最优化原理与方法 第二次实验课作业

2151094 宋正非

一、两阶段修正单纯形法

1.题目要求

编写 Matlab 程序实现两阶段修正单纯形法。输入为线性规划问题的A, b, c，输出为最优值 x，最优函数值 $c^T x$，最优值对应的基向量序号，以及迭代中间过程的修正单纯形表。利用例 16.5 对该 Matlab 函数进行测试。

2. 程序

```
function two_phase_revised_simplex(A,b,c)
% 实现两阶段修正单纯形法求解线性规划问题
% 输入：（标准型形式）
% A          约束矩阵
% b          约束向量
% c          目标向量
% 输出：
% 打印求解过程中的修正单纯形表，以及最优解和最优值

%目标向量转换
minimize=1;
if minimize==1
    c=-c;
end

%获取目标向量，约束向量和约束矩阵的维度信息
n=length(c);
m=length(b);
j=max(abs(c));

num=n; %变量计数
nbv=1:n; %非基变量的下标向量
bv=zeros(1,m); %基变量的标记向量
hv=zeros(1,m); %人工变量的标记向量

%构造初始的修正单纯形表
```

```
for i=1:m
    if b(i)<0
        A(i,:)=-A(i,:); b(i)=-b(i);
    end
    num=num+1;
    c(num)=-10*j;
    A(i,num)=1;
    bv(i)=num;
    hv(i)=num;
end
A=[-c;A];

% 初始化基变量集合和解向量
B_inv=eye(m+1,m+1);
B_inv(1,2:m+1)=c(bv);
x_b=B_inv*[0; b'];
flag=0; %flag: 算法是否收敛
iteration=0; %count: 迭代次数
of_curr=0; %of_curr: 上一次迭代的目标函数数值

while(flag~=1)
  [s,t]=min(B_inv(1,:)*A(:,nbv));%找到最小的非基变量目标函数系数
  y=B_inv*A(:,nbv(t));%计算新的解向量
  iteration=iteration+1;
  %如果新的解向量仍然有正元素，则继续迭代
  if(any(y(2:m+1)>0))
      fprintf( '第%d 次迭代,修正单纯形表: \n',iteration);
      print_table(B_inv, x_b, bv, y);
      %如果迭代过程中出现无法消除人工变量的情况，则输出错误提示
      if iteration>1 && of_curr==x_b(1)
          flag=1;
          x_b(1)=-x_b(1);
          fprintf( '给定问题存在退化可行解。\n');
          fprintf( '当前目标函数数值 cTx = %d\n',x_b(1));
          fprintf( '当前解为: \n');
          for i=1:n
              found=0;
              for j=1:m
                  if bv(j)==i
                      fprintf( 'x%u = %d\n',i,x_b(1+j));
                      found=1;
                  end
              end
              if found==0
                  fprintf( 'x%u = %d\n',i,0);
              end
          end
      %如果最终解向量中人工变量的值不为 0，则输出错误提示
      else
        of_curr=x_b(1);
        if(s>=0)
            flag=1;
            for i=1:length(hv)
                for j=1:m
                    if hv(i)==bv(j)
                        fprintf( '错误提示: 给定问题无可行解。\n');
                        return
                    end
```

```
                end
            end
            %如果已经达到最优解，则输出最优解和最优值
            if minimize==1
                x_b(1)=-x_b(1);
            end
            fprintf( '已经迭代到最优解。\n');
            fprintf( '最优目标函数值 cTx = %-10.1f\n',x_b(1));
            fprintf( '最优可行解为：\n');
            for i=1:n
                found=0;
                for j=1:m
                    if bv(j)==i
                        fprintf('x%u = %-10.1f\n',i,x_b(1+j));
                        found=1;
                    end
                end
                if found==0
                    fprintf('x%u = %d\n',i,0);
                end
            end
            %如果最终解向量中有多个非基变量的目标函数系数相等，则输出提示
            if (s==0 && any(y(2:m+1)>0))
                fprintf( '提示：本问题不止一个最优可行解。\n');
            end
        %找到约束条件中限制最紧的非基变量
        else
            u=10*j;
            for i=2:m+1
               if y(i)>0
                  if (x_b(i)/y(i))<u
                      u=(x_b(i)/y(i));
                      v=i-1;
                  end
               end
            end
            %交换该非基变量和当前基变量
            temp=bv(v);
            bv(v)=nbv(t);
            nbv(t)=temp;
            %枢轴变换
            E=eye(m+1,m+1);
            E(:,1+v)=-y/y(1+v);
            E(1+v,1+v)=1/y(1+v);
            B_inv=E*B_inv;
            x_b=B_inv*[0; b'];
        end
      end
  %解向量中所有元素都小于等于 0，则输出错误提示
  else
      fprintf( '错误提示：给定问题有无界解。\n')
      return
  end
end


function print_table(B_inv, x_b, bv, y)
% 打印修正单纯形表
```

```
fprintf('%-12s', '决策变量');
fprintf('%-15s', 'B^-1');
fprintf('%-10s', 'y0');
fprintf('\n----------------------------------------\n');
for i = 2:size(B_inv, 1)
    if i > 1
        fprintf('%-10s', sprintf('x%d', bv(i-1)));
    end
    for j = 2:size(B_inv, 2)
        fprintf('%-10.1f', B_inv(i, j));
    end
    fprintf('%-10.1f\n', x_b(i));
end
fprintf('增广的修正单纯形表中 y1:\n%.1f\n%.1f\n', y(2),y(3));
end
```

