第3章 灵敏度分析

线性规划模型都假定参数是已知的或确定的，然而有时候很难确切地知道这些参数。这些参数也会随着市场、技术的变化相应变化，并最终导致模型的最优解发生变化。因此，分析参数变化对模型最优解的影响非常重要，这种分析称为灵敏度分析。

3.1 本章内容

本章主要介绍以下内容。

(1) 灵敏度分析的概念和思路。
(2) 资源变量的分析及其全局依赖。
(3) 价值变量的分析及其全局依赖。
(4) 增加变量。
(5) 改变约束系数矩阵。
(6) 增加约束条件。

3.2 灵敏度分析的概念和思路

在了解灵敏度分析方法之前，必须先了解一下相关的概念和思路。

3.2.1 灵敏度分析的概念

灵敏度分析指一个系统(或模型)的状态或输出变化对系统参数或周围条件变化的敏感程度的分析。

在第 1 章和第 2 章中，我们将向量 $\boldsymbol{b}$、$\boldsymbol{c}$，系数矩阵 $\boldsymbol{A}$ 作为已知量来使用，但是在实际的生产运作中，这些量往往是估计或预测的结果，随着生产条件和市场状态的变化，未必符合原来的预期。这些量中的一个或几个发生变化时，线性规划问题的最优解是否会发生变化，如果变化，变化的结果又是什么，这就是灵敏度分析要解决的问题。

我们当然可以选择在有参量发生变化时从头用单纯形法进行计算，但这样做增加了不必要的计算量。更好的做法是，将个别参数的变化反映在原来得到最优解的最终单纯形法表中，看是否依然满足最优解的条件。如果不满足，再从这个表开始一步步计算。

3.2.2 灵敏度分析的实现思路

灵敏度分析的实现思路包括以下几步。

(1) 将参数的改变反映到改变前的最终单纯形表上来。

在第 2 章中介绍过线性规划问题中的各个量在最初单纯形法表和最终单纯形法表中的关系，这里需要使用的是

$$\boldsymbol{b}' = \boldsymbol{B}^{-1}\boldsymbol{b}$$

$$\boldsymbol{P}'_j = \boldsymbol{B}^{-1}\boldsymbol{P}_j$$

$$(c_j - z_j)' = c_j - \sum_{i=1}^{m} a_{ij} y_i^*$$

(2) 检查原问题是否仍为可行解。

(3) 检查对偶问题是否仍为可行解。

(4) 按表 3-1 所列情况得出结论或继续计算方法。

表 3-1 结论或继续计算方法

原问题	对偶问题	结论或继续计算方法
可行解	可行解	问题的最优基不变
可行解	非可行解	用单纯形法继续迭代求最优解
非可行解	可行解	用对偶单纯形法继续迭代求最优解
非可行解	非可行解	通过人工变量，在新的单纯形表中重新计算

3.3 资源向量 b 的变化分析与全局依赖

本节探讨资源向量 $\boldsymbol{b}$ 对最优解和目标函数值的影响。

3.3.1 资源向量 b 的变化分析原理

在原问题已经达到最优解的情况下，改变 $\boldsymbol{b}$ 列数字，可以发现检验数并不会因此而发生变化。因此在 $\boldsymbol{b}$ 变化时，要么最终单纯形法表中 $\boldsymbol{B}^{-1}\boldsymbol{b}$ 的非负性保持不变，即原最优解对应的仍然是最优基，变化后的 $\boldsymbol{b}$ 列为最优解；要么 $\boldsymbol{B}^{-1}\boldsymbol{b}$ 中出现负数，此时原最优解是原问题的非可行解，但是其对偶问题的可行解，可以通过对偶单纯形法继续求解。

