Chapter 1 Solutions of Equations and Optimization Methods

1-1 Fixed-Point Iteration Method

Fixed-point theorem Suppose $g(x) \in C[a,b]$ and $g'(x)$ exists on $[a,b]$ with $|g'(x)| \leqq k<1$, $\forall x \in (a,b)$, and then the sequence can be defined by $x_n=g(x_{n-1})$, $x_0 \in [a,b]$ converges to the unique fixed point x in $[a,b]$.

(Proof) $|x_n - x| = |g(x_{n-1}) - g(x)| = |g'(\varepsilon)| \cdot |x_{n-1} - x| \le k\,|x_{n-1} - x|$

$$|x_n - x| \le k\,|x_{n-1} - x| \le k^2\,|x_{n-2} - x| \le \cdots \le k^n\,|x_0 - x|$$

$$\because\ k<1,\ \therefore\ |x_n - x| \to 0 \Leftrightarrow \{x_n\}_{n=0}^{\infty} \text{ converges to } x.$$

Eg. Solve x^2-2x-3=0.

(Sol.) 1. Rearrange x^2-2x-3=0 into $x=\sqrt{2x+3}$ =$g_1(x)$

$$|g_1'(x)| = \frac{1}{\sqrt{2x+3}} < 1 \Rightarrow x > -1$$

Select $x_0 = 4 \Rightarrow x_1 = 3.316 \Rightarrow x_2 = 3.104 \Rightarrow \cdots \Rightarrow x_n \to 3$

2. Rearrange x^2-2x-3=0 into $x=\dfrac{3}{x-2}$ =$g_2(x)$

$$|g_2'(x)| = \left|-\frac{3}{(x-2)^2}\right| < 1 \Rightarrow \begin{matrix} x > 2+\sqrt{3} \\ x < 2-\sqrt{3} \end{matrix}$$

Select $x_0 = 4 \Rightarrow x_1 = 1.5 \Rightarrow x_2 = -6 \Rightarrow \cdots \Rightarrow x_8 = -1.003 \Rightarrow \cdots \Rightarrow x_n \to -1$

3. Rearrange x^2-2x-3=0 into $x=\dfrac{x^2-3}{2}$ =$g_3(x)$

$$|g_3'(x)| = |x| < 1 \Rightarrow -1 < x < 1$$

Select $x_0 = 4 \Rightarrow x_1 = 6.5 \Rightarrow x_2 = 19.635 \Rightarrow x_3 = 191.0$, …, $\therefore$ diverges

A C++ Program (developed by K. –Y. Lee) of solving x^2-2x-3=0 is listed as follows.

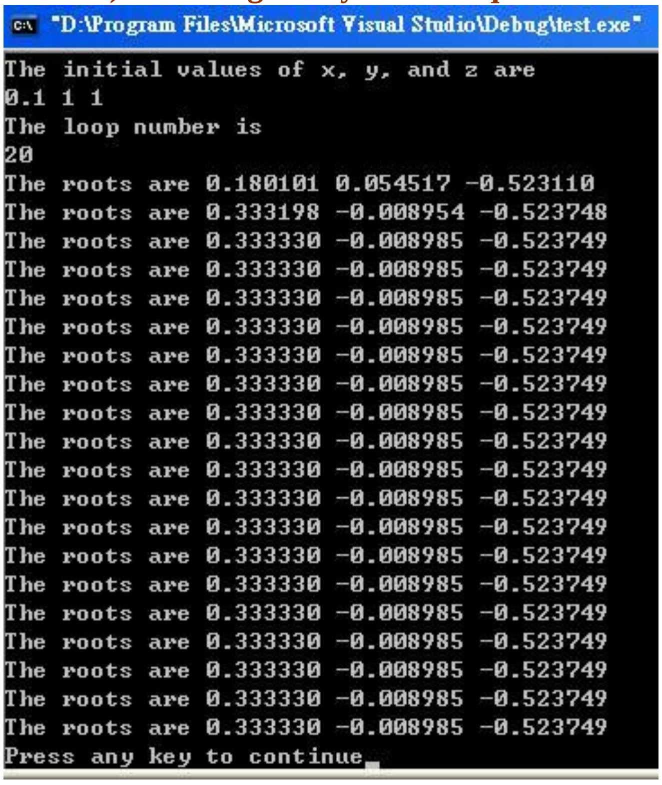
```
#include <stdio.h>
#include <math.h>
main()
{
 int i,lop; float x;
 printf("The initial value of x is\n"); scanf("%f",&x);
 printf("The loop number is\n"); scanf("%d",&lop);
 for (i=1;i<=lop;i++)
 {
  x=3/(x-2);
  printf("The root is %f \n",x);
 }

}
```

Eg. Solve $\begin{cases} 3x_1 - \cos(x_2x_3) - \dfrac{1}{2} = 0 \\ x_1^2 - 81(x_2+0.1)^2 + \sin x_3 + 1.06 = 0 \\ e^{-x_1x_2} + 20x_3 + \dfrac{10\pi - 3}{3} = 0 \end{cases}$.

(Sol.) $\begin{cases} x_1 = \dfrac{1}{3}\cos(x_2x_3) + \dfrac{1}{6} \\ x_2 = \dfrac{1}{9}\sqrt{x_1^2 + \sin x_3 + 1.06} - 0.1 \\ x_3 = -\dfrac{1}{20}e^{-x_1x_2} - \dfrac{10\pi - 3}{60} \end{cases}$

A C++ Program (developed by K. –Y. Lee) of solving the system of equations is listed as follows.

```
#include <stdio.h>

#include <math.h>

main()

{

int i,lop; float x,y,z;

printf("The initial values of x, y,
        and z are\n");

scanf("%f %f %f",&x,&y,&z);

printf("The loop number is\n");

scanf("%d",&lop);

for (i=1;i<=lop;i++)

  {

   x=cos(y*z)/3+1/6; y=sqrt(x*x+sin(z)+1.06)/9-0.1;

   z=-exp(-x*y)/20-(10*4*atan(1)-3)/60;

   printf("The roots are %f %f %f \n",x,y,z);

  }

}
```



```
The initial values of x, y, and z are
0.1 1 1
The loop number is
20
The roots are 0.180101 0.054517 -0.523110
The roots are 0.333198 -0.008954 -0.523748
The roots are 0.333330 -0.008985 -0.523749
The roots are 0.333330 -0.008985 -0.523749
The roots are 0.333330 -0.008985 -0.523749
The roots are 0.333330 -0.008985 -0.523749
The roots are 0.333330 -0.008985 -0.523749
The roots are 0.333330 -0.008985 -0.523749
The roots are 0.333330 -0.008985 -0.523749
The roots are 0.333330 -0.008985 -0.523749
The roots are 0.333330 -0.008985 -0.523749
The roots are 0.333330 -0.008985 -0.523749
The roots are 0.333330 -0.008985 -0.523749
The roots are 0.333330 -0.008985 -0.523749
The roots are 0.333330 -0.008985 -0.523749
The roots are 0.333330 -0.008985 -0.523749
The roots are 0.333330 -0.008985 -0.523749
The roots are 0.333330 -0.008985 -0.523749
The roots are 0.333330 -0.008985 -0.523749
The roots are 0.333330 -0.008985 -0.523749
Press any key to continue
```

1-2 Newton's Method

Newton's method: Solve $f(x)=0$ by $x_n=x_{n-1}-f(x_{n-1})/f'(x_{n-1})$, then $\{x_n\}$ converges to the root if an appropriate initial value is selected.

Eg. Solve $x^3+4x^2-10=0$ on [1,2].

(Sol.) $f(x)=x^3+4x^2-10$, $f'(x)=3x^2+8x$, $x_n = x_{n-1} - \dfrac{f(x_{n-1})}{f'(x_{n-1})}$

Choose $x_0 = 1.5 \Rightarrow x_1 = \cdots \Rightarrow x_2 = 1.36523001$

Secant method: Solve $f(x)=0$ by $x_n = x_{n-1} - \dfrac{f(x_{n-1})(x_{n-1}-x_{n-2})}{f(x_{n-1})-f(x_{n-2})}$.

Newton's method of solving a system of equations: Solve $F(x)=0$ by $x_k=x_{k-1}-J^{-1}(x_{k-1})F(x_{k-1})$, where $J(x)=$ $\begin{bmatrix} \partial f_1(x)/\partial x_1 \cdots \partial f_1(x)/\partial x_n \\ \vdots \qquad\qquad \vdots \\ \partial f_n(x)/\partial x_1 \cdots \partial f_n(x)/\partial x_n \end{bmatrix}$**, and $x=[x_1,x_2,\ldots,x_n]^t$.**

Eg. Solve $\begin{cases} 3x-\cos(yz)-\dfrac{1}{2}=0 \\ x^2-81(y+0.1)^2+\sin(z)+1.06=0 \\ e^{-xy}+20z+\dfrac{10\pi-3}{3}=0 \end{cases}$.

(Sol.) $J(x,y,z)=\begin{bmatrix} 3 & z\sin(yz) & y\sin(yz) \\ 2x & -162(y+0.1) & \cos(z) \\ -ye^{-xy} & -xe^{-xy} & 20 \end{bmatrix}$ and

$$F(x,y,z)=\begin{bmatrix} 3x-\cos(yz)-\dfrac{1}{2} \\ x^2-81(y+0.1)^2+\sin(z)+1.06 \\ e^{-xy}+20z+(10\pi-3)/3 \end{bmatrix}$$

$$\begin{bmatrix} x_k \\ y_k \\ z_k \end{bmatrix} = \begin{bmatrix} x_{k-1} \\ y_{k-1} \\ z_{k-1} \end{bmatrix} - \begin{bmatrix} 3 & z_{k-1}\sin(y_{k-1}z_{k-1}) & y_{k-1}\sin(y_{k-1}z_{k-1}) \\ 2x & -162(y_{k-1}+0.1) & \cos(z_{k-1}) \\ -y_{k-1}e^{-x_{k-1}y_{k-1}} & -x_{k-1}e^{-x_{k-1}y_{k-1}} & 20 \end{bmatrix}^{-1} \begin{bmatrix} 3x_{k-1}-\cos(y_{k-1}z_{k-1})-\dfrac{1}{2} \\ x_{k-1}^{\ 2}-81(y_{k-1}+0.1)^2+\sin(z_{k-1})+1.06 \\ e^{-x_{k-1}y_{k-1}}+20z_{k-1}+(10\pi-3)/3 \end{bmatrix}$$

Select $\begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix} = \begin{bmatrix} 0.1 \\ 0.1 \\ -0.1 \end{bmatrix} \Rightarrow \cdots \Rightarrow \begin{bmatrix} x_5 \\ y_5 \\ z_5 \end{bmatrix} = \begin{bmatrix} 0.33333333 \\ 0 \\ -0.52359877 \end{bmatrix}$

1-3 Complex Roots of an Equation

Conventional method of obtaining the complex roots of equations:

Eg. Solve $x^2+x+1=0$.

(Sol.) Set $x=a+bi$, $a, b\in R$

$$x^2+x+1=(a^2-b^2+2abi)+(a+bi)+1$$
$$=(a^2+a-b^2+1)+(2ab+b)i=0$$
$$\Rightarrow \begin{cases} a^2+a-b^2+1=0 \\ 2ab+b=0 \end{cases}$$ is a system of nonlinear equations.

The system of equations can be solved by the preceding mentioned method.

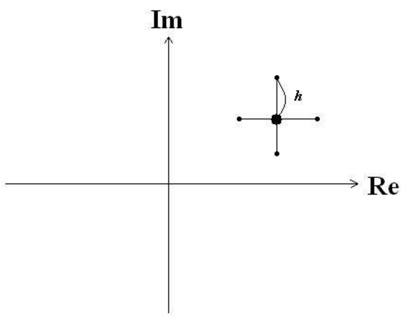
Eg. Solve $(i+1)x^2+x=i$.

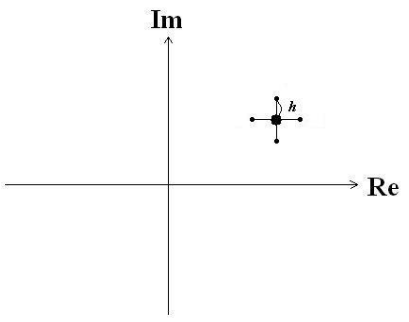
(Sol.) Set $x=a+bi$, $a, b\in R$

$$(i+1)x^2+x=(a^2-b^2+a-2ab)+i(2ab+a^2-b^2+b)=i$$
$$\Rightarrow \begin{cases} a^2-b^2-2ab+a=0 \\ 2ab+a^2-b^2+b-1=0 \end{cases}$$

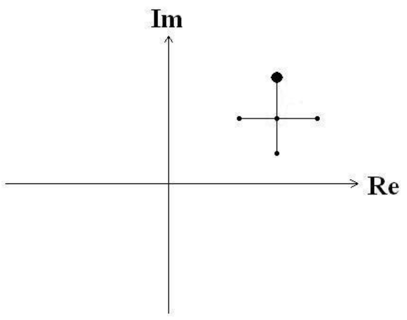
Five-point (SOS) method and nine-point method to obtain the complex roots of $f(x)=0$:

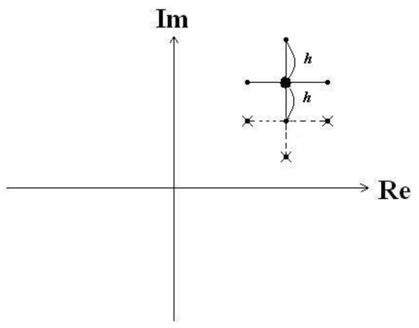
(a) Shrink: If the central point $x_c=a+ib$ such that $|f(x_c)|$ is the smallest of all grid points, and then shrink the distances between the grid points.





