最优化方法课程论文

基于 MATLAB 的单纯形法及最速下降法的实现

小组成员：张　慧　[redacted]

江　泓　[redacted]

指导老师：[redacted]

基于 MATLAB 的单纯形算法的实现

一、算法简述

考虑标准的线性规划问题：

$$\begin{aligned} \min \quad & c^T x \\ \text{s.t.} \quad & \mathrm{Ax} = \mathrm{b} \\ & \mathrm{x} \geq 0 \end{aligned}$$

求解上述线性规划标准型的单纯形算法步骤如下：

(1) 给定一初始基本可行解$\bar{x}$，确定基阵B，解B$x_B = b$,求得$x_B = B^{-1}b$。

(2) 对于非基变量，计算判别数 $\Delta = c_N^T - c_B^T B^{-1} N$，等价于计算 $\Delta = \mathrm{c} - c_B^T B^{-1} A$，记$\Delta_k = \max\{\Delta_1, \dots, \Delta_n\}$，若 $\Delta_k \geq 0$，则已得到一个最优基本可行解，停止；否则进行下一步。

(3) 解B$y_k = a_k$, 得$y_k = B^{-1}a_k$, 若$y_k \leq 0$, 即y_k的每个分量均小于或等于零，则该问题不存在有限最优解；否则进行下一步。

(4) 求i_r满足：

$$\frac{(B^{-1}\mathrm{b})_{i_r}}{(B^{-1}a_k)_{i_r}} = \min\left\{ \frac{(B^{-1}\mathrm{b})_{i_j}}{(B^{-1}a_k)_{i_j}} \middle| (B^{-1}a_k)_{i_j} > 0 \right\}$$

(5) 以a_k代替B中的a_{i_r}，转步骤(2)。

二、MATLAB 程序

```
clear                    %清空工作区
clc                      %清空命令输入框
A=input('A=');           %输入原始数据
b=input('b=');
c=input('c=');
format   rat             %结果可以用分数表示
[m,n]=size(A);           %取A的维数
E=1:m;E=E';
```

```
F=n-m+1:n;F=F';
D=[E,F];                    %创建一个一一映射，使得结果能够标准输出
X=zeros(1,n);               %产生1*n的double类零矩阵，用以初始化X
if(n<m)                     %判断最优化问题是否为标准型
    fprintf('不符合标准形式，需引入松弛变量或剩余变量')
    flag=0;
else
    flag=1;
    B=A(:,n-m+1:n);        %找基矩阵
    cB=c(n-m+1:n);         %找基变量对应目标函数中的系数
   while flag
      w=cB/B                  %cB/B=cB*inv(B),右除相当于求逆
      Delta=c-w*A             %计算检验数Delta
      [z,k]=min(Delta)        %k作为进基变量的下标
      fprintf('确定下标，选择进基变量和出基变量为\n',k);
      (B\b')./(B\A(:,k))           %这个式子是为了确定基变量的下标
      if(z>-0.000000000001)     %为了使判别数尽可能趋近于零
         flag=0;                   %所有判别数都大于0时达到最优解
         fprintf('已找到最优解!\n');
         xB=(B\b')';
         f=cB*xB';                   %得到最优解与目标函数值
         for i=1:n
            mark=0;
               for j=1:m
                  if (D(j,2)==i)          %找到基变量对应的指标
                     mark=1;
                     X(i)=xB(D(j,1))      %得到基变量的值
                  end
               end
               if mark==0
                  X(i)=0;                 %如果基指标集中没有i,则X(i)为非基变量，故X(i) = 0
               end
         end
            fprintf('基向量为:') ;   X
            fprintf('目标函数值为:') ;   f
      else
         if(B\A(:,k)<=0)           %如果B\A(:,k)中的每一个分量都小于零
```

```
                flag=0;
                fprintf(' \n  该问题不存在最优解! \n');
                 %如果B\A(:,k)的第k列中每一个分量都小于0，则该问题不存在最优解
            else
                b1=B\b';
                temp=inf;           %定义一个正无穷大的temp
                for i=1:m
                    if ((A(i,k)>0) && (b1(i)/(A(i,k)+eps))<temp )
                        temp=b1(i)/A(i,k);
                        r=i;        %找到出基变量对应的指标
                    end
                end
                fprintf('x(%d)进基，x(%d)出基\n',k,D(r,2));       %显示进基变量和出基变量
                B(:,r)=A(:,k)
                cB(r)=c(k)
                 %确定进基离基变量后，相应的基矩阵及新基对应的目标值的c也相应改
变
                D(r,2)=k;           %改变D中的映射关系
            end
        end
    end
end
```