3.3.2 资源向量 b 的全局依赖

这一部分我们讨论资源向量 $\boldsymbol{b}$ 与目标函数值之间的关系。

首先，定义 $P(\boldsymbol{b})=\{\boldsymbol{x}\mid \boldsymbol{A}\boldsymbol{x}=\boldsymbol{b},\boldsymbol{x}\geqslant 0\}$，$S=\{\boldsymbol{b}\mid P(\boldsymbol{b})\text{非空}\}$，$F(\boldsymbol{b})=\max\limits_{x\in P(\boldsymbol{b})}\boldsymbol{c}'\boldsymbol{x}$。

由于在原问题有最优解的情况下，变化 $\boldsymbol{b}$，对偶问题总有可行解，即集合$\{\boldsymbol{y}\mid \boldsymbol{y}'\boldsymbol{A}\geqslant \boldsymbol{c}'\}$非空，那么根据对偶原理，当 $\boldsymbol{x}$ 属于 S，$F(\boldsymbol{b})$总是有限数。

对于 S 中给定的一个 $\boldsymbol{b}^*$，如果存在非退化最优解，即基解全为正（请思考为什么需要非退化），采用 2.3.1 节的记号，有 $\boldsymbol{x}_B=\boldsymbol{B}^{-1}\boldsymbol{b}^*$，且检验数均非正。对于 $\boldsymbol{b}^*$ 足够小的邻域中的 $\boldsymbol{b}$，我们总能让 $\boldsymbol{x}_B$ 保持非负，而由于检验数不变，因此最优解不变，矩阵 $\boldsymbol{B}$ 不变。此时 $F(\boldsymbol{b})=\boldsymbol{c}_B'\boldsymbol{B}^{-1}\boldsymbol{b}=\boldsymbol{y}'\boldsymbol{b}$，其中 $\boldsymbol{y}$ 是对偶问题的最优解。这就说明在 $\boldsymbol{b}^*$ 足够小的邻域内，$F(\boldsymbol{b})$是 $\boldsymbol{b}$ 的线性函数，且梯度由 $\boldsymbol{y}$ 决定。

下面我们证明，$F(\boldsymbol{b})$是关于 $\boldsymbol{b}(\boldsymbol{b}\in S)$的凹函数。

任取 S 中两元素 $\boldsymbol{b}_1$、$\boldsymbol{b}_2$，$\boldsymbol{x}_1$、$\boldsymbol{x}_2$ 为其对应的最优解，则有 $F(\boldsymbol{b}_1)=\boldsymbol{c}'\boldsymbol{x}_1$，$F(\boldsymbol{b}_2)=\boldsymbol{c}'\boldsymbol{x}_2$。注意到对任意 $\lambda\in[0,1]$，若令 $\boldsymbol{y}=\lambda\boldsymbol{x}_1+(1-\lambda)\boldsymbol{x}_2$，$\boldsymbol{b}=\lambda\boldsymbol{b}_1+(1-\lambda)\boldsymbol{b}_2$，则有 $\boldsymbol{A}\boldsymbol{y}=\boldsymbol{b}$，那么

$$\begin{aligned}F(\lambda\boldsymbol{b}_1+(1-\lambda)\boldsymbol{b}_2)=F(\boldsymbol{b})&\geqslant \boldsymbol{c}'\boldsymbol{y}=\boldsymbol{c}'(\lambda\boldsymbol{x}_1+(1-\lambda)\boldsymbol{x}_2)\\&=\lambda\boldsymbol{c}'\boldsymbol{x}_1+(1-\lambda)\boldsymbol{c}'\boldsymbol{x}_2=\lambda F(\boldsymbol{b}_1)+(1-\lambda)F(\boldsymbol{b}_2)\end{aligned}$$

