№ 2335

А.П. Смирнов

Методы оптимизации

Алгоритмические основы задач оптимизации

Курс лекций

№ 2335 МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ «МИСиС»

Кафедра автоматизированных систем управления

А.П. Смирнов

Методы оптимизации

Алгоритмические основы задач оптимизации

Курс лекций

Рекомендовано редакционно-издательским
советом университета

МИСиС
ИЗДАТЕЛЬСКИЙ ДОМ

Москва 2014

УДК 004
С50

Рецензент
канд. экон. наук, доц. *И.П. Ильичев*

Смирнов А.П.

С50 Методы оптимизации : алгоритмические основы задач оптимизации : курс лекций / А.П. Смирнов. – М. : Изд. Дом МИСиС, 2014. – 135 с.

ISBN 978-5-87623-781-1

В курсе лекций рассматриваются классические методы оптимизации. Особое внимание уделено построению алгоритмов поиска экстремума, что позволит студентам самостоятельно разрабатывать соответствующие программные средства в случаях, когда использование стандартных пакетов программ невозможно. Для сложных алгоритмов приведены подробные примеры решения задач. В этих примерах показана вся процедура построения алгоритма, поясняющая переход от исходного текста к ее описанию в виде алгоритма, а затем численная реализация алгоритма, обеспечивающая решение задачи.

Курс лекций предназначен для студентов, обучающихся по направлениям подготовки 09.03.01, 09.03.02, 09.03.03, а также для магистров, изучающих курсы моделирования, оптимизации и управления в технологических и информационных системах.

УДК 004

ISBN 978-5-87623-781-1

ОГЛАВЛЕНИЕ

Предисловие

В курсе лекций представлены наиболее распространенные методы оптимизации, используемые при решении задач моделирования и управления.

При изложении методов безусловной оптимизации использована классическая последовательность методов. При изложении методов условной оптимизации, в отличие от классического порядка, вначале рассматриваются задачи нелинейного программирования, а затем линейное программирование, что позволяет более логично преподнести проблему линейного программирования. Логика подачи материала усовершенствована также и в самой теме нелинейного программирования. В сложных случаях подробно рассмотрены решения соответствующих задач. Кроме задач поиска экстремума функций на непрерывном множестве значений аргументов изложены проблемы дискретной оптимизации. Показаны принципы построения и реализация алгоритма динамического программирования, основных алгоритмов дискретного линейного программирования и алгоритмов оптимизации на графах и сетях.

Последовательность изложения материала позволит учащимся легко переходить от простых проблем к более сложным. Учебный материал представлен в виде четырех разделов.

В первом разделе рассмотрены общие проблемы поиска экстремума функций.

Во втором разделе представлены классические методы безусловной оптимизации – поиск экстремума функций без ограничений, накладываемых на область определения аргументов.

В третьем разделе изложены методы условной оптимизации. Этот раздел имеет два подраздела, посвященных нелинейному и линейному программированию. Названия связаны с соответствующей формой целевой функции и уравнений-ограничений.

Четвертый раздел посвящен изложению алгоритмов дискретной оптимизации, применяемых в случаях, когда возможные значения аргументов целевых функций заданы на дискретном множестве.

1. ОСНОВНЫЕ ПОНЯТИЯ И ПРОБЛЕМЫ ОПТИМИЗАЦИИ

1.1. Задача оптимизации

При разработке проектов систем различного рода возникает необходимость наилучшего в определенном смысле выбора параметров системы. При этом имеются задачи двух основных типов.

Задача первого типа ставится следующим образом: имеется система A с фиксированным входным вектором $x(t)$, управляемым входным вектором $u(t)$, выходным вектором $z(t)$, связанным с векторами $x(t)$ и $u(t)$ некоторым оператором (рис. 1.1), где t – время.

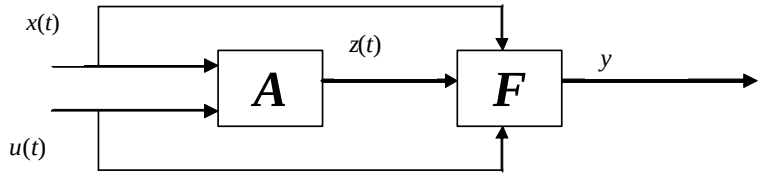

Рис. 1.1. Структура модели динамической оптимизации

Определен некоторый целевой функционал $y = F(x(t), u(t), z(t))$. Требуется найти такую векторную функцию $u(t)$, которая обеспечивает экстремум y.

Такие задачи обычно рассматриваются в курсе теории управления. Они относятся к классу задач динамической оптимизации.

Имеется обширный класс задач исследования и проектирования систем, которые можно отнести ко второму типу. В этих задачах входные и выходные векторные переменные не зависят от времени. При этом обычно говорят, что рассматривается статическая модель системы, представленная на рис. 1.2.

