姓名：王旋宇
学号：2012049010022
班级：数理基科班
老师：赖生建

2015-10-27

1 实验目的

1. 理解单变量函数求极值的两种分类搜索方法，即黄金分割搜索法和斐波那契搜索法。并能编程实现。
2. 理解单变量函数求极值的利用倒数求极小值法。并能编程实现。
3. 理解内德-米德方法的步骤，并能编程实现。
4. 理解最速下降法，并能编程实现。

2 实验原理

2 .1 准备定理

定理1 . 一阶导数测试　设$f(x)$在$I=[a,b]$上连续，并设除$x=p$处外，$f'(x)$对所有$x\in(a,b)$都有定义

1. 若在(a,p)上$f'(x)<0$，而在(p,b)上$f'(x)>0$，则$f(p)$是局部极小值。
2. 若在(a,p)上$f'(x)>0$，而在(p,b)上$f'(x)<0$，则$f(p)$是局部极大值。

定理2 . 二阶导数测试　设$f(x)$在$I=[a,b]$上连续，并且f',f''在区间(a,b)上有定义。又设$p\in(a,b)$是关键点，即$f'(p)=0$

1. 若$f''(p)>0$，则$f(p)$是f的一个局部极小值。
2. 若$f''(p)<0$，则$f(p)$是f的局部极大值。
3. 若$f''(p)=0$，结果不确定。

2 .2 黄金分割搜索

如果已知$f(x)$在(a,b)上是单峰的，则选择两个内点$c<d$。此时需要考虑两种情况

1. 如果$f(c)\le f(d)$，则从右侧压缩区间，使用[a,d]
2. 如果$f(c)>f(d)$，则从左侧压缩区间，使用[c,b]

同时，c、d满足$b-d=c-a$，其中

$$c=a+(1-r)(b-a)=ra+(1-r)b \tag{1}$$
$$d=b-(1-r)(b-a)=(1-r)a+rb \tag{2}$$

并且$1/2<r<1$。要求旧的内点中的一个应该是新的子区间中的一个内点，而另一个则应该是新子区间的一个端点。于是得出$r=\frac{-1\pm\sqrt{5}}{2}$

2 .3 斐波那契搜索法

设给定函数$f(x)$在区间$[a_0, b_0]$上是单峰。选择一个值$r_0(1/2 < r_0 < 1)$，使得内点c_0, d_0可以在下一个子区间上使用，从而只需一次新的函数求值计算。它满足$a_1 = a_0, b_1 = d_0, d_1 = c_0$，如果只能进行一次新的函数求值计算，则为子区间$[a_1, b_1]$选择$r_1(1/2 < r_1 < 1)$，则有

$$r_1 = \frac{1 - r_0}{r_0} \tag{3}$$

令$r_0 = F_{n-1}/F_n$代入(3)得$r_1 = \frac{F_{n-1}}{F_n}$。依此类推第k歌子区间按因子$r_k = \frac{F_{n-1-k}}{F_{n-k}}$缩减，得到下一个子区间，最后一个子区间的长度为$\frac{b_0 - a_0}{F_n}$。由公式

$$c_k = a_k + (1 - \frac{F_{n-1-k}}{F_{n-k}})(b_k - a_k) \tag{4}$$

$$d_k = a_k + \frac{F_{n-1-k}}{F_{n-k}}(b_k - a_k) \tag{5}$$

需要注意的是，当$\frac{F_{n-1-k}}{F_{n-k}} = \frac{1}{2}$时两个内点将在区间重点重合，为了区分它们，引入一个小的区别常数。当使用公式(4)和(5)时$(b_k - a_k)$的系数分别是1/2-e和1/2+e。

2 .4 利用导数求极小值

设f(x)在区间[a,b]是单峰的，并在$x = p$处有唯一极小值。并设$f'(x)$在(a,b)上所有的点处有定义。令初始点p_0在(a, b)内。

1. **对极小值分类**
   首先求出三个测试值

   $$p_0, \qquad p_1 = p_0 + h, \qquad p_2 = p_0 + 2h \tag{6}$$

   使得

   $$f(p_0) > f(p_1), \qquad f(p_1) < f(p_2) \tag{7}$$

2. **求极小值p的二次逼近方法**
   得到三个满足(7)的点，并且可以用来二次插值得到拉格朗日多项式为Q(x)及其导数Q'(x)，以$Q'(p + h_{min})$的形式解$Q'(x) = 0$，得

   $$h_{min} = \frac{h(4y_1 - 3y_0 - y_2)}{4y_1 - 2y_0 - 2y_2} \tag{8}$$

   $p_0 + h_{min}$比p_0更好的逼近p，因此用$p_0 + h_{min}$替代p_0，并重复计算以上计算过程，求出新的h_{min}，重复迭代这一过程，直到得到所需的精度。

2 .5　内德-米德方法

概念介绍

1. 初始三角形　首先给定三角形的三个顶点$\mathbf{V_k} = (x_k, y_k),\ k = 1, 2, 3$。有

$$\mathbf{B} = (x_1, y_1) \qquad \mathbf{G} = (x_2, y_2) \qquad \mathbf{W} = (x_3, y_3) \tag{9}$$

并且，$z_k = f(x_k, y_k)$

$$z_1 \leq z_2 \leq z_3 \tag{10}$$

2. 良边的中点$\mathbf{M} = \frac{\mathbf{B}+\mathbf{G}}{2} = (\frac{x_1+x_2}{2}, \frac{y_1+y_2}{2})$
3. 反射点R　$\mathbf{R} = \mathbf{M} + (\mathbf{M} - \mathbf{W}) = 2\mathbf{M} - \mathbf{W}$
4. 开拓点E　$\mathbf{E} = \mathbf{R} + (\mathbf{R} - \mathbf{M}) = 2\mathbf{R} - \mathbf{M}$
5. 收缩点C　$\overline{WM}$和$\overline{MR}$上的中点函数值最小的点为C。
6. 向B方向收缩　点G置换为M，点W置换为S，后者是连接B和W的线段中点。

内德-米德方法的逻辑判断　判断逻辑如下所示

IF $f(R) < f(G)$, THEN Perform Case (i) {either reflect or extend}
ELSE Perform Case (ii) {either contract or shrink}

```
BEGIN {Case (i).}                    BEGIN {Case (ii).}
IF f(B) < f(R) THEN                  IF f(R) < f(W) THEN
   replace W with R                     replace W with R
ELSE                                 Compute C = (W + M)/2
                                     or C = (M + R)/2 and f(C)
   Compute E and f(E)                IF f(C) < f(W) THEN
   IF f(E) < f(B) THEN                  replace W with C
      replace W with E               ELSE
   ELSE                                 Compute S and f(S)
      replace W with R                replace W with S
   ENDIF                              replace G with M
ENDIF                                ENDIF
END {Case (i).}                      END {Case (ii).}
```

Fig 1: 内德-米德方法的逻辑判断

2 .6　最速下降法（梯度方法）

定义1 . 设$z = f(\mathbf{X})$是$\mathbf{X}$，对$k = 1, 2, \cdots, N, \partial f(\mathbf{X})/\partial x_K$存在。$f$的梯度，记为$\nabla f(\mathbf{X})$，是向量

$$\nabla f(\mathbf{X}) = (\frac{\partial f(\mathbf{X})}{\partial x_1}, \frac{\partial f(\mathbf{X})}{\partial x_2}, \cdots, \frac{\partial f(\mathbf{X})}{\partial x_N},) \tag{11}$$