3.3.3 资源向量 b 灵敏度分析的 MATLAB 实现

在 MATLAB 代码实现的部分，都会首先用单纯形法生成原来的最优解以及对应的参量。需要注意的是，在实际灵敏度分析操作中，这一步是已经做好了的。所以资源向量 $\boldsymbol{b}$ 灵敏度分析的 MATLAB 实现的部分是紧接着单纯形法生成原来最优解的部分，这里仅介绍灵敏度分析部分的代码。具体实现代码如下：

```
A0 = input('请输入约束条件的系数矩阵');
b0 = input('请输入该矩阵原来对应的资源列向量');
c = input('请按照顺序输入价值系数行向量');
b1 = input('请输入该矩阵改变后对应的资源列向量');
format rat
anscell = simplex_method(A0,b0,c);
A = anscell{1};
b = anscell{2};
base = anscell{6};
B = [];
for i = 1:1:length(base)
   B = [B A0(:,base(i))];
```

```
end

b2 = B\b1;
if min(b2)>=0
    ansx = zeros(1,length(c));
    for i = 1:1:length(b2)
        ansx(base(i)) = b2(i);
    end
    disp('最优解变为')
    disp(ansx)
    disp(dot(ansx,c))
else
    anscell = duality(A,b2,c);
    disp('最优解变为')
    disp(anscell{3})
    disp(anscell{4})
end
%b列仍为非负时,采用原来的最优解,出现负值时,使用对偶单纯形法得到新的最优解
```

3.3.4 *b* 对目标函数值和最优解的影响

在饲料厂的例子中,我们保持其他量不变,单一地改变 $\boldsymbol{b}$,看一看对目标函数值和最优解造成的影响。

将 $\boldsymbol{b}_2$ 从 7 递增到 12,目标函数值和最优解的变化如图 3-1 所示。

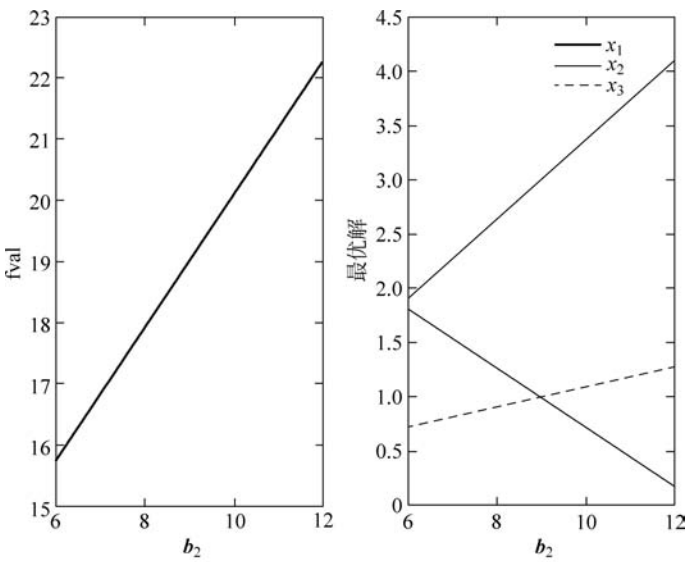

图 3-1 $\boldsymbol{b}$ 的变化分析

3.4 价值向量 c 的变化分析与全局依赖

本节将讨论价值向量 c 的变化原理和对最优解、目标函数值的影响。

3.4.1 价值向量 c 的变化分析原理

价值向量 c 的变化只会影响检验数。对于一行新的价值向量，要么检验数依然全部非正，则最优解不变，但目标函数值会因为 c 的变化而变化，要么检验数中出现正数，则此时原最优解是问题的可行解，继续使用单纯形法迭代可以得到新的最优解。

3.4.2 价值向量 c 的全局依赖

在分析资源向量 b 的全局依赖时，从原问题的对偶问题仍为可行解出发，研究变化后的目标函数值。本节将从原问题仍为可行解出发，研究原问题的对偶问题的有关性质。

定义 $Q(c)=\{y \mid A'y \geqslant c\}$，$T=\{c \mid Q(c)$非空$\}$，下面证明 T 是凸集。

如果 $c_1 \in T$ 且 $c_2 \in T$，即存在 y_1、y_2 使得 $A'y_1 \geqslant c_1$、$A'y_2 \geqslant c_2$，则对任意 $\lambda \in [0,1]$，有

$$A'(\lambda y_1 + (1-\lambda)y_2) \geqslant \lambda c_1 + (1-\lambda)c_2$$

即存在 $y=\lambda y_1+(1-\lambda)y_2$，使得 $A'y \geqslant \lambda c_1+(1-\lambda)c_2$，即 $\lambda c_1+(1-\lambda)c_2 \in T$，所以 T 为凸集。

