9. Оптимизация линейных систем

Библиотека “simplex” – предназначена для оптимизации линейных систем с использованием симплексного алгоритма. Особенность ее в том, что имеется возможность выполнять оценки промежуточных этапов симплексного алгоритма, например, определять базисные переменные и т.п.

После подключения библиотеки командой with(simplex) пользователю становятся доступны функции и опции, указанные в следующей таблице.

ДОСТУПНЫЕ ФУНКЦИИ	НАЗНАЧЕНИЕ
basis	Находит базисные переменные
convexhull	Построение выпуклой оболочки
cterm	Выводит список элементов вектора ресурсов
define_zero	Устанавливает абсолютное значение погрешности вычисления
display	Представляет систему в матричной форме
dual	Преобразует данную задачу в двойственную задачу линейного программирования
equality	Преобразует неравенства системы в равенства
feasible	Возвращает true - если решение существует, и false - если нет
maximize	Находит максимум целевой функции
minimize	Находит минимум целевой функции
NONNEGATIVE	Опция: указание на условие неотрицательности всех переменных
pivot	Создает новую систему уравнений, позволяющую найти опорный план (опорное решение)
pivoteqn	Возвращает список уравнений, задающий опорный план
pivotvar	Возвращает переменную с положительным коэффициентом
ratio	Для определения переменной, исключаемой из опорного плана
setup	Приводит систему ограничений к стандартной форме
standardize	Превращает систему ограничений в пары неравенств

Допускается как сокращенная, так и полная форма вызова команд. Последняя используется, если в рабочем документе имеются совпадающие имена разных команд.

Известно, что симплекс–метод применяется для нахождения максимума или минимума линейной целевой функции при наличии некоторых линейных ограничений.

Поясним сказанное на следующем примере. Пусть необходимо максимизировать целевую функцию f (x1, x2)= 3*x1 + 2*x2 при наличии следующей системы ограничений

-x1+2*x2<=4,
3*x1+2*x2<=14,
x1-x2<=3,
x1>=0, x2>=0.

Тогда значения переменных, доставляющих максимум целевой функции находятся достаточно простыми командами:

```
> with(simplex):
> maximize( 3*x1+2*x2, {-x1+2*x2<=4,
  3*x1+2*x2<=14, x1-x2<=3, x1>=0,  x2>=0, x1>=0,
  x2>=0});
Warning, new definition for maximize
Warning, new definition for minimize
```

$$\{x1 = \frac{5}{2}, x2 = \frac{13}{4}\}$$

Условие неотрицательности переменных удобно указать опцией NONNEGATIVE, например, вышеприведенная команда будет выглядеть так:

```
> Z:=3*x1+2*x2:
> Ogran:={-x1+2*x2<=4,3*x1+2*x2<=14,x1-x2<=3,
  x1>=0, x2>=0}:
> maximize(Z, Ogran, NONNEGATIVE );
```

$$\{x1 = \frac{5}{2}, x2 = \frac{13}{4}\}$$

Обычно, задача линейного программирования с **m** ограничениями и **n** переменными приводится к стандартной или канонической форме, которая имеет следующий вид:

Максимизировать или минимизировать $\mathbf{Z=c_1x_1+c_2x_2+...+c_nx_n}$ при ограничениях.

$a_{11}x_1+a_{12}x_2+\ldots+a_{1n}x_n = b_{1,}$
$a_{21}x_1+a_{22}x_2+\ldots+a_{2n}x_n = b_{2,}$
• •
• •
$a_{m1}x_1+a_{m2}x_2+\ldots+a_{mn}x_n = b_{m,}$
$x_1 \geq 0 \quad x_2 \geq 0 \ldots \quad x_n \geq 0,$
$b_1 \geq 0 \quad b_2 \geq 0 \ldots \quad bx_m \geq 0$

Это же в матричной форме можно записать более компактно.
Минимизировать или максимизировать **Z=Cx** при ограничениях
$Ax=b, \ \ x \geq 0, \ b \geq 0,$
где **A** – матрица размерности **m∗n; b** – вектор столбец размерности **m∗1;**
x – вектор столбец размерности **n∗1; C** – вектор-строка размерности **1∗m;**

Обычно **A** называется матрицей коэффициентов, **x** – вектором переменных, **b** – вектором ресурсов, **C** – вектором оценок задачи линейного программирования.

