10. Решение оптимизационных задач в Octave

В данной главе рассматриваются решение задач поиска минимума (максимума) в Octave. В первой главе на примерах решения практических задач рассматривается функция sqp, предназначенная для поиска минимума функции одной или нескольких переменных с ограничениями. Вторая глава целиком посвящена задачам линейного программирования.

Изучение оптимизационных задач начнём с обычных задач поиска минимума (максимума) функции одной или нескольких переменных.

10.1 Поиск экстремума функции

Для решения классических оптимизационных задач с ограничениями в Octave можно воспользоваться следующей функцией

`[x, obj, info, iter] = sqp (x0, phi, g, h, lb, ub, maxiter, tolerance),`

которая предназначена для решения следующей оптимизационной задачи.

Найти минимум функции $\varphi(x)$ при следующих ограничениях $g(x)=0, h(x)\geq 0, lb\leq x\leq ub$. Функция `sqp` при решении задачи оптимизации использует метод квадратичного программирования.

Аргументами функции `sqp` являются

`x0` – начальное приближение значения `x`,

`phi` – оптимизируемая функция $\varphi(x)$,

`g` и `h` – функции ограничений $g(x)=0$ и $h(x)\geq 0$ соответственно,

`lb` и `ub` – верхняя и нижняя границы ограничения $lb\leq x\leq ub$,

`maxiter` – максимальное количество итераций, используемое при решении оптимизационной задачи, по умолчанию эта величина равна 100,

`tolerance` – точность ε , определяющая окончание вычислений, вычисления прекращаются при достижении точности $\sqrt{(\varepsilon)}$.

Функция `sqp` возвращает следующие значения

`x` – точка, в которой функция, достигает своего минимального значения,

`obj` – минимальное значение функции,

`info` – параметр, характеризующий корректность решения оптимизационной задачи, (если функция sqp возвращает значение $info=101$, то задача решена правильно),

`iter` – реальное количество итераций при решении задачи.

Рассмотрим несколько примеров использования функции `sqp` при решении задач поиска экстремума функции одной переменной без ограничений.

ПРИМЕР 10.1.

Найти минимум функции $\varphi(x)=x^4+3x^3-13x^2-6x+26$

При решении задачи оптимизации с помощью функции `sqp` необходимо иметь точку начального приближения. Построим график функции $\varphi(x)$ (см. рис. 10.1). Из графика видно, что функция имеет минимум в окрестности точки *x=-4*. В качестве точки начального приближения выберем x0=-3. Решение задачи представлено на листинге 10.1.

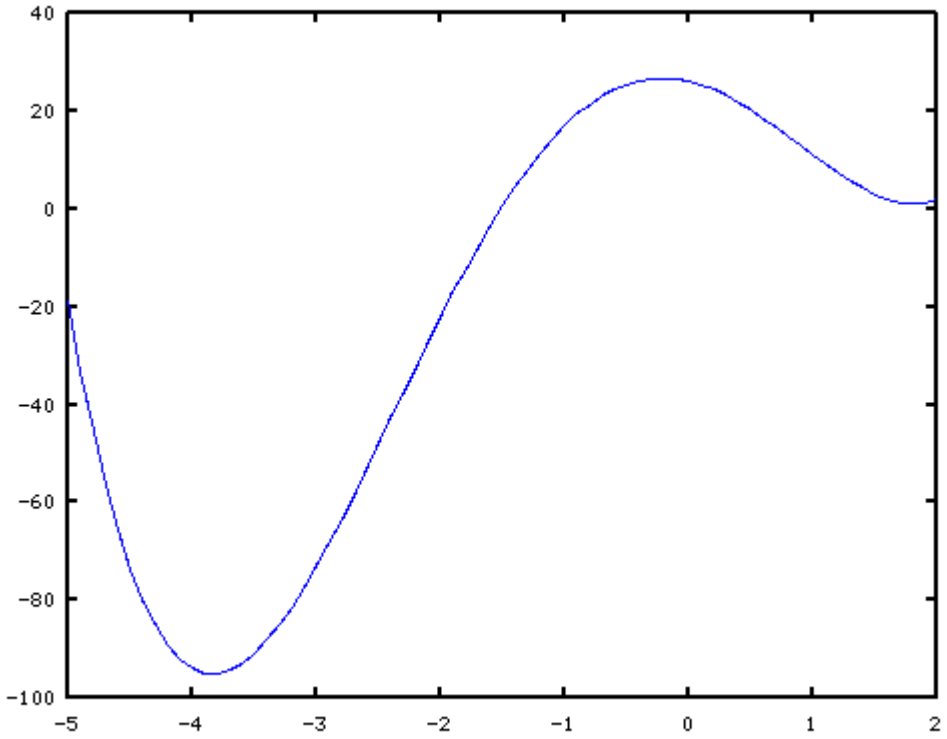
```
function obj = phi (x)
obj = x^4+3*x^3-13*x^2-6*x+26;
endfunction
[x, obj, info, iter]=sqp (-3, @phi)
```

Листинг 10.1

Результаты решения представлены ниже.

```
>>>[x, obj, info, iter]=sqp (-3, @phi)
```

```
x = -3.8407
obj = -95.089
info = 101
iter = 5
```




```
2.93069, -43.4882
```

Рисунок *10.1* График функции $\varphi(x)=x^4+3x^3-13x^2-6x+26$

Минимум функции $\varphi(x)=-95.089$ достигается в точке $x=-3.8407$, количество итераций равно 5, параметр $info=101$ свидетельствует о корректном решении задачи поиска минимума $\varphi(x)=x^4+3x^3-13x^2-6x+26$.

