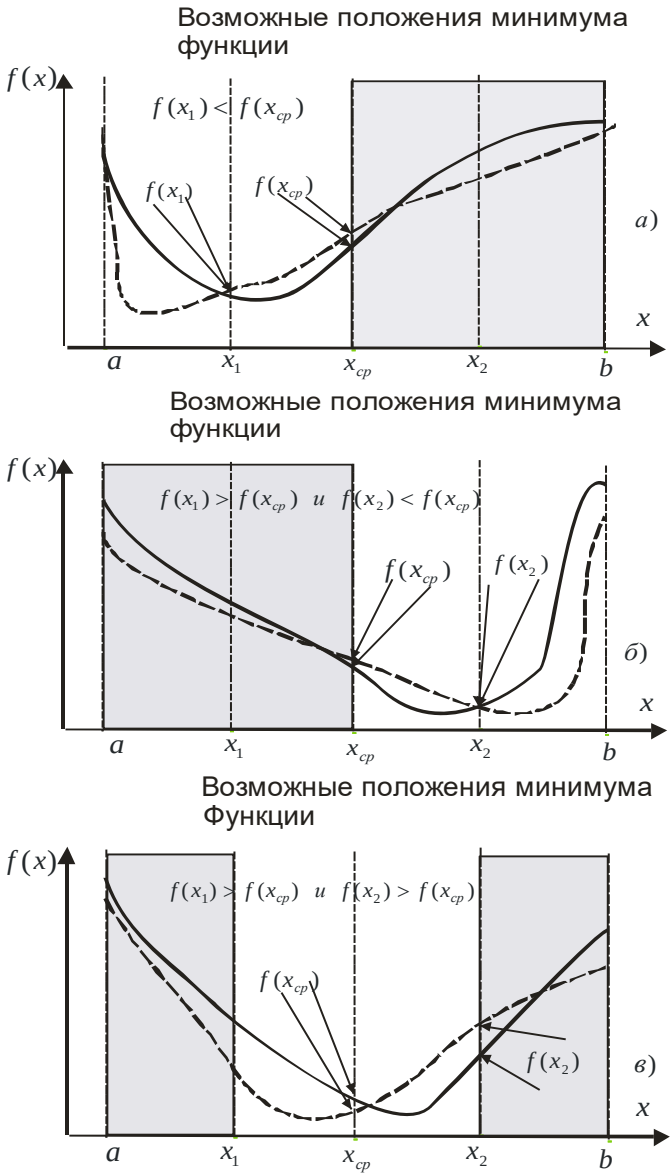


Рис.4.3. Возможные положения минимумов целевой функции.

Алгоритм поиска.

Пусть функция $f(x)$ задана на отрезке $[a,b]$ имеет только один минимум. δ – малое значения.

Шаг1: Определить середину интервала $x_{cp}=(a+b)/2$. Вычислить в этой точке значение функции $fx_{cp}= f(x_{cp})$.

Шаг2: Определить точки $x_1= x_{cp} – abs(b-a)/4$ и $x_2= x_{cp} + abs(b-a)/4$.

Шаг3: Вычислить значения функции $fx_1= f(x_1)$, $fx_2 =f(x_2)$.

Шаг4: Сравнить значения функции fx_1 и fx_{cp}.

Если $fx_1 < fx_{cp}$, то присвоить $b= x_{cp}$, $x_{cp}=x_1$, $fx_{cp}= fx_1$, вычислить точки $x_1= x_{cp} – abs(b-a)/4$ и $x_2= x_{cp} + abs(a)/4$, перейти к шагу 6. Иначе перейти к шагу 5.

Шаг5: Сравнить значения функции fx_2 и fx_{cp}.

Если $fx_2 < fx_{cp}$, то присвоить $a= x_{cp}$, $x_{cp}=x_2$, $fx_{cp}= fx_2$, вычислить точки $x_1= x_{cp} – abs(b-a)/4$ и $x_2= x_{cp} + abs(b-a)/4$, перейти к шагу 6. Иначе присвоить $a= x_1$, $b= x_2$, вычислить точки $x_1= x_{cp} – abs(b-a)/4$ и $x_2= x_{cp} + abs(b-a)/4$

Шаг6: Завершения счета. Если $abs(b-a)< \delta$, то перейти к шагу 7, иначе перейти к шагу 3.

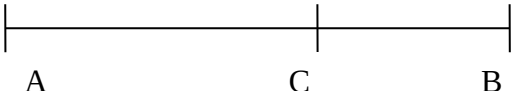
Шаг7: Вычислить $x_{min}=(a+b)/2$ и $f(x_{min})$

Золотое сечение.

Из рассмотрения выше приведенных методов можно сделать выводы позволяющие повысить эффективность поиска:

1. Если количество пробных точек равно двум, то их следует размещать на одинаковых расстояниях от середины интервала.
2. В соответствии с общей минимаксной стратегией пробные точки должны размещаться по симметричной схеме таким образом, чтобы отношение длины исключаемого подынтервала к величине интервала поиска оставалось постоянным.
3. На каждой итерации поиска должна вычисляться только одна пробная точка и значение целевой функции в этой точке.

Геометрическая интерпретация принципа метода золотого сечения заключается в следующем. Дана прямая АСВ



Отношение отрезков прямой определяется соотношением AC/AB=CB/AC, т.е. отношение большего отрезка к целому равно отношению меньшего отрезка к большему отрезку. Если принять целый отрезок AB=1, то $AC = (-1+\sqrt{5})/2$. Приближённо AC≈0.62, а CB≈0.38.

Процесс поиска минимума целевой функции носит итерационный характер, и положение пробных точек относительно текущего размера интервала поиска на каждой итерации переопределяется. Рассмотрим исходное симметричное расположение двух пробных точек на *единичном* интервале (Рис.4.4). Пробные точки отстоят от краёв интервала на расстоянии λ. (Под λ понимается не абсолютная, а относительная величина для текущего интервала поиска). При таком симметричном расположении точек длина оставшегося подынтервала (после исключения ненужной для дальнейшего анализа части) всегда равна λ независимо от того, какое значение функции в пробных точках окажется меньшим. Предположим, что исключили правый подынтервал. Оставшийся подынтервал длины λ содержит одну пробную точку, расположенную на расстоянии (1- λ) от его левой граничной точки. Чтобы сохранить симметрию в стратегии поиска, расстояние (1- λ) предшествующего шага поиска должно составлять λ-ю часть длины нового интервала поиска (которая на текущем шаге уже составляет величину равную λ от предшествующего интервала). При таком выборе следующая пробная точка расположится на расстоянии равном λ-й части длины интервала от правой граничной точки (Рис.4.4.). Этот алгоритм реализуется при $\lambda = (-1+\sqrt{5})/2$. Поиск, при котором интервал поиска делится пробными точками в указанном отношении, называется поиском с помощью золотого сечения (Рис.4.5). На первой итерации целевая функция вычисляется в двух пробных точках, на последующих итерациях производится

однократное обращение к вычислению целевой функции.

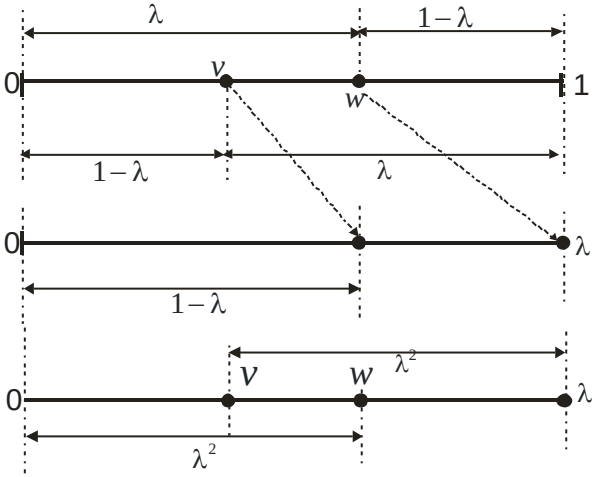


Рис.4.4. Поиск с помощью метода золотого сечения. Симметрия золотого сечения интервала.

Количество обращений к вычислению целевой функции определяется как (N+1), где N – число шагов поиска.

Пусть функция *f(x)* задана на отрезке [*a,b*] и имеет только один минимум. Параметр δ – малое значение.

Шаг1: Задать коэффициенты $\lambda 2$=0.62, $\lambda 1$=0.38, интервал *[a,b]* и длину отрезка *l=abs(b-a)*.

Шаг2: Выделить две точки $v=a+l*\lambda 1$ и $w=a+l*\lambda 2$.

Шаг3: Если abs(*b-a)* < δ , то перейти к шагу 7,
иначе перейти к шагу 4.

Шаг4: Вычислить в этих точках значения функции *fv=f(v)* и *fw=f(w)*.

Шаг5: Сравнить значения функции *fw* и *fv*.
Если *fw* < *fv* , то присвоить *a=v, v=w, fv =fw* и вычислить *w=a+abs(b-a)* λ2, f(w).* Перейти к шагу 3.
иначе перейти к шагу 6.

Шаг6: Присвоить *b=w, w=v, fw =fv* и вычислить *v=a+abs(b-a)* λ1, f(v).* Перейти к шагу 3.

Шаг7: Вычислить $x_{min}=(a+b)/2$ и $f(x_{min})$.

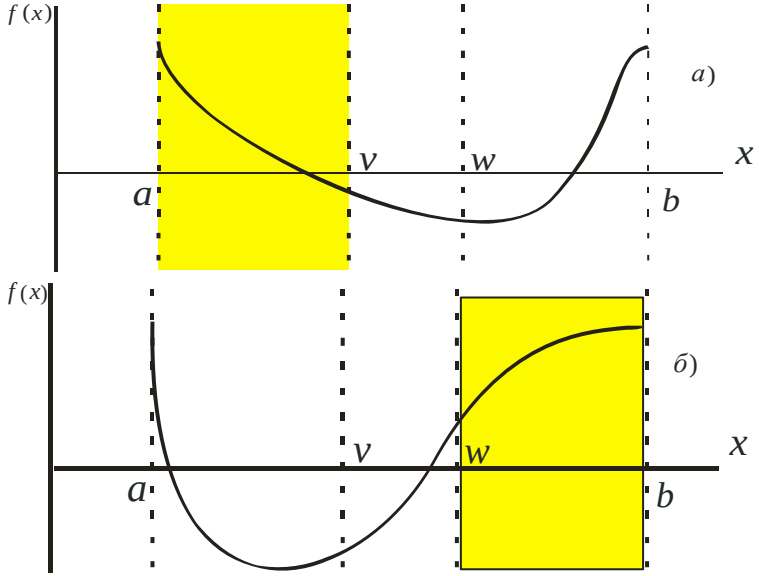


Рис.4.5. Возможные положения минимума целевой функции.

Пусть длина исходного интервала поиска L_1, а конечного интервала - L_n. Эффективность метода можно оценить по отношению L_n / L_1. В методах дихотомии и деления отрезка пополам длина конечного интервала составляет $L_1(0{,}5)^{N/2}$, а в методе золотого сечения - $L_1(0{,}618)^{N-1}$. При заданной точности вычислений ε количество обращений к вычислению целевой функции в двух первых методах составляет $N = 2\ln(\varepsilon)/\ln(0{,}5)$, а для метода золотого сечения $N = 1 + \ln(\varepsilon)/\ln(0{,}618)$.

Метод Фибоначчи.

В методе золотого сечения значение параметра λ является фиксированным. Метод Фибоначчи в принципе ничем не отличается от метода золотого сечения за исключением того, что параметр λ является вычисляемым и определяется, исходя из наперёд заданного допустимого количества обра-

щений к вычислению целевой функции m. Следовательно, если количество вычислений целевой функции m задано априори, то для этого варианта поиска параметр λ сокращения интервала поиска определяется однозначно. Пусть $\lambda = \lambda_m$, тогда алгоритм последовательного сокращения интервала поиска $[a,b]$ будет выглядеть следующим образом

$$
\begin{array}{lll}
 & \Delta_0 & \Delta_0 = b_0 - a_0 \\
k=1 & \Delta_1 = \Delta_0 * \lambda_m & \Delta_1 = b_1 - a_1 \\
k=2 & \Delta_2 = \Delta_0 - \Delta_1 = \Delta_0 * (1 - \lambda_m) & \Delta_2 = b_2 - a_2 \\
k=3 & \Delta_3 = \Delta_1 - \Delta_2 = \Delta_0 * (-1 + 2 * \lambda_m) & \Delta_3 = b_3 - a_3 \quad (4.12) \\
k=4 & \Delta_4 = \Delta_2 - \Delta_3 = \Delta_0 * (2 - 3 * \lambda_m) & \Delta_4 = b_4 - a_4 \\
k=5 & \Delta_5 = \Delta_3 - \Delta_4 = \Delta_0 * (-3 + 5 * \lambda_m) & \Delta_5 = b_5 - a_5 \\
k=6 & \Delta_6 = \Delta_4 - \Delta_5 = \Delta_0 * (5 - 8 * \lambda_m) & \Delta_6 = b_6 - a_6 \\
k=i & & \\
\Delta_i = \Delta_{i-2} - \Delta_{i-1} = \Delta_0 * (-1)^i (F_{i-1} - F_i \lambda_m) & & \Delta_i = b_i - a_i \quad (4.13)
\end{array}
$$

где F_i – числа Фибоначчи.