(b) Shift: If one of the side points $x_s=c+id$ such that $|f(x_s)|$ is the smallest of all grid points, and then let the side point be come the new central point.





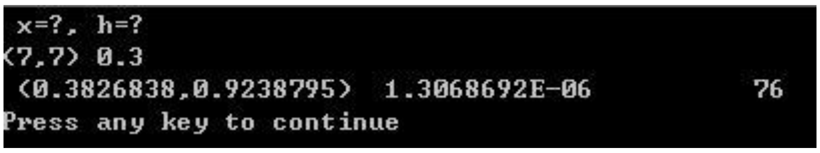
Repeat (a) and (b), the complex roots of $f(x)$ can be found.

Eg. Solve $x^4+i=0$ by the 5-point (SOS) method.

A Fortran Program (developed by K. –Y. Lee) is listed as follows.

```
      call fpoint
      stop
      end

      subroutine fpoint
        complex xc,x(5)
        complex f
        common h,x
        external f,g
        write (*,*) ‘x=?, h=?’
        read (*,*) xc,h
        x(1)=xc
        call patgen
        n=0
1       rmin=g(x(1))
          do 10 i=2,5
            if (g(x(i)).le.rmin) rmin=g(x(i))
10        continue
          if (g(x(1)).eq.rmin) then
            h=h/2.
            call patgen
          else if (g(x(2)).eq.rmin) then
            x(1)=x(2)
            call patgen
          else if (g(x(3)).eq.rmin) then
            x(1)=x(3)
            call patgen
          else if (g(x(4)).eq.rmin) then
            x(1)=x(4)
            call patgen
          else
            x(1)=x(5)
            call patgen
          endif
          n=n+1
        if (h.ge.0.000001) goto 1
        write (*,*) x(1),g(x(1)),n
        return
        end
```

```
complex function f(x)
  complex x
    f=x**4+cmplx(0.,1.)
  return
end

function g(x)
  external f
  complex x,f
    g=cabs(f(x))
  return
end

subroutine patgen
  complex x(5)
    common h,x
    x(2)=x(1)+cmplx(h,0.)
    x(3)=x(1)+cmplx(0.,h)
    x(4)=x(1)-cmplx(h,0.)
    x(5)=x(1)-cmplx(0.,h)
    return
  end
```



```
 x=?, h=?
(7,7) 0.3
 (0.3826838,0.9238795)  1.3068692E-06           76
Press any key to continue
```

1-4 Optimizations

Optimization: Find the extrema of a function or functions.

It is proved that solving an equation or the system of equations is equivalent to an optimization problem.

Eg. (a) Solve *f*(*x*)=0 $\Leftrightarrow$ Find the minimum of |*f*(*x*)|.

(b) Solve $\begin{cases} f(x,y)=0 \\ g(x,y)=0 \end{cases}$ $\Leftrightarrow$ **Find the minimum of** $|f(x,y)|^2+|g(x,y)|^2$.

1-5 Simplex Method and Shift-or-Shrink (SOS) Method in Optimizations

Simplex method: Find the local extrema of $f(x_1,x_2,\ldots,x_n)$, it needs $(n+1)$ initial guesses.

Consider a 2-dimensional case of maximization. Initial 3 guesses are ***G***, ***M***, and ***S***. Suppose $f(G)>f(M)>f(S)$.

(1) *Reflection*: Set ***new1*** is the image point of ***S*** with respect to the line connecting ***G*** and ***M***.

If $f(new1)>f(S)\Rightarrow$ *Expansion*. If $f(new1)\leqq f(S)\Rightarrow$ *Contraction*

(2) *Expansion*: Extend ***new1*** to ***new2*** by twice distance.

If $f(new1)>f(new2)\Rightarrow M, G, new1\Rightarrow$ *Reflection*

If $f(new1)\leqq f(new2)\Rightarrow M, G, new2\Rightarrow$ *Reflection*

(3) *Contraction*: Set ***new2*** is the middle point of ***S*** to the line connecting ***G*** and ***M***.

If $f(new2)>f(S)\Rightarrow M, G, new2\Rightarrow$ *Reflection*

If $f(new2)<f(S)\Rightarrow$ *Shrink*

(4) *Shrink*: Set ***new1*** is the middle point of ***S*** to ***G*** and ***new2*** is the middle point of ***M*** to ***G***. And use *new*1, *new*2, *G* $\Rightarrow$ *Reflection*

Eg. Find the minimum of $F(x,y)=x^2+y^2$.

(Sol.) Min of $F(x,y)=x^2+y^2 \Leftrightarrow$ Max of $G(x,y)=-F(x,y)=-x^2-y^2$

There are 2 variables, so we choose arbitrary 3 points: (1,0), (0,1), (1,1)

$G(1,1)=-2$ is ***S***, $G(1,0)=-1$, $G(0,1)=-1\Rightarrow$

Reflection: reject ***S***, then (0,0) is ***new1***, $G(0,0)=0>-2\Rightarrow$

Expansion: $\left(\frac{-1}{2},\frac{-1}{2}\right)$ is ***new2*** $\Rightarrow G\left(\frac{-1}{2},\frac{-1}{2}\right)=-\frac{1}{2}<0\Rightarrow$ ***new2*** is rejected

Now, 3 points are (0,0), (0,1), (1,0)$\Rightarrow$

Reflection: (0,-1) is ***new1***, $G(0,-1)=-1=G(0,1)\Rightarrow$ ***new1*** is rejected

Contraction: $\left(0,\frac{1}{2}\right)$ is ***new2***, $G\left(0,\frac{1}{2}\right)=-\frac{1}{4}>-1$

Now, 3 points are (0,0), $\left(0,\frac{1}{2}\right)$, $(1,0)\Rightarrow$

Reflection: (-1,0) is ***new1***, $G(-1,0)=-1=G(1,0)\Rightarrow$ ***new1*** is rejected

Contraction: $\left(\frac{1}{2},0\right)$ is ***new2***, $G\left(\frac{1}{2},0\right)=\frac{-1}{4}>-1$

Now, three points become (0,0), (0,1/2), (1/2.0)$\Rightarrow\cdots$

SOS method (by K. –Y. Lee *et al.*):

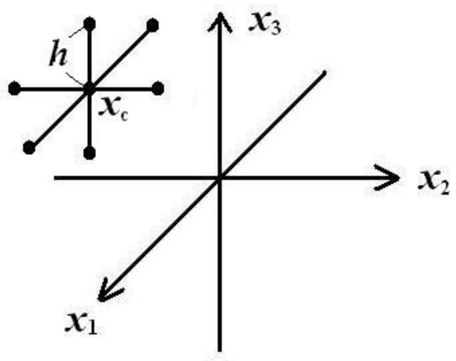
Find the local extrema of $f(x_1,x_2,\ldots,x_n)$, it needs $(2n+1)$ initial guesses as follows. Choose a central point $x_c=(x_{1c},x_{2c},\ldots,x_{nc})$, and its neighboring $2n$ grid point: $(x_{1c}\pm h,x_{2c},\ldots,x_{nc})$, $(x_{1c},x_{2c}\pm h,\ldots,x_{nc})$, …, $(x_{1c},x_{2c},\ldots,x_{nc}\pm h)$.

(a) Shift: If one of the side points makes $f(x_1,x_2,\ldots,x_n)$ has the extremum of all grid points, then let the side point become the new central point.

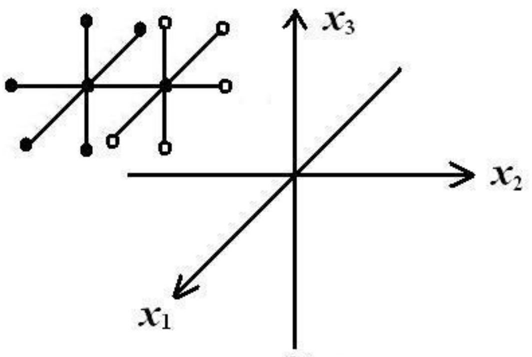
(b) Shrink: If the central point $x_c=(x_{1c},x_{2c},\ldots,x_{nc})$ makes $f(x_1,x_2,\ldots,x_n)$ has the maximum or minimum of all grid points, then let h become less.

Repeat (a) and (b), the local maximum or minimum of $f(x_1,x_2,\ldots,x_n)$ can be found.

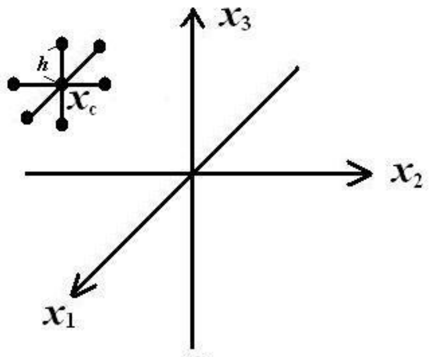
Consider a 3-dimensional SOS method in the following figures. Figure (a) shows a central point $x_c=(x_{1c},x_{2c},x_{3c})$ and its neighboring 6 grid point: $(x_{1c}\pm h,x_{2c},x_{3c})$, $(x_{1c},x_{2c}\pm h,x_{3c})$, and $(x_{1c},x_{2c},x_{3c}\pm h)$. Figure (b) presents that one of the side points makes $f(x_1,x_2,x_3)$ has the extremum of all grid points, then let the side point become the new central point. Figure (c) describes that the central point x_c makes $f(x_1,x_2,x_3)$ has the maximum or minimum of all grid points, then let h become less.



(a)



(b)



(c)

Eg. Find the minima of $f(x)=x^2-2$, $F(x,y)=x^2+y^2-1$, and $G(x,y,z)=x^2+y^2+z^2+1$.

(Sol.) (a) For $f(x)=x^2-2$, we may select $x_c=1$ and $h=1$, then we have 3 initial guesses: 1, and its neighboring 2 grid point: 0, 2.

$f(1)=-1$, $f(0)=-2$, $f(2)=2 \Rightarrow x_c=0$ and $h=1$, then we have 3 points: 0, -1, 1. (Shift)

$f(0)=-2$, $f(-1)=-1$, $f(1)=-1 \Rightarrow x_c=0$ and $h=0.5$, then we have 3 points: 0, -0.5, 0.5. (Shrink)

$f(0)=-2$, $f(-0.5)=-1.75$, $f(0.5)=-1.75 \Rightarrow x_c=0$ and $h=0.25$, then we have 3 points: 0, -0.25, 0.25. (Shrink)

$f(0)=-2$, $f(-0.25)=-1.9375$, $f(0.25)=-1.9375 \Rightarrow x_c=0$ and $h=0.125$, then we have 3 points: 0, -0.125, 0.125. (Shrink)

$\vdots$

The 3 points converge to 0 and $f(0)=-2$ is the minimum.

(b) For $F(x,y)=x^2+y^2-1$**,** we may select $x_c=(1,0)$ and $h=1$, then we have 5 initial guesses: $x_c=(1,0)$, and its neighboring 4 grid point: (0,0), (2,0), (1,-1), (1,1).

$F(1,0)=0$, $F(0,0)=-1$, $F(2,0)=3$, $F(1,-1)=1$, $F(1,1)=1 \Rightarrow x_c=(0,0)$ and $h=1$, then we have 5 points: (0,0), (1,0), (-1,0), (0,-1), (0,1). (Shift)

$F(0,0)=-1$, $F(1,0)=0$, $F(-1,0)=0$, $F(0,-1)=0$, $F(0,1)=0 \Rightarrow x_c=(0,0)$ and $h=0.5$, then we have 5 points: (0,0), (0.5,0), (-0.5,0), (0,-0.5), (0,0.5). (Shrink)

$\vdots$

The 5 points converge to (0,0) and $F(0,0)=-1$ is the minimum.

(c) For $G(x,y,z)=x^2+y^2+z^2+1$, we may select $x_c=(1,1,1)$ and $h=1$, then we have 7 initial guesses: $x_c=(1,1,1)$, and its neighboring 6 grid point: (0,1,1), (2,1,1), (1,0,1), (1,2,1), (1,1,0), (1,1,2).

$\vdots$

The 7 points converge to (0,0,0) and $G(x,y,z)=1$ is the minimum.

Eg. Minimize $f(x_1, x_2, x_3, \ldots, x_n)=x_1+x_2+x_3+\ldots+x_n$ by SOS method.