3. 例 16.5 题目

例 16.5 采用修正单纯形法求解线性规划：

$$
\begin{aligned}
\text{maximize} \quad & 3x_1 + 5x_2 \\
\text{subject to} \quad & x_1 + x_2 \leqslant 4 \\
& 5x_1 + 3x_2 \geqslant 8 \\
& x_1, x_2 \geqslant 0
\end{aligned}
$$

首先，引入 1 个松弛变量和 1 个剩余变量，将该问题改写为标准型：

$$
\begin{aligned}
\text{minimize} \quad & -3x_1 - 5x_2 \\
\text{subject to} \quad & x_1 + x_2 + x_3 = 4 \\
& 5x_1 + 3x_2 - x_4 = 8 \\
& x_1, \cdots, x_4 \geqslant 0
\end{aligned}
$$

该问题没有明显的基本可行解，因此应采用两阶段单纯形法。

4. 例 16.5 求解

```
%homework1.m
%test function with T16.5
A=[1,1,1,0;5,3,0,-1];
b=[4,8];
c=[-3,-5,0,0];
two_phase_revised_simplex(A,b,c)
```

5. 例 16.5 结果

第一阶段：

```
第1 次迭代,修正单纯形表:
决策变量            B^-1              y0
-----------------------------------------
x5          1.0         0.0         4.0
x6          0.0         1.0         8.0
增广的修正单纯形表中y1:
1.0
5.0
第2 次迭代,修正单纯形表:
决策变量            B^-1              y0
-----------------------------------------
x5          1.0         -0.2        2.4
x1          0.0         0.2         1.6
```

第二阶段:

```
第3 次迭代,修正单纯形表:
决策变量            B^-1              y0
-----------------------------------------
x3          1.0         -0.2        2.4
x1          0.0         0.2         1.6
增广的修正单纯形表中y1:
0.4
0.6
第4 次迭代,修正单纯形表:
决策变量            B^-1              y0
-----------------------------------------
x3          1.0         -0.3        1.3
x2          0.0         0.3         2.7
增广的修正单纯形表中y1:
0.3
-0.3
第5 次迭代,修正单纯形表:
决策变量            B^-1              y0
-----------------------------------------
x4          3.0         -1.0        4.0
x2          1.0         0.0         4.0
增广的修正单纯形表中y1:
-2.0
1.0
已经迭代到最优解。
```

最终结果:

```
最优目标函数值 cTx = -20.0
最优可行解为:
x1 = 0
x2 = 4.0
x3 = 0
x4 = 4.0
```

二、两阶段仿射尺度法

1. 题目要求

编写 Matlab 程序实现两阶段仿射尺度法。输入为线性规划问题的A, b, c,

及预先设定的阈值$\varepsilon > 0$，算法迭代的终止条件为

$$\frac{|\boldsymbol{c}\boldsymbol{x}^{(k+1)} - \boldsymbol{c}\boldsymbol{x}^{(k)}|}{max\{1, |\boldsymbol{c}\boldsymbol{x}^{(k)}|\}} < \varepsilon.$$

