

Решение задач линейного программирования в среде MATLAB*

А. Н. Сергеев
A_SerreeB@mail.ru

Н. А. Соловьёва
vinyo@mail.ru

Е. К. Чернэуцану
katerinache@yandex.ru

12 февраля 2011 г.

В среде MATLAB задачи линейного программирования решаются с помощью функции `linprog`. Доклад посвящён описанию её возможностей.

1°. Функция `linprog` решает задачу линейного программирования в форме

$$\begin{aligned} \mathbf{f}^T \cdot \mathbf{x} &\rightarrow \inf, \\ \mathbf{A} \cdot \mathbf{x} &\leq \mathbf{b}, \\ \mathbf{A}_{\text{eq}} \cdot \mathbf{x} &= \mathbf{b}_{\text{eq}}, \\ \mathbf{lb} &\leq \mathbf{x} \leq \mathbf{ub}. \end{aligned} \tag{1}$$

Основными входными данными `linprog` являются: вектор коэффициентов целевой функции \mathbf{f} , матрица ограничений-неравенств \mathbf{A} , вектор правых частей ограничений-неравенств \mathbf{b} , матрица ограничений-равенств \mathbf{A}_{eq} , вектор правых частей ограничений-равенств \mathbf{b}_{eq} , вектор \mathbf{lb} , ограничивающий план \mathbf{x} снизу, вектор \mathbf{ub} , ограничивающий план \mathbf{x} сверху. На выходе функция `linprog` даёт оптимальный план \mathbf{x} задачи (1) и экстремальное значение целевой функции `fval`.

ПРИМЕР 1. Решим в MATLAB задачу линейного программирования

$$\begin{aligned} f(x) &= 3x_1 + x_2 + 2x_3 \rightarrow \inf, \\ x_1 + x_2 + x_3 &\geq 1, \\ 2x_1 + x_2 - x_3 &\geq -1, \\ x_1 - x_2 + x_3 &= 0, \\ 0 &\leq x_1 \leq 1, \\ 0 &\leq x_2 \leq 1, \\ 0 &\leq x_3 \leq 1. \end{aligned}$$

*Семинар по дискретному гармоническому анализу и геометрическому моделированию «DHA & CAGD»: <http://www.dha.spb.ru/>

Соответствующая программа (m-файл)* выглядит так:

```
clear all
close all
clc % удаляются все текущие переменные из памяти MATLAB,
    закрываются все графические окна, очищается экран консоли
C = [3 1 2]; % задаётся вектор длины три
D = [1 1 1; 2 1 -1]; % строки матрицы разделяются точкой с
    запятой
B = [1 -1];
Aeq = [1 -1 1];
beq = [0];
lb = zeros(3,1); % задаётся нулевой вектор длины три
ub = [1 1 1];
f = C;
A = -D;
b = -B; % появляются знаки «-», так как ограничения-неравенства
     $Dx \geq B$  приводятся к виду  $-Dx \leq -B$ 
[x,fval] = linprog(f,A,b,Aeq,beq,lb,ub);
x
fval
```

Запустив программу, получим сообщение

```
Optimization terminated.
x =
    0
    0.5000
    0.5000
fval =
    1.5000
```

Дополнительно можно задать начальное приближение x_0 :

```
[x,fval] = linprog(f,A,b,Aeq,beq,lb,ub,x0).
```

Если какой-то из входных параметров отсутствует, на его место следует поставить квадратные скобки [], за исключением случая, когда это последний параметр в списке. Например, если нужно решить задачу без ограничений-равенств, в которой не задано начальное приближение, то оператор вызова функции `linprog` будет выглядеть так:

```
[x,fval] = linprog(f,A,b,[],[],lb,ub).
```

*Для отладки приведённых в докладе программ использовался MATLAB 7.11.0 (R2010b).

(Квадратные скобки в конце списка, соответствующие начальному приближению, не ставятся.)