梯度方法概要　设$\mathbf{P}_k$易知。

1. 求梯度向量$\nabla f(\mathbf{P}_k)$
2. 计算搜索方向$\mathbf{S}_k = -\nabla f(\mathbf{P}_k)/\| - \nabla f(\mathbf{P}_k)\|$
3. 在区间[0,b]上对$\Phi(\gamma) = f(\mathbf{P}_k + \gamma\mathbf{S}_k)$进行但参数极小化，b为一个较大值。这一过程将产生值$\gamma = h_{min}$，他是$\Phi(\gamma)$的一个局部极小值点。关系式$\Phi(hmin) = f(\mathbf{P}_k + h_{min}\mathbf{S}_k)$表明，他是$f(\mathbf{X})$ 沿搜索线$\mathbf{X} = \mathbf{P}_k + h_{min}\mathbf{S}_k$的一个极小值。
4. 构造下一个点$\mathbf{P}_{k+1} = \mathbf{P}_k + h_{min}\mathbf{S}_k$。
5. 进行极小化过程的终止判断，即函数值$f(\mathbf{P}_k)$和$f(\mathbf{P}_{k+1})$是否足够接近，并且距离$\|\mathbf{P}_{k+1} - \mathbf{P}_k\|$ 是否足够小。

3　实验内容

- P331

  a $f(x) = e^x + 2x + \frac{x^2}{2};\ [-2.4, -1.6]$

  b $f(x) = -sin(x) - x + \frac{x^2}{2};\ [0.8, 1.6]$

  c $f(x) = \frac{x^2}{2} - 4x - xcos(x);\ [0.5, 2.5]$

  d $f(x) = x^3 - 5x^2 + 23;\ [1, 5]$

  - 1.　用黄金分割搜索法求习题中各函数的局部极小值精确到小数点后6位。
  - 2.　用斐波那契法搜索法求习题中各函数的局部极小值精确到小数点后6位。
  - 3.　用2次插值求最小值求习题中各函数的局部极小值精确到小数点后6位。

- P342 2.　用内德-米德方法求以下函数的局部极小值，精确到小数点后8位。$f(x, y) = x^2 - 4x + y^2 - y - xy$。从(0,0),(1,2,0),(0,0.8)开始
- P352 7.　求表面$z = x^2 + y^2$上与点$(2, 3, 1)$距离最近的点，精确到小数点后7位。

4　实验分析

P331 1.　最后判断条件采用为两个端点函数值的之差来判断，当差值小于一定值得时候计算终止。

P331 2. 由于斐波那契搜索法的最后一个区间长度为$\frac{b_0-a_0}{F_n}$，设极小值横坐标的容差为ϵ，当

$$\frac{b_0-a_0}{F_n}<\epsilon \qquad 和 \qquad F_n>\frac{b_0-a_0}{\epsilon} \tag{12}$$

时就可以计算结束了。

P331 3. 在寻找满足条件的三个测试值的时候，算法中应该再添加一条某个测试值恰好是极值点的情况的处理。其余部分依旧按照书中所描述算法即可。

P342 2. 按照(??)中所示逻辑判断设计程序即可。

P331 7. 按照梯度法的原理设计程序即可。在计算局部极小值点的时候调用之前的二次插值寻找最小值程序。其中被调用的函数需要修改输入参数。

5 实验结果

5 .1 P331 1.

k	a_k	c_k	d_k	b_k	$f(c_k)$	$f(d_k)$
0	-2.4000000	-2.0944272	-1.9055728	-1.6000000	-1.8724010	-1.8468043
1	-2.4000000	-2.2111456	-2.0944272	-1.9055728	-1.8681337	-1.8724010
2	-2.2111456	-2.0944272	-2.0222912	-1.9055728	-1.8724010	-1.8673997
3	-2.2111456	-2.1390097	-2.0944272	-2.0222912	-1.8725667	-1.8724010
4	-2.2111456	-2.1665631	-2.1390097	-2.0944272	-1.8715577	-1.8725667
5	-2.1665631	-2.1390097	-2.1219807	-2.0944272	-1.8725667	-1.8727662
6	-2.1390097	-2.1219807	-2.1114562	-2.0944272	-1.8727662	-1.8727272
7	-2.1390097	-2.1284852	-2.1219807	-2.1114562	-1.8727283	-1.8727662
8	-2.1284852	-2.1219807	-2.1179607	-2.1114562	-1.8727662	-1.8727660

Table 1: P331 1. a的计算结果

k	a_k	c_k	d_k	b_k	$f(c_k)$	$f(d_k)$
0	0.8000000	1.1055728	1.2944272	1.6000000	-1.3881485	-1.4187088
1	1.1055728	1.2944272	1.4111456	1.6000000	-1.4187088	-1.4027625
2	1.1055728	1.2222912	1.2944272	1.4111456	-1.4151776	-1.4187088
3	1.2222912	1.2944272	1.3390097	1.4111456	-1.4187088	-1.4157937
4	1.2222912	1.2668737	1.2944272	1.3390097	-1.4185591	-1.4187088
5	1.2668737	1.2944272	1.3114562	1.3390097	-1.4187088	-1.4180569
6	1.2668737	1.2839027	1.2944272	1.3114562	-1.4188272	-1.4187088
7	1.2668737	1.2773982	1.2839027	1.2944272	-1.4187918	-1.4188272
8	1.2773982	1.2839027	1.2879227	1.2944272	-1.4188272	-1.4188076
9	1.2773982	1.2814182	1.2839027	1.2879227	-1.4188234	-1.4188272
10	1.2814182	1.2839027	1.2854382	1.2879227	-1.4188272	-1.4188234
11	1.2814182	1.2829537	1.2839027	1.2854382	-1.4188272	-1.4188272

Table 2: P331 1. b的计算结果

k	a_k	c_k	d_k	b_k	$f(c_k)$	$f(d_k)$
0	0.5000000	1.2639320	1.7360680	2.5000000	-4.6387631	-5.1516875
1	1.2639320	1.7360680	2.0278640	2.5000000	-5.1516875	-5.1604055
2	1.7360680	2.0278640	2.2082039	2.5000000	-5.1604055	-5.0806002
3	1.7360680	1.9164079	2.0278640	2.2082039	-5.1800964	-5.1604055
4	1.7360680	1.8475242	1.9164079	2.0278640	-5.1786628	-5.1800964
5	1.8475242	1.9164079	1.9589803	2.0278640	-5.1800964	-5.1756293
6	1.8475242	1.8900966	1.9164079	1.9589803	-5.1808482	-5.1800964
7	1.8475242	1.8738354	1.8900966	1.9164079	-5.1805180	-5.1808482
8	1.8738354	1.8900966	1.9001466	1.9164079	-5.1808482	-5.1807467
9	1.8738354	1.8838854	1.8900966	1.9001466	-5.1807947	-5.1808482
10	1.8838854	1.8900966	1.8939354	1.9001466	-5.1808482	-5.1808368
11	1.8838854	1.8877241	1.8900966	1.8939354	-5.1808383	-5.1808482
12	1.8877241	1.8900966	1.8915629	1.8939354	-5.1808482	-5.1808478