输出为迭代次数、估计的最优值 x 和最优函数值 $c^T x$。利用例 16.5 对程序进行测试。

2. 程序

```
function [x,N]=two_phase_affscale(A,b,c,options)
%
% TPAFFSCALE(A,b,c);
% TPAFFSCALE(A,b,c,options); %
% x=TPAFFSCALE(A,b,c);
% x=TPAFFSCALE(A,b,c,options); %
% [x,N]=TPAFFSCALE(A,b,c);
% [x,N]=TPAFFSCALE(A,b,c,options); %
% TPAFFSCALE(A,b,c)使用二阶段仿射尺度变换法求解以下线性规划问题：
% min c'x subject to Ax=b, x>=0 .
% 第二种函数形式允许定义一个可选参数向量：
% OPTIONS(1) 控制输出内容的详细程度，设置为 1 则会输出结果的表格形式（默认无输
出：0）
% OPTIONS(2) 最优解的精度
% OPTIONS(3) 最终目标函数值的精度
% OPTIONS(14) = 最大迭代次数
% OPTIONS(18) = alpha

format compact;
format short e;
% 如果没有提供选项向量，则创建一个默认选项向量
if nargin < 4
    options = [1, 1e-6, 1e-6, Inf, 0];
end
print=options(1);
n=length(c);
m=length(b);

% 第一阶段
if print
    disp("第一阶段");
```

```
    disp("-------");
end
% u 初始化为一个随机的 n 维向量，然后，v 计算为 b-Au。
% 如果 v 非零，则调用 affscale 函数来寻找一个可行的初始解，解决线性规划问题的第一阶段。
u=rand(n,1);
v=b-A*u;
if any(v) ~= 0
    u=affscale([A v],b,[zeros(1,n),1]',[u' 1]',options);
end
if print
    disp("进行第二阶段的初始条件：")
    disp(u)
end

if u(n+1)<options(2)
    % 第二阶段
    u(n+1)=[];
    if print
        disp("第二阶段");
        disp("--------");
        disp("进行第二阶段的初始条件：");
        disp(u);
    end
    [x, N] = affscale(A,b,c,u,options);
    if nargout==0
        disp("最终解：");
        disp(x');
        disp("迭代次数：");
        disp(N);
    end %if
else
    disp("终止：问题无可行解。");
end
end



function [x, N] = affscale(A,b,c,u,options)
% 输入参数：
%  c: 目标函数的系数向量 %  A: 约束条件的系数矩阵 %  b: 约束条件的右端向量 %  u: 一个可行的初始解
% 设置最大迭代次数

% 处理选项向量
if length(options)>=14
    if options(14)==0
        options(14)=1000*length(u);
    end
else
    options(14)=1000*length(u);
end

% 设置可选参数 alpha 的默认值
options(18)=0.99;
% 初始化
```

```
format compact;
format short e;
n=length(c);  % 变量个数
m=length(b);  % 约束个数
xnew = u;  % 初始解
print=options(1);  % 输出详细信息的标志
epsilon_x=options(2);  % 相邻两次迭代点的相对差的相对误差
epsilon_f=options(3);  % 相邻两次迭代目标函数值的相对误差
max_iter=options(14);  % 最大迭代次数
alpha=options(18);  % 步长

% 迭代
for k=1:max_iter
    xcurr=xnew;
    D=diag(xcurr);  % 对角矩阵
    Abar=A*D;  % 变换后的约束系数矩阵
    Pbar=eye(n)-Abar'*inv(Abar*Abar')*Abar;  % 投影矩阵
    d=-D*Pbar*D*c;  % 梯度方向
    if any(d~=zeros(n, 1))
        nonzd=find(d<0);
        r=min(-xcurr(nonzd)./d(nonzd));  % 步长
    else
        disp("终止: d = 0");
        break;
    end
    xnew=xcurr+alpha*r*d;  % 更新解
    if print
        disp("迭代次数 k =")
        disp(k);  % 输出迭代次数 k
        disp("alpha_k =");
        disp(alpha * r);  % 输出步长 alpha_k
        disp("新解 =");
        disp(xnew');  % 输出新解
    end
    if norm(xnew-xcurr)<=epsilon_x*norm(xcurr)
        disp("终止: 相邻两次迭代点的相对差小于");
        disp(epsilon_x);
        break;
    end
    if abs(c'*(xnew-xcurr))<epsilon_f*abs(c'*xcurr)
        disp("终止: 相邻两次迭代目标函数值的相对误差小于");
        disp(epsilon_f);
        break;
    end
    if k==max_iter
        disp("已达到最大迭代次数");
    end
end

% 输出结果
if nargout>= 1
    x=xnew;
    if nargout==2
        N=k;
    end
else
```

```
    disp("最终解：");
    disp(xnew');
    disp("迭代次数：");
    disp(k);
end %if
end
```