三、算法测试

原问题：

$$\begin{aligned} \max \quad & 5x_1+6x_2+4x_3 \\ \text{s.t.} \quad & 2x_1+2x_2\le 5 \\ & 5x_1+3x_2+4x_3\le 15 \\ & x_1+x_2\le 10 \\ & x_1,x_2,x_3\ge 0 \end{aligned}$$

标准化：

$$\begin{aligned} \min \quad & -5x_1-6x_2-4x_3 \\ \text{s.t.} \quad & 2x_1+2x_2+x_4=5 \\ & 5x_1+3x_2+4x_3+x_5=15 \\ & x_1+x_2+x_6=10 \\ & x_i\ge 0, i=1,\dots,6 \end{aligned}$$

输入：

A=[2 2 0 1 0 0;5 3 4 0 1 0;1 1 0 0 0 1]

b=[5 15 10]

c=[-5 -6 -4 0 0 0]

运行结果：

```
w =
     0     0     0

Delta =
    -5    -6    -4     0     0     0

z =
    -6

k =
     2
```

确定下标，选择进基变量和出基变量为

```
ans =
     5/2
     5
    10
```

x(2)进基，x(4)出基

```
B =
     2     0     0
     3     1     0
     1     0     1

cB =
    -6     0     0

w =
    -3     0     0

Delta =
     1     0    -4     3     0     0

z =
    -4
```

k =

3

确定下标，选择进基变量和出基变量为

ans =

1/0
15/8
1/0

x(3)进基，x(5)出基

B =

2	0	0
3	4	0
1	0	1

cB =

-6	-4	0

w =

-3/2	-1	0

Delta =

3	0	0	3/2	1	0

z =

0

k =

2

确定下标，选择进基变量和出基变量为

ans =

5/2
1/0
1/0

已找到最优解!

X =

0	5/2	0	0	0	0

X =

0	5/2	15/8	0	0	0

X =

0	5/2	15/8	0	0	15/2

基向量为:

X =

0	5/2	15/8	0	0	15/2

目标函数值为:

f =

-45/2

基于 MATLAB 的最速下降法实现

一．原理简述

无约束优化问题指下列优化问题

$$\min f(x)$$

求解此类问题的常用的迭代格式为

$$x_{k+1} = x_k + \alpha_k d_k,\ k = 0,1,\ldots$$

x_0为初始向量，d_k为f(x)在x_k处的下降方向，$\alpha_k > 0$为步长。在最速下降法中，d_k取负梯度方向-g_k。步长采用Armijo准则进行非精确一维搜索。

Armijo准则：

设f(x)连续可微，d_k是f(x)在x_k处的下降方向，给定$\rho \in (0,\frac{1}{2}), \beta \in (0,1)$，我们寻找使得下式成立的最小正整数$m_k$:

$$f(x_k + \beta^m d_k) \le f(x_k) + \rho\beta^m {g_k}^T d_k$$

我们需要的步长$\alpha_k = \beta^{m_k}$

二．算法简述

步骤1：给出初值x_o以及精度eps

步骤2：计算gk = -∇f(x_k);如果$|g_k| < eps$,停止，输出x_k；否则转步骤3。

步骤3.由Armijo准则搜索线性步长因子α_k

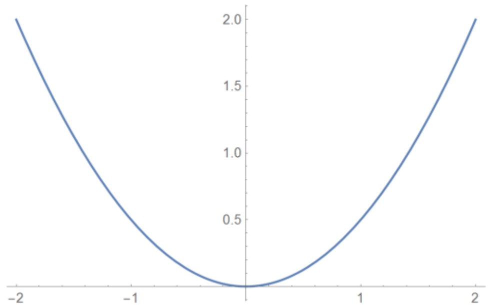
步骤4.计算$x_{k+1} = x_k + \alpha_k d_k$ ，$k = k + 1$,转步骤2.