Table 3: P331 1. c的计算结果

k	a_k	c_k	d_k	b_k	$f(c_k)$	$f(d_k)$
0	1.0000000	2.5278640	3.4721360	5.0000000	7.2028124	4.5804865
1	2.5278640	3.4721360	4.0557281	5.0000000	4.5804865	7.4677371
2	2.5278640	3.1114562	3.4721360	4.0557281	4.7167059	4.5804865
3	3.1114562	3.4721360	3.6950483	4.0557281	4.5804865	5.1829961
4	3.1114562	3.3343685	3.4721360	3.6950483	4.4814868	4.5804865
5	3.1114562	3.2492236	3.3343685	3.4721360	4.5162587	4.4814868
6	3.2492236	3.3343685	3.3869910	3.4721360	4.4814868	4.4960317
7	3.2492236	3.3018461	3.3343685	3.3869910	4.4864075	4.4814868
8	3.3018461	3.3343685	3.3544685	3.3869910	4.4814868	4.4837244
9	3.3018461	3.3219461	3.3343685	3.3544685	4.4821284	4.4814868
10	3.3219461	3.3343685	3.3420461	3.3544685	4.4814868	4.4818617
11	3.3219461	3.3296236	3.3343685	3.3420461	4.4815502	4.4814868
12	3.3296236	3.3343685	3.3373011	3.3420461	4.4814868	4.4815603
13	3.3296236	3.3325561	3.3343685	3.3373011	4.4814845	4.4814868
14	3.3296236	3.3314360	3.3325561	3.3343685	4.4814995	4.4814845
15	3.3314360	3.3325561	3.3332484	3.3343685	4.4814845	4.4814815
16	3.3325561	3.3332484	3.3336763	3.3343685	4.4814815	4.4814821

Table 4: P331 1. d的计算结果

5 .2 P331 2.

k	a_k	c_k	d_k	b_k	$f(c_k)$	$f(d_k)$
0	-2.4000000	-2.0944272	-1.9055728	-1.6000000	-1.8724010	-1.8468043
1	-2.4000000	-2.2111456	-2.0944272	-1.9055728	-1.8681337	-1.8724010
2	-2.2111456	-2.1390097	-2.0944272	-2.0222912	-1.8725667	-1.8724010
3	-2.2111456	-2.1665631	-2.1390097	-2.0944272	-1.8715577	-1.8725667
4	-2.1665631	-2.1390097	-2.1219807	-2.0944272	-1.8725667	-1.8727662
5	-2.1390097	-2.1284852	-2.1219807	-2.1114562	-1.8727283	-1.8727662
6	-2.1284852	-2.1244652	-2.1219807	-2.1179607	-1.8727573	-1.8727662
7	-2.1244652	-2.1219807	-2.1204452	-2.1179607	-1.8727662	-1.8727683
8	-2.1219807	-2.1210317	-2.1204452	-2.1194962	-1.8727678	-1.8727683
9	-2.1210317	-2.1204452	-2.1200827	-2.1194962	-1.8727683	-1.8727684
10	-2.1204452	-2.1202211	-2.1200827	-2.1198587	-1.8727684	-1.8727684
11	-2.1202211	-2.1200827	-2.1199971	-2.1198587	-1.8727684	-1.8727684
12	-2.1200827	-2.1200298	-2.1199971	-2.1199442	-1.8727684	-1.8727684
13	-2.1200827	-2.1200500	-2.1200298	-2.1199971	-1.8727684	-1.8727684
14	-2.1200500	-2.1200375	-2.1200298	-2.1200173	-1.8727684	-1.8727684
15	-2.1200375	-2.1200327	-2.1200298	-2.1200250	-1.8727684	-1.8727684
16	-2.1200327	-2.1200298	-2.1200280	-2.1200250	-1.8727684	-1.8727684
17	-2.1200298	-2.1200287	-2.1200280	-2.1200269	-1.8727684	-1.8727684
18	-2.1200287	-2.1200282	-2.1200280	-2.1200276	-1.8727684	-1.8727684
19	-2.1200287	-2.1200284	-2.1200282	-2.1200280	-1.8727684	-1.8727684
20	-2.1200284	-2.1200283	-2.1200282	-2.1200281	-1.8727684	-1.8727684
21	-2.1200283	-2.1200283	-2.1200282	-2.1200282	-1.8727684	-1.8727684
22	-2.1200283	-2.1200283	-2.1200282	-2.1200282	-1.8727684	-1.8727684
23	-2.1200283	-2.1200283	-2.1200283	-2.1200282	-1.8727684	-1.8727684
24	-2.1200283	-2.1200283	-2.1200283	-2.1200283	-1.8727684	-1.8727684
25	-2.1200283	-2.1200283	-2.1200283	-2.1200283	-1.8727684	-1.8727684
26	-2.1200283	-2.1200283	-2.1200283	-2.1200283	-1.8727684	-1.8727684
27	-2.1200283	-2.1200283	-2.1200283	-2.1200283	-1.8727684	-1.8727684
28	-2.1200283	-2.1200283	-2.1200283	-2.1200283	-1.8727684	-1.8727684
29	-2.1200283	-2.1200283	-2.1200283	-2.1200283	-1.8727684	-1.8727684
30	-2.1200283	-2.1200283	-2.1200283	-2.1200283	-1.8727684	-1.8727684
31	-2.1200283	-2.1200283	-2.1200283	-2.1200283	-1.8727684	-1.8727684
32	-2.1200283	-2.1200283	-2.1200283	-2.1200283	-1.8727684	-1.8727684

Table 5: P331 2. a的计算结果

k	a_k	c_k	d_k	b_k	$f(c_k)$	$f(d_k)$
0	0.8000000	1.1055728	1.2944272	1.6000000	-1.3881485	-1.4187088
1	1.1055728	1.2222912	1.2944272	1.4111456	-1.4151776	-1.4187088
2	1.2222912	1.2668737	1.2944272	1.3390097	-1.4185591	-1.4187088
3	1.2668737	1.2839027	1.2944272	1.3114562	-1.4188272	-1.4187088
4	1.2668737	1.2773982	1.2839027	1.2944272	-1.4187918	-1.4188272
5	1.2773982	1.2814182	1.2839027	1.2879227	-1.4188234	-1.4188272
6	1.2814182	1.2829537	1.2839027	1.2854382	-1.4188272	-1.4188272
7	1.2829537	1.2835402	1.2839027	1.2844892	-1.4188274	-1.4188272
8	1.2829537	1.2833162	1.2835402	1.2839027	-1.4188274	-1.4188274
9	1.2833162	1.2834546	1.2835402	1.2836787	-1.4188274	-1.4188274
10	1.2833162	1.2834018	1.2834546	1.2835402	-1.4188274	-1.4188274
11	1.2834018	1.2834344	1.2834546	1.2834873	-1.4188274	-1.4188274
12	1.2834018	1.2834220	1.2834344	1.2834546	-1.4188274	-1.4188274
13	1.2834220	1.2834297	1.2834344	1.2834422	-1.4188274	-1.4188274
14	1.2834220	1.2834267	1.2834297	1.2834344	-1.4188274	-1.4188274
15	1.2834267	1.2834285	1.2834297	1.2834315	-1.4188274	-1.4188274
16	1.2834267	1.2834279	1.2834285	1.2834297	-1.4188274	-1.4188274
17	1.2834279	1.2834283	1.2834285	1.2834290	-1.4188274	-1.4188274
18	1.2834283	1.2834285	1.2834287	1.2834290	-1.4188274	-1.4188274
19	1.2834285	1.2834286	1.2834287	1.2834288	-1.4188274	-1.4188274
20	1.2834286	1.2834287	1.2834288	1.2834288	-1.4188274	-1.4188274
21	1.2834287	1.2834287	1.2834288	1.2834288	-1.4188274	-1.4188274
22	1.2834287	1.2834287	1.2834287	1.2834288	-1.4188274	-1.4188274
23	1.2834287	1.2834287	1.2834287	1.2834287	-1.4188274	-1.4188274
24	1.2834287	1.2834287	1.2834287	1.2834287	-1.4188274	-1.4188274
25	1.2834287	1.2834287	1.2834287	1.2834287	-1.4188274	-1.4188274
26	1.2834287	1.2834287	1.2834287	1.2834287	-1.4188274	-1.4188274
27	1.2834287	1.2834287	1.2834287	1.2834287	-1.4188274	-1.4188274
28	1.2834287	1.2834287	1.2834287	1.2834287	-1.4188274	-1.4188274
29	1.2834287	1.2834287	1.2834287	1.2834287	-1.4188274	-1.4188274
30	1.2834287	1.2834287	1.2834287	1.2834287	-1.4188274	-1.4188274
31	1.2834287	1.2834287	1.2834287	1.2834287	-1.4188274	-1.4188274
32	1.2834287	1.2834287	1.2834287	1.2834287	-1.4188274	-1.4188274