(Sol.) **A Fortran Program (developed by K. –Y. Lee) is listed as follows.**

```
dimension gg(0:100),xc(50),y(100,50)
common x(50),h
call ASOS
stop
end

subroutine ASOS
dimension gg(0:100),xc(50),y(100,50)
common x(50),h
external g
write (*,*) 'N=?(N<=50)'
read (*,*) n
do 100 i=1, n
```

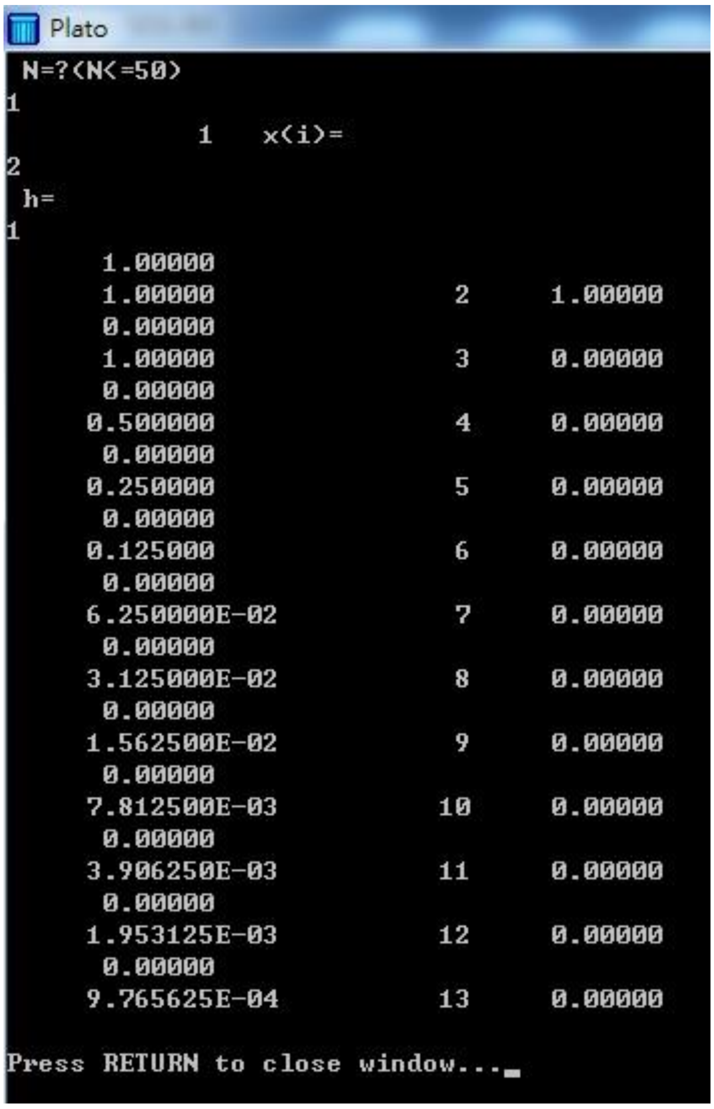
```
              write (*,*) i,'      x(i)=',
              read (*,*) xc(i)
  100      continue
  write (*,*) 'h='
  read (*,*) h
  do 150 j=1,n
      x(j)=xc(j)
  150       continue
         m=1
1        gg(0)=g(n)
         rmin=gg(0)
         do 300 i=1,2*n
                do 350 j=1,n
                      if ((2*j).eq.(i+1)) then
                             y(i,j)=x(j)+h
                      elseif    ((2*j).eq.i) then
                             y(i,j)=x(j)-h
                      else
                             y(i,j)=x(j)
                      endif
350          continue
300        continue
do      400 i=1,2*n
                    do 450 j=1,n
                          x(j)=y(i,j)
450                 continue
                    gg(i)=g(n)
                    if (gg(i).le.rmin) rmin=gg(i)
400      continue
           if (gg(0).eq.rmin) then
              h=h/2
              do 500 j=1,n
                        x(j)=xc(j)
  500       continue
           else
              do 600 i=1,2*n
                        if (rmin.eq.gg(i)) then
                              do 650 j=1,n
```

```
                         xc(j)=y(i,j)
650                  continue
               endif
600        continue
           endif
           do 700 j=1,n
                 x(j)=xc(j)
                 write (*,*) x(j)
700      continue
m=m+1
write (*,*) h,m,rmin
if (h.ge.0.001) goto 1
return
end

function g(n)
common x(50),h
g=0.
      do 1 i=1,n
         g=g+x(i)**2
1      continue
return
end
```

Case 1

Minimize $f(x)=x^2$.

Set n=1, choose initial x=x(1)=2 and h=1, we obtain the minimal value is 0 at x=0 after 13 iterations.

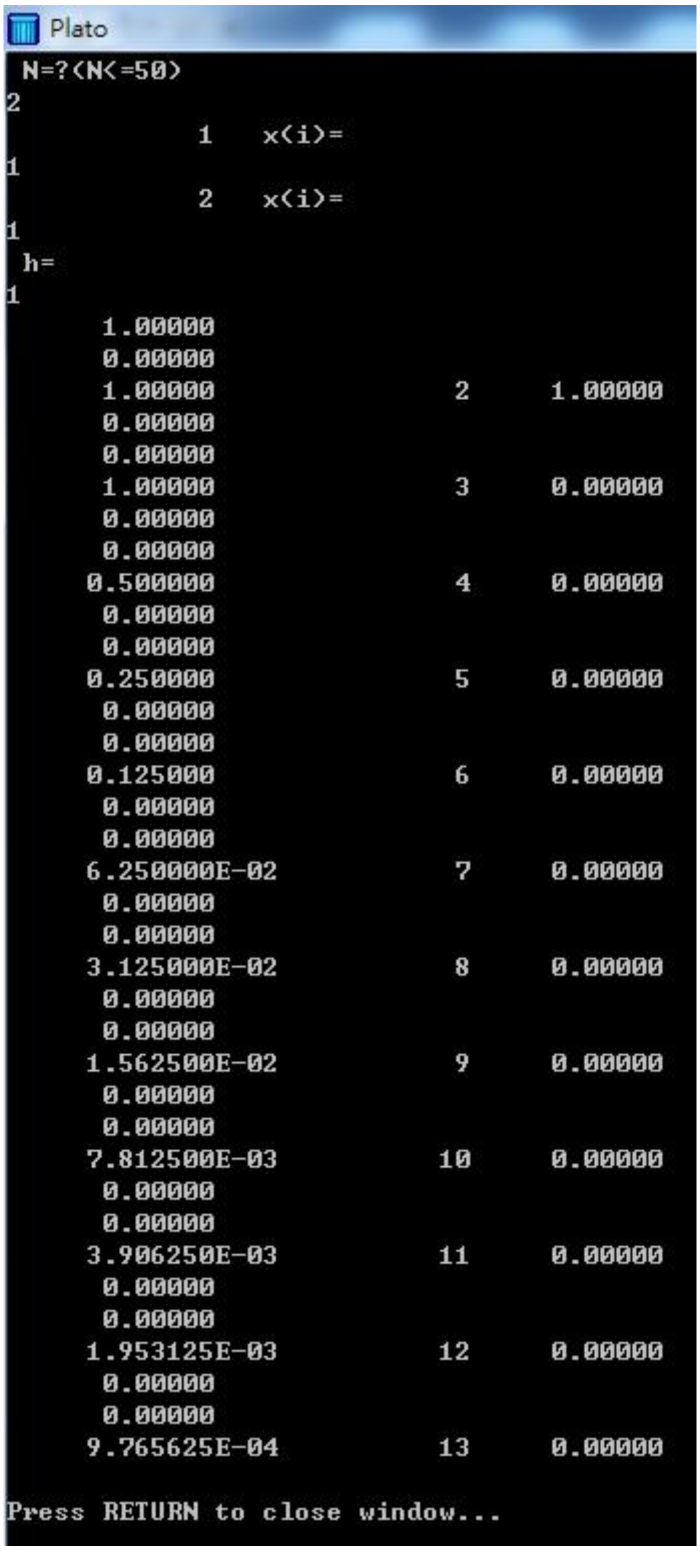


```
 N=?(N<=50)
1
            1    x(i)=
2
 h=
1
      1.00000
      1.00000              2      1.00000
      0.00000
      1.00000              3      0.00000
      0.00000
     0.500000              4      0.00000
      0.00000
     0.250000              5      0.00000
      0.00000
     0.125000              6      0.00000
      0.00000
     6.250000E-02          7      0.00000
      0.00000
     3.125000E-02          8      0.00000
      0.00000
     1.562500E-02          9      0.00000
      0.00000
     7.812500E-03         10      0.00000
      0.00000
     3.906250E-03         11      0.00000
      0.00000
     1.953125E-03         12      0.00000
      0.00000
     9.765625E-04         13      0.00000

Press RETURN to close window...
```

Case 2

Minimize $f(x,y)=x^2+y^2$.

Set n=2, choose initial $x=x(1)=1$, $y=x(2)=1$, and h=1, we obtain the minimal value is 0 at (0,0) after 13 iterations.

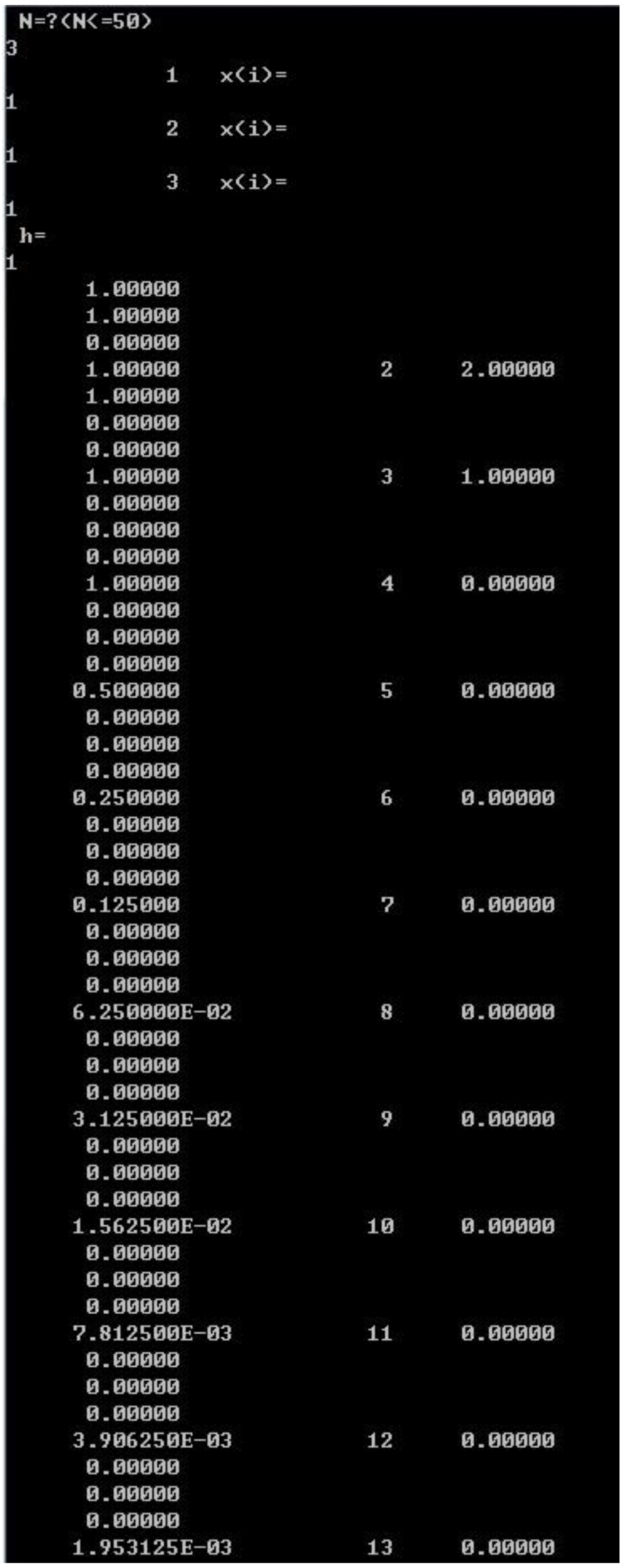


```
 N=?(N<=50)
2
            1   x(i)=
1
            2   x(i)=
1
 h=
1
      1.00000
      0.00000
      1.00000               2      1.00000
      0.00000
      0.00000
      1.00000               3      0.00000
      0.00000
      0.00000
     0.500000               4      0.00000
      0.00000
      0.00000
     0.250000               5      0.00000
      0.00000
      0.00000
     0.125000               6      0.00000
      0.00000
      0.00000
     6.250000E-02           7      0.00000
      0.00000
      0.00000
     3.125000E-02           8      0.00000
      0.00000
      0.00000
     1.562500E-02           9      0.00000
      0.00000
      0.00000
     7.812500E-03          10      0.00000
      0.00000
      0.00000
     3.906250E-03          11      0.00000
      0.00000
      0.00000
     1.953125E-03          12      0.00000
      0.00000
      0.00000
     9.765625E-04          13      0.00000

Press RETURN to close window...
```

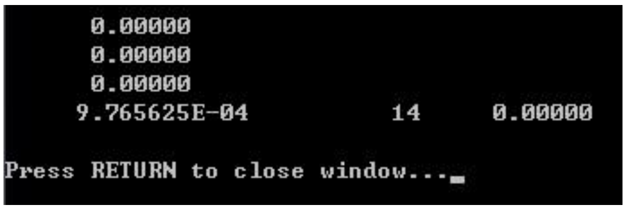
Case 3

Minimize $f(x,y,z)=x^2+y^2+z^2$.