Принцип построения последовательности чисел Фибоначчи

$$F_{i+2} = F_{i+1} + F_i \qquad (4.14)$$

Таблица 4.1. Числа Фибоначчи.

F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	…	F_i	…
1	1	2	3	5	8	13	21	34	55	…		…

Также как и в методе золотого сечения, пробные точки на k-ой итерации будем обозначать v_k и w_k.

В зависимости от анализа значений целевой функции в пробных точках v и w, границы интервала поиска формируются как

1. $a_k = a_{k-1}$, $w_k = v_{k-1}$, $b_k = w_{k-1}$

Вычисляется новое значение v_k и $f(v_k)$ или

2. $b_k = b_{k-1},\ v_k = w_{k-1},\quad a_k = v_{k-1}$

Вычисляется новое значение w_k и $f(w_k)$.

При предельном числе обращений к целевой функции равном m, последний интервал поиска можно выразить как

$$\Delta_m = (-1)^m \Delta_0 (F_{m-1} - \lambda_m F_m) \qquad (4.15)$$

Если положить $\lambda_m = F_{m-1} / F_m$, интервал поиска $\Delta_m = 0$.

Числа Фибоначчи F_k можно вычислить для любого значения индекса k, используя формулу Бинэ.

$$F_k = \frac{1}{\sqrt{5}}\left[\left(\frac{1+\sqrt{5}}{2}\right)^k - \left(\frac{1-\sqrt{5}}{2}\right)^k\right] \quad k = 1, 2, \ldots \qquad (4.16)$$

Пусть $\lambda_{k+1} = F_k / F_{k+1}, \quad k = 1,\ 2, \qquad (4.17)$

Определяем параметр

$$\lambda_{k+1} = \frac{\sqrt{5}-1}{2} * \frac{1-(-1)^k\left(\frac{3-\sqrt{5}}{2}\right)^k}{1-(-1)^{k+1}\left(\frac{3-\sqrt{5}}{2}\right)^{k+1}}, \qquad (4.18)$$

используя формулу Бинэ, в зависимости от фиксированного значения индекса k.

Таблица 4.2. Коэффициенты сокращения интервала поиска.

k	1	2	3	4	5	6
λ_k	1	0,5	0,666667	0,6	0,625	0,615385
k	7	8	9	10	11	12
λ_k	0,61905	0,61765	0,61818	0,61798	0,61806	0,61803

При $k \to \infty, \quad \lim_{k\to\infty} \lambda_k = \frac{\sqrt{5}-1}{2}$. Следовательно, параметр сокращения интервала поиска в методе Фибоначчи в пределе совпадает с параметром сокращения интервала поиска в методе золотого сечения. Предпоследний интервал поиска

$\Delta_{m-1} = b_{m-1} - a_{m-1}$ определяет точность вычисления. Последний шаг выполняется с использованием алгоритма дихотомии.

Таблица 4.3. Относительное уменьшение интервала.

Число вычислений функции	Деление отрезка пополам	Метод дихотомии	Метод золотого сечения	Метод Фибоначчи
1	1,0	1,0	1,0	1,0
2		0,5	0,618	0,5
3	0,5		0,382	0,333
4		0,25	0,236	0,200
5	0,25		0,146	0,125
6		0,125	0,090	0,077
7	0,125		0,056	0,048
8		0,0625	0,0345	0,0294
9	0,0625		0,0213	0,0182
10		0,0312	0,0132	0,0112
11	0,0312		0,00813	0,00694
12		0,0156	0,00502	0,00429
13	0,0156		0,00311	0,00265
14		0,00781	0,00192	0,00164
15	0,00781		0,00119	0,00101
16		0,00391	0,000733	0,000626
17	0,00391		0,000453	0,000387
18		0,00195	0,000280	0,000239
19	0,00195		0,000173	0,000148
20		0,000976	0,000107	0,0000913

Исследование методов сокращения интервала поиска минимума целевой функции показало, что наибольшая эффективность достигается при симметричном расположении пробных точек относительно краёв интервала. При большом числе обращений к вычислению целевой функции методы золотого сечения и Фибоначчи практически совпадают по эффективности. Различие между ними заключается в том, что метод

Фибоначчи применим для случая, когда количество обращений к вычислению целевой функции заранее фиксировано. В методе золотого сечения этого ограничения не существует. Следует отметить, что приближённые вычисления коэффициента λ приводят к накоплению ошибки, что в свою очередь может привести к нарушению принципа вложенности отрезков $[b_k, a_k]$ и расходимости процесса поиска.

Рассмотренные алгоритмы являются базовыми и допускают различные модификации.

4.2.2. Полиномиальная аппроксимация.

Рассматриваемые методы [3], в отличие от методов исключения интервалов, применимы к гладким функциям. Идея методов состоит в аппроксимации гладкой функции полиномом, для которого можно определить точку оптимума. Необходимыми условиями являются унимодальность и непрерывность исследуемой функции. Эти методы логично использовать для поиска минимума одномерной функции в тех случаях, когда направления поиска связано с вычислением производных целевой функции. Эта ситуация имеет место и при решении задач многомерной оптимизации, когда определяется минимум целевой функции по выбранному направлению спуска. Качество оценки координаты точки оптимума, полученной с помощью аппроксимирующего полинома, можно повысить двумя способами – использовать полиномы более высокого порядка или уменьшить интервал аппроксимации. Второй способ предпочтительней, т.к. построение полинома высокого порядка можем оказаться сложной процедурой. На практике ограничиваются полиномами до третьего порядка.

Метод с использованием квадратичной аппроксимации.

Простейшим вариантом полиномиальной аппроксимация является интерполяция квадратичным полиномом. (Линейная интерполяция даёт минимальные значения на краях интервала.) В качестве интерполирующей функции можно использо-

вать квадратичный полином $p(x)$, построенный по первой интерполяционной формуле Ньютона.

$$p(x) = a_0 + a_1(x - x_1) + a_2(x - x_1)(x - x_2) \qquad (4.19)$$

где x_i, y_i – узлы и значения табличной функции в узлах,

$h = (x_{i+1} - x_i)$ - шаг таблицы.

Коэффициенты полинома определяются как

$$a_0 = y_1, \quad a_i = \Delta^i y_1 / (i! * h^i), \qquad (4.20)$$

$\Delta^i y_1$ - конечная разность.

Таким образом, по трём заданным точкам (значениям функции в узлах) можно построить квадратичный полином и по первой производной определить минимум интерполяционной функции.

$$\bar{x} = (x_2 + x_1)/2 - (a_1 / 2a_2) \qquad (4.21)$$

Поскольку рассматриваемая функция и интерполирующий полином являются унимодальными функциями, то можно ожидать что, $\bar{x}$ окажется приемлемой координатой для истинного минимума x^*.

На этом принципе построен алгоритм Пауэлла [6].

Начальная точка x_1, h – шаг по оси x.

Шаг1: Вычислить $x_2 = x_1 + h$,

Шаг2: Вычислить $f(x_1)$ и $f(x_2)$.

Шаг3: Если $f(x_1) > f(x_2)$, положить $x_3 = x_1 + 2h$.

Если $f(x_1) \le f(x_2)$, положить $x_3 = x_1 - h$,

Шаг4: Вычислить $f(x_3)$ и найти

$$f_{\min} = \min\{f_1, f_2, f_3\},$$

$$x_{\min} = точка\ x_i, которя соответствует\ f_{\min}$$

Шаг5: По точкам x_1, x_2, x_3 вычислить $\bar{x}$, используя процедуру интерполяции.

Шаг6: Проверка на окончание поиска. Если $f_{\min} - f(\bar{x})$ и $x_{\min} - \bar{x}$ оказываются достаточно малыми, то поиск закончен. Иначе перейти к шагу 7.

Шаг7: Выбрать «наилучшую» точку $x_{\min}$ *или* $\bar{x}$ и две точки по обе стороны от неё, пронумеровав их в естественном порядке, и перейти к шагу 4.

Примечание: При первой реализации шага 5 границы интервала, содержащего точку минимума, не обязательно оказываются установленными. Полученная точка $\bar{x}$ может находиться за точкой x_3. Для того, чтобы исключить возможность слишком большого экстраполяционного перемещения, следует провести после шага 5 дополнительную проверку и в случае, когда точка $\bar{x}$ находится слишком далеко от x_3, заменить $\bar{x}$ точкой, координата которой вычисляется заранее установленной длиной шага.

Методы с использованием производных.

Если рассматриваемые целевые функции являются не только непрерывными, но и дифференцируемыми, то эффективность процедур поиска можно повысить.

Метод Ньютона – Рафсона.

Пусть функция $f(x)$ дважды дифференцируема, тогда для функции $f'(x) = \varphi(x)$ условием нахождения минимума (экстремума) является нахождение такого x^*, для которого $\varphi'(x^*) = 0$. Решение этого уравнения производится методом касательных. Процесс носит итерационный характер. Выберем в качестве стартовой точки значение (первое приближение значения корня уравнения $\varphi'(x) = 0$) x_k. Из точки A (Рис. 4.6) проведем касательную до пересечения с осью абсцисс в точке x_{k+1}. Уравнение касательной имеет вид

$$\widehat{\varphi}(x) - \varphi(x_k) = \varphi'(x_k)(x - x_k), \qquad (4.22)$$

где x – текущее значение координаты, а $\widehat{\varphi}(x)$ - соответствующее значение линейной функции. Полагая $\widehat{\varphi}(x) = 0$, получаем

$$x_{k+1} = x_k - \frac{\varphi(x_k)}{\varphi'(x_k)} \qquad (4.23)$$

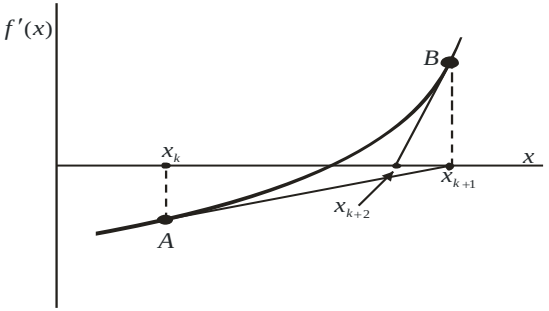


Рис.4.6. Процесс поиска минимума $f'(x) = \varphi(x)$

Вычислим значение функции $\varphi(x_{k+1})$ - точка *B* (второе приближённое значение корня). Из точки *B* проведем касательную до пересечения с осью абсцисс в точке x_{k+2}. Процесс продолжается до тех пор, пока не реализуется условие $\left|\varphi\left(x_{k+i}\right)\right| < \varepsilon$ (ε - малое число).

В этом алгоритме большое значение имеет выбор начальной точки. При неудачном выборе процесс поиска корня может расходиться. Рассмотренный алгоритм используется в разделе численных методов для решения нелинейных и трансцендентных уравнений. В поиске минимума функции $\varphi(x)$ можно использовать и метод секущих (хорд).

Метод поиска с использованием кубической аппроксимации.

Метод по своей сути схож с методом, использующим квадратичную аппроксимацию, но для построения аппроксимирующего полинома третьего порядка требуется меньшее число точек, т.к. в каждой точке вычисляется не только значение целевой функции, но и её производная. Начальная точ-

ка x_1 выбирается произвольно, точка x_2 выбирается таким образом, чтобы производные $f'(x_1)$ и $f'(x_2)$ имели разные знаки. В этом случае в интервале $[x_1, x_2]$ будет находиться точка x^*, содержащая минимум функции ($f'(x^*) = 0$) . Аппроксимирующий полином третьего порядка запишется в виде

$$p(x) = a_0 + a_1(x - x_1) + a_2(x - x_1)(x - x_2) + a_3(x - x_1)^2(x - x_2) \qquad (4.24)$$

Параметры этого уравнения подбираются так, чтобы значения $p(x)$ и $p'(x)$ в точках x_1 и x_2 совпадали со значениями $f(x)$ и $f'(x)$ в этих же точках. Выполнение этих условий приводит к нахождению коэффициентов a_0, a_1, a_2, a_3 из решения системы уравнений

$$\begin{aligned} f(x_1) &= a_0, \\ f(x_2) &= a_0 + a_1(x_2 - x_1), \\ f'(x_1) &= a_1 + a_2(x_1 - x_2), \\ f'(x_2) &= a_1 + a_2(x_1 - x_2) + a_3(x_2 - x_1)^2. \end{aligned} \qquad (4.25)$$

После нахождения коэффициентов, приравнивая первую производную полинома $p(x)$ нулю, можно получить выражение для определения точки минимума.

$$x^* = \begin{cases} x_2 & \text{если } \mu < 0, \\ x_2 - \mu(x_2 - x_1), & \text{если } 0 \le \mu \le 0, \\ x_1, & \text{если } \mu > 1, \end{cases} \qquad (4.26)$$

где

$$\mu = \frac{f'(x_2) + w - z}{f'(x_2) - f'(x_1) + 2w},$$

$$z = \frac{3(f(x_1) - f(x_2))}{x_2 - x_1} + f'(x_2) + f'(x_1), \qquad (4.27)$$

$$w = \begin{cases} (z^2 - f'(x_2) * f'(x_1))^{1/2} & \text{если} \quad x_1 < x_2, \\ -(z^2 - f'(x_2) * f'(x_1))^{1/2} & \text{если} \quad x_1 > x_2, \end{cases}$$

Получаемая точка x^* расположена между точками x_1 и x_2. Затем снова выбираются две точки, одна из которых - x^*, а другая – либо x_1, либо x_2 (значения производных в паре выбранных точек должны быть противоположны по знаку). Далее процесс кубической аппроксимации продолжается до достижения необходимой точности определения минимума.