Table 6: P331 2. b的计算结果

k	a_k	c_k	d_k	b_k	$f(c_k)$	$f(d_k)$
0	0.5000000	1.2639320	1.7360680	2.5000000	-3.8400010	-3.6447215
1	0.5000000	0.9721360	1.2639320	1.7360680	-3.4913294	-3.8400010
2	0.9721360	1.2639320	1.4442719	1.7360680	-3.8400010	-3.8734148
3	1.2639320	1.3753882	1.4442719	1.5557281	-3.8769150	-3.8734148
4	1.2639320	1.3328157	1.3753882	1.4442719	-3.8690640	-3.8769150
5	1.3328157	1.3753882	1.4016994	1.4442719	-3.8769150	-3.8779315
6	1.3753882	1.3916494	1.4016994	1.4179607	-3.8778879	-3.8779315
7	1.3916494	1.3978607	1.4016994	1.4079107	-3.8779650	-3.8779315
8	1.3916494	1.3954882	1.3978607	1.4016994	-3.8779547	-3.8779650
9	1.3954882	1.3969545	1.3978607	1.3993270	-3.8779639	-3.8779650
10	1.3969545	1.3975145	1.3978607	1.3984207	-3.8779650	-3.8779650
11	1.3975145	1.3977285	1.3978607	1.3980746	-3.8779651	-3.8779650
12	1.3975145	1.3976468	1.3977285	1.3978607	-3.8779651	-3.8779651
13	1.3976468	1.3976973	1.3977285	1.3977790	-3.8779651	-3.8779651
14	1.3976468	1.3976780	1.3976973	1.3977285	-3.8779651	-3.8779651
15	1.3976780	1.3976973	1.3977092	1.3977285	-3.8779651	-3.8779651
16	1.3976973	1.3977046	1.3977092	1.3977165	-3.8779651	-3.8779651
17	1.3976973	1.3977018	1.3977046	1.3977092	-3.8779651	-3.8779651
18	1.3977018	1.3977035	1.3977046	1.3977064	-3.8779651	-3.8779651
19	1.3977035	1.3977042	1.3977046	1.3977053	-3.8779651	-3.8779651
20	1.3977042	1.3977046	1.3977049	1.3977053	-3.8779651	-3.8779651
21	1.3977046	1.3977048	1.3977049	1.3977050	-3.8779651	-3.8779651
22	1.3977046	1.3977047	1.3977048	1.3977049	-3.8779651	-3.8779651
23	1.3977047	1.3977048	1.3977048	1.3977048	-3.8779651	-3.8779651
24	1.3977048	1.3977048	1.3977048	1.3977048	-3.8779651	-3.8779651
25	1.3977048	1.3977048	1.3977048	1.3977048	-3.8779651	-3.8779651
26	1.3977048	1.3977048	1.3977048	1.3977048	-3.8779651	-3.8779651
27	1.3977048	1.3977048	1.3977048	1.3977048	-3.8779651	-3.8779651
28	1.3977048	1.3977048	1.3977048	1.3977048	-3.8779651	-3.8779651
29	1.3977048	1.3977048	1.3977048	1.3977048	-3.8779651	-3.8779651
30	1.3977048	1.3977048	1.3977048	1.3977048	-3.8779651	-3.8779651
31	1.3977048	1.3977048	1.3977048	1.3977048	-3.8779651	-3.8779651
32	1.3977048	1.3977048	1.3977048	1.3977048	-3.8779651	-3.8779651
33	1.3977048	1.3977048	1.3977048	1.3977048	-3.8779651	-3.8779651
34	1.3977048	1.3977048	1.3977048	1.3977048	-3.8779651	-3.8779651

Table 7: P331 2. c的计算结果

k	a_k	c_k	d_k	b_k	$f(c_k)$	$f(d_k)$
0	1.0000000	2.5278640	3.4721360	5.0000000	7.2028124	4.5804865
1	2.5278640	3.1114562	3.4721360	4.0557281	4.7167059	4.5804865
2	3.1114562	3.3343685	3.4721360	3.6950483	4.4814868	4.5804865
3	3.1114562	3.2492236	3.3343685	3.4721360	4.5162587	4.4814868
4	3.2492236	3.3018461	3.3343685	3.3869910	4.4864075	4.4814868
5	3.3018461	3.3219461	3.3343685	3.3544685	4.4821284	4.4814868
6	3.3219461	3.3296236	3.3343685	3.3420461	4.4815502	4.4814868
7	3.3296236	3.3325561	3.3343685	3.3373011	4.4814845	4.4814868
8	3.3296236	3.3314360	3.3325561	3.3343685	4.4814995	4.4814845
9	3.3314360	3.3325561	3.3332484	3.3343685	4.4814845	4.4814815
10	3.3325561	3.3329840	3.3332484	3.3336763	4.4814821	4.4814815
11	3.3329840	3.3332484	3.3334118	3.3336763	4.4814815	4.4814815
12	3.3332484	3.3333494	3.3334118	3.3335128	4.4814815	4.4814815
13	3.3332484	3.3333108	3.3333494	3.3334118	4.4814815	4.4814815
14	3.3333108	3.3333347	3.3333494	3.3333733	4.4814815	4.4814815
15	3.3333108	3.3333256	3.3333347	3.3333494	4.4814815	4.4814815
16	3.3333256	3.3333312	3.3333347	3.3333403	4.4814815	4.4814815
17	3.3333312	3.3333333	3.3333347	3.3333368	4.4814815	4.4814815
18	3.3333312	3.3333325	3.3333333	3.3333347	4.4814815	4.4814815
19	3.3333325	3.3333330	3.3333333	3.3333339	4.4814815	4.4814815
20	3.3333330	3.3333332	3.3333333	3.3333335	4.4814815	4.4814815
21	3.3333332	3.3333333	3.3333333	3.3333334	4.4814815	4.4814815
22	3.3333333	3.3333333	3.3333333	3.3333334	4.4814815	4.4814815
23	3.3333333	3.3333333	3.3333333	3.3333334	4.4814815	4.4814815
24	3.3333333	3.3333333	3.3333333	3.3333333	4.4814815	4.4814815
25	3.3333333	3.3333333	3.3333333	3.3333333	4.4814815	4.4814815
26	3.3333333	3.3333333	3.3333333	3.3333333	4.4814815	4.4814815
27	3.3333333	3.3333333	3.3333333	3.3333333	4.4814815	4.4814815
28	3.3333333	3.3333333	3.3333333	3.3333333	4.4814815	4.4814815
29	3.3333333	3.3333333	3.3333333	3.3333333	4.4814815	4.4814815
30	3.3333333	3.3333333	3.3333333	3.3333333	4.4814815	4.4814815
31	3.3333333	3.3333333	3.3333333	3.3333333	4.4814815	4.4814815
32	3.3333333	3.3333333	3.3333333	3.3333333	4.4814815	4.4814815
33	3.3333333	3.3333333	3.3333333	3.3333333	4.4814815	4.4814815
34	3.3333333	3.3333333	3.3333333	3.3333333	4.4814815	4.4814815
35	3.3333333	3.3333333	3.3333333	3.3333333	4.4814815	4.4814815
36	3.3333333	3.3333333	3.3333333	3.3333333	4.4814815	4.4814815