Set n=3, choose initial x=x(1)=1, y=x(2)=1, z=x(3)=1, and h=1, we obtain the minimal value is 0 at (0,0,0) after 14 iterations.



```
 N=?(N<=50)
3
            1    x(i)=
1
            2    x(i)=
1
            3    x(i)=
1
 h=
1
      1.00000
      1.00000
      0.00000
      1.00000              2      2.00000
      1.00000
      0.00000
      0.00000
      1.00000              3      1.00000
      0.00000
      0.00000
      0.00000
      1.00000              4      0.00000
      0.00000
      0.00000
      0.00000
     0.500000              5      0.00000
      0.00000
      0.00000
      0.00000
     0.250000              6      0.00000
      0.00000
      0.00000
      0.00000
     0.125000              7      0.00000
      0.00000
      0.00000
      0.00000
     6.250000E-02          8      0.00000
      0.00000
      0.00000
      0.00000
     3.125000E-02          9      0.00000
      0.00000
      0.00000
      0.00000
     1.562500E-02         10      0.00000
      0.00000
      0.00000
      0.00000
     7.812500E-03         11      0.00000
      0.00000
      0.00000
      0.00000
     3.906250E-03         12      0.00000
      0.00000
      0.00000
      0.00000
     1.953125E-03         13      0.00000
```

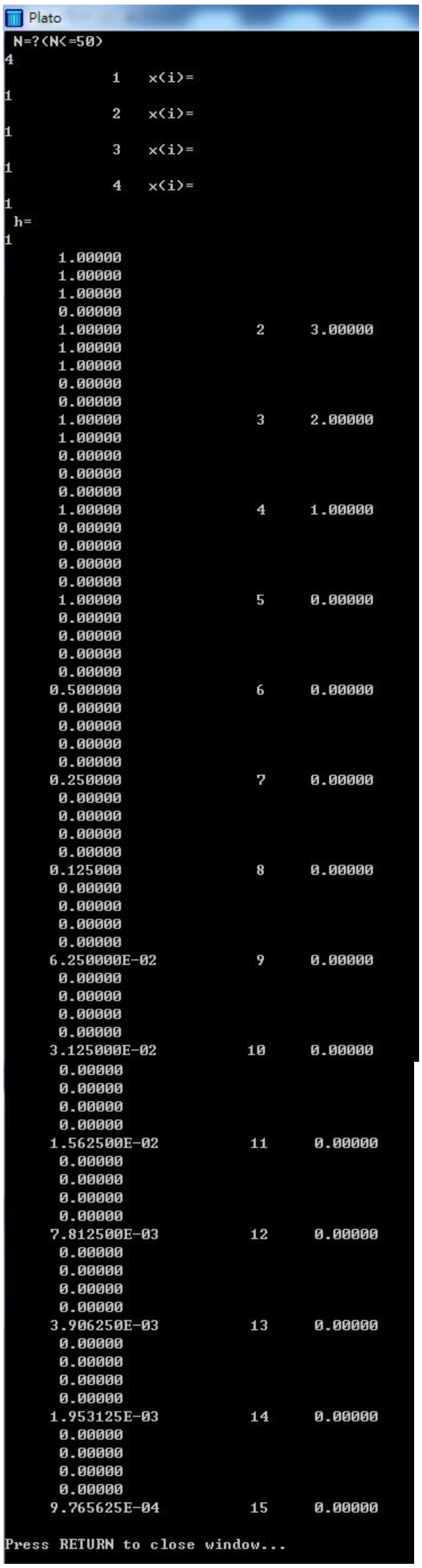


```
      0.00000
      0.00000
      0.00000
     9.765625E-04         14      0.00000

Press RETURN to close window...
```

Case 4

Minimize $f(x,y,z,w)=x^2+y^2+z^2+w^2$.

Set n=4, choose initial $x=x(1)=1$, $y=x(2)=1$, $z=x(3)=1$, $w=x(4)=1$, and h=1, we obtain the minimal value is 0 at (0,0,0,0) after 15 iterations.

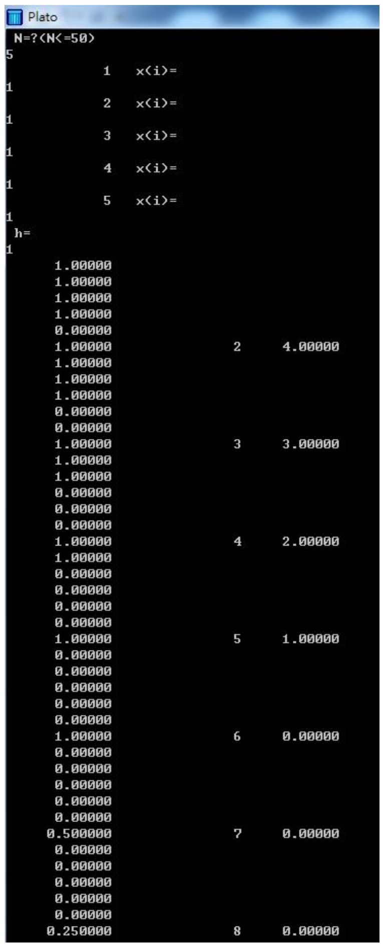


```
N=?(N<=50)
4
            1   x(i)=
1
            2   x(i)=
1
            3   x(i)=
1
            4   x(i)=
1
 h=
1
      1.00000
      1.00000
      1.00000
      0.00000
      1.00000              2     3.00000
      1.00000
      1.00000
      0.00000
      0.00000
      1.00000              3     2.00000
      1.00000
      0.00000
      0.00000
      0.00000
      1.00000              4     1.00000
      0.00000
      0.00000
      0.00000
      0.00000
      1.00000              5     0.00000
      0.00000
      0.00000
      0.00000
      0.00000
     0.500000              6     0.00000
      0.00000
      0.00000
      0.00000
      0.00000
     0.250000              7     0.00000
      0.00000
      0.00000
      0.00000
      0.00000
     0.125000              8     0.00000
      0.00000
      0.00000
      0.00000
      0.00000
     6.250000E-02          9     0.00000
      0.00000
      0.00000
      0.00000
      0.00000
     3.125000E-02         10     0.00000
      0.00000
      0.00000
      0.00000
      0.00000
     1.562500E-02         11     0.00000
      0.00000
      0.00000
      0.00000
      0.00000
     7.812500E-03         12     0.00000
      0.00000
      0.00000
      0.00000
      0.00000
     3.906250E-03         13     0.00000
      0.00000
      0.00000
      0.00000
      0.00000
     1.953125E-03         14     0.00000
      0.00000
      0.00000
      0.00000
      0.00000
     9.765625E-04         15     0.00000

Press RETURN to close window...
```

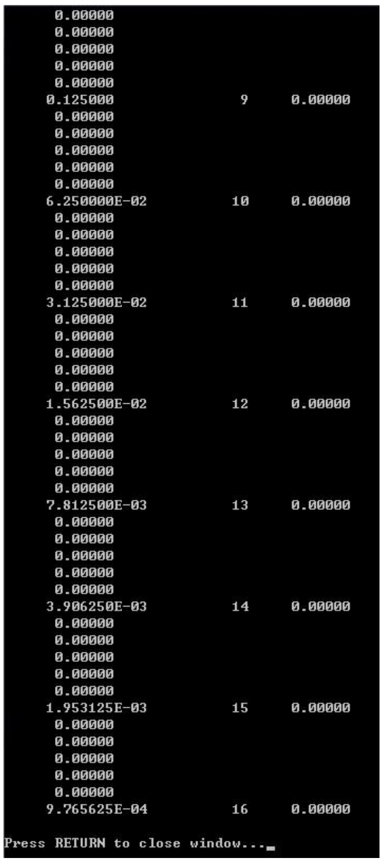
Case 5

Minimize $f(x,y,z,w)=x^2+y^2+z^2+w^2+v^2$.

Set n=4, choose initial $x=x(1)=1$, $y=x(2)=1$, $z=x(3)=1$, $w=x(4)=1$, $v=x(5)=1$,and h=1, we obtain the minimal value is 0 at (0,0,0,0,0) after 16 iterations.



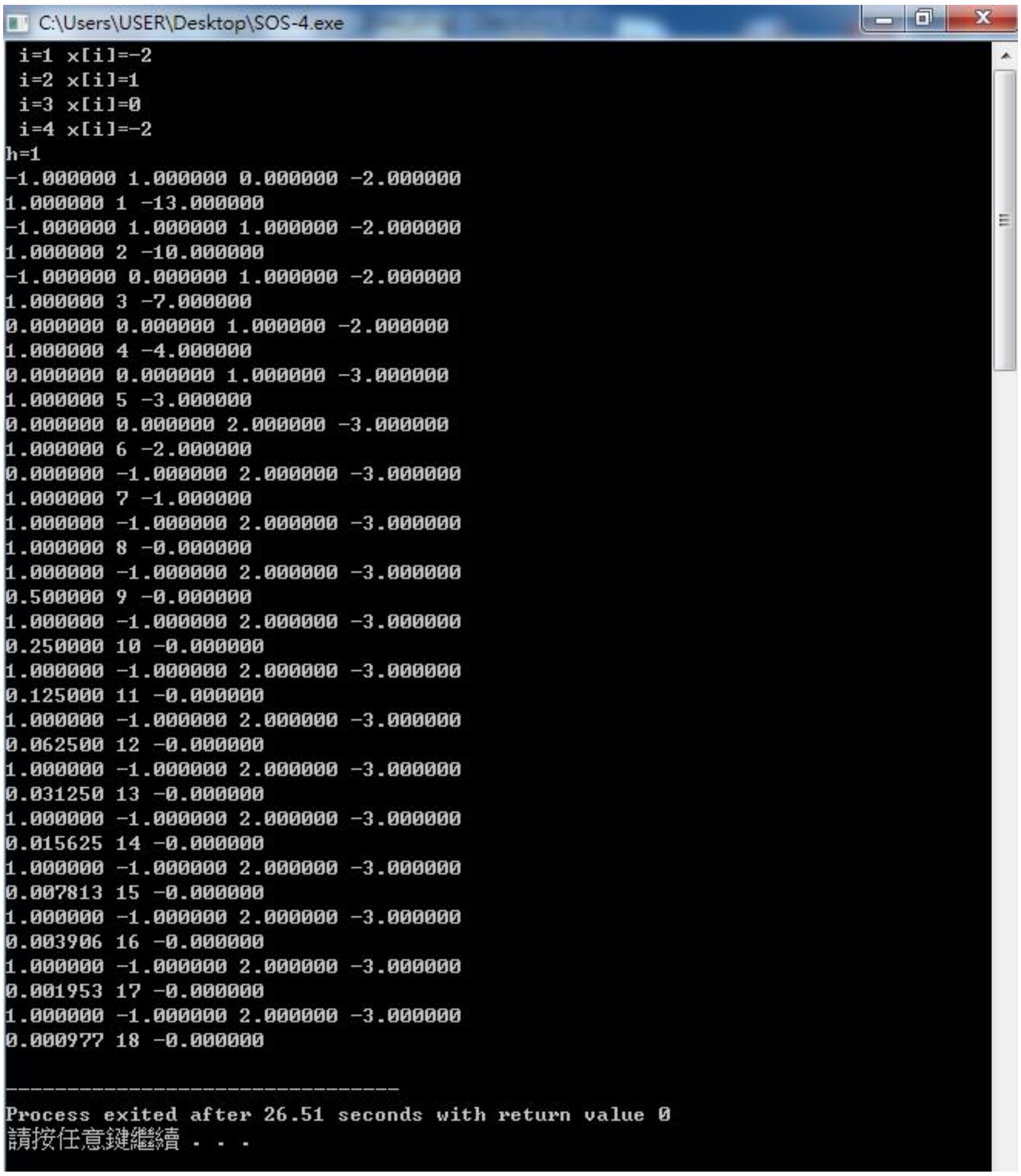
```
 N=?(N<=50)
5
            1    x(i)=
1
            2    x(i)=
1
            3    x(i)=
1
            4    x(i)=
1
            5    x(i)=
1
 h=
1
      1.00000
      1.00000
      1.00000
      1.00000
      0.00000
      1.00000                2      4.00000
      1.00000
      1.00000
      1.00000
      0.00000
      0.00000
      1.00000                3      3.00000
      1.00000
      1.00000
      0.00000
      0.00000
      0.00000
      1.00000                4      2.00000
      1.00000
      0.00000
      0.00000
      0.00000
      0.00000
      1.00000                5      1.00000
      0.00000
      0.00000
      0.00000
      0.00000
      0.00000
      1.00000                6      0.00000
      0.00000
      0.00000
      0.00000
      0.00000
      0.00000
     0.500000                7      0.00000
      0.00000
      0.00000
      0.00000
      0.00000
      0.00000
     0.250000                8      0.00000
```



```
      0.00000
      0.00000
      0.00000
      0.00000
      0.00000
     0.125000                9      0.00000
      0.00000
      0.00000
      0.00000
      0.00000
      0.00000
     6.250000E-02           10      0.00000
      0.00000
      0.00000
      0.00000
      0.00000
      0.00000
     3.125000E-02           11      0.00000
      0.00000
      0.00000
      0.00000
      0.00000
      0.00000
     1.562500E-02           12      0.00000
      0.00000
      0.00000
      0.00000
      0.00000
      0.00000
     7.812500E-03           13      0.00000
      0.00000
      0.00000
      0.00000
      0.00000
      0.00000
     3.906250E-03           14      0.00000
      0.00000
      0.00000
      0.00000
      0.00000
      0.00000
     1.953125E-03           15      0.00000
      0.00000
      0.00000
      0.00000
      0.00000
      0.00000
     9.765625E-04           16      0.00000

Press RETURN to close window..._
```