Алгоритм метода кубической аппроксимации может быть следующим:

Выбрать начальную точку x_0, h – шаг по переменной, ε_1 и ε_2 - параметры сходимости.

Шаг 1: Вычислить $f'(x_0)$.

Если $f'(x_0) < 0$, то вычислить $x_{K+1} = x_K + 2^K h$,

K=0, 1, 2 ...

Если $f'(x_0) > 0$, то вычислить $x_{K+1} = x_K - 2^K h$,

K=0, 1, 2 ...

Шаг 2: Вычислить $f'(x)$ в точках x_{K+1},

K=0, 1, 2 ... вплоть до точки x_m, в которой

$f'(x_{m-1}) * f'(x_m) < 0$.

Затем положить $x_1 = x_{m-1}$, $x_2 = x_m$, вычислить $f(x_1)$, $f'(x_1)$, $f(x_2)$ и $f'(x_2)$.

Шаг 3: Найти точку x^* аппроксимирующего полинома (по формуле (4.26)).

Шаг 4: Если $f(x) < f(x_1)$, перейти к шагу 5. Иначе вычислять x^* по формуле $x^* = x^* * \frac{1}{2}(x^* - x_1)$ до тех пор, пока не будет выполнено условие $f(x^*) \le f(x_1)$.

Шаг 5: Условие окончания поиска.

Если $\left| f(x^*) \right| \le \varepsilon_1$ и $\left| (x^* - x_1) / x^* \right| \le \varepsilon_2$, то поиск завершён.

Иначе положить, либо

$$x_2 = x_1,\ x_1 = x^* \quad \text{если} \quad f'(x^*) * f'(x_1) < 0,$$

либо

$$x_1 = x^* \quad \text{если} \quad f'(x^*) * f'(x_2) < 0.$$

Затем перейти к шагу 3.

В случае, когда значения производной вычисляются непосредственно, рекомендуется использовать метод кубической аппроксимации. Если производная вычисляется методом численного дифференцирования, то эффективней использовать метод Пауэлла. Все выше приведённые методы поиска минимума целевой функции допускают различного рода модификации, позволяющие улучшить его быстродействие или повысить точность вычислений. Выбор нужного алгоритма является своего рода искусством и зависит от характерных особенностей решаемой задачи.

4.3. Поиск минимума функций многих переменных.

Разделение оптимизационных задач на классы удобно проводить по свойствам целевой функции и функциям ограничения. Целевая функция может быть квадратичной, полиномом, сепарабельной, однородной, составной, дробной и т.д. Функции ограничения могут быть нелинейными, квадратичными, полиномами, линейными и т.д. Ограничения могут быть односторонними, двусторонними или вообще отсутствовать. Используя различные сочетания видов целевой функции и ограничений, можно выделить различного типа задачи без-

условной оптимизации и различные типы задач с ограничениями (условная оптимизация). Методы поиска минимума целевой функции различаются способами построения базовых и пробных точек в пространстве поиска, определением направления и способом движения к минимуму целевой функции.

В настоящее время существует большое число базовых методов оптимизации функций многих переменных и их модификаций. Рассмотрим наиболее известные и часто употребляемые методы, связанные с безусловной оптимизацией (т.е. без ограничений) целевых функций многих переменных с самых общих положений.

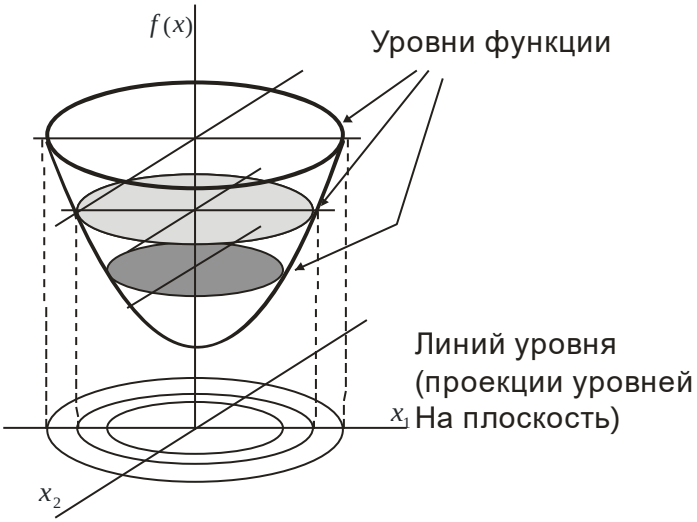


Рис. 4.7. Линии уровня целевой функции.

В дальнейшем рассмотрении используется понятие – линии уровня целевой функции. На рис.4.7. показаны несколько уровней простой целевой функции (параболоид вращения). Каждый уровень характеризуется одним значением целевой функции. Верхний обод параболоида, а также его сечения (затененные эллипсы) проецируются на плоскость $\{x_1, x_2\}$. Эти проекции являются линиями уровня (линиями, каждая точка которых имеет одно и то же численное значение соот-

ветствующего уровня целевой функции). Подобным способом в картографии указываются глубины морей и высота местности. При рассмотрении задач многомерной оптимизации могут быть полезны некоторые геометрические представления. Целевая функция представляется в виде так называемой поверхности отклика. В одномерном случае (целевая функция – функция одной переменной) – она вырождается в кривую на плоскости. В двумерном случае целевую функцию можно представить в трёхмерном пространстве (как, например, на рис.4.7.). При большей размерности пространства поиска минимума целевой функции поверхности отклика являются гиперповерхностями, которые невозможно отобразить обычными средствами.

Запишем вектор x, определённый в евклидовом пространстве R^n через его компоненты $x = \left(x^1, x^2, \ldots x^n\right)$. Функцию многих переменных можно разложить в ряд Тейлора

$$f(x) = f(x^*) + \nabla f(x^*)^T \Delta x + 1/2 \Delta x^T \nabla^2 f(x^*)^T \Delta x + O_3(x) \qquad (4.28)$$

где x^* - точка разложения функции из пространства R^n, $\Delta x = x - x^*$ - величина изменения x, $\nabla f(x)$ - n-мерный вектор-столбец первых производных функции $f(x)$ вычисленных в точке x^*, $\nabla^2 f\left(x^*\right) = H_j\left(x^*\right)$ - симметрическая матрица размерности $n \times n$ вторых частных производных функции $f(x)$ вычисленных в точке x^* (матрица Гессе).

Пренебрегая в (4.28) членами высших порядков, представим целевую функцию как квадратичную функцию, которая занимает значительное место в дальнейшем рассмотрении материала. Проведем анализ возможного поведения целевой функции в окрестности её экстремума.

Для произвольного значения x изменение целевой функции можно записать как

$$\Delta f(x) = f(x) - f(x^*) = \nabla f(x^*)^T \Delta x + 1/2 \Delta x^T \nabla^2 f(x^*)^T \Delta x \qquad (4.29)$$

В окрестности точки минимума выполняется условие

$$\Delta f(x) = f(x) - f(x^*) \geq 0 \qquad (4.30)$$

Если условие (4.30) выполняется для всех $x \in R^n$, то обозначим $x^* \to x^{**}$ как точку *глобального* минимума функции многих переменных. Если это условие справедливо только в окрестности x^*, эта точка соответствует *локальному* минимуму.

Если $\Delta f \leq 0$, то имеет место локальный максимум, если Δf принимает положительные, отрицательные значения или равна нулю в δ-окрестности, то x^* - *седловая* точка.

В предположении, что $f(x), \nabla f(x)$ и $\nabla^2 f(x),$ существуют и непрерывны для всех $x \in R^n$, знак Δf не меняется при произвольном варьировании Δx только в случае $\Delta f(x^*) = 0$.

Точка x^* является *стационарной* точкой. Следовательно, условие стационарности

$$\Delta f(x^*) = 0 \qquad (4.31)$$

и уравнение (4.29) принимает вид

$$\Delta f(x) = 1/2 \Delta x^T \nabla^2 f(x^*)^T \Delta x \quad (4.32)$$

Знак $\Delta f(x)$ определяется квадратичной формой

$$Q(x) = 1/2 \Delta x^T \nabla^2 f(x^*)^T \Delta x \quad (4.33)$$

или $Q(z) = z^T A z$

Стационарная точка x^* - точка минимума, если матрица **A** является положительно полуопределенной матрицей, т.е. $Q(z) \geq 0$ для любых z.

Существование минимума функции нескольких переменных в точке x^* определяется условиями необходимости и достаточности. Чтобы точка x^* являлась точкой минимума функции нескольких переменных *необходимо,* чтобы

$$\Delta f(x^*) = 0 \qquad (4.34)$$

и матрица $\nabla^2 f(x^*)$ была положительно полуопределенная, и *достаточно,* чтобы

$$\Delta f(x^*) = 0 \qquad (4.35)$$

и матрица $\nabla^2 f(x^*)$ была положительно определенная (т.е. $Q(z) > 0$).

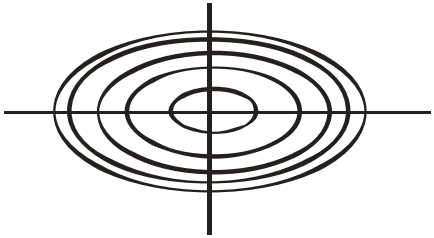


Рис. 4.8. Линии уровня «чаши», матрица $Q(z)$ положительно определена.

При выполнении условий необходимости и достаточности точка x^* является точкой изолированного (строгого) локального минимума $f(x)$. Поскольку априори, в общем случае, неизвестно поведение $f(x)$ во всей области поиска минимума, приходится ограничиваться нахождением локального минимума. Но, если можно показать, что $x^T \nabla^2 f(x^*)^T x \geq 0$ для всех x, то $f(x)$ является выпуклой функцией и найденный минимум является глобальным.

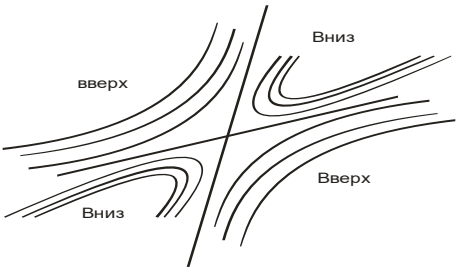


Рис.4.9. Линии уровня «седла», матрица $Q(z)$ не определена положительно.

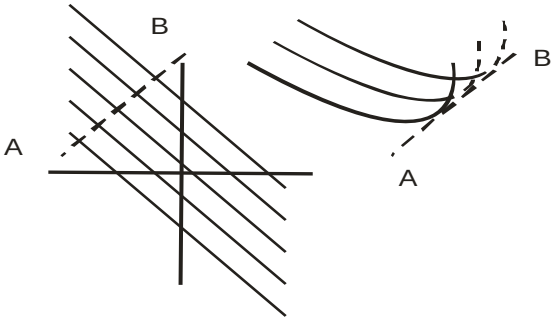


Рис.4.10. Линии уровня «оврага», матрица $Q(z)$ вырождена. Целевая функция квадратична относительно линии $A-B$.

Задачи, решаемые с использованием методов оптимизации функции многих переменных весьма разнообразны по своей сути, и поэтому каждому решению должен соответствовать адекватный математический метод. Методы решения задач безусловной оптимизации можно разделить на несколько широких классов:

1. Методы прямого поиска, построенные только на вычислении значений целевой функции.
2. Градиентные методы, в которых используются точные значения первых производных целевой функции.
3. Методы второго порядка, в которых используются первые и вторые производные целевой функции.
4. Стохастические методы.

Перед написанием программы реализующей любой алгоритм поиска полезно рассмотреть возможности преобразования задачи с целью улучшения поведения целевой функции в области минимума, сокращения времени вычисления функции и её градиента, уменьшения влияния погрешностей вычисления на процесс оптимизации. Каких-либо универсальных рекомендаций не существует и каждый случай требует индивидуального подхода.

Преобразование переменных. Бывает полезным произвести замену переменных с целью изменить характер линий уровня (поверхностей уровня в многомерном случае), придавая им менее изогнутый характер и вытягивая вдоль координаты. В зависимости от конкретного случая, замены могут носить как линейный, так и нелинейный характер.

Масштабирование целевой функции. Решение не измениться, если целевую функцию умножить на некую положительную константу или добавить константу, чтобы значения целевой функции находились в области близкой к единице. Эта операция называется масштабированием целевой функции, которая непосредственно связана с преобразованием переменных. Размерности компонент вектора аргумента целевой функции x могут оказать существенное влияние на анализ условий окончания поиска $\left|x^{(i+1)} - x^{(i)}\right| < \delta$ - сравнение двух соседних итераций поиска – *i*-ой и (*i+1*)-ой. Если размерности координат различаются на порядки, то определяющим фактором будет координата с наибольшими значениями.

Масштабирование ограничений. Изменение масштаба ограничений должно согласовываться с принципом преобразования переменных и масштабированием целевой функции.

Независимо от выбранного метода, необходимо решить задачи:

1. Выбор базовой (стартовой) точки алгоритма
2. Выбор направления поиска минимума целевой функции.
3. Выбор способа движения по направлению к минимуму.

4.3.1. Методы прямого поиска. Симплекс-метод.