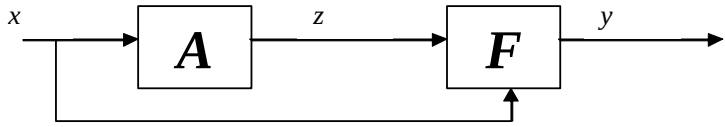

Рис. 1.2. Структура модели статической оптимизации

В рамках такой модели бывает определена целевая функция $y = F(x, z)$. При этом считается, что все входные переменные x системы могут выбираться из некоторой заданной области, а все неизменяемые входные параметры включены в оператор A. Переменные z при этом могут рассматриваться как функции $z(x)$, и тогда y является функцией только x.

Требуется выбрать значения x, соответствующие экстремуму функции $y = F(x)$.

Такая задача называется задачей статической оптимизации.

Если область, из которой можно выбирать значения x, составляет все пространство значений переменных (выбор не ограничен), соответствующая задача называется задачей безусловной оптимизации.

В противном случае область выбора ограничена с помощью некоторой системы неравенств. В частности, это могут быть равенства. В этом случае говорят, что имеется задача условной оптимизации, понимая под условиями соответствующие уравнения или неравенства, ограничивающие область выбора.

1.2. Геометрическое представление процесса поиска экстремума

Процесс функционирования алгоритмов поиска экстремума функции удобно иллюстрировать с помощью изображения траектории поиска в плоскости линий уровня функции двух переменных $y = f(x_1, x_2)$.

Заметим предварительно, что если рассматривается алгоритм поиска минимума функции, то для поиска максимума с помощью того же алгоритма достаточно изменить знак функции на обратный. Поэтому в дальнейшем при изложении курса все алгоритмы будут конструироваться для поиска минимума и отдельных оговорок об их изменении для поиска максимума не будет.

Линией уровня функции двух переменных называется ортогональная проекция сечения поверхности функции плоскостью, параллельной плоскости (x_1, x_2) и лежащей на некотором расстоянии c от нее (рис. 1.3). В случае функции n переменных понятие линии уровня сохраняется (вместо секущей плоскости имеется в виду соответствующая гиперплоскость), однако изображение теряет наглядность.

Уравнение линий уровня функции $y = f(x_1, x_2)$ записывается следующим образом:

$$f(x_1, x_2) = c.$$

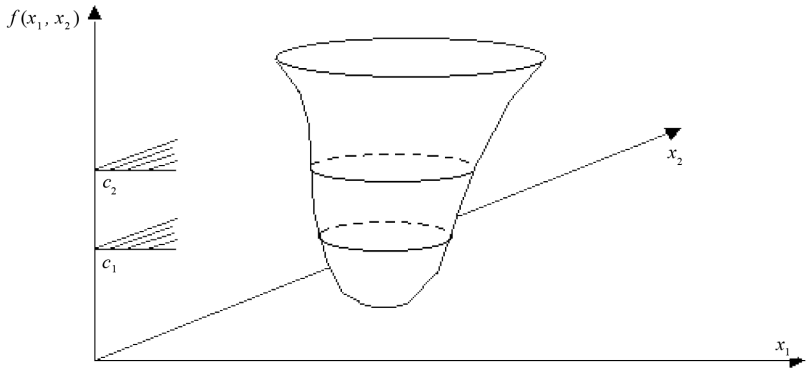

Рис. 1.3. Построение линий уровня

Если из этого уравнения можно выразить x_2, то уравнение линии уровня можно записать в явном виде:

$$x_2 = V(x_1, c).$$

Проекции линий уровня на плоскость, определяемую координатами x_1 и x_2, образуют семейство линий уровня на этой плоскости.

Алгоритмы поиска минимума обеспечивают пошаговое движение к минимуму начиная от некоторой произвольной начальной точки в области определения функции. Пример такого движения на плоскости линий уровня показан на рис.1.4. Траектория поиска заканчивается в районе точки минимума x^*.