С помощью входного параметра `options` устанавливаются некоторые дополнительные настройки, в частности, выбирается алгоритм решения. MATLAB решает задачи линейного программирования двумя способами: алгоритмом внутренней точки (`Large-Scale Algorithm`) и вариантом симплекс-метода (`Medium-Scale Algorithm`). По умолчанию используется алгоритм внутренней точки. Чтобы выбрать симплекс-метод, нужно написать

```
options = optimset('LargeScale','off','Simplex','on');
[x,fval] = linprog(f,A,b,Aeq,beq,lb,ub,[],options).
```

Разберёмся с выходными данными. MATLAB позволяет выводить информацию о том, как завершилось решение задачи. За это отвечает параметр `exitflag`. Если значение `exitflag` равно 1, то найдено решение задачи, если равно 0, то превышено допустимое число итераций, если равно -2 — множество планов задачи пусто, если равно -3 — целевая функция не ограничена снизу на множестве планов. Интерпретация других значений параметра `exitflag` приведена в MATLAB Help. Для симплекс-метода допустимое число итераций (`MaxIter`) по умолчанию в 10 раз больше количества переменных. Значение `MaxIter` можно изменить. Чтобы установить допустимое число итераций равным, к примеру, 10, нужно написать

```
options =
optimset('LargeScale','off','Simplex','on','MaxIter',10);
[x,fval] = linprog(f,A,b,Aeq,beq,lb,ub,[],options).
```

Если после выполнения десятой итерации решение не будет найдено, параметр `exitflag` станет нулевым и на экране появится сообщение

```
Maximum number of iterations exceeded;
increase options.MaxIter.
```

Параметр `output` содержит информацию о процессе оптимизации, в частности, число итераций (`iterations`) и используемый алгоритм (`algorithm`). Другие поля параметра `output` описаны в MATLAB Help. Запустим с данными из примера 1 следующую программу:

```
options = optimset('LargeScale','off','Simplex','on');
[x,fval,exitflag,output] =
linprog(f,A,b,Aeq,beq,lb,ub,[],options);
exitflag
output.iterations
output.algorithm
```

На выходе получим:

```

Optimization terminated.
exitflag =
    1
ans =
    1
ans =
    medium scale: simplex

```

Это означает, что симплекс-метод успешно завершил работу, для нахождения решения потребовалось одна итерация.

Наконец, в выходном параметре `lambda` содержится решение двойственной задачи линейного программирования. Параметр `lambda` состоит из четырёх массивов: `lambda.ineqlin`, `lambda.eqlin`, `lambda.upper`, `lambda.lower`. В этих массивах находятся двойственные переменные, приписанные ограничениям-неравенствам, ограничениям-равенствам, ограничениям на план сверху и снизу соответственно. Подробное обсуждение этого вопроса отложим до п. 3°.

2°. При решении задачи линейного программирования возможны три выхода из процесса: найдено решение задачи, множество планов пусто, целевая функция не ограничена снизу на множестве планов. Продемонстрируем эти варианты на примерах.

ПРИМЕР 2. Решим в MATLAB задачу линейного программирования

$$\begin{aligned}
 f(x) &= 4x_1 + x_2 \rightarrow \inf, \\
 x_1 + x_2 &\geq 2, \\
 x_1 - x_2 &\geq 1, \\
 x_1 &\geq 0, \quad x_2 \geq 0.
 \end{aligned}
 \tag{2}$$

Соответствующая программа будет выглядеть так:

```

clear all
close all
clc
C = [4 1];
D = [1 1; 1 -1];
B = [2 1];
lb = zeros(2,1);
f = C;
A = -D;
b = -B;
options = optimset('LargeScale','off','Simplex','on');

```

```
[x,fval,exitflag] = linprog(f,A,b,[],[],lb,[],[],options);
x
fval
exitflag
```

В результате работы программы получим:

```
Optimization terminated.
x =
    1.5000
    0.5000
fval =
    6.5000
exitflag =
    1
```

Найдено решение задачи (2).

ПРИМЕР 3. Решим в MATLAB задачу линейного программирования

$$\begin{aligned}
 f(x) &= x_1 + x_2 \rightarrow \inf, \\
 -x_1 - x_2 &\geq -1, \\
 x_1 + 4x_2 &\geq 8, \\
 x_1 &\geq 0, \quad x_2 \geq 0.
 \end{aligned}
 \tag{3}$$

Приведём результат работы соответствующей программы:

```
Exiting: The constraints are overly stringent; no feasible
starting point found.
x =
    0
    1
fval =
    1
exitflag =
   -2
```

Множество планов задачи (3) пусто.