Рассмотрим пример поиска минимума функции нескольких переменных.

ПРИМЕР 10.2.

Найти минимум функции Розенброка $f(x,y)=N(y-x^2)^2+(1-x^2)^2$

Построим график функции Розенброка (см. листинг 10.2).

```
[x y]=meshgrid(-2:0.1:2,2:-0.1:-2);
z=20*(y-x.^2).^2+(1-x).^2;
surf(x,y,z);
```

Листинг 10.2

График полученной поверхности при N=20 приведён на рис. 10.2.

Как известно, функция Розенброка имеет минимум в точке (1,1) равный 0. В виду своей специфики функция Розенброка является тестовой для алгоритмов минимизации. Найдём минимум этой функции с помощью функции `sqp` (см. листинг 10.3). При решении задач на экстремум функций многих переменных следует следующие особенности синтаксиса при определении оптимизируемой функции. Аргументом функции многих переменных (в нашем случае – её имя `r`) является массив *x*, первая переменная имеет имя *x(1)*, вторая *x(2)* и т. д. Если имя аргумента функции многих переменных будет другим – допустим *m*, то изменятся и имена переменных: *m(1)*, *m(2)*, *m(3)* и т.д.

```
function y=r(x)
```

```
y=20*(x(2)-x(1)^2)^2+(1-x(1))^2;
endfunction
x0=[0;0];
[x, obj, info, iter]=sqp(x0,@r)
```

Листинг 10.3

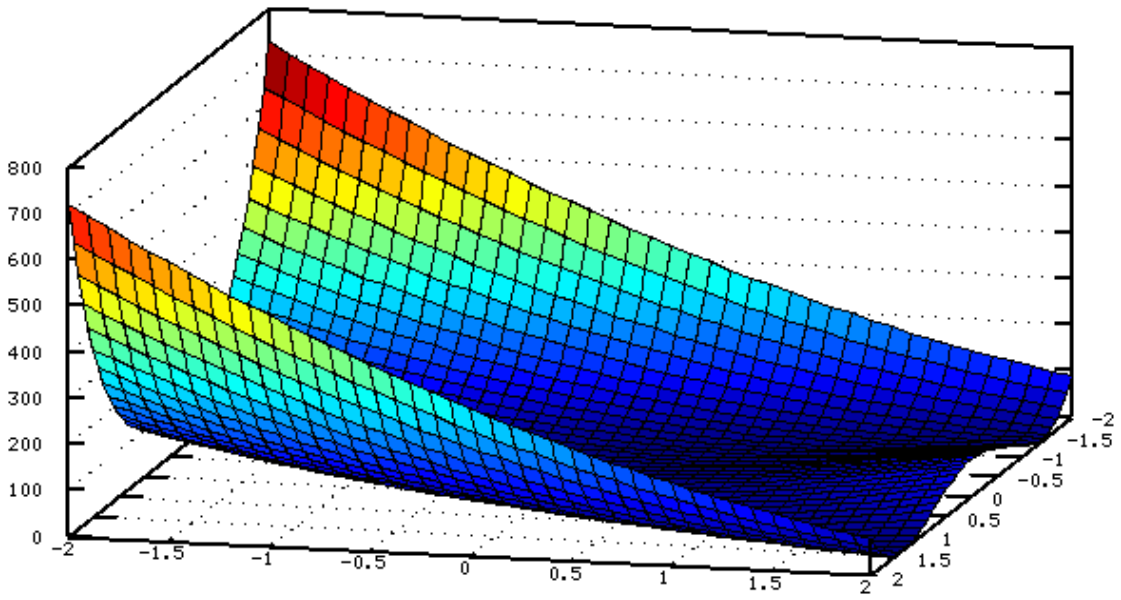


Рисунок *10.2:* График функции Розенброка

Решение задачи представлено ниже.

```
>>>x =
 1.00000
 1.00000
obj = 7.1675e-13
info = 101
iter = 14
```

Как и следовало ожидать, функция `sqp` нашла минимум в точке (1,1), само значение 0 найдено достаточно точно ($7.2 \cdot 10^{-13}$). Значение `info=101` говорит о корректном решении задачи, для нахождения минимального значения функции Розенброка потребовалось `14` итераций.

Таким образом, функция `sqp` предназначена для поиска минимума функций (как одной, так и нескольких переменных) с различными ограничениями.

Рассмотрим несколько задач поиска экстремума с ограничениями

ПРИМЕР 10.3.

Найти максимум и минимум функции $F=(x-3)^2-(y-4)^2$ при ограничениях

$$\begin{cases} 3x+2y\geq 7 \\ 10x-y\leq 8 \\ -18x+4y\leq 12 \\ x\geq 0 \\ y\geq 0 \end{cases}$$ [1].

В функции `sqp` все ограничения должны быть вида ≥ 0. Поэтому второе и третье ограничение на *-1*, и перенесём всё в левую часть неравенств. В результате этих несложных

преобразований система ограничений примет вид:

$$\boldsymbol{g}(x)=\begin{cases}3x+2y-7\geq 0\\ -10x+y+8\geq 0\\ 18x-4y+12\geq 0\\ x\geq 0\\ y\geq 0\end{cases}.$$

Последовательно рассмотрим задачу на минимум и максимум.