Table 8: P331 2. d的计算结果

5 .3 P331 3.

P_0	P_1	P_2	$f(P_0)$	$f'(P_0)$
-2.122212	-1.722212	-1.322212	-1.872766	-0.002445
-2.122212	-1.922212	-1.722212	-1.872766	-0.002445
-2.122212	-2.022212	-1.922212	-1.872766	-0.002445
-2.122212	-2.072212	-2.022212	-1.872766	-0.002445
-2.122212	-2.097212	-2.072212	-1.872766	-0.002445
-2.122212	-2.109712	-2.097212	-1.872766	-0.002445
-2.122212	-2.115962	-2.109712	-1.872766	-0.002445

Table 9: P331 3. a的计算结果: $h_{min} = 0.002184$

P_0	P_1	P_2	$f(P_0)$	$f'(P_0)$
1.279582	1.679582	2.079582	-1.418813	-0.007534
1.279582	1.479582	1.679582	-1.418813	-0.007534
1.279582	1.379582	1.479582	-1.418813	-0.007534
1.279582	1.329582	1.379582	-1.418813	-0.007534
1.279582	1.304582	1.329582	-1.418813	-0.007534
1.279582	1.292082	1.304582	-1.418813	-0.007534
1.283428	1.289678	1.295928	-1.418827	-0.000001
1.283428	1.286553	1.289678	-1.418827	-0.000001
1.283428	1.284991	1.286553	-1.418827	-0.000001
1.283428	1.284209	1.284991	-1.418827	-0.000001
1.283428	1.283819	1.284209	-1.418827	-0.000001
1.283428	1.283624	1.283819	-1.418827	-0.000001
1.283428	1.283526	1.283624	-1.418827	-0.000001
1.283428	1.283477	1.283526	-1.418827	-0.000001
1.283428	1.283453	1.283477	-1.418827	-0.000001
1.283428	1.283440	1.283453	-1.418827	-0.000001
1.283428	1.283434	1.283440	-1.418827	-0.000001
1.283428	1.283431	1.283434	-1.418827	-0.000001
1.283428	1.283430	1.283431	-1.418827	-0.000001

Table 10: P331 3. b的计算结果:$h_{min} = 0.000001$

P_0	P_1	P_2	$f(P_0)$	$f'(P_0)$
1.960751	1.230376	0.500000	-5.175359	0.154449
1.960751	1.595564	1.230376	-5.175359	0.154449
1.960751	1.778158	1.595564	-5.175359	0.154449
1.892054	1.800757	1.709460	-5.180847	0.003070
1.892054	1.846406	1.800757	-5.180847	0.003070
1.892054	1.869230	1.846406	-5.180847	0.003070
1.892054	1.880642	1.869230	-5.180847	0.003070
1.892054	1.886348	1.880642	-5.180847	0.003070
1.892054	1.889201	1.886348	-5.180847	0.003070

Table 11: P331 3. c的计算结果:$h_{min} = -0.001333$

P_0	P_1	P_2	$f(P_0)$	$f'(P_0)$
2.875000	4.875000	6.875000	5.435547	-3.953125
2.875000	3.875000	4.875000	5.435547	-3.953125
3.309451	3.809451	4.309451	4.484320	-0.237110
3.309451	3.559451	3.809451	4.484320	-0.237110
3.309451	3.434451	3.559451	4.484320	-0.237110
3.309451	3.371951	3.434451	4.484320	-0.237110
3.333252	3.364502	3.395752	4.481482	-0.000810
3.333252	3.348877	3.364502	4.481482	-0.000810
3.333252	3.341065	3.348877	4.481482	-0.000810
3.333252	3.337159	3.341065	4.481482	-0.000810
3.333252	3.335205	3.337159	4.481482	-0.000810
3.333252	3.334229	3.335205	4.481482	-0.000810
3.333252	3.333741	3.334229	4.481482	-0.000810
3.333252	3.333496	3.333741	4.481482	-0.000810

Table 12: P331 3. d的计算结果:$h_{min} = 0.000081$

5 .4 内德米德方法实验

k	B	G	W
0	1.800000,1.200000	1.200000,0.000000	0.000000,0.800000
1	1.800000,1.200000	3.000000,0.400000	1.200000,0.000000
2	3.600000,1.600000	1.800000,1.200000	3.000000,0.400000
3	3.600000,1.600000	2.400000,2.400000	1.800000,1.200000
4	3.000000,2.000000	3.900000,2.200000	3.600000,1.600000
5	3.000000,2.000000	3.300000,2.600000	3.900000,2.200000
6	3.000000,2.000000	3.150000,2.300000	3.450000,2.100000
7	3.000000,2.000000	3.075000,2.150000	3.225000,2.050000
8	3.000000,2.000000	3.037500,2.075000	3.112500,2.025000
9	3.000000,2.000000	3.018750,2.037500	3.056250,2.012500
10	3.000000,2.000000	3.009375,2.018750	3.028125,2.006250
11	3.000000,2.000000	3.004687,2.009375	3.014062,2.003125
12	3.000000,2.000000	3.002344,2.004688	3.007031,2.001563
13	3.000000,2.000000	3.001172,2.002344	3.003516,2.000781
14	3.000000,2.000000	3.000586,2.001172	3.001758,2.000391
15	3.000000,2.000000	3.000293,2.000586	3.000879,2.000195
16	3.000000,2.000000	3.000146,2.000293	3.000439,2.000098
17	3.000000,2.000000	3.000073,2.000146	3.000220,2.000049
18	3.000000,2.000000	3.000037,2.000073	3.000110,2.000024
19	3.000000,2.000000	3.000018,2.000037	3.000055,2.000012
20	3.000000,2.000000	3.000009,2.000018	3.000027,2.000006
21	3.000000,2.000000	3.000005,2.000009	3.000014,2.000003
22	3.000000,2.000000	3.000002,2.000005	3.000007,2.000002
23	3.000000,2.000000	3.000001,2.000002	3.000003,2.000001

Table 13: 计算结果

5 .5 P351 7.