ПРИМЕР 4. Решим в MATLAB задачу линейного программирования

$$\begin{aligned}
 f(x) &= -x_1 - 3x_2 \rightarrow \inf, \\
 2x_1 - x_2 &\geq 0, \\
 -x_1 + x_2 &\geq -1, \\
 x_1 &\geq 0, \quad x_2 \geq 0.
 \end{aligned}
 \tag{4}$$

Запустив программу, решающую задачу (4), получим:

```

Exiting: The solution is unbounded and at infinity;
the constraints are not restrictive enough.
x =
  1.0e+016*
  1.0000
  2.0000
fval =
 -7.0000e+016
exitflag =
 -3

```

Целевая функция задачи (4) не ограничена снизу на множестве планов.

3°. Рассмотрим задачу линейного программирования в общем виде

$$\begin{aligned}
 f(x) &:= c[N] \times x[N] \rightarrow \inf, \\
 D[M_1, N] \times x[N] &\geq B[M_1], \\
 D[M_2, N] \times x[N] &= B[M_2], \\
 -x[N] &\geq -v[N], \\
 x[N] &\geq w[N].
 \end{aligned}$$

Переходя к обозначениям MATLAB, имеем:

$$\begin{aligned}
 \mathbf{f} &= c[N], \quad \mathbf{A} = -D[M_1, N], \quad \mathbf{b} = -B[M_1], \\
 \mathbf{Aeq} &= D[M_2, N], \quad \mathbf{beq} = B[M_2], \\
 \mathbf{ub} &= v[N], \quad \mathbf{lb} = w[N].
 \end{aligned}$$

Пусть $N = 1 : n$, $M_1 = 1 : s$, $M_2 = s + 1 : t$.

Вначале рассмотрим случай, когда $w[N] = \mathbb{O}$ или $w[N]$ не задан. Переменных в двойственной задаче будет $t + n$. Двойственные переменные содержатся в параметре `lambda`. Их можно найти по формулам

$$\begin{aligned}
 u_i^* &= \text{lambda.ineqlin}(i), & i = 1, \dots, s, \\
 u_i^* &= \text{lambda.eqlin}(i - s), & i = s + 1, \dots, t, \\
 u_i^* &= \text{lambda.upper}(i - t), & i = t + 1, \dots, t + n.
 \end{aligned} \tag{5}$$

Заметим, что

- 1) если $M_1 = \emptyset$, то считаем $s = 0$ и не используем первое соотношение из (5) (массив `lambda.ineqlin` пустой);

- 2) если $M_2 = \emptyset$, то считаем $t = s$ и не используем второе соотношение из (5) (массив `lambda.eqlin` пустой);
- 3) если не задан вектор верхних границ $v[N]$, то третье соотношение (5) не используется (массив `lambda.upper` нулевой).

Теперь предположим, что $w[N] \neq \mathbb{O}$. Пусть

$$w[k_i] \neq 0, \quad i = 1, \dots, l, \quad l \leq n.$$

Тем самым выделяются индексы знаковых ограничений $N \setminus \{k_1, \dots, k_l\}$. Переменных в двойственной задаче будет $t + n + l$. Их можно найти по формулам

$$\begin{aligned} u_i^* &= \text{lambda.ineqlin}(i), & i &= 1, \dots, s, \\ u_i^* &= \text{lambda.eqlin}(i - s), & i &= s + 1, \dots, t, \\ u_i^* &= \text{lambda.upper}(i - t), & i &= t + 1, \dots, t + n, \\ u_i^* &= \text{lambda.lower}(k_{i-t-n}), & i &= t + n + 1, \dots, t + n + l. \end{aligned} \tag{6}$$