下面我们研究 c 与目标函数值的关系。在第 1 章中我们曾经证明，若线性规划问题有最优解，则一定存在一个基可行解是最优解。显然 c 的变化并不影响基可行解，记全部基可行解为 $x^1, x^2, \cdots, x^N$，对于每个 c，定义

$$G(c)=\max_{x \in Q(c)} c'x = \max_{i=1,2,\cdots,N} c'x^i$$

对于每个给定的变化方向而言，这是一个总取最大值的分段函数，显然是凸函数（如图 3-2 所示）。

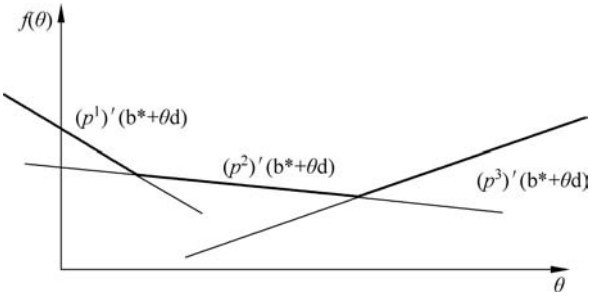

图 3-2 c 的变化原理

对于某个给定的 $\boldsymbol{c}^*$，如果存在唯一(请思考唯一性为什么是必要的)最优解 $\boldsymbol{x}^i$，即对任意 $j \neq i$，有 $\boldsymbol{c}^{*\prime}\boldsymbol{x}^i > \boldsymbol{c}^{*\prime}\boldsymbol{x}^j$，那么对于 $\boldsymbol{c}^*$ 足够小邻域内的 $\boldsymbol{c}$，依然可以保证有 $\boldsymbol{c}'\boldsymbol{x}^i > \boldsymbol{c}'\boldsymbol{x}^j$，即 $\boldsymbol{x}^i$ 仍为唯一最优解，即在 $\boldsymbol{c}^*$ 的某个邻域内，$G(\boldsymbol{c})$是 $\boldsymbol{c}$ 的线性函数且梯度为 $\boldsymbol{x}^i$，而 $G(\boldsymbol{c})$的转折点则对应了多个最优解。

3.4.3 价值向量 c 变化的 MATLAB 实现

求解 $\boldsymbol{c}$ 变化时线性规划问题，代码如下：

```
A0 = input('请输入约束条件的系数矩阵');
b = input('请输入该矩阵对应的资源列向量');
c0 = input('请按照顺序输入原来价值系数行向量');
c1 = input('请按照顺序输入改变后价值系数行向量');
format rat
anscell = simplex_method(A0,b,c0);
A = anscell{1};
b = anscell{2};
base = anscell{6};

c_z = [];
cb = [];
for t = 1:1:length(base)
   cb = [cb,c1(base(t))];
end

for j = 1:1:length(c1)
   z(j) = dot(cb',A(:,j));
   c_z = [c_z c1(j) - z(j)];
end
% 对新的检验数加以判断
if max(c_z)<= 0
   disp('最优解仍为')
   disp(anscell{3})
   disp(dot(anscell{3},c1))
else
   anscell = simplex_method(A,b,c1);
   disp('最优解变为')
   disp(anscell{3})
   disp(anscell{4})
end
```