В задаче на минимум функция, в которой хранится F(x) будет такой

```
function y=f(x)
y=(-x(1)-3)^2+(-x(2)-4)^2;
endfunction
```

Вектор-функцию ограничений **g(x)** можно записать так

```
function r = g (x)
r=[3*x(1)+3*x(2)-7;
-10*x(1)+x(2)+8;
18*x(1)-4*x(2)+12;
x(1);
x(2)];
endfunction
```

Решение задачи минимум представлено на листинге 10.4.

```
function y=f(x)
y=(x(1)-3)^2+(x(2)-4)^2;
endfunction
function r = g (x)
r=[3*x(1)+3*x(2)-7;
-10*x(1)+x(2)+8;
18*x(1)-4*x(2)+12;
x(1);
x(2)];
endfunction
x0=[0;0];
[x, obj, info, iter]=sqp(x0,@f,[],@g)
```

Листинг 10.4

Результаты представлены ниже

```
>>>x =
2.0000
12.0000
obj = -281.00
info = 101
iter = 3
>>>maximum = 65.000
```

Минимум *3.2079* достигается в точке (*1.2178*, *4.1782*), значение `info=101` говорит о корректном решении задачи, для нахождения минимального значения потребовалось всего `5` итерации.

Теперь рассмотрим решение задачи на максимум. Функция `sqp` может искать только минимум. Поэтому вспомним, как задача на максимум сводится к задаче на минимум $max\, f(x)=-min\, f(-x)$.

Введём дополнительную функцию

```
function y=f1(x)
y=-f(-x);
endfunction
```

Полный текст решения задачи на максимум представлен на листинге 10.5.

```
function y=f(x)
y=(x(1)-3)^2+(x(2)-4)^2;
endfunction
function y=f1(x)
y=-f(-x);
endfunction
function r = g (x)
r=[3*x(1)+2*x(2)-7;
-10*x(1)+x(2)+8;
18*x(1)-4*x(2)+12;
x(1);
x(2)];
endfunction
x0=[0;0];
[x, obj, info, iter]=sqp(x0,@f1,[],@g)
maximum=f(x)
```

Листинг 10.5

Результаты работы программы можно увидеть ниже.

```
>>>x =
2.0000
12.0000
obj = -281.00
info = 101
iter = 3
>>>maximum = 65.000
```

Максимум достигается в точке (2,12), его величина равна 65.

ПРИМЕР 10.4.

План производства изделий трёх типов составляет 120 деталей(x_1 – количество изделий первого вида, x_2 – количество изделий второго вида, x_3 – количество изделий третьего вида). Изделия можно изготовить тремя способами. При первом технологическом способе производят изделия первого типа и затраты составляют $4x_1+x_1^2$. Второй технологический способ предназначен для производства изделий второго типа и затраты составляют $8x_2+x_2^2$. Третий способ позволяет производить изделия третьего типа и затраты в нём можно рассчитать по формуле x_3^2 . Определить, сколько изделий каждого типа надо изготовить, чтобы затраты были минимальными [1].

Сформулируем эту задачу, как задачу оптимизации. Найти минимум функции $f(x_{1,}x_2)=4x_1+x_1^2+8x_2+x_2^2+x_3+x_3^2$ при следующих ограничениях $x_1+x_2+x_3=120, x_1\geq 0, x_2\geq 0, x_3\geq 0$.

Функция цели f в Octave будет представлена следующим образом

```
function y=f(x)
y=4*x(1)+x(1)*x(1)+8*x(2)+x(2)*x(2)  +x(3)*x(3);
endfunction
```

Две функции ограничений $g(x)=0$ и $\varphi(x)\geq 0$ можно будет записать так.

```
function z=g(x)
z=x(1)+x(2)+x(3)-120;
endfunction
function u=fi(x)
fi=[x(1);x(2);x(3)];
```

```
endfunction
```

На листинге 10.6 представлена программа решения задачи 10.4.

```
function y=f(x)
y=4*x(1)+x(1)*x(1)+8*x(2)+x(2)*x(2)  +x(3)*x(3);
endfunction
function z=g(x)
z=x(1)+x(2)+x(3)-120;
endfunction
function u=fi(x)
fi=[x(1);x(2);x(3)];
endfunction
x0=[0;0;0];
[x, obj, info, iter]=sqp(x0,@f,@g)
```

Листинг 10.6

Результаты решения можно увидеть ниже.

```
>>>x =
40.000
38.000
42.000
obj = 5272.0
info = 101
iter = 8
```

Минимальные затраты составят 5272 денежных единицы, при этом будет произведено 40 изделий первого вида, 38 – второго и 42 – третьего. Для решения задачи было проведено 8 итераций.

ПРИМЕР 10.5.

Найти максимум функции $f=-x_1^2-x_2^2$ при ограничениях $(x_1-7)^2+(x_2-7)^2\le 18$, $x_1\ge 0$, $x_2\ge 0$ [1].

В этой задаче необходимо свести задачу на максимум к задаче на минимум, а также путём умножения на -1 заменить знак в неравенстве.

Текст программы решения задачи в Octave представлен на листинге

```
function y=f(x)
y=-x(1)*x(1)-x(2)*x(2) ;
endfunction
function y=f1(x)
y=-f(-x) ;
endfunction
function u=fi(x)
u=[-(x(1)-7)^2-(x(2)-7)^2+18;
x(1);
x(2)];
endfunction
x0=[0;0];
[xopt, obj, info, iter]=sqp(x0,@f1,[],@fi)
f(xopt)
```

Листинг 10.7.