3. 例 16.5 题目

例 16.5 采用修正单纯形法求解线性规划：

$$\begin{array}{ll} \text{maximize} & 3x_1 + 5x_2 \\ \text{subject to} & x_1 + x_2 \leqslant 4 \\ & 5x_1 + 3x_2 \geqslant 8 \\ & x_1, x_2 \geqslant 0 \end{array}$$

首先，引入 1 个松弛变量和 1 个剩余变量，将该问题改写为标准型：

$$\begin{array}{ll} \text{minimize} & -3x_1 - 5x_2 \\ \text{subject to} & x_1 + x_2 + x_3 = 4 \\ & 5x_1 + 3x_2 - x_4 = 8 \\ & x_1, \cdots, x_4 \geqslant 0 \end{array}$$

该问题没有明显的基本可行解，因此应采用两阶段单纯形法。

4. 例 16.5 求解

```
%homework2.m
%test function with T16.5
A = [1,1,1,0;5,3,0,-1];
b = [4;8];
c = [-3;-5;0;0];
options = [1, 1e-6, 1e-6, 1000 * length(c), 0.99];
[x,N] = two_phase_affscale(A,b,c,options);
disp("最终解：");
disp(x');
disp("迭代次数：");
disp(N);
disp("最优函数值：");
disp(x'*c);
```

5. 例 16.5 结果

第一阶段：

```
>> homework2
第一阶段
-------
迭代次数 k =
     1
alpha_k =
   3.5922e+00
新解 =
   8.8099e-01   1.3345e+00   1.7663e+00   4.3572e-01   1.0000e-02
迭代次数 k =
     2
alpha_k =
   9.9006e+01
新解 =
   8.8316e-01   1.3399e+00   1.7768e+00   4.3575e-01   1.0000e-04
迭代次数 k =
     3
alpha_k =
   9.9000e+03
新解 =
   8.8319e-01   1.3399e+00   1.7769e+00   4.3575e-01   1.0000e-06
迭代次数 k =
     4
alpha_k =
   9.9000e+05
新解 =
   8.8319e-01   1.3399e+00   1.7769e+00   4.3575e-01   1.0000e-08
终止：相邻两次迭代点的相对差小于
   1.0000e-06
进行第二阶段的初始条件：
   8.8319e-01
   1.3399e+00
   1.7769e+00
   4.3575e-01
   1.0000e-08
```

第二阶段：

```
第二阶段
---------
进行第二阶段的初始条件：
   8.8319e-01
   1.3399e+00
   1.7769e+00
   4.3575e-01
迭代次数 k =
     1
alpha_k =
   4.9340e-01
新解 =
   8.8319e-03   2.8277e+00   1.1635e+00   5.2715e-01
迭代次数 k =
     2
alpha_k =
   7.6469e+00
新解 =
   5.7137e-03   3.9827e+00   1.1635e-02   3.9765e+00
迭代次数 k =
     3
alpha_k =
   1.7019e+01
新解 =
   4.6024e-03   3.9953e+00   1.1635e-04   4.0089e+00
```

```
迭代次数 k =
     4
alpha_k =
   1.0755e+02
新解 =
   4.6024e-05   3.9998e+00   1.0907e-04   3.9998e+00
迭代次数 k =
     5
alpha_k =
   1.8153e+03
新解 =
   3.8333e-05   4.0000e+00   1.0907e-06   4.0001e+00
迭代次数 k =
     6
alpha_k =
   1.2913e+04
新解 =
   3.8333e-07   4.0000e+00   1.0139e-06   4.0000e+00
迭代次数 k =
     7
alpha_k =
   1.9529e+05
新解 =
   3.2594e-07   4.0000e+00   1.0139e-08   4.0000e+00
终止：相邻两次迭代点的相对差小于
   1.0000e-06
```

最终结果：

```
终止：相邻两次迭代点的相对差小于
   1.0000e-06
最终解:
   3.2594e-07   4.0000e+00   1.0139e-08   4.0000e+00
迭代次数:
     7
最优函数值:
  -2.0000e+01
```