Case 1 Find the maximum of $f(c,w,u,v)=-(c-1)^2-(w+1)^2-(u-2)^2-(v+3)^2$ by C/C++ program.

```
#include<stdio.h>

#include<stdlib.h>

#include<math.h>

float f4(float c, float w, float u, float v)

{

float f4=-pow(c-1,2)-pow(w+1,2)-pow(u-2,2)-pow(v+3,2);

return (f4);

}

void ASOS()

{

int n,m;

float x[50],xc[50],h, rmax, gg[100],y[100][50];

             n=4;

         for    (int i=1; i<= n; i++)

         {

             printf(" i=%d x[i]=", i);

             scanf ("%f", &xc[i]);

         }

       printf("h=");

       scanf ("%f", &h);

         for    (int j=1; j<=n; j++)

         {
               x[j]=xc[j];

       }

m=0;

     while (h>0.001)

     {

gg[0]=f4(x[1],x[2],x[3],x[4]) ;

rmax=gg[0];


     for ( int i=0; i<=2*n; i++)

     {

             for (int j=1; j<=n; j++)

           {

               if (2*j==(i+1))

                     { y[i][j]=x[j]+h; }

             else if    ((2*j)==i)

                   { y[i][j]=x[j]-h; }

             else

                     { y[i][j]=x[j]; }

             }

     }


         for (int i=0; i<=2*n; i++)

       {
              for (int j=1; j<=n; j++)
             {
                  x[j]=y[i][j];
             }
            gg[i]=f4(x[1],x[2],x[3],x[4]) ;
            if (gg[i]>=rmax)
        { rmax=gg[i]; }
        }
   if (gg[0]==rmax)
    { h=h/2; }
     for (int j;    j<=n; j++)
     {
            x[j]=xc[j];
     }
   if (gg[0]!=rmax)
 {
        for (int i=1; i<=2*n; i++)
     {
            if (rmax==gg[i])
          {
                for (int j=1; j<=n; j++)
                {
                    xc[j]=y[i][j];
                    }
                 }
               }
              }
              for (int j=1;    j<=n; j++)
              {
                 x[j]=xc[j];
              }
                    m=m+1;
                      printf("%f %f %f %f \n", x[1],x[2],x[3],x[4]);
                      printf("%f %d %f \n", h,m,rmax);
             }
}


main ()
{
ASOS();
}
```


C:\Users\USER\Desktop\SOS-4.exe

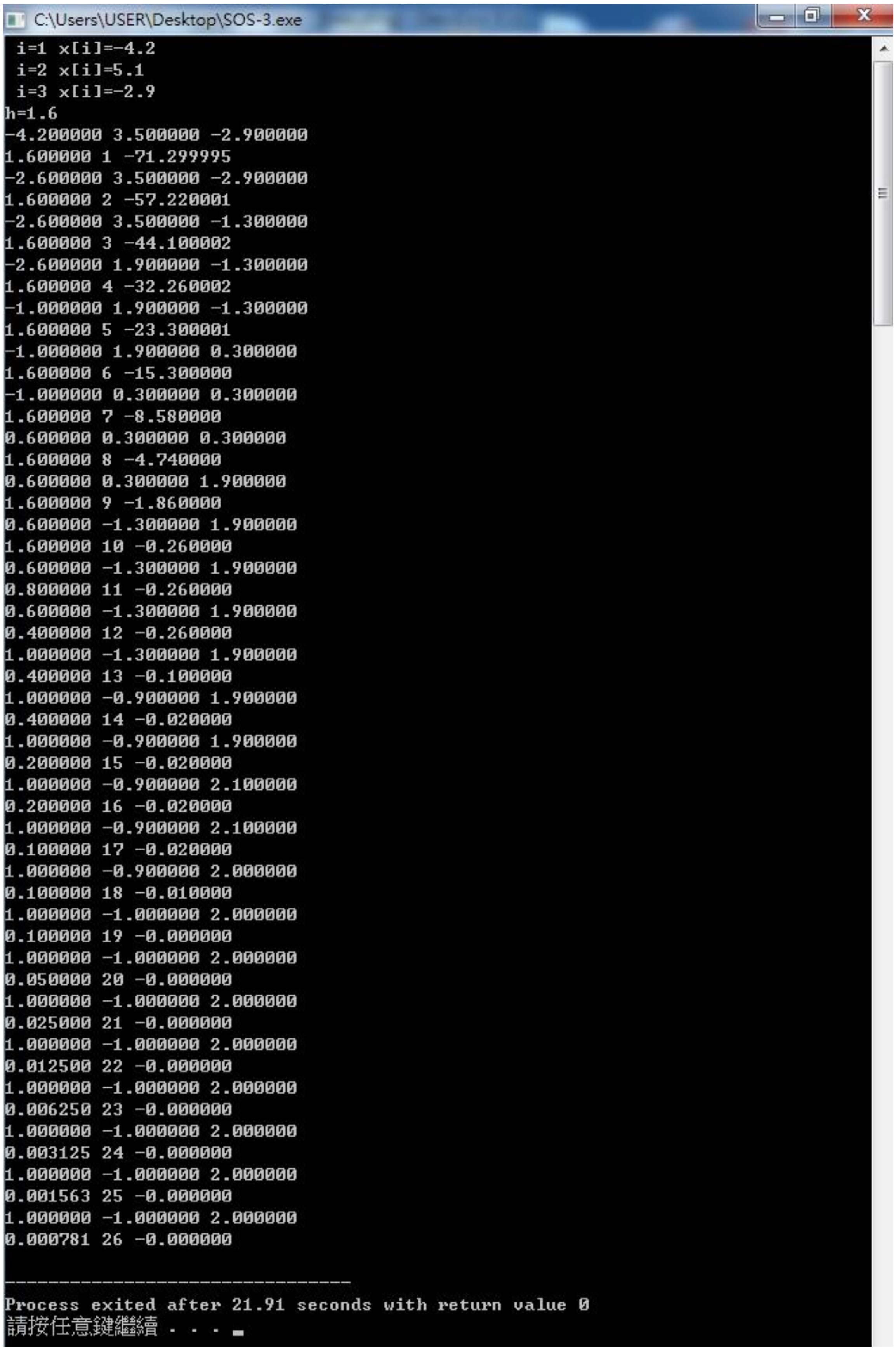
```
 i=1 x[i]=-2
 i=2 x[i]=1
 i=3 x[i]=0
 i=4 x[i]=-2
h=1
-1.000000 1.000000 0.000000 -2.000000
1.000000 1 -13.000000
-1.000000 1.000000 1.000000 -2.000000
1.000000 2 -10.000000
-1.000000 0.000000 1.000000 -2.000000
1.000000 3 -7.000000
0.000000 0.000000 1.000000 -2.000000
1.000000 4 -4.000000
0.000000 0.000000 1.000000 -3.000000
1.000000 5 -3.000000
0.000000 0.000000 2.000000 -3.000000
1.000000 6 -2.000000
0.000000 -1.000000 2.000000 -3.000000
1.000000 7 -1.000000
1.000000 -1.000000 2.000000 -3.000000
1.000000 8 -0.000000
1.000000 -1.000000 2.000000 -3.000000
0.500000 9 -0.000000
1.000000 -1.000000 2.000000 -3.000000
0.250000 10 -0.000000
1.000000 -1.000000 2.000000 -3.000000
0.125000 11 -0.000000
1.000000 -1.000000 2.000000 -3.000000
0.062500 12 -0.000000
1.000000 -1.000000 2.000000 -3.000000
0.031250 13 -0.000000
1.000000 -1.000000 2.000000 -3.000000
0.015625 14 -0.000000
1.000000 -1.000000 2.000000 -3.000000
0.007813 15 -0.000000
1.000000 -1.000000 2.000000 -3.000000
0.003906 16 -0.000000
1.000000 -1.000000 2.000000 -3.000000
0.001953 17 -0.000000
1.000000 -1.000000 2.000000 -3.000000
0.000977 18 -0.000000

--------------------------------
Process exited after 26.51 seconds with return value 0
請按任意鍵繼續 . . .
```

Case 2 Find the maximum of $f(c,w,u)=-(c-1)^2-(w+1)^2-(u-2)^2$ by C/C++ program.

```
#include<stdio.h>

#include<stdlib.h>

#include<math.h>

float f3(float c, float w, float u)

{

float f3=-pow(c-1,2)-pow(w+1,2)-pow(u-2,2);

return (f3);

}


void ASOS()

{

int n,m;

float x[50],xc[50],h, rmax, gg[100],y[100][50];

           n=3;

        for   (int i=1; i<= n; i++)

        {

           printf(" i=%d x[i]=", i);

           scanf ("%f", &xc[i]);

        }

      printf("h=");

      scanf ("%f", &h);

        for   (int j=1; j<=n; j++)
```

```
     {
              x[j]=xc[j];
       }
m=0;
     while (h>0.001)
     {
gg[0]=f3(x[1],x[2],x[3]) ;
rmax=gg[0];
     for ( int i=0; i<=2*n; i++)
     {
             for (int j=1; j<=n; j++)
           {
               if (2*j==(i+1))
                     { y[i][j]=x[j]+h; }
             else if    ((2*j)==i)
                   { y[i][j]=x[j]-h; }
             else
                     { y[i][j]=x[j]; }
             }
     }


         for (int i=0; i<=2*n; i++)
       {
              for (int j=1; j<=n; j++)
             {
                     x[j]=y[i][j];
             }
            gg[i]=f3(x[1],x[2],x[3]) ;
            if (gg[i]>=rmax)
        { rmax=gg[i]; }
        }
   if (gg[0]==rmax)
    { h=h/2; }
     for (int j;    j<=n; j++)
     {
            x[j]=xc[j];
     }
   if (gg[0]!=rmax)
 {
       for (int i=1; i<=2*n; i++)
     {
            if (rmax==gg[i])
          {
                for (int j=1; j<=n; j++)
                {
                    xc[j]=y[i][j];
```

```
                        }
                   }
                }
               }
               for (int j=1;    j<=n; j++)
               {
                   x[j]=xc[j];
               }
                       m=m+1;
                         printf("%f %f %f \n", x[1],x[2],x[3]);
                         printf("%f %d %f \n", h,m,rmax);
             }
}


main ()
{
ASOS();
}
```


C:\Users\USER\Desktop\SOS-3.exe

```
 i=1 x[i]=-4.2
 i=2 x[i]=5.1
 i=3 x[i]=-2.9
h=1.6
-4.200000 3.500000 -2.900000
1.600000 1 -71.299995
-2.600000 3.500000 -2.900000
1.600000 2 -57.220001
-2.600000 3.500000 -1.300000
1.600000 3 -44.100002
-2.600000 1.900000 -1.300000
1.600000 4 -32.260002
-1.000000 1.900000 -1.300000
1.600000 5 -23.300001
-1.000000 1.900000 0.300000
1.600000 6 -15.300000
-1.000000 0.300000 0.300000
1.600000 7 -8.580000
0.600000 0.300000 0.300000
1.600000 8 -4.740000
0.600000 0.300000 1.900000
1.600000 9 -1.860000
0.600000 -1.300000 1.900000
1.600000 10 -0.260000
0.600000 -1.300000 1.900000
0.800000 11 -0.260000
0.600000 -1.300000 1.900000
0.400000 12 -0.260000
1.000000 -1.300000 1.900000
0.400000 13 -0.100000
1.000000 -0.900000 1.900000
0.400000 14 -0.020000
1.000000 -0.900000 1.900000
0.200000 15 -0.020000
1.000000 -0.900000 2.100000
0.200000 16 -0.020000
1.000000 -0.900000 2.100000
0.100000 17 -0.020000
1.000000 -0.900000 2.000000
0.100000 18 -0.010000
1.000000 -1.000000 2.000000
0.100000 19 -0.000000
1.000000 -1.000000 2.000000
0.050000 20 -0.000000
1.000000 -1.000000 2.000000
0.025000 21 -0.000000
1.000000 -1.000000 2.000000
0.012500 22 -0.000000
1.000000 -1.000000 2.000000
0.006250 23 -0.000000
1.000000 -1.000000 2.000000
0.003125 24 -0.000000
1.000000 -1.000000 2.000000
0.001563 25 -0.000000
1.000000 -1.000000 2.000000
0.000781 26 -0.000000


--------------------------------
Process exited after 21.91 seconds with return value 0
請按任意鍵繼續 . . .
```