Симплекс-метод, применяемый для решения задач нелинейного программирования, никак не связан с симплексным методом линейного программирования, кроме созвучности названий. Он относится к методам, основанным на вы-

числениях только значений целевой функции. Прототипом этого метода являлся метод *эволюционной оптимизации*, стратегия поиска в котором заключается в следующем – задаётся произвольная базовая точка в многомерной области поиска и исследуются определенным образом выделенные близлежащие к ней другие точки. Из этой группы точек выбирается точка с наилучшим (наименьшим) значением целевой функции и процесс повторяется. Подобная стратегия проста, но в случае большого числа переменных (большой размерности пространства поиска) требует больших вычислительных ресурсов и далеко не эффективна. Развитие подобной, но более эффективной стратегии, реализовано в *методе поиска по симплексу*. Основу этого метода составляет регулярный симплекс, который представляет собой многогранник, рёбра которого - прямые линии, пересекающиеся в (n+1)-ой равноотстоящих друг от друга вершинах, определённый в n-мерном пространстве. Например, в случае двумерного пространства, симплексом является равносторонний треугольник (Рис.4.11.), а трехмерном - тетраэдр.

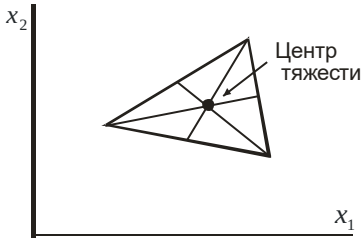


Рис.4.11. Пример симплекса в двумерном пространстве.

Процедура поиска начинается с построения начального симплекса в произвольной базовой точке пространства. Следует отметить, выбор месторасположения остальных вершин многогранника определяет размер симплекса. В выбранных вершинах симплекса вычисляются значения целевой функции.

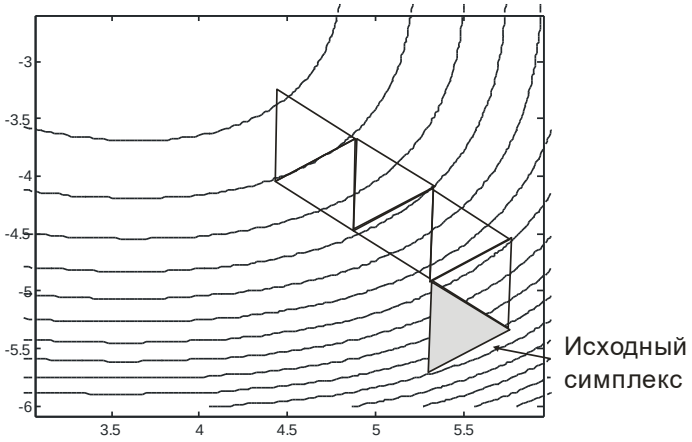


Рис.4.12. Процесс трансформации симплекса.

Выбирается вершина, в которой значение целевой функции наибольшее (наихудшая вершина). Определяется центр тяжести симплекса и прямая, соединяющая наихудшую вершину и центр тяжести симплекса. По направлению, определяемому этой прямой, строится вершина нового симплекса (Рис.4.12.). Итерационный процесс продолжается до тех пор, пока очередной симплекс не захватит точку минимума целевой функции или не произойдёт «зацикливание» алгоритма – циклическое движение по двум соседним положениям симплексов. Разрешение этой проблемы заключается в следующих действиях:

1. Если в новой полученной вершине значение целевой функции окажется худшим, то осуществляется возврат к вершине прежнего симплекса. Вместо неё берётся другая вершина, которой соответствует следующее по величине значение целевой функции и отыскивается точка, которая является её дополнением (на соответствующей ей прямой).
2. Если какая-либо вершина симплекса не исключается на протяжении более чем m итераций, то необходимо уменьшить размеры симплекса с помощью коэффици-

ента *редукции* и построить новый симплекс, выбрав опорной точкой вершину симплекса, имеющую наименьшее значение целевой функции. (Рис.4.13.).

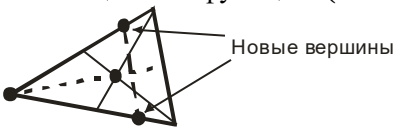


Рис.4.13. Редукция симплекса.

Значение m выбирается из соотношения

$$m = 1{,}65 * n + 0{,}05 * n^2 \qquad (4.36)$$

где n – размерность пространства поиска, m – округляется до ближайшего целого числа.

3. Поиск завершается, когда размеры симплекса или разности между значениями целевой функции в вершинах симплекса станут достаточно малыми. Степень малости задаётся начальными условиями, определяющими требуемую точность решения задачи.

Построение симплекса начинается с задания координат базовой точки $x^{(0)}$ и масштабного множителя α. Верхний индекс $(\cdot)$ означает номер базовой точки или вершины симплекса. Координаты остальных вершин симплекса определяются как

$$x^{(i)} = \begin{cases} x_j^{(0)} + \delta_1 & для \quad j \neq i \\ x_j^{(0)} + \delta_2 & для \quad j = i \end{cases} \qquad (4.37)$$

$$i, j = 1, 2, 3, ..., n$$

Величины приращений координат δ_1 и δ_2, зависящие от n и α, определяются по формулам

$$\delta_1 = \left[\frac{(n+1)^{1/2} + n - 1}{n\sqrt{2}} \right] * \alpha$$

$$\delta_2 = \left[\frac{(n+1)^{1/2} - 1}{n\sqrt{2}} \right] * \alpha \qquad (4.38)$$

Величина α выбирается исходя из характеристик решаемой задачи. При α=1 ребра регулярного симплекса имеют единичную длину.

Центр тяжести симплекса

$$x_c = \frac{1}{n}\sum_{\substack{i=0 \\ i \neq j}}^{n} x^{(i)} \qquad (4.39)$$

Все точки прямой, проходящие через центр тяжести симплекса и $x^{(j)}$ вершину симплекса, лежат на прямой

$$x = x^{(j)} + \lambda * (x_c - x^{(j)}). \qquad (4.40)$$

Исходной точке $x^{(j)}$ соответствует значение λ=0, центру тяжести x_c соответствует λ=1. Новая вершина регулярного симплекса получается при λ=2 и определяется соотношением

$$x_{нов}^{(j)} = 2 * x_c - x_{предыдущ}^{(j)} \qquad (4.41)$$

Рассмотренный алгоритм поиска минимума целевой функции симплекс-методом является базовым алгоритмом подобного типа, который обладает своими достоинствами и недостатками.

К достоинствам можно отнести следующие свойства:

1. Алгоритм достаточно прост с точки зрения написания программы расчёта.
2. Алгоритм не требует большого объёма памяти – хранится информация о координатах (n+1)-ой вершине симплекса.
3. Алгоритм мало чувствителен к ошибкам вычисления значений целевой функции, так как при его реализации оперируют с наибольшими значениями целевой функции в вершинах симплекса, а не с наименьшими.

К недостаткам следует отнести следующее:

1. Алгоритм достаточно медленный, так как информация, полученная на предшествующих итерациях, никак не используется на текущей итерации.

2. В алгоритме не предусмотрен простой способ расширения симплекса, не требующего пересчёта значений целевой функции во всех вершинах образца. Уменьшение размеров симплекса может происходить при движении в узком «овраге» или по «хребту» поверхности отражающей целевую функцию. При выходе из «оврага» (или «хребта») дальнейший поиск ведется в условиях уменьшенного шага.
3. Возможны трудности с масштабированием переменных. Желательно все переменные отмасштабировать таким образом, чтобы они имели сопоставимые по величине значения.

Модификацией симплекс-метода является ***алгоритм Нелдера-Милда***, который позволяет при отражении симплекса осуществлять операции его сжатия и растяжения (т.е. отказ от сохранения свойств регулярности симплекса в процессе поиска).

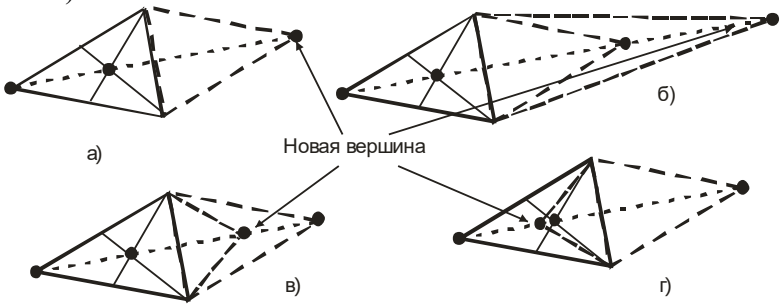


Рис.4.14. Преобразование симплекса в методе Нелдера-Мида.

В методе Нелдера-Мида преобразование симплекса – растяжение, сжатие и редукция – производятся на основании анализа трёх вершин симплекса (двумерная задача). Первая вершина $x^{(\max)}$, в которой значение целевой функции $f(x^{(\max)})$ наибольшее (наихудшая вершина). Вторая вершина $x^{(2)}$, в

которой целевая функция $f(x^{(2)})$ имеет второе по величине значение (после $f(x^{(\max)})$). Третья вершина $x^{(\min)}$, в которой значение целевой функции $f(x^{(\min)})$ наименьшее (наилучшая вершина). Отражение вершины симплекса производится вдоль прямой

$$x = x^{(\max)} + \lambda(x_c - x^{(\max)}) \quad \text{или} \quad x = x^{(\max)} + (1+\theta)(x_c - x^{(\max)}) \qquad (4.42)$$

Эти уравнения прямой эквивалентны уравнению (4.40). Параметр θ определяет характер преобразования симплекса.

- При $f(x^{(\min)}) < f(x^{(нов)}) < f(x^{(2)})$ и $\theta = 1$ происходит нормальное отражение вершины симплекса (Рис.4.14а.).
- При $f(x^{(нов)}) < f(x^{(\min)})$ и $\theta > 1$ – растяжение симплекса (Рис.4.14б.).
- При $f(x^{(нов)}) > f(x^{(2)})$, $f(x^{(нов)}) \geq f(x^{(\max)})$ и $\theta < 0$ - сжатие симплекса (Рис.4.14в.).
- При $f(x^{(2)}) < f(x^{(нов)}) < f(x^{(\max)})$ и $\theta > 0$ - сжатие симплекса (Рис.4.14г.).

Рекомендуется в качестве удовлетворительных значений параметра θ принять следующие величины: θ=1- нормальное отражение, θ=0.5 – сжатие, θ=2 – растяжение.
Последовательность действий следующая:
Шаг 1: Задать значения параметра преобразования симплекса θ и δ – точность вычислений.
Шаг 2: Построить исходный симплекс.
Шаг 3: Вычислить x_c, $x^{(\max)}$, $x^{(\min)}$, $x^{(2)}$.

Шаг 4: Произвести анализ целевой функции в точках $x^{(\max)}$, $x^{(\min)}$, $x^{(2)}$ и вычислить положение новой вершины, используя приведенные выше условия преобразования симплекса и уравнение прямой (4.42).

Шаг 5: Сравнить значения целевой функции в вершинах симплекса на двух последовательных итерациях – если значения различаются на величину меньшую или равную δ, то закончить поиск, иначе перейти к шагу 3.

Паркинсон и Хатчинсон [3] исследовали влияние значений коэффициента преобразования симплекса и способа построения исходного симплекса на эффективность поиска. Они установили, что ориентация исходного симплекса влияет на эффективность поиска значительно сильнее, чем его размеры, и предложили выбрать значения коэффициента преобразования симплекса: θ=2 – нормальное отражение, θ=0.25 – сжатие, θ=2.5 – растяжение. Ориентация симплекса в пространстве влияет на выбор направления поиска (4.42) и, следовательно, быстроту достижения минимума.

В литературе имеются рекомендации по увеличению числа вершин симплекса $(n+k)\ \ k=1,2,..$, так как при сложной конфигурации линей уровня возможно вырождение симплекса. Например, в двумерном случае симплекс может выродится в прямую. Наличие дополнительных вершин позволяет избежать этой опасности, но приводит к увеличению времени расчёта.

Методы спуска.

Различные методы спуска, применяемые для решения задач оптимизации, различаются либо выбором направления спуска, либо способом движения вдоль направления спуска или тем и другим.

В области поиска задается начальная точка $x^{(0)}=\left(x_1^{(0)},x_2^{(0)},x_3^{(0)},...\ x_n^{(0)}\right)$ и выбирается направление спуска, на котором определяется точка $x^{(1)}$, выбирается новое направление и определяется точка $x^{(2)}$ и так далее (n - размерность пространства). В результате строится последовательность точек $x^{(i)}$, сходящая к точке минимума целевой функции.

Выбор направления. Во всех методах, независимо от способа определения направления спуска, должно соблюдаться условие $f\left(x^{(i)}\right) > f\left(x^{(i+1)}\right)$, т.е. значение целевой функции на последующей итерации меньше, чем на предыдущей итерации.

Метод покоординатного спуска.

Метод покоординатного спуска относится к методам нулевого порядка, в которых процесс поиска базируется только на вычислении значений целевой функции в выбранных соответствующим образом пробных точках. Существует много вариантов метода покоординатного спуска. Простейшим алгоритмом, реализующим этот метод, является метод циклического изменения переменных, при котором каждый раз изменяется только одна переменная. В качестве направлений поиска выбираются координатные направления. Вдоль каждого из координатных направлений последовательно производится поиск минимума целевой функции с использованием выше описанных методов поиска минимума одномерной функции.