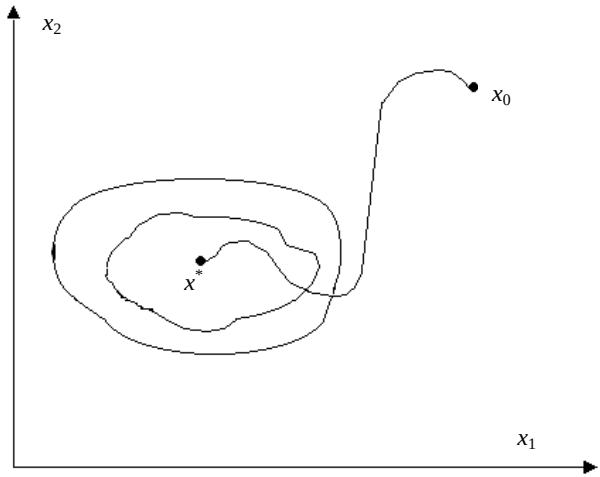

Рис. 1.4. Типовая траектория поиска минимума функции

В алгоритмах поиска минимума часто используется понятие градиента функции. Напомним, что градиентом называется вектор, компонентами которого являются частные производные функции по ее аргументам. Для функции двух переменных (прил. 1) этот вектор перпендикулярен касательной к линии уровня в рассматриваемой точке и указывает направление максимального возрастания функции, т.е. максимальное значение приращения функции Δf при фиксированном модуле вектора Δx.

1.3. Условия существования минимума

Запишем разложение функции $f(x)$ в ряд Тейлора в окрестности точки x^* с точностью до малых 3-го порядка (заметим, что под x и dx везде будут подразумеваться векторы за исключением отдельно оговариваемых случаев, а Δx^T означает операцию транспонирования вектора):

$$f(x) = f(x^*) + G^T(x^*)\Delta x + \frac{1}{2}\Delta x^T H(x^*)\Delta x + O(\Delta x), \qquad (1.1)$$

где $G(x^*)$ – вектор-градиент в точке x^*,

$H(x^*)$ – матрица вторых производных (матрица Гессе);

$O(\Delta x)$ – бесконечно малые компоненты.

Напомним, что условие стационарности точки x^* есть равенство нулю градиента: $G(x^*) = 0$.

В точке минимума приращение функции $\Delta f(x) = f(x) - f(x^*)$ должно быть больше нуля при любом векторе x, удовлетворяющем условию $\|x - x^*\| < \varepsilon$.

Это условие записывают также следующим образом: x принадлежит $\mathrm{sph}(x^*, \varepsilon)$, т.е. точка x принадлежит множеству точек сферы радиуса ε с центром в точке x^*. Тогда из соотношения (1.1) следует положительность квадратичной формы:

$$\Delta x^T H(x^*)\Delta x > 0. \qquad (1.2)$$

Последнее выражение означает положительную определенность матрицы H в точке x^*.

Заметим, что при построении алгоритмов поиска минимума должно быть введено условие останова. Это условие для гладких функций обычно состоит в проверке близости модуля градиента к нулю. С другой стороны, алгоритмы поиска должны быть рассчита-

ны на то, что в точке минимума градиент может не существовать. Тогда правило останова должно быть реализовано в виде непосредственной проверки условия $\Delta f(x) > 0$.

1.4. Общий вид алгоритма поиска минимума функции

Алгоритм поиска минимума некоторой функции $f(x)$ в общем виде представляет собой итерационную процедуру вида

$$x(k+1) = x(k) + \alpha(k) S(k), \qquad (1.3)$$

где $x(k)$ – предыдущая точка в пространстве переменных;
$x(k + 1)$ – следующая точка;
$S(k)$ – вектор направления перемещения;
$\alpha(k)$ – скалярный множитель, определяющий величину шага в направлении $S(k)$;
k – номер итерации.

Операции алгоритма могут быть представлены следующим образом.

1. Задать $k = 0$ и начальную точку $x(0)$.
2. Вычислить значение функции $f(x(k))$.
3. Вычислить направление $S(k)$ из условия уменьшения функции $f(x)$.
4. Вычислить очередную точку $x(k + 1) = x(k) + \alpha(k)\, S(k)$.
5. Вычислить значение $f(x(k + 1))$.
6. Вычислить и проверить выполнение условия останова. Если условие выполнено, идти к 9.
7. $k = k + 1$.
8. Идти к 3.
9. Вывод результатов. STOP.

Методы поиска минимума отличаются один от другого:
– способом вычисления направления;
– способом вычисления множителя $\alpha(k)$;
– условиями останова.

Наиболее общим условием останова является сравнение нормы разности $\Delta x(k) = x(k + 1) – x(k)$ с заданным числом ε_x.

Останов алгоритма в общем случае еще не означает, что найдена точка минимума. Чтобы это утверждать, нужно проверить классическое условие минимума:

$$\Delta f(x) = f(x) - f(x^*) > 0$$

для всех x из sph($x(k + 1)$, ε), т.е. из ε-окрестности точки $x(k + 1)$.

Одним из распространенных способов вычисления множителя $\alpha(k)$ является следующий:

$$\alpha(k) = \arg\min f(x(k) + \lambda S(k)),$$

где минимум берется по переменной λ.