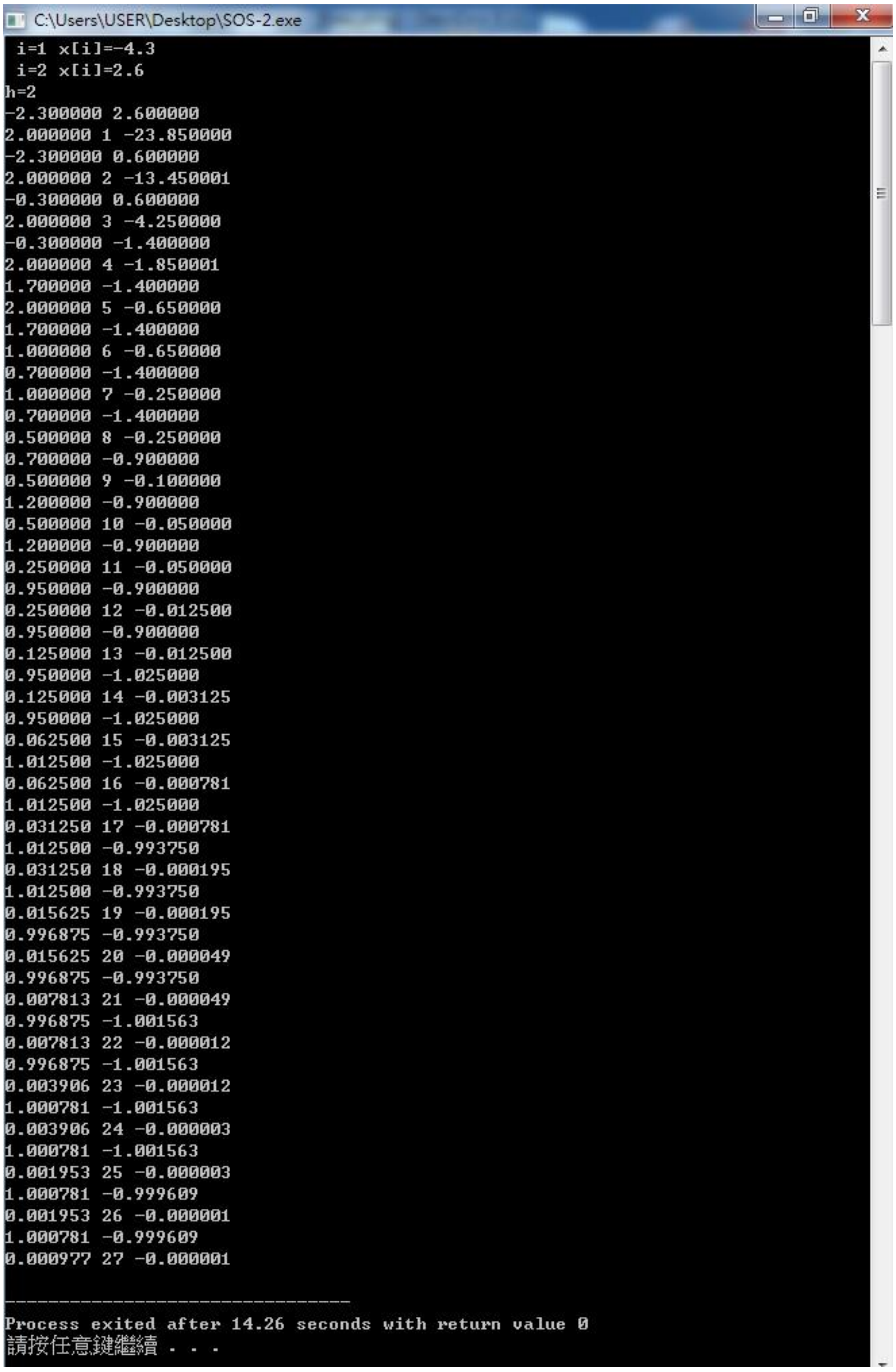
Case 3 Find the maximum of $f(c,w)=-(c-1)^2-(w+1)^2$ by C/C++ program.

```
#include<stdio.h>

#include<stdlib.h>

#include<math.h>

float f2(float c, float w)

{

float f2=-pow(c-1,2)-pow(w+1,2);

return (f2);

}


void ASOS()

{

int n,m;

float x[50],xc[50],h, rmax, gg[100],y[100][50];

            n=2;

        for    (int i=1; i<= n; i++)

        {

            printf(" i=%d x[i]=", i);

            scanf ("%f", &xc[i]);

        }

      printf("h=");

      scanf ("%f", &h);
```

```
    for    (int j=1; j<=n; j++)
    {
            x[j]=xc[j];
      }
m=0;
    while (h>0.001)
    {
gg[0]=f2(x[1],x[2]) ;
rmax=gg[0];
    for ( int i=0; i<=2*n; i++)
    {
          for (int j=1; j<=n; j++)
         {
            if (2*j==(i+1))
                 { y[i][j]=x[j]+h; }
          else if    ((2*j)==i)
               { y[i][j]=x[j]-h; }
          else
                 { y[i][j]=x[j]; }
          }
    }
```

```
   for (int i=0; i<=2*n; i++)
  {
            for (int j=1; j<=n; j++)
          {
                x[j]=y[i][j];
          }
         gg[i]=f2(x[1],x[2]) ;
         if (gg[i]>=rmax)
     { rmax=gg[i]; }
     }


  if (gg[0]==rmax)
   { h=h/2; }
   for (int j;    j<=n; j++)
   {
         x[j]=xc[j];
   }
  if (gg[0]!=rmax)
 {
      for (int i=1; i<=2*n; i++)
   {
         if (rmax==gg[i])
       {
```

```
                              for (int j=1; j<=n; j++)
                              {
                                  xc[j]=y[i][j];
                              }
                         }
                     }
                   }
                   for (int j=1;    j<=n; j++)
                   {
                        x[j]=xc[j];
                   }
                            m=m+1;
                              printf("%f %f \n", x[1],x[2]);
                              printf("%f %d %f \n", h,m,rmax);
                 }
}


main ()
{
ASOS();
}
```

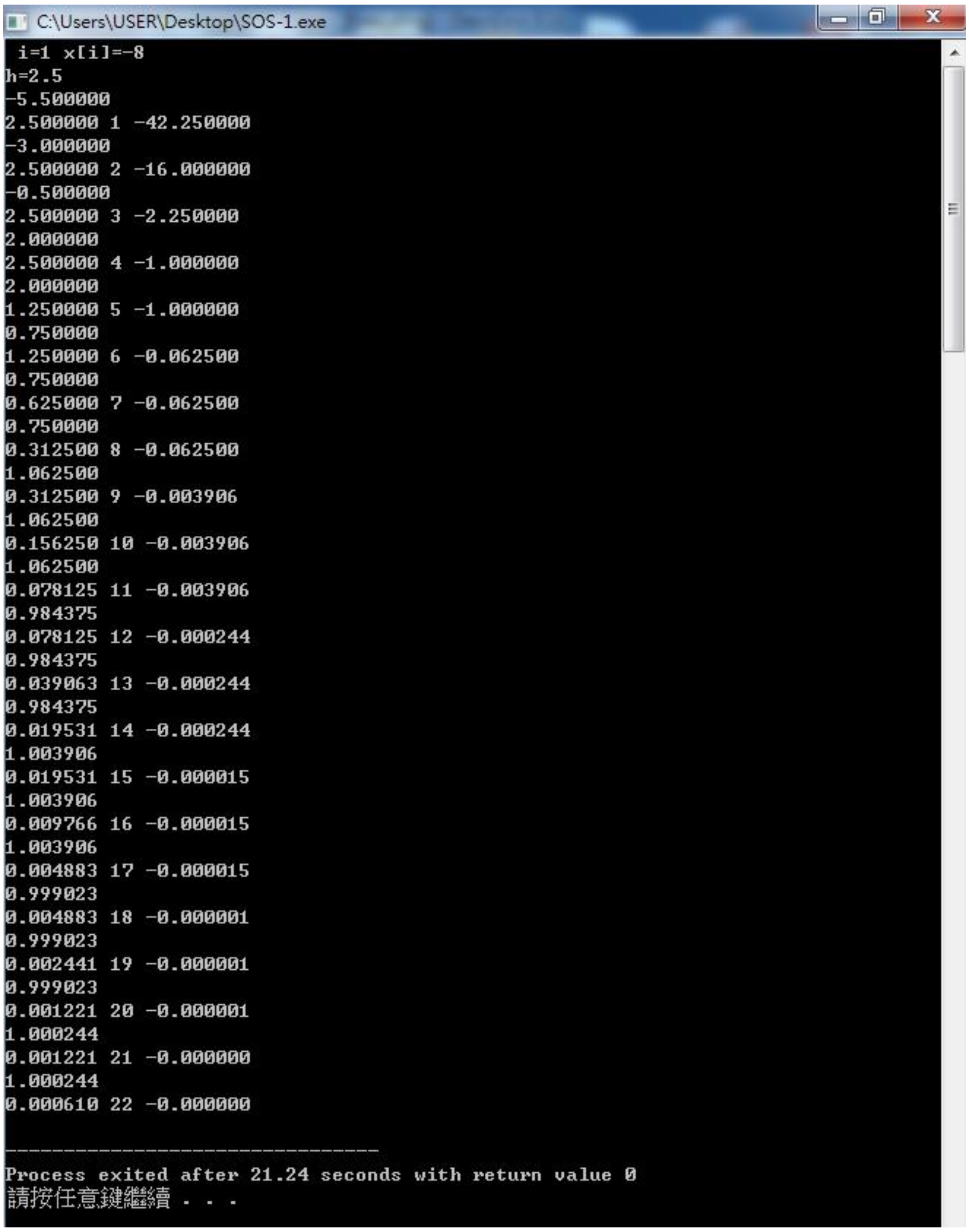


```
 i=1 x[i]=-4.3
 i=2 x[i]=2.6
h=2
-2.300000 2.600000
2.000000 1 -23.850000
-2.300000 0.600000
2.000000 2 -13.450001
-0.300000 0.600000
2.000000 3 -4.250000
-0.300000 -1.400000
2.000000 4 -1.850001
1.700000 -1.400000
2.000000 5 -0.650000
1.700000 -1.400000
1.000000 6 -0.650000
0.700000 -1.400000
1.000000 7 -0.250000
0.700000 -1.400000
0.500000 8 -0.250000
0.700000 -0.900000
0.500000 9 -0.100000
1.200000 -0.900000
0.500000 10 -0.050000
1.200000 -0.900000
0.250000 11 -0.050000
0.950000 -0.900000
0.250000 12 -0.012500
0.950000 -0.900000
0.125000 13 -0.012500
0.950000 -1.025000
0.125000 14 -0.003125
0.950000 -1.025000
0.062500 15 -0.003125
1.012500 -1.025000
0.062500 16 -0.000781
1.012500 -1.025000
0.031250 17 -0.000781
1.012500 -0.993750
0.031250 18 -0.000195
1.012500 -0.993750
0.015625 19 -0.000195
0.996875 -0.993750
0.015625 20 -0.000049
0.996875 -0.993750
0.007813 21 -0.000049
0.996875 -1.001563
0.007813 22 -0.000012
0.996875 -1.001563
0.003906 23 -0.000012
1.000781 -1.001563
0.003906 24 -0.000003
1.000781 -1.001563
0.001953 25 -0.000003
1.000781 -0.999609
0.001953 26 -0.000001
1.000781 -0.999609
0.000977 27 -0.000001

--------------------------------
Process exited after 14.26 seconds with return value 0
請按任意鍵繼續 . . .
```

Case 4 Find the maximum of $f(c)=-(c-1)^2$ by C/C++ program.

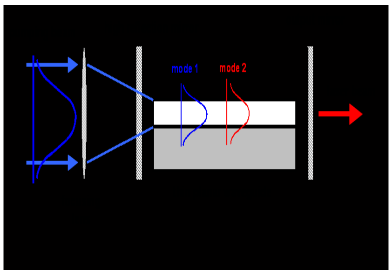
```
#include<stdio.h>

#include<stdlib.h>

#include<math.h>

float f1(float c)

{

float f1=-pow(c-1,2);

return (f1);

}



void ASOS()

{

int n,m;

float x[50],xc[50],h, rmax, gg[100],y[100][50];

           n=1;

        for   (int i=1; i<= n; i++)

        {

           printf(" i=%d x[i]=", i);

           scanf ("%f", &xc[i]);

        }

      printf("h=");

      scanf ("%f", &h);

        for   (int j=1; j<=n; j++)
      {
                x[j]=xc[j];
        }
m=0;
      while (h>0.001)
      {
gg[0]=f1(x[1]) ;
rmax=gg[0];
      for ( int i=0; i<=2*n; i++)
      {
              for (int j=1; j<=n; j++)
            {
                if (2*j==(i+1))
                      { y[i][j]=x[j]+h; }
              else if    ((2*j)==i)
                    { y[i][j]=x[j]-h; }
              else
                      { y[i][j]=x[j]; }
              }
      }


          for (int i=0; i<=2*n; i++)
        {
                for (int j=1; j<=n; j++)
              {
                      x[j]=y[i][j];
              }

            gg[i]=f1(x[1]) ;
            if (gg[i]>=rmax)
        { rmax=gg[i]; }
        }
    if (gg[0]==rmax)
     { h=h/2; }
      for (int j;    j<=n; j++)
      {
            x[j]=xc[j];
      }
    if (gg[0]!=rmax)
  {
        for (int i=1; i<=2*n; i++)
      {
            if (rmax==gg[i])
          {
                for (int j=1; j<=n; j++)
                {
```

```
                         xc[j]=y[i][j];
                      }
                 }
              }
            }
            for (int j=1;    j<=n; j++)
            {
                x[j]=xc[j];
            }
                   m=m+1;
                     printf("%f    \n", x[1]);
                     printf("%f %d %f \n", h,m,rmax);
          }
}


main ()
{
ASOS();
}
```



C:\Users\USER\Desktop\SOS-1.exe

```
 i=1 x[i]=-8
h=2.5
-5.500000
2.500000 1 -42.250000
-3.000000
2.500000 2 -16.000000
-0.500000
2.500000 3 -2.250000
2.000000
2.500000 4 -1.000000
2.000000
1.250000 5 -1.000000
0.750000
1.250000 6 -0.062500
0.750000
0.625000 7 -0.062500
0.750000
0.312500 8 -0.062500
1.062500
0.312500 9 -0.003906
1.062500
0.156250 10 -0.003906
1.062500
0.078125 11 -0.003906
0.984375
0.078125 12 -0.000244
0.984375
0.039063 13 -0.000244
0.984375
0.019531 14 -0.000244
1.003906
0.019531 15 -0.000015
1.003906
0.009766 16 -0.000015
1.003906
0.004883 17 -0.000015
0.999023
0.004883 18 -0.000001
0.999023
0.002441 19 -0.000001
0.999023
0.001221 20 -0.000001
1.000244
0.001221 21 -0.000000
1.000244
0.000610 22 -0.000000


--------------------------------
Process exited after 21.24 seconds with return value 0
請按任意鍵繼續 . . .
```



Eg. How can we obtain the optimal direct coupling efficiency between an optical waveguide and a fiber? It is desired to maximize

$$\eta=\frac{\left[\iint\varphi(x,y)\phi(x,y)dxdy\right]^2}{\iint\varphi^2(x,y)dxdy\cdot\iint\phi^2(x,y)dxdy}$$

Waveguide mode: $\varphi(x,y)=C_w\cdot\varphi_x(x)\varphi_y(y)$, where $\varphi_x(x)=\exp\left(-x^2\big/a_1^2\right)$ and $\varphi_y(y)=\exp\left[-(y-b)^2\big/a_i^2\right]$, $i=\begin{cases}2, & y\le b\\ 3, & y>b\end{cases}$.