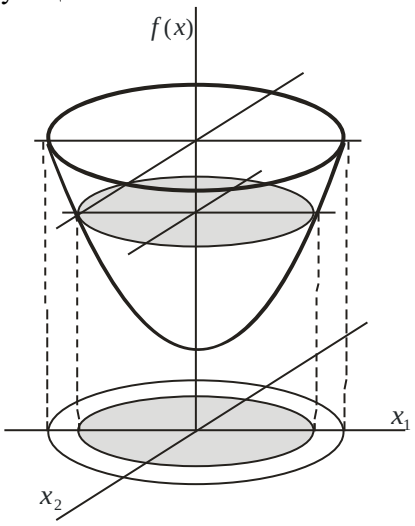


Рис.4.15. Двумерная целевая функция, проекция линий уровня на плоскость $\{x_1, x_2\}$.

На Рис.4.15 двумерная функция - параболоид, построенный в координатах $\{x_1, x_2, f(x)\}$. При более сложных целевых функциях линии уровня могут иметь весьма сложную конфигурацию. На Рис.4.16. представлено сечение двумерной функции. Параболоид рассечён вертикальной плоскостью $A - A' - B' - B$ параллельной оси x_2 (значение x_1 - фиксировано). На поверхности параболоида сечение представлено параболой $A - C - B$, проекцией которой на плоскость $\{x_1, x_2\}$ является прямая $A' - C' - B'$. Минимум параболы находится в точке C. На прямой $A' - C' - B'$ точке минимума соответствует точка C'.

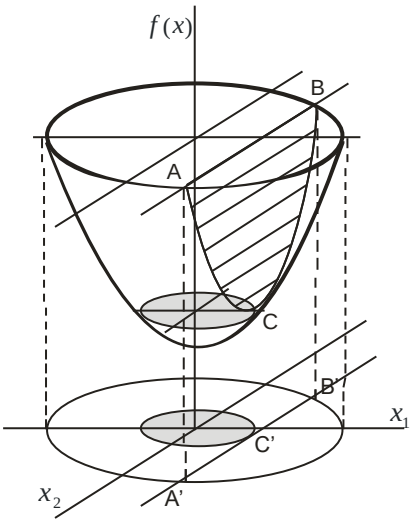


Рис.4.16. Сечение целевой функции.

На Рис.4.17. представлен алгоритм поиска минимума двумерной функции. В заданной области задаётся стартовая точка *A* и по линии «1-1» (параллельной координатной оси x_1

) при фиксированном значении x_2 определяется точка *B*, в которой значение целевой функции по этому направлению минимально. Затем из точки *B* по линии «2-2» (параллельной координатной оси x_2) при фиксированном значении x_1, найденном на предшествующем поиске, определяется точка *C*, в которой значение целевой функции по этому направлению минимально. Поскольку в данном примере размерность пространства равна 2, данная итерация завершается. Из точки *C* производится выполнение нового итерационного циклического перебора координатного поиска и так до тех пор, пока значения целевой функции, полученные на двух соседних итерациях, не будут отличаться друг от друга на заданную ранее малую величину.

Рис.4.17. Поиск минимума двумерной функции методом покоординатного спуска.

Подобный алгоритм хорошо работает, если целевая функция обладает свойством сферической симметрии (или достаточно близка к ней). При сложной конфигурации линий уровня конечный результат поиска может оказаться зависимым от порядка перебора координатных направлений. Однако, если линии уровня сильно искривлены и имеют сложную

конфигурацию, то алгоритм может превратиться в бесконечную последовательность итерации («зациклится»).

Рассмотрим по шагам работу алгоритма на примере двумерной функции (Рис. 4.17.).

Шаг1: Выбрать начальную точку A с координатами $(x_1^{(0)}, x_2^{(0)})$. Запомнить $x^{предш} = x^{(0)}$.

Шаг2: Организовать циклический просмотр переменных. Зафиксировать значение $x_2^f = x_2^{(0)}$, тогда целевая функция $f(x_1, x_2)$ будет функцией одной переменной $f(x_1)$ с параметром x_2^f. Найти минимум $f(x_1)$ по направлению 1-1 любым методом одномерной оптимизации (точка B) с координатами $(x_1^{(1)} = x_1^f, x_2^f)$. Зафиксировать значение x_1^f, тогда целевая функция $f(x_1, x_2)$ будет функцией одной переменной $f(x_2)$ с параметром x_1^f. Найти минимум $f(x_2)$ по направлению 2-2 любым методом одномерной оптимизации (точка C) с координатами $(x_1^f, x_2^2 = x_2^f)$. *[В случае многомерной функции эти процедуры повторяются для каждого координатного направления.]* Итерация заканчивается определением промежуточного минимума $\left(x_1^{(i)} = x_1^f, x_2^{(i)} = x_2^f\right)$.

Шаг3: Проверить условия достижения минимума – значения целевой функции $f(x_1, x_2)$, т.е. $\left\| x^f - x^{предш} \right\| < \delta$.

Если «да», то закончить поиск, иначе положить $x_1^{(0)} = x_1^f$, $x_2^{(0)} = x_2^f$, $x^{предш} = x^f$ и перейти к шагу 2.

Метод Хука-Дживса.

Этот метод является модификацией метода покоординатного спуска, компенсирующий некоторые его недостатки. Суть изменений заключается в том, что помимо покоординатного спуска периодически проводится спуск по направле-

нию $d = x^{(i+1)} - x^{(i)}$, которое определяется точками промежуточного определения минимума целевой функции на двух последовательных итерациях поиска. Следовательно, в отличие от метода циклического перебора координатных направлений, здесь для определения направления спуска используется информация, полученная на предшествующих итерациях. Метод Хука-Дживса является комбинацией *исследующего поиска, циклического изменения переменных* и *поиска по образцу* (Рис. 4.18).

Исследующий поиск заключается в последовательности следующих действий:

- Задаются величины шагов по всем координатным направлениям (они могут быть различными).
- Задаётся исходная точка.
- По одной из координат делается пробный шаг. Если значение целевой функции в пробной точке не превышает её значения в исходной точке, то шаг удачный. В противном случае из исходной точки делается шаг в обратном направлении.
- Перебор осуществляется по всем координатам. Полученная точка называется базовой точкой.

Поиск по образцу. В данном алгоритме (4.43) приняты следующие обозначения:

$x^{(i)}$ - текущая базовая точка,

$x^{(i-1)}$ - предыдущая базовая точка,

$x_p^{(i+1)}$ - точка, построенная при движении по образцу,

$x^{(i+1)}$ - следующая (новая) базовая точка.

- Производится единственный шаг вдоль прямой соединяющей две базовые точки - полученную и предыдущую. Новая базовая точка на этом направлении определяется как

$$x_p^{(i+1)} = x^{(i)} + \left(x^{(i)} - x^{(i-1)}\right) \qquad (4.43)$$

- Как только движение по образцу не приводит к уменьшению значения целевой функции, точка $x_p^{(i+1)}$ фиксируется в качестве временной базовой точки и относительно неё проводится исследующий поиск. Если в результате поиска определяется точка, в которой значение целевой функции меньше чем точке $x^{(i)}$, то она рассматривается как новая базовая точка $x^{(i+1)}$.
- Если исследующий поиск неудачен, то нужно вернуться в точку $x^{(i)}$ и провести исследующий поиск с целью определения нового направления минимизации.
- Если в конечном итоге (т.е. после просмотра всех возможных направлений) поиск не приводит к успеху, то нужно уменьшить величину шага и повторить исследующий поиск.
- Поиск завершён, когда величина шага становится достаточно малой.

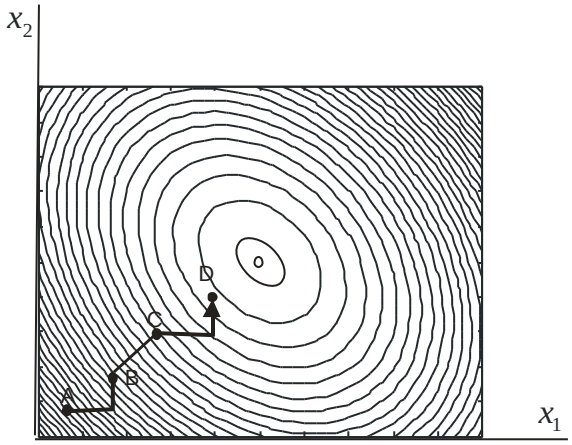


Рис.4.18. Поиск минимума целевой функции методом Хука-Дживса.

Алгоритм Хука-Дживса по шагам:
Шаг 1: Задать

начальную точку $x^{(0)}$,

приращения по координатам Δ_k, $k = 1,2,3, \ldots n$,

Коэффициент уменьшения шага $\alpha > 1$,

Параметр окончания поиска (малое значение) $\varepsilon > 0$.

Шаг 2: Провести исследующий поиск в текущей базовой точке, начиная с $x^{(0)}$.

Шаг 3: Проверить результаты исследующего поиска на успешный результат:

- Успешный – перейти к шагу 5.
- Неуспешный – перейти к шагу 4.

Шаг 4: Проверить условие окончания поиска – если величина шага $\|\Delta x\| < \varepsilon$, то прекратить поиск, последняя базовая точка является точкой минимума.

Иначе уменьшить приращение по координатам $\Delta_k = \Delta_k / \alpha$, $k = 1,2,3, \ldots n$ и перейти к шагу 2.

Шаг 5: Провести поиск по образцу $x_p^{(i+1)} = x^{(i)} + \left(x^{(i)} - x^{(i-1)}\right)$

Шаг 6: Провести исследующий поиск, используя $x_p^{(i+1)}$ в качестве базовой точки и считать, что найденная $x^{(i+1)}$ - следующая (новая) базовая точка.

Шаг 7: Проверить условие $f\left(x^{(i+1)}\right) < f\left(x^{(i)}\right)$. Если условие – «правда», то положить $\left(x^{(i-1)} = x^{(i)}\right)$, $\left(x^{(i)} = x^{(i+1)}\right)$ и перейти к шагу 5. Иначе перейти к шагу 4.

Алгоритм имеет недостатки:

- При сложной конфигурации линий уровня на этапе циклического движения по координатам возможно преждевременное прекращение поиска.
- При значительных нелинейных эффектах возможно вырождение – последовательность исследующих поисков не переходит в движение по образцу.

Метод имеет различные модификации связанные исследующим поиском, выбором направления и длины шага.

4.3.2. Метод сопряженных направлений.

Наиболее эффективными из методов прямого поиска являются метод сопряженных направлений, который, также как и методы других видов поиска, имеет различные реализуемые алгоритмы. Основным достоинством этого метода является то, что при работе алгоритма для построения направлений поиска используется информация, полученная на предшествующей итерации. Метод ориентирован на решение задач с квадратичными целевыми функциями.
В общем виде квадратичная функция запишем как

$$q(x) = a + b^T x + \frac{1}{2} x^T Q x \qquad (4.44)$$

Рассмотрим понятие сопряжённых направлений. Пусть Q - симметрическая матрица размерности $n \times n$.

Направления $d^{(1)}, d^{(2)}, d^{(3)}, \ldots . d^{(r)}$, $r \le n$ называются *сопряженными* (Q-*сопряженными*), если эти направления линейно независимые и

$$\left(d^{(i)}\right)^T Q d^{(i)} = 0 \text{ для всех } i \ne j. \qquad (4.45)$$

Если задана произвольная точка $x^{(1)}$, положение точек, находящихся на направлении d задается формулой

$$x = x^{(1)} + \lambda d \qquad (4.46)$$

Минимум функции $q(x)$ (4.44) вдоль направления d определяется нахождением такого значения λ^*, при котором $\partial q / \partial \lambda = 0$. Следовательно, полагая, что минимум функции находится в точке $y^{(1)}$, можно записать

$$\left[\left(y^{(1)}\right)^T Q + b^T\right] d = 0 \qquad (4.47)$$

Для другой произвольной точки $x^{(2)}$ минимум функции находится в точке $y^{(2)}$, можно записать

$$\left[\left(y^{(2)}\right)^T Q + b^T\right] d = 0 \qquad (4.48)$$

Вычитая (15.19) из (15.20) получим

$$\left(y^{(2)} - y^{(1)}\right)^T Q\, d = 0\,, \qquad (4.49)$$

Это означает, что направления d и $\left(y^{(2)} - y^{(1)}\right)$ являются Q-сопряженными. Данное построение отражает *свойство параллельного подпространства* для квадратичных функций. На рис.4.19. показаны сопряженные направления для двумерного случая. Видно, что для квадратичной функции можно построить систему сопряжённых направлений и найти точку минимума на основании трёх одномерных поисков.

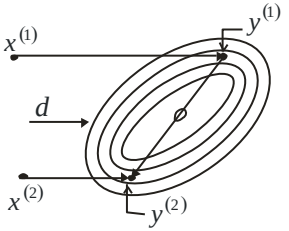


Рис. 4.19. Сопряжённые направления на плоскости.

В рассмотренном выше алгоритме для построения системы сопряжённых направлений использовались начальные точки и некоторое направление. Это может быть не всегда удобно, поэтому лучше строить сопряжённые направления исходя из одной начальной точки.

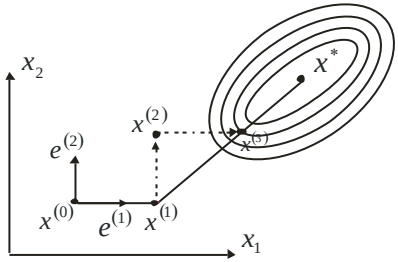


Рис 4.19. Поиск минимума методом сопряжённых направлений.