三. Matlab实现

程序包含 4 部分，分别是最速下降法主函数 steepest1.m；求梯度函数 fun_grad1.m；测试函数 fun1.m;Armijo 求步长因子函数 armijo1.m

1.测试函数：f(t) = 1/2*t^2

函数大致图像：



具体实现：

armijo1.m × fun1.m × fun_grad1.m × steepest1.m × +

```
function [ f ] = fun1( t )
%一元测试函数

f = 1/2*t^2;

end
```

armijo1.m × fun1.m × fun_grad1.m × steepest1.m × +

```
function [ gk ] = fun_grad1( t0 )
%对函数求导

syms t

gk = subs(diff(fun1(t)),t,t0);

end
```

armijo1.m × fun1.m × fun_grad1.m × steepest1.m × +

```
function [ alpha ] = armijo1( t0 )
%用armijo准则搜索合适的步长因子alpha
%如果输出的aplha=beta^mmax，步长可能过小，输出提醒

beta=0.5;rho=0.2; %规定参数

m=0;mmax=20; %m为搜索次数，mmax为最大搜索次数

while(m<=mmax)

    gk=fun_grad1(t0); %利用fun_grad1()函数计算导数

    dk=-gk; %下降方向为导数负值

    t1=t0+(beta^m)*dk; %计算下一点的取值

    if(fun1(t1) <= fun1(t0) + rho*(beta^m)*gk*dk')
        %检验此时的m是否使判断式成立

       alpha=beta^m;return; %如果成立的话则输出此时的alpha并结束程序

    end

    t0=t1; m=m+1;
end

alpha=beta^m;

disp('alpha is already (1/2)^11,the step may be too little');
%进行步长过小的提醒

end
```

armijo1.m fun1.m fun_grad1.m steepest1.m

```matlab
function [ tk ] = steepest1( t0,eps )
%最速下降法主函数
%   t0是初值，eps是规定的精度，当某个点的梯度值小于eps，迭代结束

k=0;kmax=5000; %规定最大循环次数、

while(k<kmax)

    gk=fun_grad1(t0); %算导数

    if(abs(gk)<eps) %判断是否达到精度

       tk=t0;

       disp(tk);return; %输出此时的t点

    else

        dk=-gk; %计算此时的导数

        alpha=armijo1(t0); %计算次点的步长因子

        t0=t0+alpha*dk; %计算新的一点

        k=k+1; %计数变量k+1;

    end

end
tk=t0; disp('达到最大搜索次数，但还未达到精度');

end
```

测试结果：

Command Window

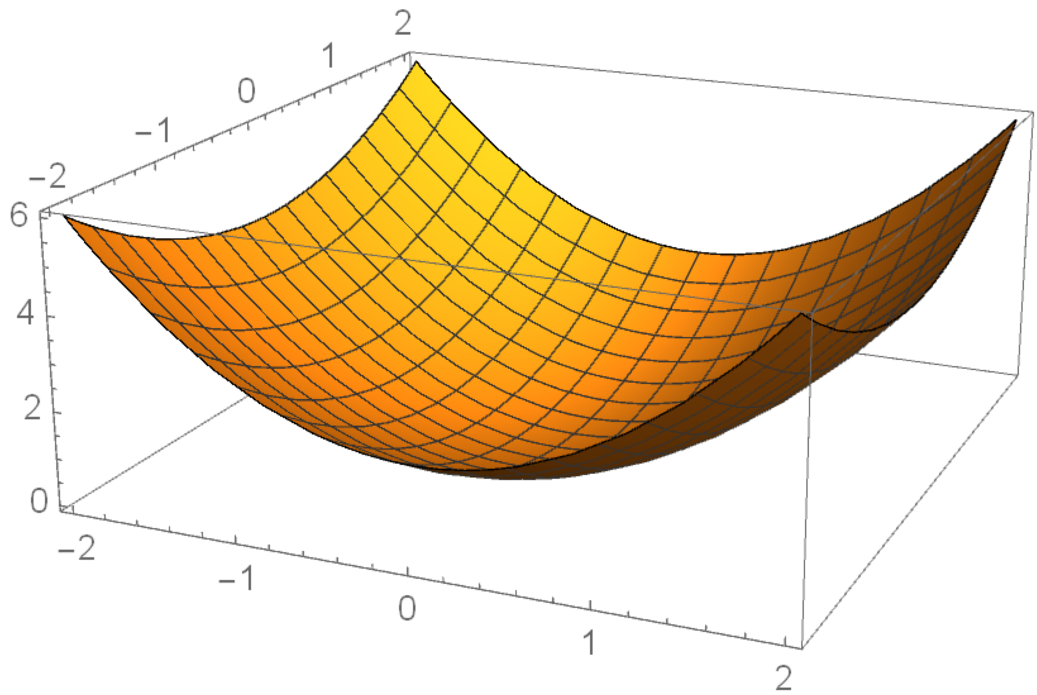
```
>> steepest1(2,1e-6)
0


ans =

0
```

