**Лабораторная работа № 3**

**Оптимизация**

**Линейное и нелинейное программирование**

В состав MATLAB входит Optimization Toolbox, предназначенный для решения линейных и нелинейных оптимизационных задач. Функции этого Toolbox реализуют основные алгоритмы оптимизации, причем понимание алгоритма позволяет настроить выбранную функцию на эффективное решение поставленной задачи, что продемонстрировано на примере системы нелинейных уравнений. Optimization Toolbox не имеет приложений с графическим интерфейсом. Последний раздел данной главы содержит пример приложения, облегчающего доступ к нужной функции Toolbox и управление вычислительным процессом.

**Optimization Toolbox**

При исследовании функций вы использовали ряд функций Optimization Toolbox для решения уравнений (fzero), минимизации функции одной переменной на отрезке (fminbnd) и минимизации функций нескольких переменных без ограничений на независимые переменные (fminsearch). Данный раздел посвящен более сложным задачам с ограничениями, которые могут быть решены с использованием вычислительных функций Optimization Toolbox. Интерфейс этих функций достаточно гибкий, все они допускают обращение с переменным числом входных и выходных аргументов в зависимости от данных задачи и искомых величин. Кроме того, большинство функций Toolbox позволяют исследование задач, зависящих от одного или нескольких параметров.

**Линейное и нелинейное программирование**

**Линейное программирование**

Задача линейного программирования состоит в нахождении вектора ***х***, который минимизирует целевую линейную функцию

|  |  |  |
| --- | --- | --- |
|  | *fTx*, | (16.1) |

где *f* – вектор коэффициентов, и удовлетворяет заданным линейным ограничениям: неравенствам

|  |  |  |
| --- | --- | --- |
|  | *Ах* < *b* | (16.2) |

и равенствам

|  |  |  |
| --- | --- | --- |
|  | *Aeqx = beq*. | (16.3) |

Кроме того, могут быть поставлены двусторонние покомпонентные ограничения в векторной форме

|  |  |  |
| --- | --- | --- |
|  | *lb* < *x* < *ub*. | (16.4) |

В задачах оптимизации могут быть заданы не все типы ограничений, например, ограничения-равенства могут отсутствовать.

Для решения задач линейного программирования предназначена функция linprog. Первым аргументом linprog всегда является вектор *f*, далее задаются матрица *А* и вектор *b*. При наличии ограничений в виде равенств дополнительными аргументами могут быть *Aeq* и *beq*, наконец, двусторонние ограничения являются шестым и седьмым аргументами linprog.

Решите классическую задачу линейного программирования о составлении рациона питания. Имеются три продукта Ш, П2, ПЗ разной цены, каждый из которых содержит определенное количество питательных ингредиентов И1, И2, ИЗ, И4 (табл. 16.1). Известно, что в день требуется: И1 – не менее 250, И2– не менее 60, ИЗ– не менее 100 и И4– не менее 220. Требуется минимизировать затраты на приобретение продуктов. Очевидно, что количество приобретаемых продуктов не может быть отрицательным.

Таблица 16.1. Питательность и цена продуктов

|  |  |  |  |
| --- | --- | --- | --- |
|  | **П1** | **П2** | **П3** |
| **И1** | 4 | 6 | 15 |
| **И2** | 2 | 2 | 0 |
| **ИЗ** | 5 | 3 | 4 |
| **И4** | 7 | 3 | 12 |
| **Цена** | 44 | 35 | 100 |

Запишите целевую функцию *f*, матрицу *А*, векторы *b* и *lb* ограничений в соответствии с требованиями Toolbox, обозначив искомые количества продуктов через *х*1, *х*2 и *х*3 соответственно. Поскольку линейные ограничения содержат "меньше или равно", а количество ингредиентов в рационе не должно быть менее заданных величин, то следует изменить знаки обеих частей системы.



Для решения задачи составьте файл-программу ration. При вызове linprog вместо неиспользуемых аргументов (нет ограничений в виде равенств) задайте пустые массивы, обозначаемые в MATLAB квадратными скобками.

Верхнее ограничение вида *x* < *ub* отсутствует, а функция linprog поддерживает обращение с переменным числом входных аргументов, поэтому седьмой входной аргумент не нужен.