Результаты решения можно увидеть ниже

```
>>>xopt =
4.0000
4.0000
obj=32.000
info=101
```

```
iter=8
>>>ans=-32.000
```

Функция достигает своего максимального значения -32 в точке (4,4).

Следующим классом оптимизационных задач, рассматриваемых в этой главе, будут задачи линейного программирования (ЗЛП).

10.2 Решение задач линейного программирования

Эти задачи встречаются во многих отраслях знаний. Алгоритмы их решения хорошо известны. Эти алгоритмы реализованы во многих, как проприетарных, так и свободных, математических пакетах. Не является исключением и Octave. Но перед тем, как рассмотреть решение задач линейного программирования в Octave, давайте вспомним, что такое задача линейного программирования.

10.2.1 Задача линейного программирования

Знакомство с задачами линейного программирования начнем на примере задачи об оптимальном рационе.

Задача об оптимальном рационе. Имеется четыре вида продуктов питания: *П1*, *П2*, *П3*, *П4*. Известна стоимость единицы каждого продукта c_1, c_2, c_3, c_4 . Из этих продуктов необходимо составить пищевой рацион, который должен содержать не менее b_1 единиц белков, не менее b_2 единиц углеводов, не менее b_3 единиц жиров. Причем известно, в единице продукта *П1* содержится a_{11} единиц белков, a_{12} единиц углеводов и a_{13} единиц жиров и т.д. (см. таблицу 10.1).

Таблица *10.1.* Содержимое белков, углеводов и жиров в продуктах

Элемент	белки	углеводы	жиры
П1	a_{11}	a_{12}	a_{11}
П2	a_{21}	a_{22}	a_{23}
П3	a_{31}	a_{32}	a_{33}
П4	a_{41}	a_{42}	a_{43}

Требуется составить пищевой рацион, чтобы обеспечить заданные условия при минимальной стоимости.

Пусть x_1, x_2, x_3, x_4 – количества продуктов *П1*, *П2*, *П3*, *П4*. Общая стоимость рациона равна

$$L = c_1 x_1 + c_2 x_2 + c_3 x_3 + c_4 x_4 = \sum_{i=1}^{4} c_i x_i \qquad (10.1)$$

Сформулируем ограничение на количество белков, углеводов и жиров в виде неравенств. В одной единице продукта *П1* содержится a_{11} единиц белков, в x_1 единицах - $a_{11} x_1$, в x_2 единицах продукта *П2* содержится $a_{21} x_2$ единиц белка и т.д. Следовательно общее количество белков во всех четырех типов продукта равно $\sum_{j=1}^{4} a_{j1} x_j$ и должно быть не больше b_1 . Получаем первое ограничение

$$a_{11}x_1+a_{21}x_2+a_{31}x_3+a_{41}x_4\le b_1 \qquad (10.2)$$

Аналогичные ограничения для жиров и углеводов имеют вид:

$$a_{12}x_1+a_{22}x_2+a_{32}x_3+a_{42}x_4\le b_2 \qquad (10.3)$$

$$a_{13}x_1+a_{23}x_2+a_{33}x_3+a_{43}x_4\le b_3 \qquad (10.4)$$

Принимаем во внимание, что x_1, x_2, x_3, x_4 положительные значения, получим еще четыре ограничения

$$x_1\ge 0, x_2\ge 0, x_3\ge 0, x_4\ge 0 \qquad (10.5)$$

Таким образом задачу о рациональном рационе можно сформулировать следующим образом: найти значения переменных x_1, x_2, x_3, x_4, удовлетворяющие системе ограничений (10.2) – (10.5), при которых линейная функция (10.1) принимала бы минимальное значение.

Задача об оптимальном рационе является задачей линейного программирования, функция (10.1) называется функцией цели, а ограничения (10.2) – (10.5) системой ограничений задачи линейного программирования.

В задачах линейного программирования функция цели функция цели *L* и система ограничений являются линейными.

В общем случае задачу *линейного программирования* можно сформулировать следующим образом. Найти такие положительные значения $x_1, x_2, \ldots, x_n$, при которых *функция цели L* (10.6) достигает своего минимального значения и удовлетворяет системе линейных ограничений (10.7).

$$L=c_1x_1+c_2x_2+\ldots+c_nx_n=\sum_{i=1}^{n} c_ix_i \qquad (10.6)$$

$$\sum_{j=1}^{n} a_{ij}x_j\le b_i, i=1,\ldots m. \qquad (10.7)$$

Если в задачу линейного программирования добавляется ограничение целочисленности значений *x*, то мы получаем задачу *целочисленного программирования*.

Octave позволяет решать задачи линейной оптимизации с ограничениями в более общей формулировке.

Найти такие положительные значения $x_1, x_2, \ldots, x_n$, при которых *функция цели L* (10.6) достигает своего минимального (максимального) значения и удовлетворяет системе линейных ограничений. Система ограничений может быть представлена неравенствами (10.8) или (10.9). При этом значения *x* могут быть, как вещественными, так и целочисленными, как положительными, так и отрицательными.

$$\sum_{j=1}^{n} a_{ij}x_j\le b_i, \qquad (10.8)$$

$$\sum_{j=1}^{n} a_{ij} x_j \geq b_i, \qquad (10.9)$$

$$i = 1, \ldots, m.$$

Рассмотрим решение задач линейного программирования в Octave.