P_k	S_k	h_{min}
(-3.0000000,-2.0000000)	(0.8287239,0.5596577)	4.93130314
(1.0866887,0.7598417)	(-0.9929252,0.1187415)	0.15447160
(0.9333099,0.7781839)	(0.3459625,0.9382484)	0.18935353
(0.9988192,0.9558445)	(-0.7612119,-0.6485032)	0.04236865
(0.9665676,0.9283683)	(-0.7792922,-0.6266607)	0.00227798
(0.9647924,0.9269408)	(-0.7807214,-0.6248792)	0.00018421
(0.9646486,0.9268257)	(-0.7808400,-0.6247310)	0.00001530
(0.9646367,0.9268161)	(-0.7808499,-0.6247187)	0.00000127
(0.9646357,0.9268154)	(-0.7808507,-0.6247177)	0.00000011
(0.9646356,0.9268153)	(-0.7808508,-0.6247176)	0.00000001

Table 14

6 实验结论

- 在进行计算量比较大的计算的时候不要使用迭代而是使用循环。因为迭代的压栈出栈比较耗时间。在一些较小的计算中可以使用迭代，这样代码可以简洁一些。
- 集中求最小值的算法都是迭代实现。数值计算中大部分算法都是迭代。这正是利用了计算机善于处理规范化并且大规模重复运算的长处。在设计算法的时候也要注意这一点。
- 内德米德单纯性算法计算的比较快，但是选择初始值还是对计算有影响的。
- 在利用二次插值求最小值的时候，在寻找合适的测试值的时候要考虑测试点正好在最小值的情况，即测试点导数为零。在实际运算中出现了这个问题。

7 实验代码

7 .1 P331 1.

main.c

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

double f(double x);

int main()
```

```
{
    double left_end, right_end;//the endpoint of interval
    double c, d;//the interior points
    double R;//the ration
    double tol = 1e-6;
    int k=0;
/*********initialize the coefficients*********/
    left_end = 1;
    right_end = 5;
    R = (sqrt(5) - 1) / 2;
/*********golden ratio*******************/
    do{
        //find interior points
        c = left_end + (1 - R) * (right_end - left_end);
        d = right_end - (1 - R) * (right_end - left_end);

        k++;//record the number of cycle
        //if find the minimization, exit the recycle
        if(fabs(f(d) - f(c)) < (tol)){
            break;
        }
        //find a new interval
        if(f(c) < f(d)){
            right_end = d;
        }
        else{
            left_end = c;
        }
    }while(1);

    printf("result = %.9lf\n",f(c));
    return 0;
}
/*********************************
function name : f
description:
Input: the variable x
Output:f(x)
*********************************/
double f(double x)
{
    //return pow(x, 2) - sin(x);
    //return exp(x) + 2*x + pow(x,2)/2;
    //return -sin(x) - x +pow(x,2)/2;
    //return pow(x,2)/2 -4*x - x*cos(x);
    return pow(x,3)-5*pow(x,2)+23;
```

```
}
```

7 .2 P331 2.

```
  #include <stdio.h>
#include <stdlib.h>
#include <math.h>

//#define rational (*(Fibonacci +count - k - 1) / *(Fibonacci +count - k))

double generate_Fb(double *Fibonacci, int n);
double f(double x);

int main()
{
    double *Fibonacci, F_try, f_b, f_bb;
    double tol, e;
    double a, b;
    int count;
/*****initialize the coefficients****************/
    a = 1;
    b = 5;
    tol = 1e-7;
    e = 0.01;

    f_b = 1;
    f_bb = 0;
    count = 2;
/***************iterate to find length of sequence********************/
    do{
        F_try = f_bb + f_b;
        f_bb = f_b;
        f_b = F_try;
        if(F_try > (b - a) / tol){
            break;
        }
        else{
            count++;
        }
    }while(1);

/*******build a fibonacci suquence********/
    Fibonacci = malloc(count * sizeof(double));
    generate_Fb(Fibonacci, count);

/**************************search**************************/
```

```
    int k;
    double d, c, rational;
    rational = *(Fibonacci +count - 1) / *(Fibonacci +count);
    c = a + (1 - rational) * (b-a);
    d = a + rational * (b-a);
    for(k = 1; k < count-1; k++){
        //if F(n-1)/F(n)=1/2
        if(rational == 1/2){
            rational = 1/2 - e;
        }
        //get a new interval
        if(f(c) > f(d)){
            a = c;
            c = d;
            d = a + (*(Fibonacci +count - k - 1) / *(Fibonacci +count - k)) * (b-a);
        }
        if(f(c) < f(d)){
            b = d;
            d = c;
            c = a + (1 - (*(Fibonacci +count - k - 1) / *(Fibonacci +count - k))) * (b-a);
        }
    }
    printf("result = %.8lf\n",f(c));
    return 0;
}
/*********************************
function name : generate_Fb
description:generate the sequence of Fibonacci.
Input: the length of the sequence and the point of sequence.
Output: the number of F(n)
/*********************************/
double generate_Fb(double *Fibonacci, int n)
{
    //iterate to find F(n)
    if(n > 1){
        *(Fibonacci + n) = generate_Fb(Fibonacci, n-1) + generate_Fb(Fibonacci, n-2);
        return *(Fibonacci + n);
    }
    //set F(0) and F(1)
    else if(n == 1){
        *(Fibonacci + n) = 1;
        return 1;
    }
    else if(n == 0){
        *(Fibonacci + n) = 0;
        return 0;
```

```
    }
}
/*******************************
function name : f
description:
Input: the variable x
Output:f(x)
/*******************************/
double f(double x)
{
    //return pow(x,2) - sin(x);
    //return exp(x) + 2*x + pow(x,2)/2;
    //return -sin(x) -x+pow(x,2)/2;
    //return pow(x,2)-4*x-x*cos(x);
    return pow(x,3)-5*pow(x,2)+23;
}
```

7 .3 P331 3.

```
  main.c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "operation.h"

int main()
{
    double P[3]={0};
    double left_end, right_end, h;
    double tol;

    //scanf("%lf",&left_end);
    left_end = 1;
    //scanf("%lf",&right_end);
    right_end = 5;
    h = ((right_end - left_end)) / 2;
    tol = 1e-6;
    printf("h = %lf\n",h);
    P[0] = left_end;

    show_P(P);
    do{
        h = search_P(P, h, &left_end, &right_end);
                show_P(P);
        if(P[1] == P[0] && P[2] == P[0]){
            printf("out\n");
```

```
            break;
        }
        reset_P(P, h, &left_end, &right_end);
    }while(fabs(df(P[0])) > tol && P[0] < 10);

    return 0;
}

operation.h
#ifndef OPERATION_H_INCLUDED
#define OPERATION_H_INCLUDED

double f(double x);
double df(double x);

/*******************************
function name : search_P
description:search the test points's step length.
Input: endpoints of interval, step length, point of test points
Output:the step length
*******************************/
double search_P(double* P, double h, double* left_end, double* right_end)
{
    if(fabs(df(P[0])) < 1e-6){
        P[1] = P[0];
        P[2] = P[0];
        return h;
    }
    //getchar();
/********************pre*************************************/
    if(df(P[0]) > 0){
        //printf("change to minus\n");
        if(h > (double)((P[0] - *left_end)) / 2){
            h = (double)((P[0] - *left_end)) / 2;
        }
        h = -fabs(h);
    }

    P[1] = P[0] + h;
    P[2] = P[0] + 2 * h;
/*************************************************************/
    if(f(P[0]) > f(P[1]) && f(P[2]) > f(P[1])){
        return h;
    }
    if(f(P[0]) <= f(P[1])){
        show_P(P);
```