2.测试函数：f (x1,x2) = x1^2+2*x1^2

函数图像：



具体实现：

armijo2.m　fun2.m　grad.m　steepest2.m

```
function [ g ] = fun2( v )
% 进行无约束优化的目标函数表达式

x1=0;x2=0;%先分配初值

x1=v(1);x2=v(2);

g=x1^2-2*x1*x2+4*x2^2+x1-3*x2; %函数表达式

end
```

armijo2.m × | fun2.m × | grad.m × | steepest2.m × | +

```matlab
function [ j ] = grad(v2)
 %用diff求目标函数对于x1,x2的一阶偏导数，再带入值求该点梯度
syms x1 x2

v=[x1,x2];

v1=v2;

j1=[diff(fun2(v),x1),diff(fun2(v),x2)]; %计算f关于x1, x2的偏导数

j=[subs(j1(1),v,v1),subs(j1(2),v,v1)]; %带入值计算该点的梯度

end
```

armijo2.m × | fun2.m × | grad.m × | steepest2.m × | +

```matlab
function [ alpha ] = armijo2( v )
% armijo2用于求函数f(x1,x2) = (x1+x2)/(3+x1^2+x2^2+x1*x2)的步长因子
%如果输出的aplha=beta^mmax, 步长可能过小, 输出提醒
v1=v;

beta=0.5;rho=0.2;  %beta应在0-1之间, rho应在0-1/2之间

m=1;mmax=20;% (1/2)^10已经太小了, 在进行搜索步长就不合适了

while(m<=mmax)

    gk=grad(v1); %用grad函数求v1点的梯度

    dk=-gk; %下降方向取负梯度

    v2=v1+(beta^m)*dk; %计算下一点

    if(fun2(v2)<=fun2(v1)+rho*(beta^m)*gk*dk') %判断m是否使该式成立

        alpha=beta^m;return;  %若成立则输出alpha=beta^m
    end

    v1=v2;m=m+1;

end
alpha=beta^m; %若在10次之后还没有找到合适的beta, 直接输出

disp('alpha is already (1/2)^11,the step may be too little');
%输出步长过小的提醒

end
```

armijo2.m fun2.m grad.m steepest2.m

```
function [ vk ] = steepest2( v,eps )
% 求二元函数的极小值点
% 为了方便表示，变量的输入使用向量形式

v1=v;

dk=-grad(v1);

k=0;kmax=5000;%kmax为最大循环次数

while(k<kmax)

if (norm(dk)<eps) %判断梯度的范数是否已经达到精度

    disp('此时已经找到符合精度的点');
    vk=vpa(v1);return; %若达到精度的话输出此时的xk还有梯度值gk=-dk

else

    dk=-grad(v1); %若不满足精度，则计算此处下降方向dk

    alpha=armijo2(v1); %计算此处的步长alpha

    v1=v1+alpha*dk; %计算xk+1

    dk=-grad(v1); %计算xk+1的梯度，用于与精度比较

    k=k+1; %循环次数+1

    disp(k);disp(vpa(dk)); %用于观察进程
end
end
end
```

测试结果：

```
此时已经找到符合精度的点

ans =

[ -0.16666638851165771484375, 0.3333334922790527734375]
```