3.4.4 c 对目标函数值和最优解的影响

在饲料厂的例子中，令 c_3 从 0 变化到 5，得到最优解和对应目标函数的变化如图 3-3 所示。

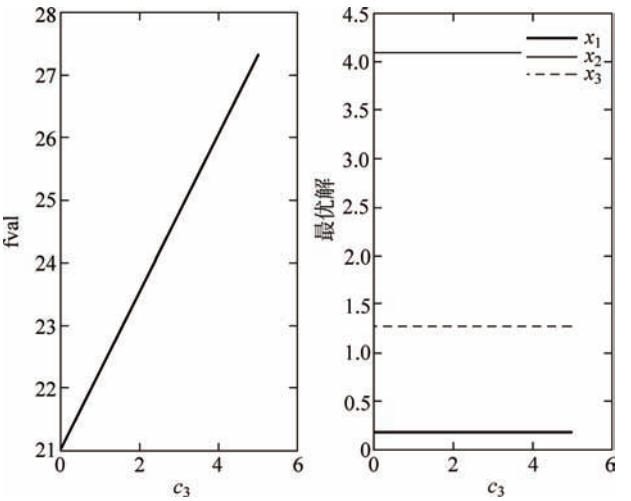

图 3-3 c 的变化分析

3.5 增加变量的分析

增加一个变量，相当于约束矩阵增加一列，分析这个变化带来的影响。

3.5.1 增加变量的分析原理

记增加的变量在 $\boldsymbol{A}$ 中对应的列向量为 $\boldsymbol{A}_j$，价值为 c_j。首先计算在原问题的最终单纯形法表中新增的 $\boldsymbol{x}$ 对应的检验数 $\sigma = c_j - \boldsymbol{C}_B\boldsymbol{B}^{-1}\boldsymbol{A}_j$，如果该检验数非正，那么原问题的最优解保持不变，新增的 $\boldsymbol{x}$ 取为 0；否则，计算出 $\boldsymbol{A}_j' = \boldsymbol{B}^{-1}\boldsymbol{A}_j$ 并写入原问题的最终单纯形法表中（注意此时满足使用单纯形法的条件，继续用单纯形法迭代即可）。

3.5.2 增加变量分析的 MATLAB 实现

实现代码如下：

```
clear all;
A0 = input('请输入原来约束条件的系数矩阵');
```

```
b = input('请输入该矩阵对应的资源列向量');
c0 = input('请按照顺序输入原来价值系数行向量');
p = input('请输入新变量 x 对应的系数列向量');
addc = input('请输入新变量 x 对应的价值系数');
format rat
anscell = simplex_method(A0,b,c0);
A = anscell{1};
b = anscell{2};
base = anscell{6};
B = [];
for i = 1:1:length(base)
   B = [B A0(:,base(i))];
end
cb = [];
for t = 1:1:length(base)
   cb = [cb,c0(base(t))];
end
sigma = addc - dot(cb/B,p);
% 计算原问题最终单纯形法表中的 sigma
if sigma <= 0
   disp('最优解不变,新增的 x 取为 0')
else
   c = [c0,addc];
   A = [A B\p];
   anscell = simplex_method(A,b,c);
   disp('最优解变为')
   disp(anscell{3})
   disp(anscell{4})
end
```

3.6 改变约束系数矩阵的分析

本节将介绍改变约束系数矩阵的分析原理及 MATLAB 实现过程。

3.6.1 改变约束系数矩阵的分析原理

本节依然采用 2.4 节中的记号。讨论当某个 x_i 对应的系数列向量及其价值发生变化时,对最优解的影响。

如果 $\boldsymbol{N}$ 中的某列发生了变化,那么除了这一列之外,其他列的检验数都不变。只需要采取与 2.4 节中相同的方法计算检验数和新的最终单纯形法表中该 $\boldsymbol{x}$ 对应的列向量,如果检验数保持非正,则最优解不变;否则用单纯形法继续计算。

如果 $\boldsymbol{B}$ 中某列发生了变化，则最终单纯形法表中的 b 列和检验数行都会发生变化，因此可能出现原问题和其对偶问题均为非可行解的情况。此时应设法将原问题转换为可行解，此时为使单纯形法表中有基可行解，需要添加人工变量，并同第 1 章，使用两阶段法求解。