Листинг 16.1 содержит операторы файл-программы ration.

>> A = [4 6 15

2 2 0

5 3 4

7 3 12];

>> A = -A;

>> b = [250; 60; 100; 220];

>> b=-b;

>> % Определение коэффициентов целевой функции

f = [44; 35; 100];

>> % Задание ограничений снизу на переменные

>> lb =[0; 0; 0] ;

>> % Решение и вывод результата в командное окно

>> x = linprog(f, A, b, [ ], [ ], lb)

Optimization terminated.

x =

 13.2143

 16.7857

 6.4286

>>

Обращение к linprog с двумя выходными аргументами позволяет не только получить вектор решения, но и значение целевой функции, т. е. минимальную стоимость рациона в рассматриваемой задаче:

>> [x, p] = linprog(f, A, b, [],[], lb, [ ]);

Optimization terminated.

>> p

p =

 1.8118e+003

**Квадратичное программирование**

В задачах квадратичного программирования целевая функция имеет вид

|  |  |  |
| --- | --- | --- |
|  |  | (16.5) |

а ограничения в общем случае совпадают с ограничениями (16.2 – 16.4) в задаче линейного программирования. Для решения задач квадратичного программирования предназначена функция quadprog. Интерфейс quadprog практически не отличается от linprog, за исключением того, что первыми двумя входными параметрами являются массив н и вектор-столбец f, соответствующие матрице *Н* и вектору *f* целевой функции. Вместо матриц и векторов отсутствующих ограничений задаются пустые массивы.

Проиллюстрируем использование функции quadprog на примере задачи Марковица об определении состава инвестиционного портфеля рискованных ценных бумаг. Инвестор предполагает вложить свободные денежные средства в рыночные активы (ценные бумаги, акции) с целью получения дохода в будущем периоде. Для уменьшения рисков он выбрал для вложения четыре различных акции, которые обозначим А1, А2, A3, А4. Перед инвестором стоит задача определить, какую часть своих средств вложить в каждый актив так, чтобы получить желаемую доходность портфеля с наименьшим риском. Портфель определяется вектором долей от суммы инвестиций для покупки акций:

*x* = (*x*l, *x*2, *x*3, *x*4)*T*, *х*1+*х*2+*х*3 + *х*4=1, *хi*> 0 (*i* = 1,2,3,4).

Риск оценивается как величина среднеквадратического отклонения ожидаемой доходности. Будем считать, что инвестор каким-либо способом произвел оценку ожидаемой доходности для каждой ценной бумаги, т. е. построил вектор доходностей *у* = (*у*1, *y*2, *y*3, *y*4)*T* и вычислил матрицу ковариации *V*.

Если вектор у и матрица V известны, то доходность портфеля и его дисперсия вычисляются по формулам:



Формальная постановка задачи Марковица приводит к задаче квадратичного программирования: найти минимум функции



при ограничениях:



Величина желаемой доходности портфеля а должна быть не меньше минимальной и не больше максимальной доходности выбранных для инвестирования ценных бумаг. Неравенство (16.9) – одностороннее покомпонентное ограничение типа (16.4). Ограничения (16.7) и (16.8) – это ограничения вида (16.3), поэтому их следует объединить в одно, построив матрицу



и вектор



Пусть заданы следующие значения для матрицы *V*, вектора *y* и желаемой

доходности портфеля *a*:



Тогда в обозначениях, принятых для описания функции quadprog, построим матрицы и вектора, связанные с ограничениями:



Для решения задачи о формировании портфеля с фиксированной доходностью составьте файл-программу risk\_asset. Возможный вариант исходного текста файл-программы risk\_asset приведен в листинге 16.2.:

% Задание матрицы коэффициентов целевой функции

V =[102.0 27.1 -52.3 66.5;

27.1 148.8 42.1 -66.4;

-52.3 42.1 246.5 56.9;

66.5 -66.4 56.9 272.3];

% Задание ограничений типа равенств

Aeq = [11.3 13.2 16.1 17.4;

1 1 1 1];

beq = [15; 1];

% Задание ограничений снизу на переменные

lb = [0; 0; 0; 0] ;

% Решение и вывод результата в командное окно

x = quadprog( V, [],[], [ ], Aeq, beq, lb)

Warning: Large-scale method does not currently solve this problem formulation, using medium-scale method instead.