```
        printf("%lf & %lf\n",f(P[0]),df(P[0]));
        h /= 2;
        h = search_P(P,h,left_end,right_end);
        return h;
    }
    if(f(P[0]) > f(P[1]) && f(P[1]) > f(P[2])){
        show_P(P);
        printf("%lf & %lf\n",f(P[0]),df(P[0]));
        h *= 2;
        h = search_P(P,h,left_end,right_end);
        return h;
    }
}
/*********************************
function name : reset_P
description: set the new research interval
Input: old interval's endpoint and step length h, and P.
Output:
/*********************************/
void reset_P(double* P, double h, double* left_end, double *right_end)
{
    double h_min = 0;

    *right_end = P[2];

    h_min = h * (4 * f(P[1]) - 3 * f(P[0]) - f(P[2]))
        / (4 * f(P[1]) - 2 * f(P[0]) - 2 * f(P[2]));
    printf("h_min = %lf\n",h_min);
    P[0] += h_min;
    if(df(P[0]) <= 0){
        *left_end = P[0];
    }
    if(df(P[0]) > 0){
        *right_end = P[0];
    }
}
/*********************************
function name : f
description:
Input: the variable x
Output:f(x)
/*********************************/
double f(double x)
{
    //return pow(x, 2) - sin(x);
    //return exp(x)+2*x+pow(x,2)/2;
```

```
    //return -sin(x)-x+pow(x,2)/2;
    //return pow(x,2)/2-4*x-x*cos(x);
    return pow(x,3)-5*pow(x,2)+23;
}
/*******************************
function name : df
description:
Input: the variable x
Output:df(x)/dx
/*******************************/
double df(double x)
{
    //return exp(x)+2+x;
    //return -cos(x) - 1 +x;
    //return x-4-cos(x)+x*sin(x);
    return 3*pow(x,2)-10*x;
}

void show_P(double *P)
{
    printf("%lf & ",P[0]);
    printf("%lf & " ,P[1]);
    printf("%lf & ",P[2]);
}


#endif // OPERATION_H_INCLUDED
```

7 .4 内德米德方法

main.v

```
  #include <stdio.h>
#include <stdlib.h>
#include "operation.h"
#include <math.h>

int main()
{
    vtx *B, *G, *W;
    vtx *M, *R, *C;
    double tol = 1e-11;
    int count = 0;

    B = initialize_vertex(B);
    G = initialize_vertex(G);
    W = initialize_vertex(W);
```

```
    M = initialize_vertex(M);
    R = initialize_vertex(R);

    start_BGW(B,G,W);
    sort(B,G,W);
    midof(M,B,G);
    expansion(R,M,W);

    printf("\nbest point\t\tGood point\t\tWorst point\n");

    do{
        if(f(R) < f(G)){
            situation_1(B,R,W,M);
        }
        else{
            situation_2(B,R,W,M,G);
        }
        sort(B,G,W);
        printf("%d & ",count++);
        show_triangle(B,G,W);
        midof(M,B,G);
        expansion(R,M,W);
    }while(fabs(f(B) - f(W)) > tol);

    return 0;
}
```

operation.h

```
#ifndef OPERATION_H_INCLUDED
#define OPERATION_H_INCLUDED

struct vertex{
    double coord_x;
    double coord_y;
};
typedef struct vertex vtx;

vtx* initialize_vertex(vtx* B)
{
    B = (vtx*)malloc(sizeof(vtx));
    B->coord_x = 0;
    B->coord_y = 0;
    //show_vertex(B);
    return B;
}
```

```
void show_vertex(vtx *B)
{
    printf("%lf,%lf & ",B->coord_x,B->coord_y);
}

void start_BGW(vtx* B,vtx* G,vtx* W)
{
    B->coord_x = 0;
    B->coord_y = 0;

    G->coord_x = 1.2;
    G->coord_y = 0;

    W->coord_x = 0;
    W->coord_y = 0.8;
    printf("Start with the three vertices:\n");
    show_vertex(B);
    show_vertex(G);
    show_vertex(W);
}

double f(vtx * B)
{
    double x = B->coord_x;
    double y = B->coord_y;
    return pow(x,2) - 4 * x + pow(y,2) - y - x * y;
}

void midof(vtx* M, vtx* B, vtx* G)
{
    M->coord_x = (B->coord_x + G->coord_x) / 2;
    M->coord_y = (B->coord_y + G->coord_y) / 2;
    //printf("\nM = \n");
    //show_vertex(M);
}

void expansion(vtx* R, vtx* M, vtx* W)
{
    R->coord_x = 2 * M->coord_x - W->coord_x;
    R->coord_y = 2 * M->coord_y - W->coord_y;
    //printf("\nR = \n");
    //show_vertex(R);
}

void compute_E(vtx* E, vtx* R, vtx* M)
{
```

```
    E->coord_x = 2 * R->coord_x - M->coord_x;
    E->coord_y = 2 * R->coord_y - M->coord_y;
}

void situation_1(vtx* B, vtx* R, vtx* W,vtx* M)
{
    vtx* E;
    E = initialize_vertex(E);
    if(f(B) < f(R)){
        replace(W,R);
    }
    else{
        compute_E(E, R, M);
        if(f(E) < f(B)){
            replace(W,E);
        }
        else{
            replace(W,R);
        }
    }
    return;
}

void situation_2(vtx*B, vtx* R, vtx* W, vtx* M, vtx* G)
{
    if(f(R) < f(W)){
        replace(W,R);
    }

    midof(M,B,G);
    expansion(R,M,W);
    vtx *C, *C2, *S;
    C = initialize_vertex(C);
    C2 = initialize_vertex(C2);
    S = initialize_vertex(S);

    compute_C(C,W,M);
    compute_C2(C2,M,R);

    if(f(C) > f(C2)){
        replace(C,C2);
    }

    if(f(C) < f(M)){
        replace(W,C);
    }
```

```
    else{
        compute_S(S,B,W);
        replace(W,S);
        replace(G,M);
    }
}

void compute_S(vtx* S, vtx* B, vtx* W)
{
    S->coord_x = (W->coord_x + B->coord_x) / 2;
    S->coord_y = (W->coord_y + B->coord_y) / 2;
}

void compute_C(vtx* C,vtx* W,vtx* M)
{
    C->coord_x = (W->coord_x + M->coord_x) / (double)2;
    C->coord_y = (W->coord_y + M->coord_y) / (double)2;
    return;
}

void compute_C2(vtx* C2,vtx* M,vtx* R)
{
    C2->coord_x = (R->coord_x + M->coord_x) / 2;
    C2->coord_y = (R->coord_y + M->coord_y) / 2;
}

void replace(vtx* A, vtx* B)
{
    A->coord_x = B->coord_x;
    A->coord_y = B->coord_y;
}

void show_triangle(vtx* B, vtx* G, vtx* W)
{
    show_vertex(B);
    show_vertex(G);
    show_vertex(W);
    printf("\n");
}

void sort(vtx* B,vtx* G,vtx* W)
{
    vtx *temp;
    temp = initialize_vertex(temp);

    if(f(G) < f(B)){
```

```
        replace(temp,B);
        replace(B,G);
        replace(G,temp);
    }

    if(f(W) < f(B)){
        replace(temp,B);
        replace(B,W);
        replace(W,temp);

        replace(temp,W);
        replace(W,G);
        replace(G,temp);
        return;
    }
    if(f(G) < f(W)){
        return;
    }
    if(f(W) < f(G) && f(W) > f(B)){
        replace(temp,G);
        replace(G,W);
        replace(W,temp);
        return;
    }
}
#endif // OPERATION_H_INCLUDED
```

7 .5 P351 7.

