**同 济 大 学**

**实验报告**

线性规划



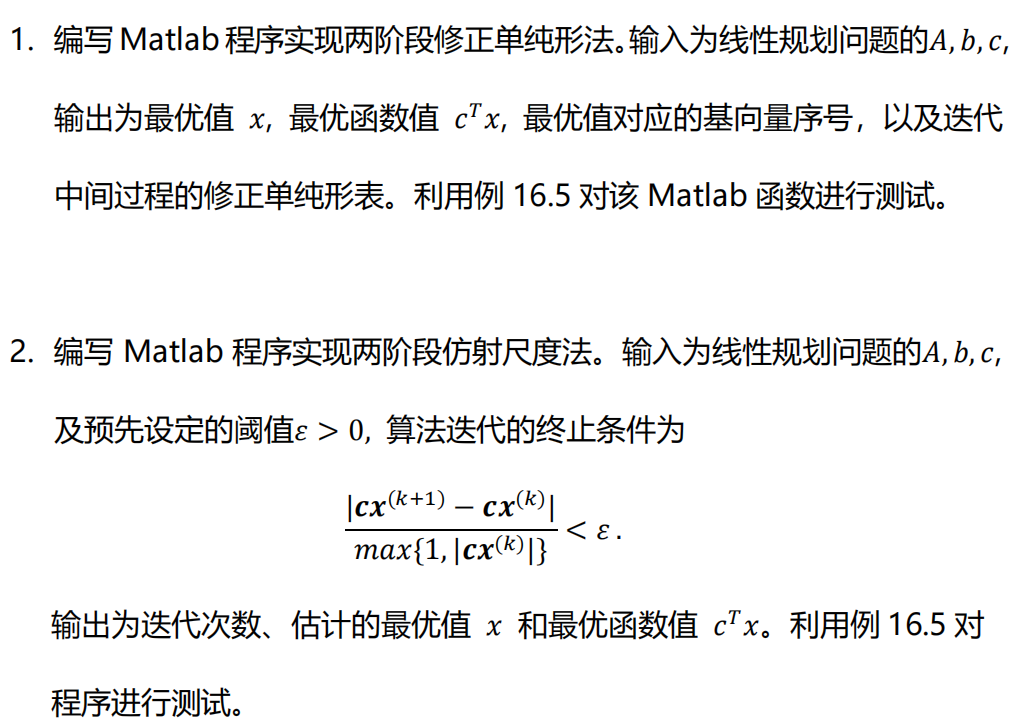
**课程名称**   最优化原理

**学生姓名**  姚 天 亮

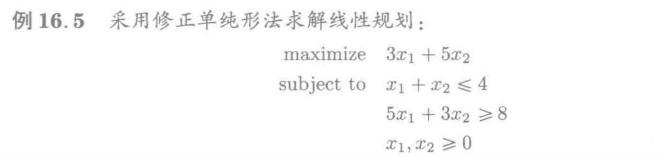
**学 号**  2150248

**学 院** 电子与信息工程学院

1. **作业要求**



例16.5：



1. **理论推导**

本作业题的目的分别利用两阶段修正单纯形法与两阶段仿射尺度法对线性规划问题进行求解。

两阶段修正单纯形法是一种求解线性规划问题的方法，它主要用于处理约束系数矩阵不包含单位矩阵（没有明显的基本可行解）的情况，也就是无法直接得到初始基可行解的情况。它分为两个阶段：

第一阶段：引入人工变量，构造一个只含有人工变量的目标函数，并求其最小值。如果最小值为零，则说明原问题有基可行解，可以进入第二阶段；如果最小值不为零，则说明原问题无可行解，算法终止。

第二阶段：去掉人工变量，恢复原目标函数，用单纯形法求解原问题的最优解。

两阶段仿射尺度法的基本原理同两阶段修正单纯形法，只不过将单纯形法计算的模块替换为仿射尺度的计算模块。

修正单纯形法是一种改进的单纯形法，它可以避免对大部分非基变量的计算，从而提高求解线性规划问题的效率。修正单纯形法的基本思想是，给定一个初始的可行基矩阵和其逆矩阵，通过不断地修正旧的可行基矩阵的逆矩阵，获得新的可行基矩阵的逆矩阵，进而完成单纯形法所需要的其他运算。修正单纯形法的主要步骤如下：