> In quadprog at 262

 In risk\_asset at 13

Optimization terminated.

x =

 0.0626

 0.4359

 0.1439

 0.3575

>>

При этом в командное окно вывелось сообщение о применении Medium-scale алгоритма вместо Large-scale, принятого по умолчанию (см. разд. ''Параметры оптимизации").

**Нелинейное программирование**

Optimization Toolbox позволяет решать ряд оптимизационных задач, в которых минимизируемая функция нелинейна, и к линейным ограничениям добавляются нелинейные. Общая постановка задачи нелинейного программирования такова: требуется разыскать

|  |  |  |
| --- | --- | --- |
|  | min *f*(*x*) | (16.10) |

среди всех векторов х, удовлетворяющих системе линейных ограничений (16.2–16.4) и дополнительных неравенств и равенств

|  |  |  |
| --- | --- | --- |
|  | *c*(*x*) < 0; *ceq*(*x*) = 0. | (16.11) |

Решение поставленной задачи производится при помощи функции fmincon. Основное отличие интерфейса fmincon от linprog и quadprog состоит в том, что нелинейные ограничения *с*(*х*) < 0 и *ceq*(*x*) = 0 задаются в файл-функции. Обращение к fmincon в достаточно общем случае выглядит следующим образом:

х = fmincon (fun, х0, A, b, Aeq, beq, lb, ub, nonlcon, options, P1, P2, ...)

Указание второго дополнительного выходного аргумента позволяет получить значение функции в точке минимума, а третьего – информацию о результате. Если третий аргумент больше нуля, то результат найден с требуемой точностью, ноль – достигнуто максимальное число итераций или количество вызовов исследуемой функции в процессе решения, меньше нуля – решение не найдено. Первый входной аргумент fun является указателем на файл-функцию (или ее именем), вычисляющую минимизируемую функцию *f*(*x*), которая может зависеть от нескольких параметров. Значения параметров, в случае их наличия, передаются в последних аргументах P1, P2,... начиная с 11-ой позиции в списке входных аргументов.

Входным аргументом файл-функции fun должен быть вектор, длина которого совпадает с числом переменных, т. е. компонент вектора х. Неиспользуемые векторы и матрицы ограничений заменяются в списке входных аргументов fmincon квадратными скобками (пустым массивом). Начальное приближение к решению указывается в х0. Список входных аргументов fmincon содержит управляющую структуру options, предназначенную для задания опций вычислительных алгоритмов. Для большинства функций Toolbox Optimization набор опций существенно больше, чем для функций fzero, fminbnd и fminsearch.

Нелинейные ограничения (16.11) (неравенства и равенства) программируются в файл-функции, указатель на нее (или ее имя) указывается в аргументе nonlcon. Входным аргументом nonlcon является вектор *х*, соответствующий искомому вектору х, а двумя выходными аргументами – векторы с и ceq левых частей нелинейных ограничений *c* и *ceq*. Последние идущие подряд входные аргументы функции fmincon могут быть опущены, если они не используются. Например, при отсутствии нелинейных ограничений и параметров применяется следующий вызов fmincon:

х = fmincon (fun, х0, A, b, Aeq, beq, lb, ub)

Поскольку в данном случае управляющая структура options не задана, то вычисления будут производиться с принятыми по умолчанию опциями, узнать о которых можно при помощи функции optimset. Для этого достаточно вызвать ее от строки с именем вычислительной функции Optimization Toolbox, например, выполнение команды

» optimset('fmincon')

приводит к выводу информации о всех настройках алгоритма функции fmincon, в том числе и точности 10-6 по аргументу и функции.