10.2.2 Решение задач линейного программирования в Octave

Для решения задач линейного программирования в Octave существует функция

```
[xopt, fmin, status, extra] = glpk (c, a, b, lb, ub, ctype, vartype, sense, param)
```

Здесь

`c` – вектор-столбец, включающий в себя коэффициенты при неизвестных функции цели, размерность вектора `c` равна количеству неизвестных `n` в задаче линейного программирования;

`a` – матрица при неизвестных из левой части системы ограничений, количество строк матрицы равно количеству ограничений `m`, а количество столбцов совпадает с количеством неизвестных `n`;

`b` – вектор-столбец содержит свободные члены системы ограничений, размерность вектора равна количеству ограничений *m*.

`lb` – вектор-столбец размерности `n`, содержащий верхнюю систему ограничений (**x**>**lb**), по умолчанию `lb` – вектор столбец, состоящий из нулей;

`ub` – вектор-столбец размерности `n`, содержащий нижнюю систему ограничений (**x<ub**), по умолчанию верхняя система ограничений отсутствует, подразумевается, что все знаения вектора `ub` равны +∞;

`ctype` – массив символов размерности `n`, определяющий тип ограничения (например, (10.7) или (10.9)), элементы этого вектора могут принимать одно из следующих значений:

`"F"` – ограничение будет проигнорировано,

`"U"` – ограничение с верхней границей (`(A(i,:)*x <= b(i))`),

`"S"` – ограничение в виде равенства (`(A(i,:)*x = b(i))`),

`"L"` – ограничение с верхней границей (`(A(i,:)*x >= b(i))`),

`"D"` – двойное ограничение (`(A(i,:)*x <= b(i)` и `(A(i,:)*x >= b(i))`;

`vartype` – массив символов размерности `n`, который определяет тип переменной x_i

"C" – вещественная переменная, "I" – целочисленная переменная;

`sense` – значение, определяющее тип задачи оптимизации:

- 1 – задача минимизации,
- -1 – задача максимизации;

`param` – структура, определяющая параметры оптимизационных алгоритмов, при обращении к функции `glpk`.

Во многих случаях достаточно значений структуры `param` по умолчанию, в этом случае последний параметр в функции `glpk` можно не указывать. Подробное описание структуры `param` выходит за рамки книги, в случае необходимости его использования авторы рекомендуют обратиться ко встроенной справке Octave.

Функция `glpk` возвращает следующие значения:

`xopt` – массив значений x, при котором функция цели `L` принимает оптимальное значение;

`fmin` – оптимальное значение функции цели;

`status` – переменная, определяющая как решена задача оптимизация, при

`status=180`, решение найдено и задача оптимизации решена полностью[1];

`extra` – структура, включающая следующие поля:

`lambda` – множители Лагранжа;

`time` – время в секундах, затраченное на решение задачи;

`mem` – память в байтах, которая была использована, при решении задачи (значение недоступно, если была использована библиотека линейного программирования GLPK 4.15[2] и выше).

Рассмотрим несколько примеров решения задач линейного программирования.

ПРИМЕР 10.6.

Найти такие значения переменных x_1, x_2, x_3, x_4 при которых функция цели L $L=-x_2-2x_3+4x_4$ достигает своего минимального значения и удовлетворяются ограничения:

$$\begin{aligned} &3x_1-x_2\le 2\\ &x_2-2x_3\le -1\\ &4x_3-x_4\le 3\\ &5x_1+x_4\ge 6\\ &x_1\ge 0, x_2\ge 0, x_3\ge 0, x_4\ge 0 \end{aligned} \quad .$$

Сформируем параметры функции `gplk`

$c=\begin{bmatrix} 0\\ -1\\ -2\\ 4 \end{bmatrix}$ – коэффициенты при неизвестных функции цели,

$a=\begin{bmatrix} 3 & -1 & 0 & 0\\ 0 & 1 & -2 & 0\\ 0 & 0 & 4 & -1\\ 5 & 0 & 0 & 1 \end{bmatrix}$ – матрица системы ограничений,

$b=\begin{bmatrix} 2\\ -1\\ 3\\ 6 \end{bmatrix}$ – свободные члены системы ограничений,

ctype="UUUL" – массив символов, определяющий тип ограничения[3],

vartype="CCCC" – массив, определяющий тип переменной, в данном случае все переменные вещественные,

sense=1 – задача на минимум.

Текст программы решения задачи приведен на листинге 10.8.

```
c=[0;-1;-2;4];
a=[3 -1 0 0; 0 1 -2 0;0 0 4 -1;5 0 0 1];
b=[2; -1; 3; 6];
ctype="UUUL";
vartype="CCCC";
sense=1;
[xmin, fmin, status] =glpk (c, a, b, [0 ;0 ;0; 0], [], ctype, vartype, sense)
```

1 Если $status \neq 180$, то полученному решению доверять нельзя, подробнее о значениях переменной status в этом случае можно прочитать в справке.

2 В последней на момент написания книги версии octave использовалась библиотека GLPK версии 4.38.

3 Первые три ограничения типа «меньше», четвертое – типа «больше»

Листинг 10.8

Результаты решения задачи представлены ниже.

```
>>>xmin =
1.00000
1.00000
1.00000
1.00000
fmin = 1.00000
status = 180
```

Минимальное значение $L=1$ достигается при $x=\begin{bmatrix}1\\1\\1\\1\end{bmatrix}$. Значение переменной *status* равно 180, что свидетельствует о корректном решении задачи линейного программирования.

ПРИМЕР 10.7.