Для получения информации о векторе оценок достаточно подать команду cterm(C). Например, для вышеописанной системы под идентификатором Ogran:

```
> cterm(Ogran);
```

$$\{0, 3, 4, 14\}$$

Ограничения в виде неравенств можно преобразовать в равенства при помощи введения так называемых искусственных переменных, что на практике, как правило, вызывает трудности. Это же в Maple выполняется простой командой setup. Для вышеприведенной задачи она вводит три дополнительные переменные: _SL1, _SL2 и _SL3.

```
> dp:= setup({-x1+2*x2<=4,3*x1+2*x2<=14,x1-
  x2<=3});
```

$$dp := \{\_SL1 = 4 + x1 - 2\,x2,\ \_SL2 = 14 - 3\,x1 - 2\,x2,\ \_SL3 = 3 - x1 + x2\}$$

Используя стандартную форму представления ограничений и команду basis, можно получить список базисных переменных, применяемых в алгоритме симплекс–метода. Подчеркнем еще раз, что множество ограничений, для которого находится базис, должно быть представлено в специальной форме, выдаваемой функцией simplex[setup].

Функция simplex[setup] гарантирует, что каждая из этих переменных находится только в одном уравнении, и только в его левой части.

```
> basis(dp);
```

$$\{ \_SL1 , \_SL2 , \_SL3 \}$$

Команда convert перемещает все константы в правую часть уравнения (неравенства) для каждого ограничения, например:

```
> convert( {4>=-x1+2*x2, 3*x1+2*x2<=14, x1-
  x2<=3}, std );
```

$$\{-x1 + 2\,x2 \le 4, 3\,x1 + 2\,x2 \le 14, x1 - x2 \le 3\}$$

Функция standardize(C) возвращает набор или список неравенств формы <=, эквивалентный исходному множеству ограничений, со всеми константами в правых частях. Эта функция отличается от convert(C,std) тем, что уравнения превращаются в пары неравенств. Эта функция эквивалентна convert(C,stdle).

Функция convexhull(ps) определяет выпуклую область, прилегающую к данным точкам. Каждая точка в **ps** - это список (или множество) **x** и **y** координат в виде чисел. Результат, возвращаемый convexhull, есть список точек в порядке положительно ориентированных (против часовой стрелки). Например:

```
> convexhull( {[-1,0], [0,0], [1,1], [2,0],
  [1,0], [1,1/2],[1,-1]});
```

$$[[-1, 0], [1, -1], [2, 0], [1, 1]]$$

Функция define_zero(err) устанавливает абсолютный размер минимальной ошибки. По умолчанию она равна значению, определенному константой Digits.

Функция display(C) отображает на дисплее линейную систему в матричной форме. Возможен и такой формат вызова: display(C,[x, y, z])

Функция display(C) создает матричное уравнение **Ax rel B** для ограничений, определяющих линейную систему. Множество линейных уравнений C, отображаемых на дисплее, должно быть записано в специальной форме, производимой функцией simplex[setup].

Функция dual(f, C, y) преобразует данную задачу в двойственную задачу линейного программирования. Выражение **f** - целевая линейная функция; **C -** система, записанная в специальной форме, получаемой по команде simplex[convert/stdle]. Переменная **y** используется, чтобы создать имена **y1**, **y2**,... для двойственных переменных. Результат выполнения dual возвращается как последовательность в виде: цель, ограничения. Приведем пример:

```
> dual( 3*x1+2*x2, {-x1+2*x2 <= 4,
  3*x1+2*x2<=14, x1-x2<=3}, z);
```

$$4\,z1 + 14\,z2 + 3\,z3,\ \{2 \le 2\,z1 + 2\,z2 - z3,\ 3 \le -z1 + 3\,z2 + z3\}$$