3.6.2 改变约束系数矩阵分析的 MATLAB 实现

针对以上情况，改变约束矩阵后继续求解的代码如下：

```
A0 = input('请输入原来约束条件的系数矩阵');
b = input('请输入该矩阵对应的资源列向量');
c0 = input('请按照顺序输入原来价值系数行向量');
count = input('请输入 A 的第几列被改变了');
p = input('请输入改变后的列向量');
changec = input('请输入改变后的价值系数');
format rat
anscell = simplex_method(A0,b,c0);
A = anscell{1};
b = anscell{2};
base = anscell{6};
c = c0;
c(count) = changec;
B = [];
for i = 1:1:length(base)
   B = [B A0(:,base(i))];
end
cb = [];
for t = 1:1:length(base)
   cb = [cb,c0(base(t))];
end
sigma = changec - dot(cb/B,p);
p1 = B\p;
if ismember(count,base) % 改变 B
   A = [A(:,1:count) p1 A(:,count + 1:end)];
   bA = [b A];

   out = find(A(:,count));
   bA(out,:) = bA(out,:)/bA(out,count + 2);

   for i = 1:1:size(A,1)
      if i == out
         continue
      end
```

```
        d = bA(i,count + 2);
        bA(i,:) = bA(i,:) - d * (bA(out,:));
    end

    b = bA(:,1);
    A = bA(:,2:end);
A = [A(:,1:count - 1) A(:,count + 1:end)];
% 让新的 x 对应单纯形法表中的列向量仍为原来形式的标准单位列向量
    if min(b)>= 0
        anscell = simplex_method(A,b,c);
        disp(anscell{3})
        disp(anscell{4})
% 原问题有可行解,直接用单纯形法
    else
        for i = 1:1:length(b)
            if b(i)< 0
                bA(i,:) = - bA(i,:);
            end
        end
        b = bA(:,1);
        A = bA(:,2:end);
        A = [A(:,1:count - 1) A(:,count + 1:end)];
        [RANGE10,RANGE20] = size(A);
        findj0 = [];
        for i = 1:1:RANGE20
            if length(find(A(:,i) == 1)) == 1&&length(find(A(:,i))) == 1
                if isempty(findj0)
                    ind0 = find(A(:,i) == 1);
                    addfindj0 = [ind0;i];
                    findj0 = [findj0 addfindj0];
                elseif~ismember(find(A(:,i) == 1),findj0(1,:))
                    ind0 = find(A(:,i) == 1);
                    addfindj0 = [ind0;i];
                    findj0 = [findj0 addfindj0];
                end
            end
        end

        if isempty(findj0)
            base0 = [];
            add = (1:RANGE10);
            findj10 = [];
        else
            findj10 = sortrows(findj0');
            base0 = findj10(:,2);
```

```
        end

        if~isempty(base0)
           add = setdiff((1:RANGE10),findj10(:,1));
        end

        for i = 1:1:length(add)
           l = zeros(RANGE10,1);
           l(add(i)) = 1;
           A = [A l];
        end

        c00 = zeros(1,length(add) + RANGE20);
        for i = 1:1:length(add)
           c00(RANGE20 + i) = - 1;
        end

        anscell = simplex_method(A,b,c00);
        optans = anscell{4};
        A = anscell{1};
        b = anscell{2};

        if optans~ = 0
           fprintf('无可行解')
        else
           A = A(:,1:RANGE20);
           anscell = simplex_method(A,b,c);
           disp(anscell{5})
           disp(anscell{3})
           disp(anscell{4})
        end
    end
    %原问题不是可行解时,用两阶段法
    else
       if sigma < = 0
          disp('最优解仍为')
          disp(anscell{4})
          disp(dot(anscell{4},c))
       else
          A(:,count) = p1;
          anscell = simplex_method(A,b,c);
          disp(anscell{5})
          disp(anscell{3})
          disp(anscell{4})
       end
    end
    %改变 N
```