Процедура построения заключается в следующем (например, для двумерного случая).

Пусть алгоритм начинается из точки $x^{(0)}$ (Рис.4.20) и в дальнейшем построении используются единичные координатные векторы $e^{(1)}, e^{(2)}$ (в многомерном случае $e^{(1)}, e^{(2)}, e^{(3)}, ..., e^{(n)}$).

Для заданной начальной точки $x^{(0)}$ вычисляется значение $\lambda^{(0)}$, которому соответствует минимальное значение функции $f\left(x^{(0)} + \lambda^{(0)} e^{(1)}\right)$. Определим

$$x^{(1)} = x^{(0)} + \lambda^{(0)} e^{(1)} \qquad (4.50)$$

и вычислим $\lambda^{(1)}$, которому соответствует минимальное значение функции $f\left(x^{(1)} + \lambda^{(1)} e^{(2)}\right)$.

Определим

$$x^{(2)} = x^{(1)} + \lambda^{(1)} e^{(2)}. \qquad (4.51)$$

Далее вычислим $\lambda^{(2)}$, минимизирующее $f\left(x^{(2)} + \lambda^{(2)} e^{(1)}\right)$ и положим

$$x^{(3)} = x^{(2)} + \lambda^{(2)} e^{(1)}. \qquad (4.52)$$

Направления $\left(x^{(3)} - x^{(1)}\right)$ и $e^{(1)}$ оказываются сопряжёнными. Этот принцип можно применить и *n*-мерному пространству. В этом случае для нахождения минимума квадратичной функции требуется проведение n^2 одномерных поисков.

Метод сопряженных направлений Пауэла.

Шаг 1: Задать начальную точку $x^{(0)}$ и систему *n* линейно независимых направлений $d^{(i)}$; возможен вариант, когда эти направления совпадают с единичными координатными векторами $d^{(i)} = e^{(i)}, \quad i = 1,2,...,n$.

Шаг 2: Найти минимум функции $f(x)$ при последовательном движении по $(n+1)$ направлению. При первом поиске начальная, а при последующих проходах по-

лученная ранее точка минимума, берётся в качестве исходной точки. Направление $d^{(n)}$ используется как на первом, так и на последнем поиске.

Шаг 3: Определить новое сопряжённое направление с помощью обобщённого свойства параллельного подпространства (см. выше).

Шаг 4: Заменить направление $d^{(1)}$ на $d^{(2)}$ и т.д. Заменить $d^{(n)}$ полученным сопряжённым направлением. Перейти к шагу 2.

Для квадратичной функции алгоритм приводит к нахождению точки минимума за *n* циклов. Если целевая функция не квадратичная, то для нахождения минимума требуется более чем *n* циклов (после прохождения *n* циклов требуется *обновление* метода). Алгоритм чувствителен к ошибкам счёта, так как они могут привести к нарушению свойств ортогональности.

4.3.3. Градиентные методы.

Рассмотренные прямые методы поиска для получения решения требуют очень большое количество вычислений целевой функции. Кроме того, в них никак не используются свойства стационарных точек (т.е. точек удовлетворяющих необходимому условию первого порядка – 4.31). Эти два обстоятельства привели к необходимости рассмотрения методов поиска, связанных с градиентом целевой функции. Предполагается, что компоненты градиента могут быть вычислены либо аналитически, либо при помощи численных методов с высокой степенью точности. Поскольку антиградиент функции указывает направление, по которому функция убывает наиболее быстро, то естественно спуск нужно проводить в этом направлении. Все градиентные методы носят итеративный характер, и связь между итерациями определяется соотношением

$$x^{(i+1)} = x^{(i)} + \alpha^{(i)} d\left(x^{(i)}\right) \qquad (4.53)$$

где $x^{(i)}$ - текущее приближение (итерация) к решению x^*, $\alpha^{(i)}$ - параметр, определяющий длину шага в направлении поиска $d^{(i)} = d\left(x^{(i)}\right)$ в *n*-мерном пространстве переменных x_j, $j = 1,2,\ldots,n$. Необходимо решить вопрос о выборе $\alpha^{(i)}$ и направления $d\left(x^{(i)}\right)$. Выбор $\alpha^{(i)}$ определяется методами одномерной оптимизации при минимизации функции $f(x)$ в выбранном направлении. Направление $d\left(x^{(i)}\right)$ можно определить различными методами. Оно непосредственно связано с градиентом целевой функции. И одной из разновидностей метода выбора направления $d\left(x^{(i)}\right)$являются м*етоды наискорейшего спуска.*

Метод Коши.

Пусть в некоторой произвольной точке $\bar{x}$ многомерного пространства нужно определить направление локального наискорейшего спуска, т.е. направление по которому в окрестности точки $\bar{x}$ происходит наискорейшее уменьшение целевой функции. Процедура Коши основана на линейной аппроксимации. Разлагая функцию в ряд Тейлора и ограничиваясь первыми двумя членами ряда, можно записать

$$f(x) = f(\bar{x}) + \nabla f(\bar{x})^T \Delta x. \qquad (4.54)$$

Локальное уменьшение функции определяется вторым членом, так как значение первого члена фиксировано. Следовательно, наибольшее уменьшение целевой функции происходит в таком направлении (4.53), которому соответствует наибольшая отрицательная величина второго члена уравнения (4.54). Из свойств скалярного умножения следует, что выбор нужного направления произойдёт при условии

$$d(\bar{x}) = -\nabla f(\bar{x}) \qquad (4.55)$$

и второй член примет вид

$$-\alpha \nabla f(\bar{x})^T \nabla f(\bar{x}) \qquad (4.56)$$

Поэтому в основе простейшего градиентного метода лежит формула

$$x^{(i+1)} = x^{(i)} - \alpha^{(i)} \nabla f\left(x^{(i)}\right), \qquad (4.57)$$

где $\alpha^{(i)}$ - заданный положительный параметр вычисленный вдоль направления $\nabla f\left(x^{(i)}\right)$ методом одномерного поиска

Метод Коши требует вычисления целевой функции и её первых производных на каждой итерации. Следует отметить, что он хорошо работает, когда линии уровня являются окружностями или близки к ним и направление антиградиента, вообще говоря, не может служить глобальным направлением поиска минимума. На Рис.4.21. представлен алгоритм Коши для двумерного случая. В точке $x^{(0)}$ вычисляется градиент целевой функции, и по направлению антиградиента методами одномерной оптимизации определяется минимум– точка $x^{(1)}$. Далее процедура повторяется и определяется глобальный минимум в точке $x^{(2)}$. Если бы линии уровня были окружностями, то уже на первой итерации можно было определить положение глобального минимума двумерной функции, так как линия антиградиента проходила бы через точку $x^{(2)}$.

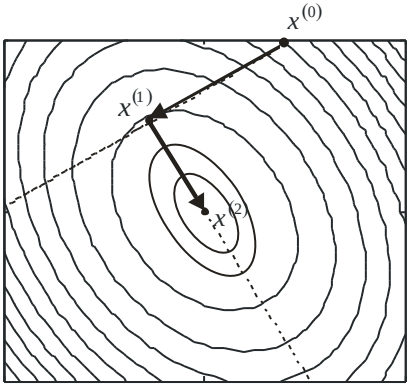


Рис. 4.21. Итерации поиска по методу Коши.

Поскольку определение минимума по направлению производится численными методами с определённой погрешностью, то для достижения глобального минимума с нужной степе-

нью точности возможно повторение итерационных циклов поиска.

Метод имеет ряд достоинств и недостатков. К недостаткам следует отнести необходимость определять $\alpha^{(i)}$ на каждой итерации, а также медленную сходимость к точке минимума из-за малых значений градиента в окрестности минимума. Достоинствами метода являются его устойчивость и хорошее продвижение из точек расположенных далеко от минимума.

Рассмотрим принципиальный алгоритм, который завершает работу при выполнении одного из условий:

- либо предельные значения модуля градиента меньше ε_1,
- либо предельные значения модуля вектора смещения Δx, между соседними итерациями - ε_1,
- либо число итераций превышает предельное значение m.

Шаг 1: Задать начальную точку $x^{(0)}$, задать размерность пространства поиска n, задать предельные значения для модуля градиента и модуля вектора смещения Δx между соседними итерациями - ε_1, задать точность определения минимума одномерного поиска - ε_2,
задать предельное число итераций m.
Положить $i = 0$.

Шаг 2: Вычислить градиент $\nabla f\left(x^{(i)}\right)$

Шаг 3: Проверить условие $\left\|\nabla f\left(x^{(i)}\right)\right\| \le \varepsilon_1$. Если «да», то перейти к шагу 9, иначе перейти к шагу 4.

Шаг 4: Проверить условие $i \ge m$. Если «да», то перейти к шагу 9, иначе перейти к шагу 5.

Шаг 5: Поиск минимума по направлению антиградиента одним из методов одномерной оптимизации (использовать значение ε_2).

Шаг 6: Положить $x^{(i+1)} = x^{(i)} - \alpha^{(i)} \nabla f\left(x^{(i)}\right)$.

Шаг 7: Проверить условие $\dfrac{\left\|x^{(i+1)}-x^{(i)}\right\|}{\left\|x^{(i)}\right\|}\le\varepsilon_1$. Если «да», то перейти к шагу 9, иначе перейти к шагу 8.

Шаг 8: Положить $i=i+1$ и перейти к шагу 2.

Шаг 9: Останов.

Метод Ньютона.

В отличие от метода Коши метод Ньютона основан на квадратичной аппроксимации целевой функции (разложение в ряд Тейлора до членов второго порядка). В алгоритме используются вторые производные целевой функции.

$$\bar{f}\left(x;x^{(i)}\right)=f\left(x^{(i)}\right)+\nabla f\left(x^{(i)}\right)^T\Delta x+\frac{1}{2}\Delta x^T\nabla^2 f\left(x^{(i)}\right)\Delta x. \quad (4.58)$$

$\bar{f}\left(x;x^{(i)}\right)$ - аппроксимирующая функция переменной x, построенной в точке $x^{(i)}$.

Процесс поиска минимума – итерационный процесс, на каждой итерации которого во вновь получаемой точке $x^{(i+1)}$ градиент аппроксимирующей функции $f(x)$ обращался в нуль. Тогда

$$\nabla\bar{f}\left(x;x^{(i)}\right)=\nabla f\left(x^{(i)}\right)+\nabla^2 f\left(x^{(i)}\right)\Delta x=0 \quad (4.59)$$

и, следовательно,

$$\Delta x=-\nabla^2 f\left(x^{(i)}\right)^{-1}\nabla f\left(x^{(i)}\right). \quad (4.60)$$

Последовательность построения точек минимума на текущей итерации определяется соотношением

$$x^{(i+1)}=x^{(i)}-\nabla^2 f\left(x^{(i)}\right)^{-1}\nabla\left(x^{(i)}\right) \quad (4.61)$$

Последовательность шагов в алгоритме Ньютона такая же, как в алгоритме Коши, различается только процесс построения точек $x^{(i+1)}$ - сравним выражения (4.57) и (4.61).

4.3.4. Методы сопряженных градиентов.

Эти методы часто используются даже, когда целевая функция не является выпуклой. В этом случае определяется

локальный минимум. Направление вектора градиента функции, определённого в точке x n-мерного пространства указывает направление возрастания этой функции. Следовательно, для достижения минимума целевой функции необходимо организовать движение в направлении обратном градиенту функции.

Пусть функция $f(x)$ строго выпукла, дважды непрерывно дифференцируема и x^* - оптимальное решение. Тогда $\nabla f(x^*) = 0$. Обозначим вектор приращения аргумента как h. Уравнение (4.28) можно записать в виде

$$f(x) - f(x^*) = \frac{1}{2}\langle x - x^*, H(x^*)(x - x^*)\rangle = \\ = -\langle H(x^*)x^*, x\rangle + \frac{1}{2}\langle x, H(x^*)x\rangle + \frac{1}{2}\langle x^*, H(x^*)x^*\rangle \quad (4.62)$$

где $H(x^*)$- симметрическая матрица размерности $n \times n$ вторых производных целевой функции $f(x)$ вычисленных в точке x^*.

Характерной чертой этих методов является положение, что минимум выпуклой функции может быть достигнут не более чем за n итераций. Это утверждение касается принципиального алгоритма, в реализуемом алгоритме для достижения минимума с заданной точностью возможно повторение циклов поиска. В конкретных разновидностях метода сопряженных градиентов это связано с погрешностями вычисления градиента, определения минимума по направлению и точности вычислений.

Алгоритм метода сопряженных градиентов (прототип)

Шаг 1: Выбрать точку $x_0 \in R^n$ и положить $i = 0$.

Шаг 2: Вычислить градиент $\nabla f(x_i)$

Шаг 3: Если $\nabla f(x_i) = 0$, то поиск закончен. Иначе вычислить h_i и перейти к шагу 4.

Шаг 4: Вычислить такое значение $\lambda_i > 0$, что

$$f(x_i + \lambda_i h_i) = min\{f(x_i + \lambda_i h_i)\}$$

Шаг 5: Положить $x_{i+1} = x_i + \lambda_i h_i$, положить $i = i + 1$ и перейти к шагу 2.

На шаге 3 производится поиск минимума по направлению (одномерный поиск). Поскольку он осуществляется численным алгоритмом то, следовательно, значение минимума вычисляется с некоторой погрешностью. Исходя из этого, приведённый алгоритм можно считать принципиальным алгоритмом, а реализуемый алгоритм требует повторения поиска, который начинается из точки предшествующего минимума.