Функция convert(**s**, equality) преобразует список или множество неравенств **s** в равенства. Заметим, что область допустимых значений, описываемая исходным множеством отношений, не сохраняется этой функцией, т.е. решение задачи будет находиться на границе области. Приведем пример:

```
> convert( {3*x+4*y <= 4, 4*x+3*y <= 5},
  equality);
```

$$\{3\,x + 4\,y = 4,\ 4\,x + 3\,y = 5\}$$

Функция simplex[feasible] – возвращает **true**, если решение линейной системы **C** существует, и **false** - в противном случае. Она имеет несколько форматов вызова:

feasible(C)

feasible(C, vartype)

feasible(C, vartype, ‘NewC’, ‘Transform’)

где C - множество линейных ограничений; vartype - опция NONNEGATIVE или UNRESTRICTED; NewC и Transform - имена.

Последние два аргумента используются, чтобы возвратить в виде множества конечную систему, обнаруженную feasible и любые преобразованные переменные. Новая система может иметь глобальные искусственные переменные (как например, _AR или _SL1).

```
> feasible({-x1+2*x2<=4, 3*x1+2*x2<=14, x1-
  x2<=3}, NONNEGATIVE);
```

$$true$$

Функция simplex[maximize] находит максимальное значение линейной системы и может иметь следующие форматы вызова:

maximize(f, C)

maximize(f , C, vartype)

maximize(f , C, vartype, ‘NewC’, ‘transform’) где **f** - целевая функция; **C** - множество или список линейных ограничений; vartype - опция NONNEGATIVE или UNRESTRICTED; NewC и transform - имена.

Функция maximize возвращает множество уравнений, описывающих оптимальное решение заданной линейной системы, или пустое множество в случае, если для ограничений **C** решения не существует, или NULL в случае, когда решение неограниченное. Уравнения, возвращаемые maximize, могут заменяться снова в функции f, чтобы получить величину оптимального решения.

Можно определять третий параметр, ограничивая переменные условием неотрицательности (NONNEGATIVE). Аналогично, UNRESTRICTED указывает, что никакое ограничение знака не должно устанавливаться для переменных.

Четвертый и пятый параметры можно включать для задания имен, в которые возвращается результаты преобразований. Например:

```
> maximize(x+y, {5*x+4*y <= 6, 2*x+4*y<= 3});
```

$$\{y = \frac{1}{4}, x = 1\}$$

```
> maximize(x-y, {2*x+5*y<= 4, 8*x+2*y<=-1});

> maximize(x-y, {2*x+5*y<= 4, 8*x+2*y<= -1},
  NONNEGATIVE);
```

$$\{\ \}$$

Функция simplex[minimize] находит минимальное значение линейной системы и имеет три формата вызова, аналогичные форматам, описанным выше для функции simplex[maximize].

Функция pivot(C, x, eqn) создает новую систему уравнений, позволяющую найти опорный план (опорное решение). Здесь C – набор или список уравнений; x – имя переменной; eqn – уравнение или список уравнений.

Вызываемая функция pivot(C,x,eqn) создает новую систему уравнений в форме, аналогичной формам, используемым в [setup], разрешая заданное уравнение относительно X, заменяя результат в C. Базис (C,x,eqn) создает новый комплект уравнений в форме совместимой с формами, использованными в simplex[setup], решая определенное уравнение eqn для x, затем заменяя результат в C. Это эквивалентно стандартному нахождению опорного плана в массиве коэффициентов.

Целевая функция f в новом описании заменяется функцией, использующей новый опорный план.

Приведем пример:

```
> pivot({_SL1 = 5-4*x-3*y, _SL2 = 4-3*x-4*y}, x,
  [_SL1 = 5-4*x-3*y]);
```

$$\{x=-\frac{1}{4}\_SL1+\frac{5}{4}-\frac{3}{4}y,\ \_SL2=\frac{1}{4}+\frac{3}{4}\_SL1-\frac{7}{4}y\}$$

Функция pivoteqn(C, var) возвращает список уравнений, задающий опорный план. Здесь **C** - система линейных уравнений; **var** - выбранная переменная для нахождения опорного плана.