```
  main.c
  #include <stdio.h>
#include <stdlib.h>
#include "ope_gradient.h"

int main()
{
    vtx **P;
    vtx *gd;
    vtx *S;
    double hmin;

    P = (vtx**)malloc(3 * sizeof(vtx*));
    gd = (vtx*)malloc(sizeof(vtx));
    S = (vtx*)malloc(sizeof(vtx));

    *(P + 0) = initialize_vertex(*(P + 0));
```

```
    *(P + 1) = initialize_vertex(*(P + 1));
    *(P + 2) = initialize_vertex(*(P + 2));
    gd = initialize_vertex(gd);
    S = initialize_vertex(S);

    (*(P + 0))->coord_x = -3;
    (*(P + 0))->coord_y = -2;

    do{
        gd = gradient(*(P + 0),gd);
        S = compute_S(gd, S);

        hmin = compute_min(*(P + 0), S);
        (*(P + 1))->coord_x = (*(P+0))->coord_x;
        (*(P + 1))->coord_y = (*(P+0))->coord_y;
        (*(P + 0))->coord_x = (*(P+0))->coord_x + hmin * S->coord_x;
        (*(P + 0))->coord_y = (*(P+0))->coord_y + hmin * S->coord_y;
    }while(distance(*(P + 0), *(P + 1)) > 1e-7);

    return 0;
}
```

ope_gradient.h

```
#ifndef OPE_GRADIENT_H_INCLUDED
#define OPE_GRADIENT_H_INCLUDED

#include <math.h>


struct vertex{
    double coord_x;
    double coord_y;
};
typedef struct vertex vtx;

#include "operation.h"

double cond(vtx* v);
double pdx(vtx *v);
double pdy(vtx *v);

vtx* initialize_vertex(vtx* B)
{
    B = (vtx*)malloc(sizeof(vtx));
    B->coord_x = 0;
```

```
    B->coord_y = 0;
    //show_vertex(B);
    return B;
}

void show_vertex(vtx *B)
{
    printf("(%.7lf,%.7lf) & ",B->coord_x,B->coord_y);
}

vtx* gradient(vtx* P, vtx* gd)
{
    gd->coord_x = pdx(P);
    gd->coord_y = pdy(P);

    return gd;
}

vtx* compute_S(vtx *gd,vtx* S)
{
    S->coord_x = -gd->coord_x / cond(gd);
    S->coord_y = -gd->coord_y / cond(gd);

    return S;
}

double cond(vtx* v)
{
    return sqrt(pow(v->coord_x,2) + pow(v->coord_y,2));
}

double f(vtx *v)
{
    double x = v->coord_x;
    double y = v->coord_y;
    return pow(x,4)+2*pow(x,2)*pow(y,2)-pow(x,2)-4*x+pow(y,4)-pow(y,2)-6*y+14;
}

double pdx(vtx *v)
{
    double x = v->coord_x;
    double y = v->coord_y;

    return 4*pow(x,3)+4*x*y*y-2*x-4;
}
```

```
double pdy(vtx *v)
{
    double x = v->coord_x;
    double y = v->coord_y;

    return 4*pow(y,3)+4*y*x*x-2*y-4;
}



double compute_min(vtx* P0, vtx* S)
{
    show_vertex(P0);
    show_vertex(S);

    double P[3]={0};
    double left_end, right_end, h;
    double tol;

    left_end = 0;
    right_end = 10;
    h = (right_end - left_end) / 2;
    tol = 1e-6;
    P[0] = left_end;

    //show_P(P);
    do{
        h = search_P(P, h, &left_end, &right_end, P0, S);
        reset_P(P, h, &left_end, &right_end, P0, S);
    }while(fabs(df_coe(P[0], P0, S)) > tol && P[0] < 10);

    return P[0];
}

double distance(vtx* A, vtx* B)
{
    double delta_x = pow(A->coord_x - B->coord_x, 2);
    double delta_y = pow(A->coord_y - B->coord_y, 2);
    return sqrt(delta_x + delta_y);
}
#endif // OPE_GRADIENT_H_INCLUDED

operation.h
#ifndef OPERATION_H_INCLUDED
#define OPERATION_H_INCLUDED
```

```
#include "ope_gradient.h"

double f_coe(double x, vtx* P,vtx* S);
double df_coe(double x, vtx* P,vtx* S);


double search_P(double* P, double h, double* left_end, double* right_end, vtx* P0, vtx* S)
{
    if(fabs(df_coe(P[0], P0, S)) < 1e-6){
        P[1] = P[0];
        P[2] = P[0];
        return h;
    }

    if(df_coe(P[0], P0, S) > 0){
        if(h > (double)((P[0] - *left_end)) / 2){
            h = (double)((P[0] - *left_end)) / 2;
        }
        h = -fabs(h);
    }

    P[1] = P[0] + h;
    P[2] = P[0] + 2 * h;

    if(f_coe(P[0], P0, S) > f_coe(P[1], P0, S)
                    && f_coe(P[2], P0, S) > f_coe(P[1], P0, S)){
        return h;
    }
    if(f_coe(P[0], P0, S) <= f_coe(P[1], P0, S)){
        h /= 2;
        h = search_P(P,h,left_end,right_end, P0, S);
        return h;
    }
    if(f_coe(P[0], P0, S) > f_coe(P[1], P0, S)
                    && f_coe(P[1], P0, S) > f_coe(P[2], P0, S)){
        h *= 2;
        h = search_P(P,h,left_end,right_end, P0, S);
        return h;
    }
}

void reset_P(double* P, double h, double* left_end, double *right_end, vtx* P0,vtx* S)
{
    double h_min = 0;

    *right_end = P[2];
```

```
    h_min = h * (4 * f_coe(P[1], P0, S) - 3 * f_coe(P[0], P0, S)
        - f_coe(P[2], P0, S)) /
        (4 * f_coe(P[1], P0, S) - 2 * f_coe(P[0], P0, S)
        - 2 * f_coe(P[2], P0, S));
    P[0] += h_min;
    if(df_coe(P[0], P0, S) <= 0){
        *left_end = P[0];
    }
    if(df_coe(P[0], P0, S) > 0){
        *right_end = P[0];
    }
}

double f_coe(double x, vtx* P,vtx* S)
{
    double xc,yc;
    xc = P->coord_x + x * S->coord_x;
    yc = P->coord_y + x * S->coord_y;

    return pow(xc,4)+2*pow(xc,2)*pow(yc,2)-pow(xc,2)-4*xc+pow(yc,4)-pow(yc,2)-6*yc+14;
}

double df_coe(double x, vtx* P,vtx* S)
{
    double xc,yc;
    xc = P->coord_x + x * S->coord_x;
    yc = P->coord_y + x * S->coord_y;

    double part1 = 4*pow(xc,3)-4-2*xc+4*xc*pow(yc,2);
    double part2 = -6+4*pow(yc,3)-2*yc+4*pow(xc,2)*yc;
    return S->coord_x*part1+S->coord_y*part2;
}

void show_P(double *P)
{
    printf("P[0] = %.7lf\n",P[0]);
    printf("P[1] = %.7lf\n",P[1]);
    printf("P[2] = %.7lf\n",P[2]);
}
#endif // OPERATION_H_INCLUDED
```