S1.针对初始基本可行解构造修正的单纯形表

S2.计算当前检验数，如果对所有非基变量都有检验数大于等于零，则停止运算，当前基本可行解即是最优解；否则进入下一步

S3.从小于零的检验数中选择一个检验数作为进基变量

S4.如果不存在正的约束系数，则停止运算，问题有无界解；否则计算出基变量和步长

S5.更新基本可行解和修正的单纯形表

S6.回到第二步继续迭代。

最为基础的单纯形法的基本思想是从一个初始的基本可行解出发，通过不断地改进基本可行解，使目标函数值不断增大或减小，直到找到最优解为止。单纯形法的基本步骤如下：

S1.将线性规划问题化为标准形式，即目标函数为最大化，约束条件为等式，变量非负。

S2.找到一个初始的基本可行解，即满足约束条件和非负性的一组解，可以通过引入松弛变量或人工变量来构造。

S3.计算检验数，即目标函数中非基变量的系数减去基变量对应列的线性组合。检验数反映了非基变量对目标函数值的影响。

S4.判断是否达到最优解。如果所有检验数都小于等于零（最大化问题）或大于等于零（最小化问题），则当前基本可行解是最优解，算法停止；否则进入下一步。

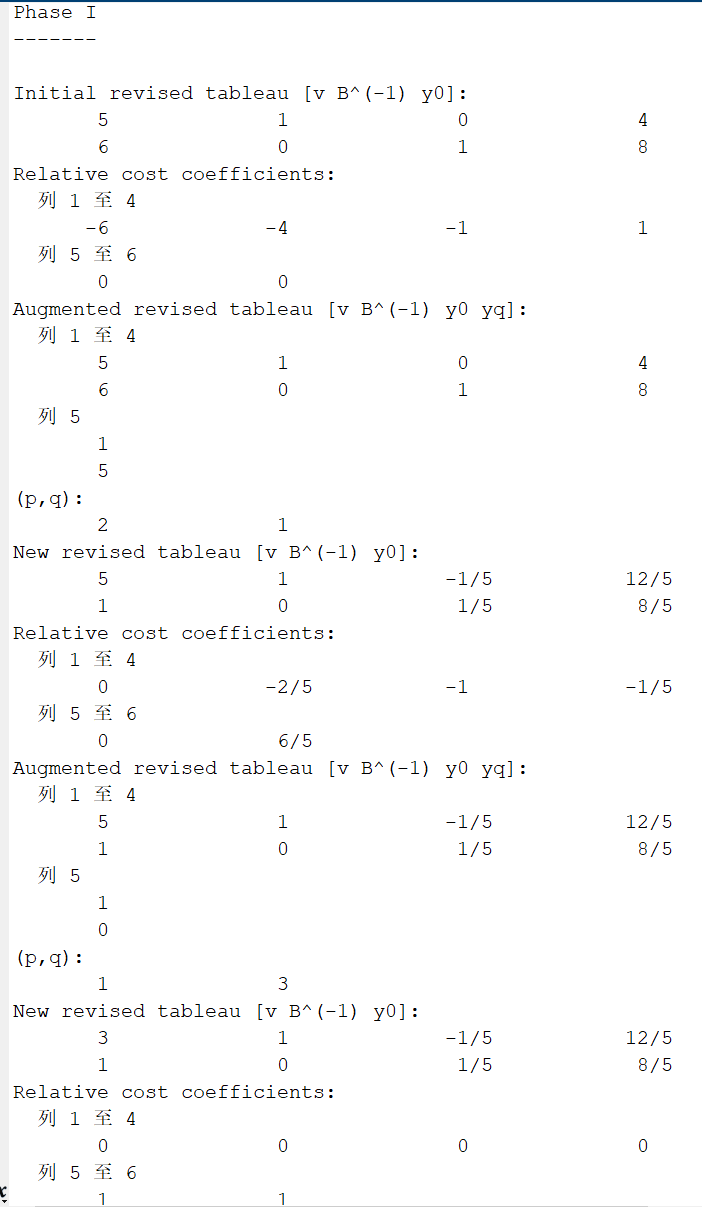
S5.选择一个进基变量和一个出基变量。进基变量是检验数为正（最大化问题）或负（最小化问题）的非基变量中的一个，可以按照最大系数法、最小下标法等规则来选取。出基变量是当前基变量中的一个，可以按照最小比率法来选取，即使得进基变量增加后不会导致其他基变量为负。

S6.进行枢轴运算，即用高斯消元法将出基变量所在行的主元（即进基变量所在列的元素）化为1，然后用该行消去其他行中该列的元素，使得进基变量成为新的基变量，而出基变量成为新的非基变量。