Fiber mode: $\phi(x,y)=C_f\cdot\exp\left\{-\left[x^2+(y-c)^2\right]\big/w^2\right\}$

Utilize the formulae: $\int_0^A e^{-t^2}dt=\frac{\sqrt{\pi}}{2}erf(A)$, $\int_0^A e^{-Bt^2}dt=\frac{\sqrt{\pi}}{2\sqrt{B}}erf(A\sqrt{B})$, $\int_{-\infty}^0 e^{-(t-A)^2}dt=\frac{\sqrt{\pi}}{2}[1-erf(A)]$, and $\int_0^\infty e^{-(t-A)^2}dt=\frac{\sqrt{\pi}}{2}[1+erf(A)]$

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\phi(x,y)\varphi(x,y)dxdy=C_fC_w\int_{-\infty}^{\infty}e^{-(\frac{1}{w^2}+\frac{1}{a_1^2})x^2}dx\cdot\left[\int_{-\infty}^{b}e^{-\frac{(y-c)^2}{w^2}-\frac{(y-b)^2}{a_2^2}}dy+\int_b^\infty e^{-\frac{(y-c)^2}{w^2}-\frac{(y-b)^2}{a_3^2}}dy\right]$$

$$\int_{-\infty}^{\infty}e^{-(\frac{1}{w^2}+\frac{1}{a_1^2})x^2}dx=\frac{wa_1\sqrt{\pi}}{\sqrt{w^2+a_1^2}},$$

$$\int_{-\infty}^{b}e^{-\frac{(y-c)^2}{w^2}-\frac{(y-b)^2}{a_2^2}}dy=\int_{-\infty}^{0}e^{-\frac{[u-(c-b)]^2}{w^2}-\frac{u^2}{a_2^2}}du=e^{-\frac{(c-b)^2}{(a_2^2+w^2)}}\cdot\int_{-\infty}^{0}e^{-\frac{a_2^2+w^2}{w^2a_2^2}\cdot[u-\frac{a_2^2(c-b)}{a_2^2+w^2}]^2}du$$

$$=\frac{a_2w\sqrt{\pi}}{2\cdot\sqrt{a_2^2+w^2}}\cdot\exp\left[-\frac{(c-b)^2}{(a_2^2+w^2)}\right]\cdot\left\{1-erf\left[\frac{a_2(c-b)}{w\cdot\sqrt{a_2^2+w^2}}\right]\right\},$$

Similarly,

$$\int_{b}^{\infty}e^{-\frac{(y-c)^2}{w^2}-\frac{(y-b)^2}{a_3^2}}dy=\int_{0}^{\infty}e^{-\frac{[u-(c-b)]^2}{w^2}-\frac{u^2}{a_3^2}}du=e^{-\frac{(c-b)^2}{(a_3^2+w^2)}}\cdot\int_{0}^{\infty}e^{-\frac{a_3^2+w^2}{w^2a_3^2}\cdot[u-\frac{a_3^2(c-b)}{a_3^2+w^2}]^2}du$$

$$=\frac{a_3w\sqrt{\pi}}{2\cdot\sqrt{a_3^2+w^2}}\cdot\exp\left[-\frac{(c-b)^2}{(a_3^2+w^2)}\right]\cdot\left\{1+erf\left[\frac{a_3(c-b)}{w\cdot\sqrt{a_3^2+w^2}}\right]\right\}$$

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\phi^2(x,y)dxdy=C_f^{\,2}\cdot\frac{w^2\pi}{2},\ \text{and}\ \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\varphi^2(x,y)dxdy=C_w^{\,2}\cdot\frac{a_1(a_2+a_3)\pi}{4},$$

$$\Rightarrow\eta=\frac{2a_1w^2}{(a_2+a_3)(a_1^2+w^2)}\cdot(I_1+I_2)^2,\ \text{where}$$

$$I_1=\frac{a_2}{\sqrt{a_2^2+w^2}}\cdot\exp\left[-\frac{(c-b)^2}{(a_2^2+w^2)}\right]\cdot\left\{1-erf\left[\frac{a_2(c-b)}{w\cdot\sqrt{a_2^2+w^2}}\right]\right\}$$

$$I_2=\frac{a_3}{\sqrt{a_3^2+w^2}}\cdot\exp\left[-\frac{(c-b)^2}{(a_3^2+w^2)}\right]\cdot\left\{1+erf\left[\frac{a_3(c-b)}{w\cdot\sqrt{a_3^2+w^2}}\right]\right\}$$

(Sol.)

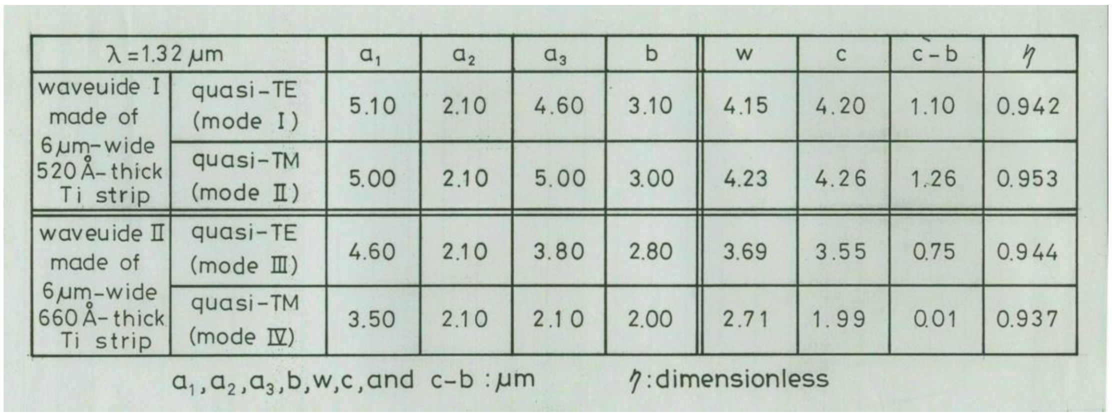
Optimal Direct Coupling from Single-Mode Fibers to Ti:LiNbO$_3$ Channel Waveguides

KEH-YI LEE
CHYI-JIEH HSIEH
JYH-ROU SZE
GWO-JIUNN JAW

Department of Electrical Engineering
Chinese Culture University
Taipei Taiwan, Republic of China

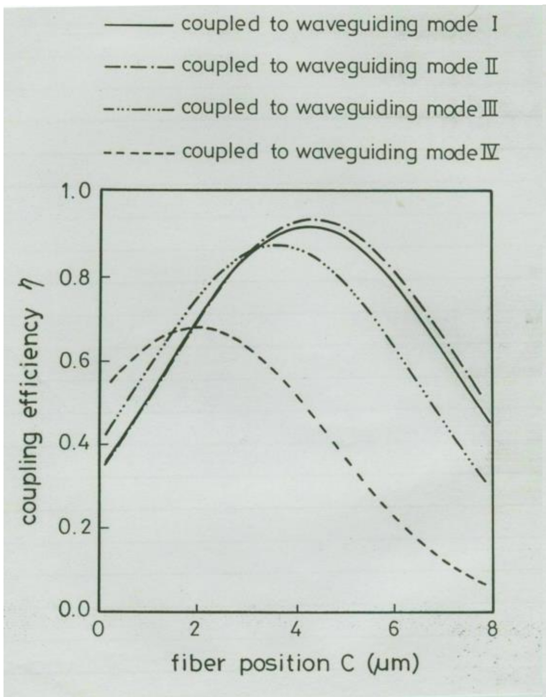
We investigate the optimal direct coupling from single-mode fibers to Ti:LiNbO$_3$ channel waveguides using a very general formula and a heuristic optimization technique in this article. The coupling efficiency of the optical power depends on both the fiber positions and the modal sizes of the incident fields. From our numerical simulation, it is found that the optimal positions of the fiber axes are not in alignment with the peaks of the waveguiding modal fields and that the coupling efficiency can be improved by microlenses.

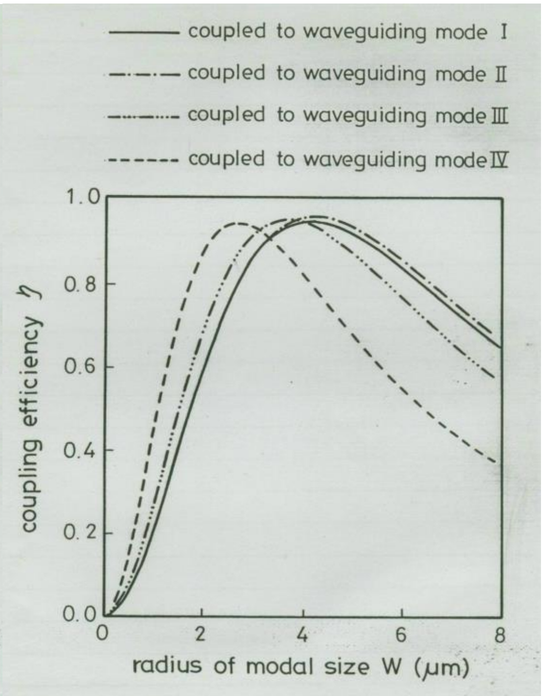
Keywords coupling efficiency, optimization



$\lambda = 1.32\ \mu m$		a_1	a_2	a_3	b	w	c	$c-b$	η
waveuide I made of 6 μm-wide 520 Å-thick Ti strip	quasi-TE (mode I)	5.10	2.10	4.60	3.10	4.15	4.20	1.10	0.942
	quasi-TM (mode II)	5.00	2.10	5.00	3.00	4.23	4.26	1.26	0.953
waveuide II made of 6 μm-wide 660 Å-thick Ti strip	quasi-TE (mode III)	4.60	2.10	3.80	2.80	3.69	3.55	0.75	0.944
	quasi-TM (mode IV)	3.50	2.10	2.10	2.00	2.71	1.99	0.01	0.937

a_1, a_2, a_3, b, w, c, and $c-b$: μm η : dimensionless





1-6 Steepest Descent Method in Optimizations

Find the minimum of $f(x_1,x_2,\dots,x_n)$: Initial guess is $x_0=(x_1^0,x_2^0,\dots,x_n^0)$, and

$$-\nabla f(x_1,\cdots,x_n)=\left(-\frac{\partial f(x_1,\cdots,x_n)}{\partial x_1},\cdots,-\frac{\partial f(x_1,\cdots,x_n)}{\partial x_n}\right)$$

Recursive formula: $x_{k+1}=x_k-\alpha_k\nabla f(x_k)$

$$\Rightarrow (x_1^{k+1},\cdots,x_n^{k+1})=\left(x_1^k-\alpha_k\cdot\frac{\partial f}{\partial x_1\Big|_{(x_1^k,\cdots,x_n^k)}},\cdots,x_n^k-\alpha_k\cdot\frac{\partial f}{\partial x_n\Big|_{(x_1^k,\cdots,x_n^k)}}\right)$$

Solve α_k of $\dfrac{d}{d\alpha_k}f(x_k-\alpha_k\nabla f(x_k))=0$

Eg. Find the minimum of $f(x,y)=x^2+y^2$.

(Sol.) $-\nabla f(x,y)=(-2x,-2y), -\alpha_k\nabla f(x,y)=(-2\alpha_k x,-2\alpha_k y)$

Initial guess: $(x^0,y^0)=(1,1)$

$$\frac{d}{d\alpha_0}f(1-2\alpha_0\cdot 1,1-2\alpha_0\cdot 1)=\frac{d}{d\alpha_0}[(1-2\alpha_0)^2+(1-2\alpha_0)^2]=0$$

$$\Rightarrow \alpha_0=\frac{1}{2}\Rightarrow (x^1,y^1)=(x_0-2\alpha_0\cdot x_0,y_0-2\alpha_0\cdot y_0)=(0,0)$$

$$\vdots$$

In fact, (0,0) is the position of the minimum position of $f(x,y)=x^2+y^2$.

1-7 Optimizations in Constrained Problems

In the previous section, only unconstrained problems are encountered. If there exist some constraints, some new methods are needed to solve them.

1. Lagrange Multiplier Method: **Find the extrema of a function with some constraint equations.**

2. Penalty Function Method: **Find the extrema of a function with some constraint inequalities.**

Eg. Find the minimum of $f(x)=\sin(x)$, $\pi/2<x<7\pi/4$.