Найти такие значения переменных x_1, x_2, x_3 при которых функция цели L $L=-5+x_1-x_2-3x_3$ достигает своего минимального значения и удовлетворяются ограничения:

$$\begin{aligned} &x_1+x_2\geq 2\\ &x_1-x_2\leq 0\\ &x_1+x_3\geq 2\\ &x_1+x_2-x_3\leq 3\\ &x_1\geq 0, x_2\geq 0, x_3\geq 0 \end{aligned} \quad .$$

Сформируем параметры функции `gplk`

$c=\begin{bmatrix}1\\-1\\-3\end{bmatrix}$ – коэффициенты при неизвестных функции цели,

$a=\begin{bmatrix}1 & 1 & 0\\1 & -1 & 0\\1 & 0 & 1\\1 & 1 & -1\end{bmatrix}$ – матрица системы ограничений (три переменных и четыре ограничения),

$b=\begin{bmatrix}2\\0\\2\\3\end{bmatrix}$ – свободные члены системы ограничений,

ctype="LULU" – массив символов, определяющий тип ограничения[4],

vartype="CCC" – массив, определяющий тип переменной, в данном случае все переменные вещественные,

sense=-1 – задача на максимум.

Текст программы решения задачи приведен на листинге .

```
c=[1;-1;-3];
a=[1 1 0; 1 -1 0;1 0 1;1 1 -1];
b=[2; 0;2; 3];
ctype="LULU";
vartype="CCC";
sense=-1;
```

4 Первое и третье ограничения типа «больше», второе четвертое – типа «меньше».

```
[xmax,fmax,status]=glpk(c, a, b, [],[], ctype, vartype, sense)
```

Листинг 10.9

Результаты решения задачи представлены ниже.

```
>>>xmax =
1.66667
1.66667
0.33333
fmax = -1.0000
status = 180
```

Минимальное значение $L=-6$[5] достигается при $x=\begin{bmatrix}1.66667\\1.66667\\0.33333\end{bmatrix}$. Значение переменной *status* равно 180, что свидетельствует о корректном решении задачи линейного программирования.

Решим задачу 10.9, как задачу *целочисленного программирования* (см. листинг 10.10)

```
c=[1;-1;-3];
a=[1 1 0; 1 -1 0;1 0 1;1 1 -1];
b=[2; 0;2; 3];
ctype="LULU";
vartype="III";
sense=-1;
[xmax,fmax,status]=glpk(c, a, b, [],[], ctype, vartype, sense)
```

Листинг 10.10

Результаты решения задачи целочисленного программирования:

```
>>>xmin =
2
2
1
fmin = -3
status = 171
```

Значение *status=171* свидетельствует о корректном решении задачи целочисленного программирования, значение $L=-8$ достигается при $x=\begin{bmatrix}2\\2\\1\end{bmatrix}$.

ПРИМЕР 10.8.

Туристическая фирма заключила контракт с двумя турбазами на одном из черноморских курортов, рассчитанных соответственно, на 195 и 165 человек. Туристам для посещения предлагается дельфинарий в городе, ботанический сад и походы в горы.

Составить маршрут движения туристов так, чтобы это обошлось возможно дешевле, если дельфинарий принимает в день 90 организованных туристов, ботанический сад – 170, а в горы в один день могут пойти 105 человек.

Стоимость одного посещения выражается таблицей 10.2.

Таблица *10.2.*

Турбаза	Дельфинарий	Ботанический сад	Поход в горы
1	5	9	20

5 Авторы обращают внимание читателей в Octave fmin было равно -1, а потом из него необходимо было вычесть -5

2	10	12	24

Для решения задачи введем следующие обозначения:

x1 – число туристов первой турбазы, посещающих дельфинарий;

x2 – число туристов первой турбазы, посещающих ботанический сад;

x3 – число туристов первой турбазы, отправляющихся в поход;

x4 – число туристов второй турбазы, посещающих дельфинарий;

x5 – число туристов второй турбазы, посещающих ботанический сад;

x6 – число туристов второй турбазы, отправляющихся в поход.

Составим функцию цели, заключающуюся минимизации стоимости мероприятий фирмы:

$Z=5x1+9x2+20x3+10x4+12x5+24x6$.

Руководствуясь условием задачи, определим ограничения:

$x1 + x4 \le 90$;

$x2 + x5 \le 170$;

$x3 + x6 \le 105$;

$x1 + x2 + x3 = 195$;

$x4 + x5 + x6 = 165$.

Кроме того, количество туристов, участвующих в мероприятиях не может быть отрицательным: $x1 \ge 0$, $x2 \ge 0$, $x3 \ge 0$, $x4 \ge 0$, $x5 \ge 0$, $x6 \ge 0$.

Кроме того, необходимо помнить, что это задача целочисленного программирования (количество туристов — число целое!!!).

В массиве *x* будут хранится значения *x1*, *x2*, *x3*, *x4*, *x5* и *x6*.

Сформируем параметры функции `gplk`

$c=\begin{bmatrix}15\\9\\20\\10\\12\\24\end{bmatrix}$ – коэффициенты при неизвестных функции цели,

$a=\begin{bmatrix}1&0&0&1&0&0\\0&1&0&0&1&0\\0&0&1&0&0&1\\1&1&1&0&0&0\\0&0&0&1&1&1\end{bmatrix}$ – матрица системы ограничений (шесть переменных и пять ограничений),

$b=\begin{bmatrix}90\\170\\105\\195\\165\end{bmatrix}$ – свободные члены системы ограничений,

ctype="UUUSS" – массив символов, определяющий тип ограничения[6],

vartype="IIIII" – массив, определяющий тип переменной, в данном случае все переменные целые (задача целочисленного программирования),

sense=1 – задача на минимум.