S7.回到第三步继续迭代，直到达到最优解或发现问题无界或无可行解。

1. **实验结果**

图1，图2分别展示了两题中迭代过程中值的变化情况。



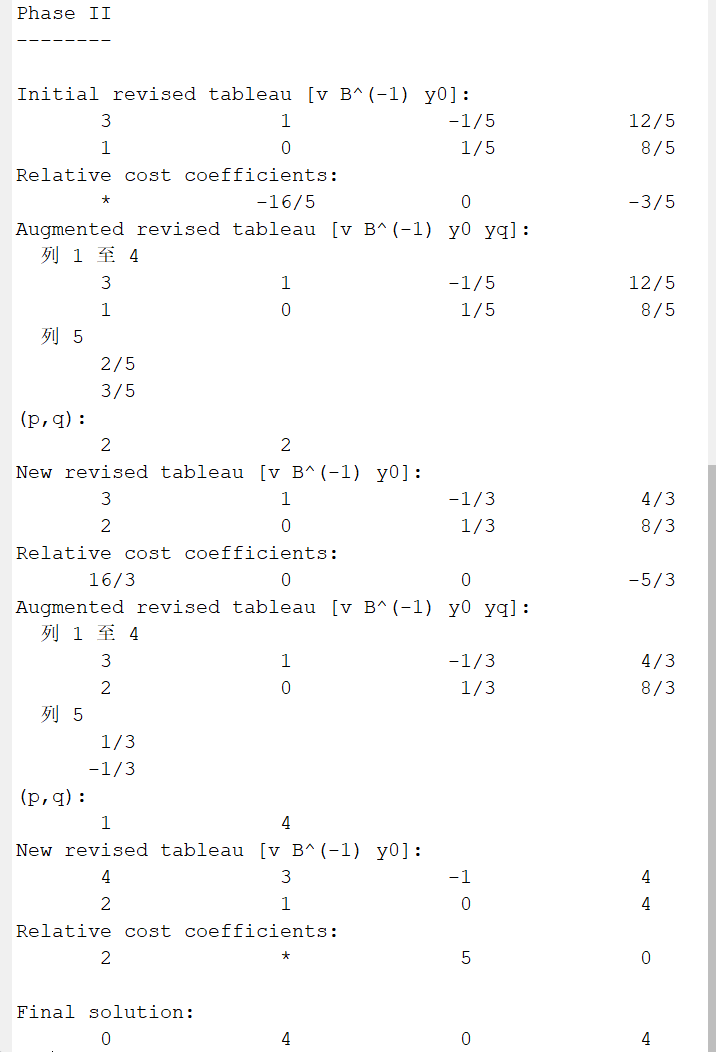


图1 第一题的迭代过程

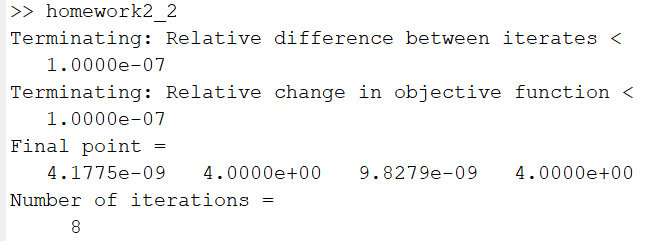


图2 第二题的迭代结果

**附录**

1. **作业一**

%2150248 姚天亮 自动化

%main1.m

A=[1 1 1 0; 5 3 0 -1];

b=[4;8];

c=[-3;-5;0;0];

options(1)=1;

format rat;

tprevsimp(c,A,b,options);

% revsimp.m

function [x,v,Binv]=revsimp(c,A,b,v,Binv,options)

if nargin ~= 6

options = [];

if nargin ~= 5

disp('Wrong number of arguments.');

return;

end

end

format compact;

%format short e;

options = foptions(options);

print = options(1);

n=length(c);

m=length(b);

cB=c(v(:));

y0 = Binv\*b;

lambdaT=cB'\*Binv;

r = c'-lambdaT\*A; %row vector of relative cost coefficients

if print

disp(' ');

disp('Initial revised tableau [v B^(-1) y0]:');

disp([v Binv y0]);

disp('Relative cost coefficients:');

disp(r);

end %if

while ones(1,n)\*(r' >= zeros(n,1)) ~= n

if options(5) == 0;

[r\_q,q] = min(r);

else

%Bland’s rule

q=1;

while r(q) >= 0

q=q+1;

end

end %if

yq = Binv\*A(:,q);

min\_ratio = inf;

p=0;

for i=1:m,

if yq(i)>0

if y0(i)/yq(i) < min\_ratio

min\_ratio = y0(i)/yq(i);

p = i;

end %if

end %if

end %for

if p == 0

disp('Problem unbounded');

break;

end %if

if print,

disp('Augmented revised tableau [v B^(-1) y0 yq]:')

disp([v Binv y0 yq]);

disp('(p,q):');

disp([p,q]);

end

augrevtabl=pivot([Binv y0 yq],p,m+2);