Каждое возвращенное уравнение **eq** имеет минимальный неотрицательный коэффициент. В том случае, когда все коэффициенты неотрицательны, функция возвращает Fail. Линейные уравнения **C** записываются в форме, задаваемой simplex[setup]. Например;

```
> pivoteqn({_SL1 = 5-4*x-3*y, _SL2 = 4-3*x-4*y},
  x);
```

$$[\_SL1=5-4x-3y]$$

Функция pivotvar(f, List) возвращает переменную с положительным коэффициентом или значение Fail. Здесь f – целевая функция, выражается через свободные переменные; List – список переменных, которые имеются в задаче. Список переменных используется, чтобы определить порядок, в котором переменные должны быть проверены. Если список не определен, то pivotvar выбирает свой порядок (симплекс–метод работает, пока имеются отрицательные значения коэффициентов). Например:

```
> pivotvar(x1 + 3*x3 - x4);
```

$$x1$$

```
> pivotvar(x1 + 3*x3 - x4 , [x4,x3,x1]);
```

$$x3$$

Функция ratio(C, x) возвращает список коэффициентов матрицы **A**, которые помогают определять, какое уравнение в **C** имеет наиболее значительное ограничение для того, чтобы построить следующий опорный план симплекс–метода; **x** - переменная, которая должна использоваться в расчете коэффициентов. Иными словами, функция ratio(C, x) используется для исключения из опорного плана переменной с минимальным положительным коэффициентом. Все отрицательные коэффициенты выдаются как бесконечность, аналогично как и те, у которых значение coeff(eq, x, 1)=0. Уравнения **C** записывается в форме, задаваемой simplex[setup]. Например:

```
> ratio([_SL1 = 5-4*x-3*y, _SL2 = 4-3*x-4*y],
  x);
```

$$\left[\frac{5}{4}, \frac{4}{3}\right]$$

Функция simplex[setup] создает множество уравнений с единственной переменной в левой части и имеет следующие форматы вызова:

setup(C)

setup(C, NONNEGATIVE)

setup(C, NONNEGATIVE, ‘t’)

где C - множество линейных уравнений; ‘t’ - имя.

В созданной системе уравнений переменные в левой части являются базисными для соответствующей линейной системы и не содержатся в правой части любого уравнения. Дополнительные переменные вида _SLi вводятся для того, чтобы иметь дело с неравенством. Получающаяся система эквивалентна исходной системе C, но при этом решения новой системы могут быть преобразованы в решение исходной системы.

Если присутствует параметр NONNEGATIVE, все переменные считаются неотрицательными. Если имеется третий параметр , то значение для данной переменной t может быть неограниченным.

Пример:

```
> setup({x+4*y = 5, 2*x+4*y<=0},NONNEGATIVE);
```

$$\{x = -4\,y + 5,\ \_SL2 = 4\,y - 10\}$$

Функция standardize(C) - преобразует множество выражений к виду <=, т.е. возвращает набор или список неравенств вида <=, эквивалентный исходному множеству ограничений C со всеми константами в правых частях. Эта функция эквивалентна convert(C,stdle) и отличается от convert(C,std) тем, что равенства заменяются парой неравенств.

```
> standardize({x+4*y = 5, 2*x+4*y<=3});
```

$$\{2\,x + 4\,y \le 3,\ x + 4\,y \le 5,\ -x - 4\,y \le -5\}$$

Функция convert(C,std) возвращает множество (список) ограничений целевой функции, полученное перемещением всех констант в правую часть уравнения (неравенства) для каждого ограничения в C. Неравенство формы >= автоматически превращается в Maple в <=.

```
> convert({2<=x+4*y, 6<=4*x+2*y}, std );
```

$$\{-x - 4\,y \le -2,\ -4\,x - 2\,y \le -6\}$$

Функция convert(C,stdle) возвращает множество или список неравенств с формой <=, эквивалентных исходному множеству ограничений, со всеми константами справа. Приведем пример:

```
> convert({2*x+4*y<=3, 4*x+3*y = 6}, stdle );
```

$$\{2\,x + 4\,y \le 3,\ 4\,x + 3\,y \le 6,\ -4\,x - 3\,y \le -6\}$$