Решение задачи представлено в листинге 10.11.

```
c=[5;9;20;10;12;24];
a=[1 0 0 1 0 0; 0 1 0 0 1 0;0 0 1 0 0 1;1 1 1 0 0 0 ;0 0 0 1 1
```

6 Первые три ограничения типа «меньше», четвертое и пятое – типа «равно».

```
1];
    b=[90; 170; 105; 195; 165];
    ctype="UUUSS";
    vartype="IIIIII";
    sense=1;
    [xmin, fmin, status]=glpk(c,a,b,[], [], ctype, vartype, sense)
```

Листинг 10.11

Результаты решения задачи 10.8 представлены ниже.

```
>>>xmin =
90
5
100
0
165
0
fmin = 4475
status = 171
```

Значение переменной *status=171* свидетельствует о корректном решении задачи целочисленного программирования.

В результате получилось следующее решение:

90 туристов первой турбазы посетят дельфинарий, 5 туристов первой турбазы и все 165 второй турбазы поедут в ботанический сад, *100* туристов первой турбазы отправятся в поход. Стоимость мероприятия составит 4475.

ПРИМЕР 10.9.

Для изготовления трех видов изделий (А, Б, В) используется токарное, фрезерное, шлифовальное и сварочное оборудование. Затраты времени на обработку одного изделия каждого типа представлены в таблице 10.3. Общий фонд рабочего времени каждого вида оборудования и прибыль от реализации изделий каждого типа представлены этой же таблице. Составить план выпуска изделий для достижения максимальной прибыли [1].

Таблица *10.3.*

Тип оборудования	Затраты времени на обработку одного изделия вида (станко-ч)			Общий фонд времени работы оборудования (ч)
	А	Б	В	
Фрезерное	2	4	5	120
Токарное	1	8	6	280
Сварочное	7	4	5	240
Шлифовальное	4	6	7	360
Прибыль (тыс. грн)	10	14	12	

Пусть x_1 – количество изделий вида А, x_2 – количество изделий вида Б, x_3 – вида В.

Прибыль от реализации всех изделий составляет

$$L = 10x_1 + 14x_2 + 12x_3 \qquad (10.10)$$

Общий фонд рабочего времени фрезерного оборудования составляет $2x_1 + 4x_2 + 5x_3$. Эта величина не должна превышать 120 часов.

$$2x_1+4x_2+5x_3\le 120 \qquad (10.11)$$

Запишем аналогичные ограничения для фонда рабочего времени токарного, сварочного, шлифовального оборудования

$$\begin{cases} x_1+8x_2+6x_3\le 280 \\ 7x_1+4x_2+5x_3<240 \\ 4x_1+6x_2+7x_3\le 360 \end{cases} \qquad (10.12)$$

Таким образом следующую задачу линейного программирования.

Найти такие положительные значения x_1 , x_2 , x_3 при которых функция цели L (10.10) достигает максимального значения и выполняются ограничения (10.11)-(10.12).

Теперь решим эту задачу в Octave с помощью функции `gplk`.

Сформируем параметры функции `gplk`

$c=\begin{bmatrix}10\\14\\12\end{bmatrix}$ – коэффициенты при неизвестных функции цели, $a=\begin{bmatrix}2&4&5\\1&8&6\\7&4&5\\4&6&7\end{bmatrix}$ – матрица системы ограничений (три переменных и четыре ограничений), $b=\begin{bmatrix}120\\280\\240\\360\end{bmatrix}$ – свободные члены системы ограничений, *ctype="UUUГ"* – массив символов, определяющий тип ограничения[7], *vartype="III"* – массив, определяющий тип переменной, в данном случае все переменные целые, *sense=-1* – задача на максимум.

Решение задачи в Octave представлено на листинге

```
c=[10;14;12];
a=[2 4 5; 1 8 6;7 4 5;4 6 7];
b=[120; 280; 240; 360];
ctype="UUUU";
vartype="III";
sense=-1;
[xmax, fmax, status] =glpk (c, a, b, [0 ;0 ;0], [], ctype, vartype, sense)
```

Листинг 10.12

Результаты представлены ниже

```
>>>xmax =
24
18
0
fmax = 492
status = 171
```

Таким образом для получения максимальной прибыли (*fmax=492*) необходимо произвести 24 единицы изделия типа А и 18 единиц изделия типа Б. Значение параметра *status=171* говорит о корректности решения задачи линейного программирования.

ПРИМЕР 10.10.

Для изготовления четырёх видов изделий используется токарное, фрезерное, сверлильное, расточное шлифовального оборудование, а также комплектующие

7 Все четыре ограничения типа «меньше».

изделия. Сборка изделий требует сборочно-наладочных работ. В таблице 10.4 представлены: нормы затрат ресурсов на изготовление различных изделий, наличие каждого из ресурсов, прибыль от реализации одного изделия, ограничения на выпуск изделий второго и третьего типа [1]. Сформировать план выпуска продукции для достижения максимальной прибыли.

Таблица *10.4*.

Ресурсы	Нормы затрат на одно изделие				Общий объём ресурсов
	1	2	3	4	
Производительность оборудования (человеко-ч)					
токарного	550		620		64270
фрезерного	40	30	20	20	4800
сверлильного	86	110	150	52	22360
расточного	160	92	158	128	26240
шлифовального		158	30	50	7900
Комплектующие изделия (шт.)	3	4	3	3	520
Сборочно-наладочные работы (человеко-ч)	4.5	4.5	4.5	4.5	720
Прибыль от реализации одного изделия (тыс. руб.)	315	278	573	370	
Выпуск					
минимальный		40			
максимальный			120		

Пусть x_1 – количество изделий первого вида, x_2 – количество изделий второго вида, x_3 и x_4 – количество изделий третьего и четвёртого вида соответственно.