3.6.3 改变 A 的影响

在饲料厂的例子中，将 $A(1,3)$从 1 递增到 3，对结果的影响如图 3-4 所示。

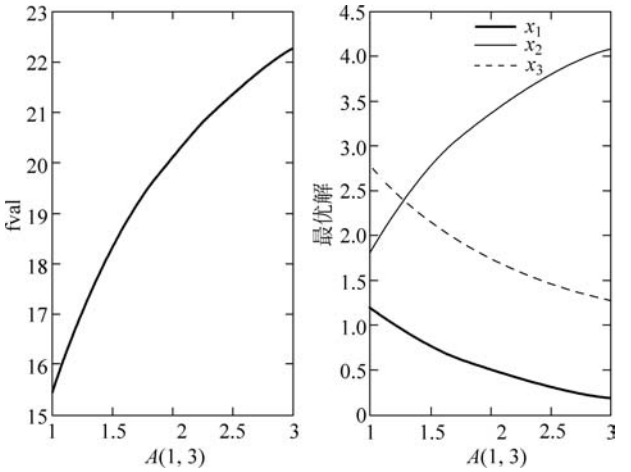

图 3-4 A 的影响

3.7 增加约束条件的分析

3.7.1 增加约束条件的分析原理

当一个新的约束条件被添加时，首先考查原最优解是否符合这个新的约束条件。如果符合，最优解保持不变；如果不符合，则将新的约束条件反映到单纯形法表中继续分析。如果新增的约束条件为小于或等于关系，那么通过增加松弛变量，只需进行简单的矩阵初等行变换，就可以得到一个单纯形法的基可行解，继续用单纯形法迭代即可。如果新增了大于或等于关系的约束条件，那么需要先添加人工变量，再把添加条件后的矩阵通过初等变化变为标准形式，从而用两阶段法求解。

3.7.2 增加约束条件分析的 MATLAB 实现

在使用这段代码前，仍然需要读者先将约束条件化为标准形式。如果新增的约束条件是不等式，那么新增条件的行向量要在原来维数的基础上增加 1(松弛变量)或减少 1(剩余变量)，例如原来的约束矩阵是 $\begin{bmatrix}1 & 3\\ 2 & 5\end{bmatrix}$，如果新增条件为 $x_1+x_2\geqslant 1$，那么应输入的行向量是[1,1,−1]。

具体增加约束条件分析的实现代码如下：

```
A = input('请输入原来约束条件的系数矩阵');
b = input('请输入原来该矩阵对应的资源列向量');
c = input('请按照顺序输入价值系数行向量');
a = input('请输入新增约束条件行向量');
addb = input('请输入新增资源限制的值');
format rat
anscell = simplex_method(A,b,c);
b = [anscell{2};addb];
if length(a) == size(A,2)
   A = [anscell{1};a];
   flag = 0;
else
   c = [c,0];
   A = [anscell{1},zeros(size(A,1),1)];
   A = [A;a];
   if a(end) == 1
      flag = 1; % 此即小于或等于的情况
   else
      flag = 0;
   end
end
base = anscell{6};
bA = [b A];
for i = 1:1:length(base)
   bA(size(A,1),:) = bA(size(A,1),:) - A(size(A,1),base(i)) * bA(i,:);
end
b = bA(:,1);
A = bA(:,2:end);
if flag == 1
   anscell = duality(A,b,c);
   disp(anscell{5})
   disp(anscell{3})
   disp(anscell{4})
else % 可参见两阶段法,这里更简单一些,由于一定是添加一列且其末数为 1
   add = zeros(size(A,1),1);
   add(end) = 1;
   A1 = [A add];
   c0 = zeros(1,length(c) + 1);
   c0(end) = - 1;
   anscell = duality(A1,b,c0);
   if anscell{4}~ = 0
      disp('无可行解')
   else
```

```
        A2 = anscell{1};
        A = A2(:,1:end - 1); % 去除人工变量
        b = anscell{2};
        anscell = simplex_method(A,b,c);
        disp(anscell{5})
        disp(anscell{3})
        disp(anscell{4})
    end
end
```