Различные варианты метода сопряженных градиентов отличаются только способом вычисления векторов h_i, по которым производится поиск минимума функции.

Рассмотрим процесс биортогонализации. Пусть дана симметрическая положительно определённая матрица размера $n \times n$. Построим две последовательности векторов $g_0, g_1, \ldots$ и $h_0, h_1, \ldots$ в R^n, таких что

$$\langle g_i, g_j \rangle = 0 \quad \text{для всех} \quad i \neq j$$

и

$$\langle h_i, Hh_j \rangle = 0 \quad \text{для всех} \quad i \neq j$$

Эти последовательности можно построить с помощью метода ортогонализации Грамма-Шмидта (приём «шнуровки»).

Пусть $g_0 \in R^n$ - произвольный вектор и $h_0 = g_0$. Далее последовательности векторов строятся следующим образом

$$g_1 = g_0 - \lambda_0 Hh_0, \qquad \lambda_0 = \frac{\langle g_0, g_0 \rangle}{\langle g_0, Hh_0 \rangle},$$

что даёт $\langle g_0, g_1 \rangle = 0$,

$$h_1 = g_1 + \gamma_0 h_0, \qquad \gamma_0 = -\frac{\langle Hh_0, g_1 \rangle}{\langle Hh_0, h_1 \rangle},$$

что даёт $\langle h_0, Hh_1 \rangle = 0$.

Последующее построение

$$g_2 = \alpha g_0 + g_1 - \lambda_1 Hh_1, \quad h_2 = g_2 + \gamma_1 h_1 + \beta h_0.$$

Параметры λ_1 и α выбираются таким образом, чтобы

$$\langle g_0, g_1 \rangle = \langle g_1, g_2 \rangle = 0,$$

а γ_1 и β выбираются таким образом, чтобы

$$\langle h_0, Hh_2 \rangle = \langle h_1, Hh_2 \rangle = 0.$$

Построение продолжается до некоторого индекса $m \le n$, при котором $g_m = h_m = 0$.

Используя этот алгоритм построения двух последовательностей векторов

$$g_{i+1} = g_i - \lambda_i Hh_i, \qquad h_{i+1} = g_{i+1} + \gamma_i h_i,$$

где $h_0 = g_0$, а λ_i и γ_i выбраны таким образом, что

$\langle g_{i+1}, g_j \rangle = 0$ и $\langle h_{i+1}, Hh_j \rangle = 0$, т.е.

$$\lambda_i = \frac{\langle g_i, g_i \rangle}{\langle g_i, Hh_i \rangle}, \quad \gamma_i = -\frac{\langle Hh_i, g_{i+1} \rangle}{\langle Hh_i, h_i \rangle} \qquad (4.63)$$

если знаменатели отличны от 0 и $\lambda_i = 0$, $\gamma_i = 0$ в противном случае. Можно показать, что все коэффициенты типа α, β и т.д. равны нулю. Векторы $g_0, g_1, \ldots g_m$ попарно ортогональны, а так как все они отличны от нуля, их общее число не может превосходить n, т.е. $m \le n$.

Алгоритм Флетчера-Ривса

Шаг 1: Выбрать точку $x_0 \in R^n$. Если $\nabla f(x_0) = 0$, то остановиться.

Шаг 2: Положить $i = 0$, $g_0 = h_0 = -\nabla f(x_0)$

Шаг 3: Вычислить такое значение $\lambda_i > 0$, что

$$f(x_i + \lambda_i h_i) = min\{f(x_i + \lambda_i h_i)\}, \ \lambda \ge 0$$

Шаг 4: Положить $x_{i+1} = x_i + \lambda_i h_i$.

Шаг 5: Вычислить $\nabla f(x_{i+1})$

Шаг 6: $\nabla f(x_{i+1}) = 0$, то остановиться. Иначе положить

$$g_{i+1} = -\nabla f(x_{i+1}), \quad h_{i+1} = g_{i+1} + \gamma_i h_i,$$

где $\gamma_i = \dfrac{\langle g_{i+1}, g_{i+1} \rangle}{\langle g_i, g_i \rangle}$,

положить $i = i + 1$ и перейти к шагу 3.

На шаге 2 вычисляется градиент целевой функции в стартовой точке. На шагах 3 и 4 определяется локальный минимум по выбранному направлению. Далее проверяются условия окончания поиска. Этот метод более эффективен, чем методы, в которых направление поиска задаётся заранее (например, метод наискорейшего спуска). Но сам алгоритм поиска является более сложным и требует разработки более сложных программ. В данном методе реализуется свойство градиентных методов, позволяющее работать с целевой функцией имеющей разрывы производной (Рис.4.22).

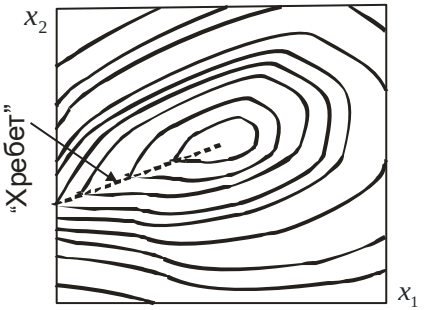


Рис. 4.22. «Хребет» - линия разрыва производной.

Алгоритм Дэвидона-Флетчера-Пауэла *(с переменной метрикой).*

Метод является наиболее широко используемым градиентным методом, обладающий хорошей устойчивостью. Относительным недостатком является необходимость хране-

ния матрицы H_i, которая в задачах с большим количеством переменных может иметь большие размеры.

Шаг 1: Выбрать точку $x_0 \in R^n$. Если $\nabla f(x_0) = 0$, то остановиться; иначе перейти к шагу 2.

Шаг 2: Положить $i = 0$, положить $H_0 = I$ (I - единичная матрица размерности $n \times n$, но можно взять любую симметрическую положительно определённую матрицу), положить $g_0 = \nabla f(x_0)$

Шаг 3: Положить $h_i = -H_i g_i$

Примечание: Матрица H_i называется *метрикой*, а метод - методом с *переменной метрикой*, поскольку матрица H_i изменяется на каждой итерации.

Шаг 4: Вычислить такое значение $\lambda_i > 0$, что

$$f(x_i + \lambda_i h_i) = min\{f(x_i + \lambda_i h_i)\}, \quad \lambda \geq 0$$

Шаг 5: Вычислить $\nabla f(x_i + \lambda_i h_i)$

Шаг 6: Если $\nabla f(x_i + \lambda_i h_i) = 0$, то остановиться, иначе положить $x_{i+1} = x_i + \lambda_i h_i$,

$g_{i+1} = \nabla f(x_{i+1})$,

$\Delta g_i = g_{i+1} - g_i$,

$\Delta x_i = x_{i+1} = x_i$.

Вычислить

$$H_{i+1} = H_i - \frac{1}{\langle \Delta g_i, H_i \Delta g_i \rangle} \Delta x_i \rangle\langle H_i \Delta g_i + \\ + \frac{1}{\langle \Delta x_i, \Delta g_i \rangle} \Delta x_i \rangle\langle \Delta x_i ,$$

и перейти к шагу 3.

[Матрица H_{i+1}, построенная таким образом, является симметрической положительно определённой, т.е. соответствует построению алгоритма поиска минимума функции методом сопряжённых градиентов.]

Шаг 7: Положить $i = i + 1$ и перейти к шагу 3.

Примечание: Запись $\cdot\rangle\langle\cdot$ - означает *диаду*. Если из двух векторов a и b составить, например, матрицу 3-го порядка, у которой (i, j)-й элемент матрицы равен $a_i b_j$.

$$\begin{bmatrix} a_1b_1 & a_1b_2 & a_1b_3 \\ a_2b_1 & a_2b_2 & a_2b_3 \\ a_3b_1 & a_3b_2 & a_3b_3 \end{bmatrix}$$

Такая матрица является диадой.

4.3.5. Методы случайного поиска (стохастическая аппроксимация)

Оригинальный подход, позволяющий обойти трудности применения детерминированных методов в случае многомерного пространства, предложен Бруксом и основан на случайном поиске. Пусть пространство проектирования представляет собой куб или гиперкуб со стороной, равной единице, и разделено на кубические ячейки путем деления на 10 равных частей каждой стороны куба, соответствующей одному из проектных параметров. При размерности пространства поиска N=2 число ячеек равно 100, при N=3 - оно равно 1000. В общем случае при N-мерном пространстве число ячеек равно 10^N. Вероятность того, что, выбранная наугад, ячейка войдет в число 10% наиболее перспективных ячеек равна 0,1. Так как при N=1 нас будет интересовать одна ячейка из 10, при N=2-одна из десяти лучших при общем количестве ячеек 100 и т. д. Вероятность того, что мы пропустим одну из 10% наиболее перспективных ячеек, составит 0,9. Если случайным образом выбрать две ячейки, то вероятность пропуска будет 0,81. Вообще вероятность нахождения,

по крайней мере, одной ячейки из наиболее перспективных, доля которых равна f, после N попыток составит

$$P = 1 - (1 - f)^N$$

В представленной ниже таблицей указано, сколько ячеек надо выбрать случайным образом, чтобы обеспечить заданную вероятность при заданной доле наиболее перспективных ячеек. Из нее видно, что при случайной выборке 44 ячеек вероятность достижения f=0,1 составит 99%. Это очень неплохо, если вспомнить, что для 100%-ного обеспечения целевую функцию в случае пяти переменных пришлось бы вычислить 2 476 099 раз.

Таблица 4.5

f	Вероятность			
	0,80	0,90	0,95	0,99
0,1	16	22	29	44
0,05	32	25	59	90
0,01	161	230	299	459
0,005	322	460	598	919

Метод случайного поиска имеет два преимущества. Во-первых, он пригоден для любой целевой функции независимо от того, является она унимодальной или нет. Во-вторых, вероятность успеха при попытках не зависит от размерности рассматриваемого пространства. Хотя этот метод не позволяет непосредственно найти оптимальное решение, он создает подходящие предпосылки для применения в дальнейшем других методов поиска. Поэтому его часто применяют в сочетании с одним или несколькими методами других типов.

4.4. Поиск глобального минимума целевой функции

4.4.1. Поиск глобального минимума функции одной переменной.

Метод ломаных.

Метод [1] применим к функциям, удовлетворяющим *условию Липшица.* Функция *f(x)* заданная на отрезке [*a, b*]

удовлетворяет условию Липшица, если существует такая постоянная $L>0$, что

$$\left| f(x_1) - f(x_{2)}) \right| \le L \left| x_1 - x_2 \right| \qquad \forall\, x_1, x_2 \in [a,b] \qquad (4.63)$$

где L – *постоянная Липшица*.

Это условие геометрически отражает положение, что на любом участке рассматриваемой функции угол наклона хорды, соединяющей точки $(f(x_1), x_1)$ и $(f(x_2), x_2)$ не превышает значения постоянной L.

Процесс построения ломаной функции (Рис.4.23.) начинается с выбора произвольной начальной точки x_0 и вычисления функции $f(x_0)$ - точка C_0. Из точки C_0 построим два отрезка C_0 - A_0 и C_0 - B_0 с угловыми наклонами L и $-L$ соответственно. Ломаная линия A_0 - C_0 - B_0 - первое приближение целевой функции. Следующая точка x_1 выбирается из условия наименьшего значения ординаты точки m_0 - пересечение линий C_0 - B_0 и C_1 - A_1. Совершенно очевидно, минимальное значение ордината точки m_0 примет лишь при условии, когда абсцисса точки C_1 совпадает с правой границей интервала [a, b]. (При другом выборе начальной точки x_0 , точка x_1 будет определяться левой границей интервала [a, b]).

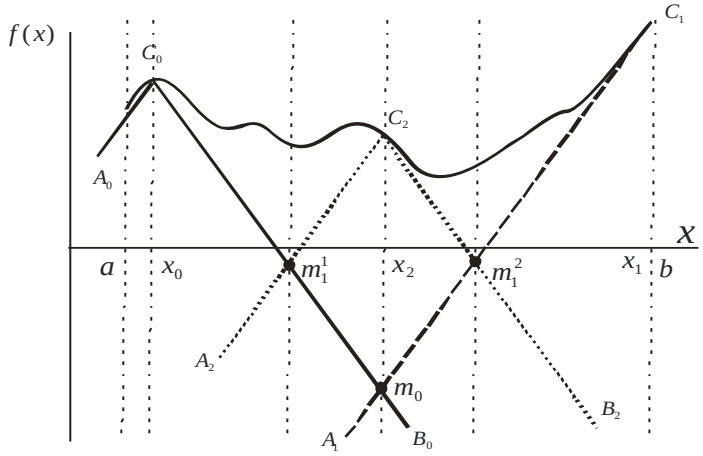


Рис.4.23. Построение ломаной функции из условия Липшица. Сплошная кривая – целевая функция.

Ломаная линия $A_0 - C_0 - m_0 - C_1$ является вторым приближением целевой функции. Абсцисса точки m_0 имеет значение x_2. Определим точку C_2, принадлежащую целевой функции, и из этой точки построим два отрезка $C_2 - A_2$ и $C_2 - B_2$ с угловыми наклонами L и $-L$ соответственно. Их пересечение с $A_0 - C_0 - m_0 - C_1$ определит следующие точки m_1^1 и m_1^2. Далее процесс построения продолжается подобным образом. В результате получаем приближение целевой функции снизу в виде ломанной линии (Рис.4.24.). Последовательность нижних точек результирующей ломаной функции $m_0, m_1, m_2 \ldots m_n$ (индексация переопределена с левого края интервала) позволяет выделить подынтервал, содержащий глобальный минимум. Достоинством этого метода является то, что он сходится при любом выборе начальной точки. Недостатком метода является необходимость заранее знать значение предельного угла наклона хорды L.