Тогда прибыль от реализации всех изделий вычисляется по формуле

$$L = 315x_1 + 278x_2 + 573x_3 + 370x_4 \quad . \tag{10.13}$$

Ограничения на фонд рабочего времени формируют следующие ограничения

$$\begin{cases} 550x_1 + 620x_3 \le 64270 \\ 30x_1 + 30x_2 + 20x_3 + 20x_4 \le 4800 \\ 86x_1 + 110x_2 + 150x_3 + 52x_4 \le 22360 \\ 160x_1 + 92x_2 + 158x_3 + 128x_4 \le 26240 \\ 158x_2 + 30x_3 + 50x_4 \le 7900 \end{cases} . \tag{10.14}$$

Ограничение на возможное использование комплектующих изделий

$$3x_1+4x_2+3x_3+3x_4\le 520 \quad . \qquad (10.15)$$

Ограничение на выполнение сборочно-наладочных работ

$$4.5x_1+4.5x_2+4.5x_3+4.5x_4\le 720 \quad . \qquad (10.16)$$

Ограничения на возможный выпуск изделий каждого вида

$$x2\ge 40, x3\le 120, x1\ge 0, x3\ge 0, x4\ge 0 \quad . \qquad (10.17)$$

Сформулируем задачу линейного программирования.

Найти значения x_1 , x_2 , x_3 и x_4 при которых функция цели L (10.13) достигает своего максимального значения и выполняются ограничения (10.14)-(10.17).

Рассматриваемая задача из широко известной книги [1] была интересна авторам в связи с тем, что еще 25 лет назад для решения задач подобной сложности использовали большие ЭВМ и специализированные пакеты решения оптимизационных задач. На подготовку данные и решение её затрачивался не один час. Мы же попробуем решить её в Octave и посмотрим сколько времени у нас на это уйдёт.

Сформируем параметры функции `gplk`.

$c=\begin{bmatrix}315\\278\\573\\370\end{bmatrix}$ – коэффициенты при неизвестных функции цели,

$a=\begin{pmatrix}550 & 0 & 620 & 0\\40 & 30 & 20 & 20\\86 & 110 & 150 & 52\\160 & 92 & 158 & 128\\0 & 158 & 30 & 50\\3 & 4 & 3 & 3\\4.5 & 4.5 & 4.5 & 4.5\\0 & 1 & 0 & 0\\0 & 0 & 1 & 0\\1 & 0 & 0 & 0\\0 & 0 & 1 & 0\\0 & 0 & 0 & 1\end{pmatrix}$ – матрица системы ограничений (четыре переменных и

двенадцать ограничений), $b=\begin{pmatrix}64270\\4800\\22360\\26240\\7900\\520\\720\\40\\120\\0\\0\\0\end{pmatrix}$ – свободные члены системы ограничений,

ctype="UUUUUUULULLL" – массив символов, определяющий тип ограничения[8],

vartype="IIII" – массив, определяющий тип переменной, в данном случае все переменные целые (задача целочисленного программирования),

sense=-1 – задача на максимум.

Программа решения задачи в Octave представлена в листинге 10.13.

```
c=[315; 278 ;573 ;370];
a=[550 0 620 0; 40 30 20 20; 86 110 150 52; 160 92 158 128; 0
158 30 50; 3 4 3 3; 4.5 4.5 4.5 4.5; 0 1 0 0; 0 0 1 0; 1 0 0 0; 0
0 1 0; 0 0 0 1];
b=[64270; 4800; 22360; 26240; 7900; 520; 720; 40; 120; 0; 0;
0];
ctype="UUUUUUULULLL";
vartype="IIII";
sense=-1;
[xmax, fmax, status]=glpk(c,a,b,[], [], ctype, vartype, sense)
```

Листинг 10.13

Результаты решения представлены ниже.

```
>>> [xmax, fmax, status]=glpk(c,a,b,[], [], ctype, vartype, sense)
xmax =
65
40
46
4
fmax = 59433
status = 171
```

Для получения максимальной прибыли (*fmax=492*) необходимо произвести 65 единиц изделий первого типа, 40 – второго, 46 – третьего и 4 – четвёртого. Значение параметра *status=171* говорит о корректности решения задачи линейного программирования.

Для написания программы и решения довольно сложной задачи в Octave понадобилось буквально пару минут.

Подобным образом можно решать всевозможные задачи линейного программирования. Кроме Octave, для решения задач линейного программирования авторы использовали электронные таблицы OpenOffice.org Calc, MS Office Excel, математические программы MathCad, Matlab, Maple, Mathematica, Scilab. На наш взгляд, именно Octave, обладает самой мощной гибкой и мощной функцией `gplk` для решения задач линейного программирования из всех свободных и проприетарных программ.

Рассмотренных двух функций (`gplk` и `sqp`) достаточно для решения очень многих

8 Первые три ограничения типа «меньше», четвертое и пятое – типа «равно».

оптимизационных задач. Если читателю встретятся оптимизационные задачи, которые невозможно решить с помощью `gplk` и `sqp`, то авторы рекомендуют ему обратиться к пакету расширений *Minimization* для GNU Octave. Краткое описание функций этого пакета на английском языке с некоторыми примерами приведено на странице http://octave.sourceforge.net/optim/overview.html.