Это означает, что точка $x(k + 1)$ определяется как минимум функции $f(x)$ в направлении $S(k)$. При этом используется один из способов поиска минимума функции одной переменной, а конкретное значение $\alpha(k)$ не используется в алгоритме. Указанный поиск называется одномерным поиском в заданном направлении.

1.5. Методы одномерного поиска в заданном направлении

Классические методы одномерного поиска предполагают, что указанная выше функция от λ имеет единственный экстремум в направлении $S(k)$. Для любого алгоритма поиска выбирается некоторая начальная точка $x(0)$ и задается так называемый шаг грубого поиска h. (Значение h обычно на порядок больше, чем заданная точность одномерного поиска ε_x.) Затем определяется $f(x(0))$, вычисляется новая точка $x(1) = x(0) + hS(k)$ и $f(x(1))$.

Если $f(x(1)) < f(x(0))$, то определяется следующая точка $x(k + 1) = x(k) + hS(k)$ и т.д., пока не будет выполнено условие $f(x(k + 1)) \geq f(x(k))$. Затем производится так называемая фиксация области минимума. Очевидно, что в общем случае минимум должен находиться между точками $x(k + 1)$ и $x(k – 1)$. Введем обозначения $a = x(k + 1)$ и $b = x(k – 1)$, где a и b есть концы отрезка, содержащего минимум.

Если при первом шаге оказалось, что $f(x(1)) \geq f(x(0))$, то делается обратный шаг: $\tilde{x}(1) = x(0) – h\ S(0)$. Если при этом $f(\tilde{x}(1)) \geq f(x(0))$, то присваиваются значения $a = \tilde{x}(1)$, $b = x(1)$.

Если же $f(\tilde{x}(1)) < f(x(0))$, то делается замена $h = –h$ и производится процедура поиска области минимума, как указано выше.

Далее выполняется алгоритм точного поиска на отрезке $[a, b]$.

Большинство методов точного поиска основаны на следующем правиле: на отрезке $[a, b]$ помещаются две точки, симметричные относительно его середины. Назовем левую точку c, а правую точку d. Вычислим $f(c)$ и $f(d)$ (рис. 1.5).

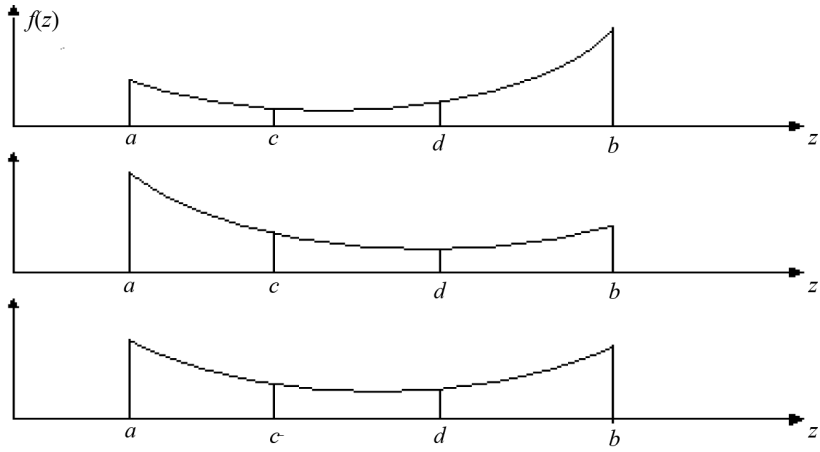

Рис. 1.5. Возможное положение минимума на отрезке [*a*, *b*]

Из рис. 1.5 следует, что дальнейшая локализация области минимума может быть выполнена по одному из следующих правил:

– если $f(c) < f(d)$, то $b = d$;

– если $f(c) > f(d)$, то $a = c$;

– если $f(c) = f(d)$, то $a = c$, $b = d$.

Полученный в результате новый отрезок [*a*, *b*] подвергается той же операции. Останов указанного цикла происходит по условию

$$||b - a|| < \varepsilon_x.$$

После этого минимум фиксируется в точке $x^* = (a + b) / 2$.

Конкретные алгоритмы поиска различаются способом задания точек *c* и *d*. Ниже приведены два типа алгоритмов.

1. Метод «золотого сечения».

Если обозначить через *L* длину отрезка [*a*, *b*], а через *m* длину отрезка [*a*, *c*], то отрезок [*a*, *b*] делится в отношении

$$\frac{m}{L-m} = \frac{L-m}{L}. \qquad (1.4)$$

Отсюда, поскольку $m < L$, следует; что $m = L(3 - \sqrt{5})/2 \approx 0{,}382L$.

Тогда $L - m = 0{,}618L$.

Поэтому расчет точек *c* и *d* производится по следующим формулам:

$$c = a + 0{,}382(b - a); \; d = a + 0{,}618(b - a).$$

Напомним, что все переменные здесь – это векторы.