Найдите решение следующей простой задачи (очевидно, что решение – нулевой вектор):



Обратите внимание, что имеется только одно нелинейное ограничение в виде неравенства, которое следует преобразовать к виду *c*(*x*) < 0, перенеся единицу в левую часть. Написание файл-функции, вычисляющей 3(*x*1)2 +2(*x*2)2, не представляет труда (листинг 16.3). При программировании нелинейного ограничения учтите, что соответствующая файл-функция возвращает два вектора левых частей нелинейных ограничений (листинг 16.4). В рассматриваемом примере первый вектор состоит только из одной компоненты, а второй должен быть пустым, поскольку нет ограничений в виде равенств.Листинг 16.3. Минимизируемая функция

function f =myfun(x)

% Вычисление минимизируемой функции

f =3\*x(1)^2 + 2\*x(2)^2;

Листинг 16.4. Файл- функция с ограничениями

function [c, ceq]= mycon(x)

% Задание ограничений

c(1) = x(1)^2 + x(2)^2 - 1; % ограничения в виде неравенства

% Правая часть ограничений-равенств является пустым массивом,

% поскольку данных ограничений нет

ceq = [];

Выполнение fmincon, например, из командной строки

>> x= fmincon (@myfun, [0.7 0.7], [], [], [],[],[],[],@mycon)

Warning: Large-scale (trust region) method does not currently solve this type of problem,

 using medium-scale (line search) instead.

> In fmincon at 317

Optimization terminated: magnitude of directional derivative in search

 direction less than 2\*options.TolFun and maximum constraint violation

 is less than options.TolCon.

No active inequalities.

x =

 1.0e-006 \*

 0.1349 0.5241

>>

Обращение к fmincon с тремя выходными аргументами позволяет получить значение функции в точке минимума:

>> [x,f,flag]= fmincon (@myfun, [0.7 0.7], [], [], [],[],[],[],@mycon)

Warning: Large-scale (trust region) method does not currently solve this type of problem,

 using medium-scale (line search) instead.

> In fmincon at 317

Optimization terminated: magnitude of directional derivative in search

 direction less than 2\*options.TolFun and maximum constraint violation

 is less than options.TolCon.

No active inequalities.

x =

 1.0e-006 \*

 0.1349 0.5241

f =

 6.0388e-013

flag =

 5

>> 1.0882e-009

Значение flag большее нуля свидетельствует о том, что решение успешно найдено.

**Самостоятельная работа**

Решить задачи линейного и нелинейного программирования

(Все переменные не отрицательные.)

1. *L* = *x*1 + *x*2 + *x*3 – *x*4 → max,

3*x*1 – *x*2 < 7,

*x*2 – 2*x*3 < –1,

4*x*3 – *x*4 < 3,

5*x*1 + 2*x*4 > 14.

1. *L* = 44*x*1 + 35*x*2 + 100*x*3 → min,

4*x*1 + 6*x*2 + 15*x*3 > 250,

2*x*1 + 2*x*2 > 60,

5*x*1 + 3*x*2 + 4*x*3 > 100,

7*x*1 + 3*x*2 + 12*x*3 > 22.

1. *L* = 4*x*1 + 3*x*2 + *x*3 → min,

*x*1 < 10,

*x*2 < 8,

*x*3 < 6,

*x*1 + 4*x*2 + 3*x*3 < 40,

*x*1 + *x*2 + *x*3 = 12.

1. *L* = 2 + *x*1 + 2*x*2 → min,

*x*1 + *x*2 > 2,

*x*1 + 2*x*2 > 3,

–2*x*1 + 2*x*2 > –4,

*x*1 + 3*x*2 > 4,

3*x*1 + *x*2 > 4.

1. *L* = *x*2 + 2*x*3 – *x*4 → min,

–*x*1 + *x*2 –2x4 > –1,

*x*1 + *x*3 + x4 > 1,

*x*2 + *x*3 – x4 > 1.

1. *F* = 2*x*12 + 5*x*22 – 9 → min,

*x*1 *x*2 = 1.

1. *F* = *x*2 – *x*12 + 6*x*1 → max,

3*x*1 + 2*x*2 < 24,

*x*2 < 4.

1. *F* = *x*12 + *x*22 – 1 → min,

*x*1 *x*2 = 4.

1. *F* = 3*x*12 + 2*x*22 → min,

*x*12 + *x*22 < 1.

Алексеев Е.Р. MAILAB 7. Самоучитель / Е.Р. Алексеев, О.В. Чеснокова. – М.: НТ Пресс, 2006. – 464 с.

Ануфриев И. Е., Смирнов А. Б., Смирнова Е. Н. MATLAB 7. - СПб.: БХВ-Петербург, 2005. - 1104 с: ил.