Binv=augrevtabl(:,1:m);

y0=augrevtabl(:,m+1);

v(p) = q;

cB=c(v(:));

lambdaT=cB'\*Binv;

r = c'-lambdaT\*A; %row vector of relative cost coefficients

if print,

disp('New revised tableau [v B^(-1) y0]:');

disp([v Binv y0]);

disp('Relative cost coefficients:');

disp(r);

end %if

end %while

x=zeros(n,1);

x(v(:))=y0;

%tprevsimp

function [x,v]=tprevsimp(c,A,b,options)

if nargin ~= 4

options = [];

if nargin ~= 3

disp('Wrong number of arguments.');

return;

end

end

clc;

format compact;

%format short e;

options = foptions(options);

print = options(1);

n=length(c);

m=length(b);

%Phase I

if print

disp(' ');

disp('Phase I');

disp('-------');

end

v=n\*ones(m,1);

for i=1:m

v(i)=v(i)+i;

end

[x,v,Binv]=revsimp([zeros(n,1);ones(m,1)],[A eye(m)],b,v,eye(m),options);

%Phase II

if print

disp(' ');

disp('Phase II');

disp('--------');

end

[x,v,Binv]=revsimp(c,A,b,v,Binv,options);

if print

disp(' ');

disp('Final solution:');

disp(x');

end

1. **作业二**

%2150248 姚天亮 自动化

% main2.m

A=[1 1 1 0; 5 3 0 -1];

b=[4;8];

c=[-3;-5;0;0];

options(1)=0;

tpaffscale(c,A,b,options);

% tpaffscale.m

function [x,N]=tpaffscale(c,A,b,options)

if nargin ~= 4

options = [];

if nargin ~= 3

disp('Wrong number of arguments.');

return;

End

end

%clc;

format compact;

format short e;

options = foptions(options);

print = options(1);

n=length(c);

m=length(b);

%Phase I

if print,

disp(' ');

disp('Phase I');

disp('-------');

end

u = rand(n,1);

v = b-A\*u;

if v ~= zeros(m,1),

u = affscale([zeros(1,n),1]',[A v],b,[u' 1]',options);

u(n+1) = [0];

end

if print

disp('')

disp('Initial condition for Phase II:')

disp(u)

end

if u(n+1) < options(2),

%Phase II

u(n+1) = [];

if print

disp(' ');

disp('Phase II');

disp('--------');

disp('Initial condition for Phase II:');

disp(u);

end

[x,N]=affscale(c,A,b,u,options);

if nargout == 0

disp('Final point =');

disp(x');

disp('Number of iterations =');

disp(N);

end %if

else

disp('Terminating: problem has no feasible solution.');

end

% affscale.m

function [x,N] = affscale(c,A,b,u,options);

if nargin ~= 5

options = [];

if nargin ~= 4

disp('Wrong number of arguments.');

return;

end

end

xnew=u;

if length(options) >= 14

if options(14)==0

options(14)=1000\*length(xnew);

end

else

options(14)=1000\*length(xnew);

end

%if length(options) < 18

options(18)=0.99; %optional step size

%end

%clc;

format compact;

format short e;

options = foptions(options);

print = options(1);

epsilon\_x = options(2);

epsilon\_f = options(3);

max\_iter=options(14);

alpha=options(18);

n=length(c);

m=length(b);

for k = 1:max\_iter,

xcurr=xnew;

D = diag(xcurr);

Abar = A\*D;

Pbar = eye(n) - Abar'\*inv(Abar\*Abar')\*Abar;

d = -D\*Pbar\*D\*c;

if d ~= zeros(n,1),

nonzd = find(d<0);

r = min(-xcurr(nonzd)./d(nonzd));

else

disp('Terminating: d = 0');

break;

end

xnew = xcurr+alpha\*r\*d;

if print,

disp('Iteration number k =')

disp(k); %print iteration index k

disp('alpha\_k =');

disp(alpha\*r); %print alpha\_k

disp('New point =');

disp(xnew'); %print new point

end %if

if norm(xnew-xcurr) <= epsilon\_x\*norm(xcurr)

disp('Terminating: Relative difference between iterates <');

disp(epsilon\_x);

break;

end %if

if abs(c'\*(xnew-xcurr)) < epsilon\_f\*abs(c'\*xcurr),

disp('Terminating: Relative change in objective function <' );

disp(epsilon\_f);

break;

end %if

if k == max\_iter

disp('Terminating with maximum number of iterations');

end %if

end %for

if nargout >= 1

x=xnew;

if nargout == 2

N=k;

end

else

disp('Final point =');

disp(xnew');

disp('Number of iterations =');

disp(k);

end %if