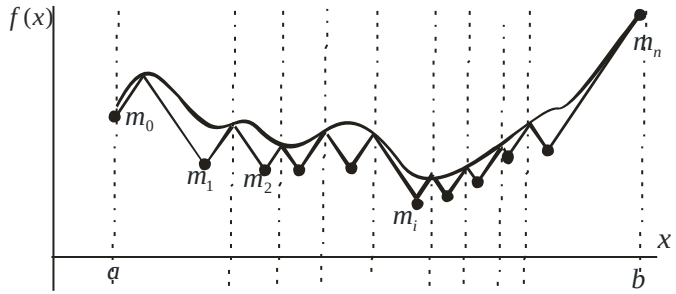


Рис.4.24. Результирующая ломаная функция.

Метод покрытий.

Существует целый ряд алгоритмов, в которых определённым образом строится последовательность отрезков $[a_i, b_i]$ полностью покрывающих интервал поиска $[a_{нач}, b_{нач}]$.

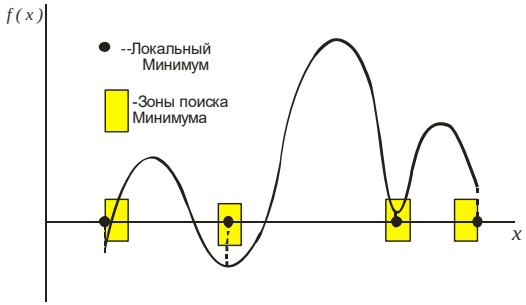


Рис.4.25. Определение областей локальных минимумов.

В точках a_i и b_i вычисляются значения целевой функции. Эти методы называются методами покрытия [1]. Самый простой алгоритм – *метод равномерного перебора точек* $x_i = a_{нач} + i * d$, где $d = x_{i+1} - x_i = const$.

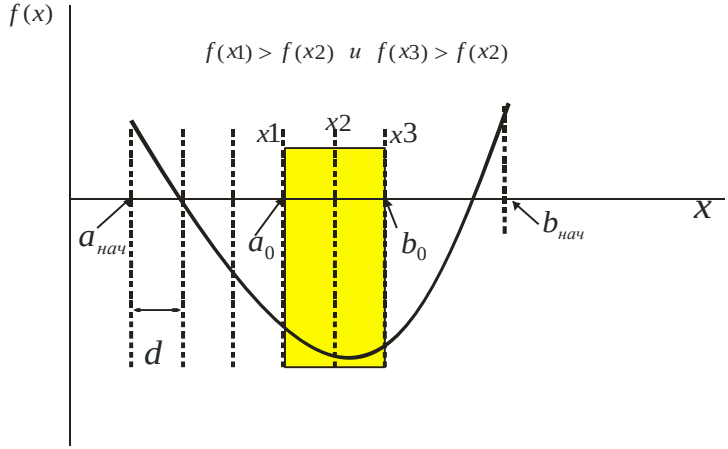


Рис.4.26. Определение границ интервала [*a*,*b*] содержащего локальный минимум, d – шаг сканирования.

Поиск глобального минимума целевой функции, удовлетворяющей условию Липшица (4.63), осуществляется методом сканирования интервала $[a_{нач}, b_{нач}]$ с достаточно большим шагом d по переменной *x* (Рис.4.25.). Выделяются области $[a_0, b_0]$, в которых содержатся минимумы функции.

Условия присутствия минимума функции на этом подынтервале определяются как $f(x1) > f(x2)$ и $f(x3) > f(x2)$. Каждая из этих областей исследуется на собственный локальный минимум (Рис.4.26.). Среди локальных минимумов определяется глобальный минимум.

4.4.2. Поиск глобального минимума функции многих переменных.

Целевая функция, в общем случае, может иметь сложную конфигурацию поверхностей (линий) уровня, которую при отсутствии достаточно полной априорной информации, даже трудно себе представить. Поэтому в решении реальных задач приходится проводить численный эксперимент для определения возможных локальных минимумов. Например, функция Химмельблау имеет четыре минимума (Рис. 4.27.).

*f(x,y)=(x*x+y-11)*(x*x+y-11)+(x+y*y-7)*(x+y*y-7),*

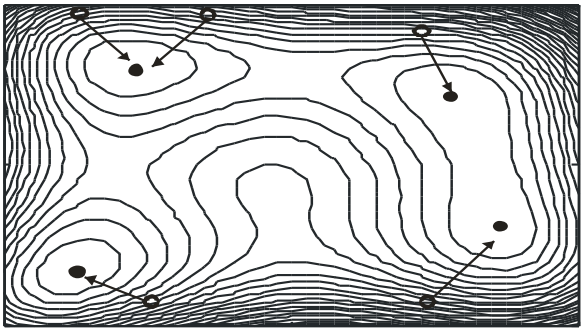


Рис. 4.27. Функция Химмельблау.

При наличии многих экстремумов нужно построить циклический алгоритм с разными стартовыми точками. В результате алгоритм поиска (в зависимости от координат стартовой точки) будет тяготеть к различным минимумам. Если в результате достаточного числа поисков новых минимумов не

обнаружится, то можно с большой вероятностью считать, что все локальные минимумы найдены. Определить глобальный минимум из ограниченного числа локальных минимумов не составит никакого труда.

При использовании алгоритмов, в которых производится вычисление производных, получить их аналитические выражения, часто, оказывается достаточно сложно. Применяемые конечно-разностные схемы вычисления производных должны быть хорошо апробированы и адаптированы к решаемой задаче. Сложности могут возникнуть при движении в направлении антиградиента в точках близких к минимуму, так как в этой области градиент целевой функции близок к нулю. Процедуры аппроксимации целевой функции при поиске одномерного минимума по выбранному направлению также вносят погрешности в принципиальный алгоритм поиска глобального минимума. Точность определения одномерного минимума определяется точностью используемых численных методов. В результате, глобальный минимум оказывается недостигнутым и требуется восстановление алгоритма, т.е. повторение поиска, для которого стартовой точкой является результат предшествующей итерации.

Задачи нелинейного программирования во многом присущи проблемам, возникающим при создании различных физических устройств или установок, а также исследовании сложных физических процессов. Целевая функция формируется на основе энергетических и динамических характеристик рассматриваемой системы. По своей форме целевая функция может иметь, например, следующий вид:

- линейная модель,

$$f(x) = \sum_{i=1}^{n} a_i \left(x_1 - x_i^* \right) \qquad (4.64)$$

- квадратичные модели

$$f(x) = \sum_{i=1}^{n} a_i \left(x_i - x_i^* \right)^2, \quad f(x) = F\left(\sum_{i=1}^{n} a_i \left(x_i - x_i^* \right) \right)^2, \qquad (4.65)$$

$$f(x)=\sum_{i=1}^{n} a_i\left(\varphi_i(x)-\varphi_i^*\right)^2,\ f(x)=\sum_{i=1}^{n} a_i\left(\frac{\left(\varphi_i(x)-\varphi_i^*\right)}{\varphi_i^{норм}}\right)^2,$$

где $x=(x_1,x_2,\ldots,x_n)$ - варьируемые параметры системы, a_i - весовые коэффициенты, x^* - параметр цели, $\varphi_i(x)$ - функция от варьируемых параметров системы, φ^* - функция параметр цели, $\varphi_i^{норм}$ - нормирующий коэффициент.

Используя весовые и нормирующие коэффициенты, можно произвести необходимое масштабирование переменных и целевой функции, а также выделить приоритетные параметры оптимизации.

В реальных физических задачах возможны разрывы целевой функции, которые заранее невозможно предугадать. Например, при расчёте динамики заряженных частиц в группирователе линейного ускорителя электронов может реализоваться такое распределение ускоряющего поля, при котором частицы не захватываются в режим ускорения, и, следовательно, нельзя сформировать целевую функцию. Решение подобных задач являются особой областью методов оптимизации [12].

4.5. Методы оптимизации в задачах с ограничениями.

Задача формулируется как нахождение минимума целевой функции $f(x)$ определённой на множестве X

$$min\{f(x):\ x\in X\} \qquad (4.66)$$

при наличии ограничений

$$r_i(x)=0 \ \text{ и } \ g_i(x)\le 0, \qquad (4.67)$$

где ($r_i(x)$, $g_i(x)$ -функции ограничения, определённые на множестве X .

Задачи условной оптимизации, как уже отмечалось выше, решаются сведением их к задачам безусловной оптимизации. Эта операция выполняется с учетом прямых

и функциональных ограничений. Устранение прямых ограничений при переходе к безусловной оптимизации осуществляется соответствующим нормированием управляемых параметров. При прямом ограничении $a < u_i < b$ нормирование можно выполнить, например, следующим образом

$$x_i = tg\left[\pi \frac{u_i - (b+a)/2}{b-a}\right]$$

преобразовав ограниченный параметр u_i в неограниченный x_i. Функциональные ограничения устраняются путем конструирования обобщенной функции оптимизации с учетом типа ограничений.

Для решения подобных задач используются два различных по своей сути метода – *метод внешних штрафных функций* и *метод внутренних (барьерных) штрафных функций*.

Задачу нелинейного программирования с ограничениями можно записать в компактном виде как

$$min\left\{f(x): x \in C \subset R^n\right\} \qquad (4.68)$$

где функция $f(x)$ непрерывно дифференцируема, C - замкнутое множество в пространстве R^n.

Основная идея методов штрафных функций состоит в том, что от решения задачи (4.68) переходим к решению задачи безусловной оптимизации

$$min\left\{f(x) + \sum_{i=0}^{n} p_i(x): x \in R^n\right\}, \quad i = 0,1,\ldots,n, \quad (4.69)$$

где $p_i(x)$ - штрафные функции.

В методе внешних штрафных функций штрафы $p_i(x)$ выбираются так, что точки вне C по мере удаления от границы становятся всё более «невыгодными», при этом внутри C выполняются условия либо $p_i(x) = 0$, либо $p_i(x) \to 0$. На Рис. 4.28. приведены семейства внешних штрафных функций.

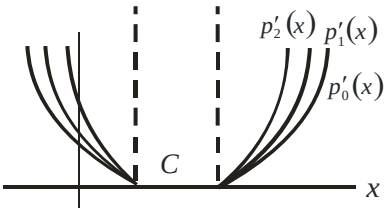


Рис.4.28. Внешние штрафные функции.

Последовательность непрерывных функций $p_i'(x)$ определённая на замкнутом множестве C называется последовательность внешних штрафных функций, если

$$\begin{aligned} &p_i'(x)=0 \quad \text{для всех} \quad x\in C, \quad i=0,1,\dots, \\ &p_i'(x)>0 \quad \text{для всех} \quad x\notin C, \quad i=0,1,\dots, \\ &p_{i+1}'(x)>p_i'(x) \quad \text{для всех} \quad x\notin C, \quad i=0,1,\dots, \\ &p_i'(x)\to\infty \quad \text{для всех} \quad x\in C, \quad i\to\infty \end{aligned} \qquad (4.70)$$

В методе внутренних штрафных функций штрафы $p_i(x)$ выбираются так, чтобы точки оптимальные для задачи (4.60) принадлежали внутренности C^0 множества C и $f(x)+p_i(x)\to f(x)$ для всех x из C^0, а по мере приближения к изнутри границе множества C $f(x)+p_i(x)\to\infty$.

Последовательность непрерывных функций $p_i''(x)$ определённая на замкнутом множестве C называется последовательность внутренних штрафных функций, если

$$0<p_{i+1}''(x)<p_i''(x) \quad \text{для всех} \quad x\in C, \quad i=0,1,\dots,$$

$$p_i''(x)\to 0 \quad \text{для всех} \quad x\in C, \quad i\to\infty$$

$$p_i''(x)\to\infty \quad i=0,1,\dots, \quad \text{для всех} \quad x \text{ стремящихся к границам множества } C$$

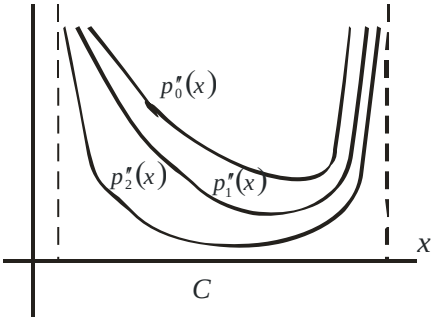


Рис. 4.29. Внутренние штрафные функции.

Исходя из этих положений, в процессе поиска происходит как бы «отталкивание» внутрь от границ области поиска. Стартовая точка поиска должна выбираться из множества C^0.

В реальных алгоритмах используются и смешенные технологии внешних и внутренних штрафных функций.

Использование методов штрафных функций позволяет преобразовать задачу условной оптимизации в задачу безусловной оптимизации. При этом предполагается, что эта новая задача решаема методами безусловной оптимизации. Кроме методов штрафных функций существуют и другие методы решения задач условной оптимизации – методы центров, методы возможных направлений, методы проекций градиента и прямые методы, которые также будут рассмотрены в дальнейшем.