Если не задан вектор верхних границ $v[N]$, то третье соотношение (6) не используется (массив `lambda.upper` нулевой), а четвёртое принимает вид

$$u_i^* = \text{lambda.lower}(k_{i-t}), \quad i = t + 1, \dots, t + l. \tag{7}$$

ПРИМЕР 5. Рассмотрим задачу линейного программирования

$$\begin{aligned} f(x) &= x_1 + x_2 \rightarrow \inf, \\ x_1 - x_2 &\geq 2, \\ x_1 - 2x_2 &\geq 1, \\ -x_1 &\geq -4, \\ -x_2 &\geq -4, \\ x_1 &\geq 0, \quad x_2 \geq 0. \end{aligned} \tag{8}$$

Составим двойственную задачу:

$$\begin{aligned} g(u) &= 2u_1 + u_2 - 4u_3 - 4u_4 \rightarrow \sup, \\ u_1 + u_2 - u_3 &\leq 1, \\ -u_1 - 2u_2 - u_4 &\leq 1, \\ u_1, u_2, u_3, u_4 &\geq 0. \end{aligned} \tag{9}$$

Решим задачу (8) в среде MATLAB. Программа будет выглядеть следующим образом:

```

clear all
close all
clc
C = [1 1];
D = [1 -1; 1 -2];
B = [2 1];
lb = zeros(2,1);
ub = [4 4];
f = C;
A = -D;
b = -B;
options = optimset('LargeScale','off','Simplex','on');
[x,fval,exitflag,output,lambda] =
linprog(f,A,b,[],[],lb,ub,[],options);
x
fval
lambda = structfun(@(t)(t.'),lambda,'UniformOutput',false)
% транспонируется содержимое lambda для лучшего отображения на
  экране консоли

```

На выходе получим оптимальный план $x_1^* = 2$, $x_2^* = 0$, минимальное значение целевой функции $f(x^*) = 2$ и `lambda`. Параметр `lambda` будет состоять из массивов

```

lambda =
ineqlin: [1 0]
eqlin: [1x0 double]
upper: [0 0]
lower: [0 2]

```

(Запись `eqlin: [1x0 double]` означает, что `eqlin` — пустой массив.)

Заметим, что в задаче (8) $t = s$ и $w[N] = \mathbb{O}$. По формулам (5)

$$u_1^* = 1, u_2^* = 0, u_3^* = 0, u_4^* = 0.$$

Легко проверить, что это план двойственной задачи (9), удовлетворяющий условиям дополнителности и соотношению двойственности $f(x^*) = 2 = g(u^*)$.

ПРИМЕР 6. Рассмотрим задачу линейного программирования

$$\begin{aligned}
 f(x) &= x_1 + x_2 \rightarrow \inf, \\
 x_1 - x_2 &\geq 2, \\
 x_1 - 2x_2 &\geq 1, \\
 x_1 &\geq 0, \\
 x_2 &\geq -1.
 \end{aligned} \tag{10}$$

Составим двойственную задачу:

$$\begin{aligned}
 g(u) &= 2u_1 + u_2 - u_3 \rightarrow \sup, \\
 u_1 + u_2 + u_3 &\leq 1, \\
 -u_1 - 2u_2 - u_3 &= 1, \\
 u_1, u_2, u_3 &\geq 0.
 \end{aligned} \tag{11}$$

Решим задачу (10) в среде MATLAB. Для этого в программе из предыдущего примера достаточно заменить `lb = zeros(2,1)` на `lb = [0 -1]` и в списке аргументов функции `linprog` вместо `ub` поставить `[]`. На выходе получим оптимальный план $x_1^* = 1$, $x_2^* = -1$, минимальное значение целевой функции $f(x^*) = 0$ и `lambda`. Параметр `lambda` будет состоять из массивов

```

lambda =
ineqlin: [1 0]
eqlin: [1x0 double]
upper: [0 0]
lower: [0 2]

```

Заметим, что в задаче (10) отсутствует $v[N]$ и $w[1] = 0$, $w[2] \neq 0$ (то есть $l = 1$, $k_1 = 2$). По формулам (6) и (7)

$$u_1^* = 1, u_2^* = 0, u_3^* = 2.$$

Легко проверить, что это план двойственной задачи (11), удовлетворяющий условиям дополнителности и соотношению двойственности $f(x^*) = 0 = g(u^*)$.