(Sol.) Original problem has infinite local minima

Under the following constraints, only one minimum point by setting

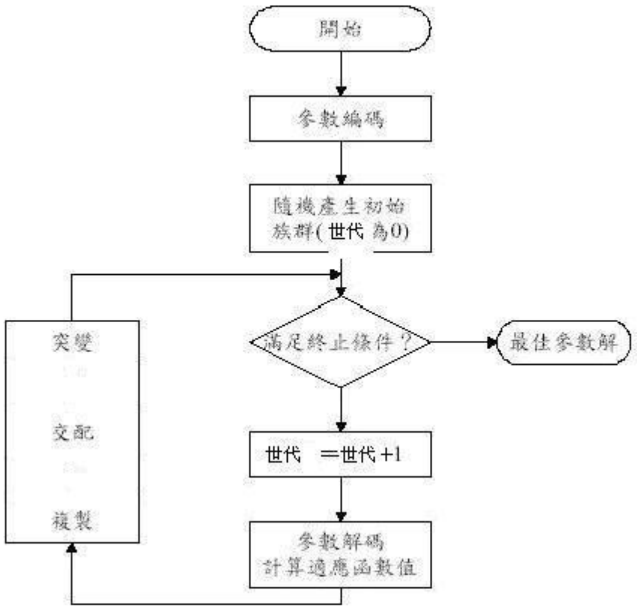
$$g(x)=\begin{cases}\sin(x) & , \dfrac{\pi}{2}<x<\dfrac{7\pi}{4}\\ 100000 & , \ elsewhere\end{cases}$$

To sum up, imposing some appropriate penalty function outside the constraint ranges, and then we can solve the constraint problem by the previous method as well as the unconstraint ones.

Appendix-Genetic Algorithm(基因演算法)

基因演算法特性

- 最佳化過程只針對適應函數處理
- 透過編碼技術可以直接在空間中找到全域最佳解



基因演算法專有名詞

- 個體 (individual)
- 世代(generation)
- 族群大小 (population size)
- 交配率 (crossover rate)
- 突變率 (mutation rate)
- 適應函數 (fitness function)

基因演算法主要運算

- 複製

L：個體數目

fi：個體適應值 (*i*=1,2,…,*L*)

一個個體被複製到下一代的機率為

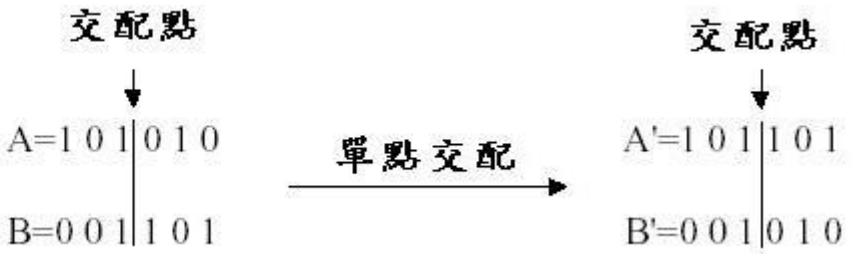
$$P_i = \frac{f_i}{\sum_{i=1}^{L} f_i} \qquad i = 1, 2, \cdots, L$$

$$n_i = L \cdot P_i \qquad i = 1, 2, \cdots, L$$

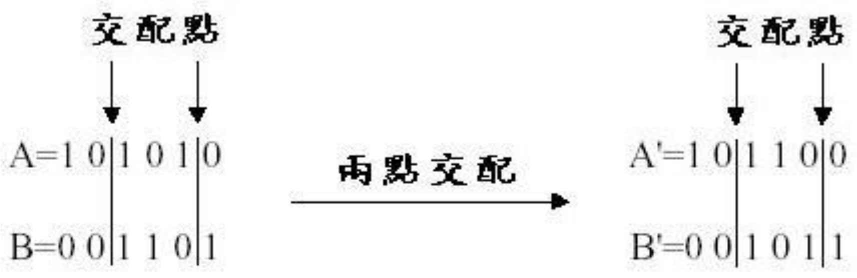
- 交配

有三種主要交配方式：(1)單點交配(one-point crossover) (2)兩點交配(two-point crossover) (3)均勻交配(uniform crossover)

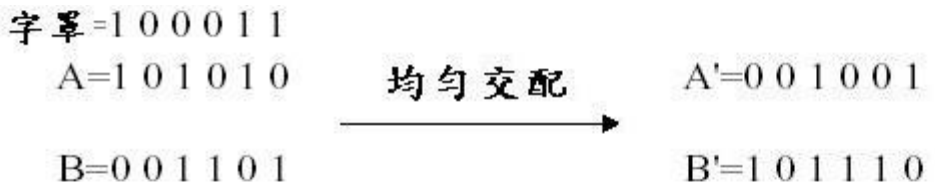
單點交配



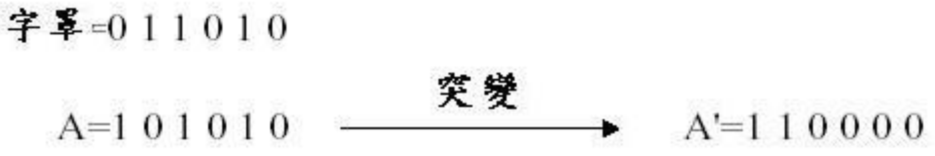
兩點交配



均勻交配



突變



由於突變不會常常發生，可以假設適當的機率函數來設定突變方式。

Eg. Find the maximum of $f(x)$=-(x-11)^2 by the genetic algorithm.

編碼方式：以 5 個 bits 來表示，例如 0=00000, …, 31=11111

演化機制：輪盤選擇法或四捨五入決定

目標函數：$f(x)=-(x-11)^2$

求極大值：$f_i(x)=-(x-11)^2$-min.

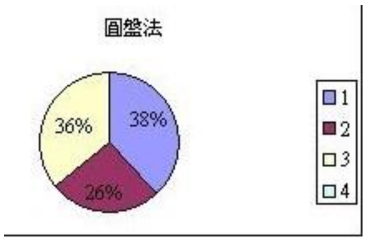
(Sol.) 以傳統數學方法可知：當 x=11 時，$f(x)$之極大值為 0

Genetic Algorithm

The first step

個體編號(假設只猜測四個起始值)	隨機產生的第一代族群(二進位)	x 值	$f(x)$	$f_i(x)=-(x-11)^2$-min	$f_i / \sum f_i$	$f_i / \overline{f}$	實際值(由輪盤或四捨五入決定)
1	01100	12	-1	48	0.38	1.52	2
2	01111	15	-16	33	0.26	1.05	1
3	01001	9	-4	45	0.36	1.43	1
4	10010	18	-49	0	0	0	0
總和 $\sum f_i$ =126							
平均 $\overline{f}$ =31.5							
Max of $f(x)$ = -1							

經由演化機制，吾人淘汰 x=18(=10010)，而繁殖 x=12(=01100)為 2 個



採用單點交配與突變：

經選擇後的個體編號	經選擇後的個體	交配的配對(隨機產生)	交配的位置點(隨機產生)	新產生的個體(二進位)	x 值(第二代族群)
1	**01100**	4	1	**01001**	9
2	**01100**	3	4	**01101**	13
3	**01111**	2	4	**01110**	**14*** → 10
4	**01001**	1	1	**01100**	12

假設第二代族群中的 x=14*(=01110)發生基因突變成為 x=10(=01010)

⇒ **The second step**

個 體 編號	第二代族群	x 值	$f(x)$	$f_i(x)$=-(x-11)^2-min	$f_i/\sum f_i$	$f_i/\overline{f}$	實際值 (由輪盤或四捨五入決定)
1	01001	9	-4	0	0	0	0
2	01101	13	-4	0	0	0	0
3	01010	10	-1	3	0.5	2	2*
4	01100	12	-1	3	0.5	2	1+1*
總和 $\sum f_i$ =6							
平均 $\overline{f}$ =1.5							
Max of $f(x)$= 0							

經由演化機制，吾人淘汰 x=9(=01001)與 x=13(=01101)，而繁殖 x=10(=01010)與 x=12(=01100) 各為 2 個，但是假設兩個 x=10(=01010) 均發生突變為 x=11(=01011)，而其中一個 x=12(=01100)突變為 x=8(=01000)

採用單點交配與突變：

經選擇後的個體編號	經選擇後的個體	交配的配對(隨機產生)	交配的位置點(隨機產生)	新產生的個體(二進位)	x 值 (第三代族群)
1	01**011**	3	2	01**100**	12
2	0101**1**	4	4	0101**0**	10
3	01**100**	1	2	01**011**	11
4	0100**0**	2	4	0100**1**	9

⇒ **The third step**

個 體 編號	第三代 族群	*x* 值	$f(x)$	$f_i(x)=-(x-11)^2$-min	$f_i/\sum f_i$	$f_i/\overline{f}$	實際值 (由輪盤或四捨五入決定)
1	01100	12	-1	3	0.3	1.2	1
2	01010	10	-1	3	0.3	1.2	1
3	01011	11	0	4	0.4	1.6	2
4	01001	9	-4	0	0	0	0
總和 $\sum f_i$ =10							
平均 $\overline{f}$ =2.5							
Max of $f(x)$= 0							

經由演化機制，吾人淘汰 *x*=9(=01001)，而繁殖 *x*=11(=01011)為 2 個

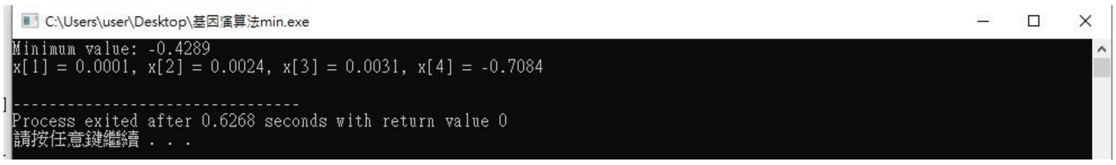
採用單點交配與突變：

經選擇後的個體編號	經選擇後的個體	交配的配對(隨機產生)	交配的位置點(隨機產生)	新產生的個體(二進位)	*x* 值 (第四代族群)
1	0110**0**	2	4	01100	12
2	0101**0**	1	4	01010	10
3	010**11**	3	3	01011	11
4	010**11**	4	3	01011	11

吾人將可發現 *x* 的四個起始值自第一代族群(12,15,9,18)而至第二代族群(9,13,10,12)，而至第三代族群(12,10,11,9)，而至第四代族群(12,10,11,11)，最後所有族群逐漸收斂至 11，而 $f(x)$的極大值則逐漸收斂至 0

基因演算法求極小值

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>
#define POPULATION_SIZE 100
#define MAX_GENERATIONS 1000
#define MUTATION_RATE 0.01
#define imax 4

// Function to calculate the fitness value of an individual
// double calculateFitness(double x, double y, double z, double w) {
   double calculateFitness(double x1, double x2, double x3, double x4) {
//      return 2 * ((x - 3) * (x - 3)) + ((y + 5.1) * (y + 5.1)) + 4 * ((z - 4.8) * (z - 4.8));
return x4*exp(-x1*x1-x2*x2-x3*x3-x4*x4);
}

// Function to generate a random number between min and max
double getRandomNumber(double min, double max) {
    return ((double)rand() / RAND_MAX) * (max - min) + min;
}

// Structure representing an individual in the population
typedef struct {
    double x[5];
    double fitness;
} Individual;

// Function to initialize an individual with random values
void initializeIndividual(Individual *individual) {
//      individual->x = getRandomNumber(-10, 10);
//      individual->y = getRandomNumber(-10, 10);
//      individual->z = getRandomNumber(-10, 10);
//      individual->w = getRandomNumber(-10, 10);
for (int i=1;i<=imax;i++)
{ individual->x[i] = getRandomNumber(-10, 10);
}
    individual->fitness = calculateFitness(individual->x[1], individual->x[2], individual->x[3],
individual->x[4]);
}

// Function to mutate an individual by adding a small random value
void mutateIndividual(Individual *individual) {
//      individual->x += getRandomNumber(-1, 1) * MUTATION_RATE;
//      individual->y += getRandomNumber(-1, 1) * MUTATION_RATE;
//      individual->z += getRandomNumber(-1, 1) * MUTATION_RATE;
//      individual->w += getRandomNumber(-1, 1) * MUTATION_RATE;
for (int i=1;i<=imax;i++)
{ individual->x[i] += getRandomNumber(-1, 1) * MUTATION_RATE;
}
    individual->fitness = calculateFitness(individual->x[1], individual->x[2], individual->x[3],
individual->x[4]);
}

// Function to perform tournament selection to choose parents for reproduction
Individual selectParent(Individual *population) {
    int tournamentSize = 5;
    int selected = rand() % POPULATION_SIZE;

    for (int i = 0; i < tournamentSize; i++) {
        int index = rand() % POPULATION_SIZE;
        if (population[index].fitness < population[selected].fitness) {
            selected = index;
        }
    }

    return population[selected];
}

// Function to perform crossover between two parents to create offspring
Individual crossover(Individual parent1, Individual parent2) {
    Individual offspring;
//      offspring.x = (parent1.x + parent2.x) / 2.0;
//      offspring.y = (parent1.y + parent2.y) / 2.0;
//      offspring.z = (parent1.z + parent2.z) / 2.0;
//      offspring.w = (parent1.w + parent2.w) / 2.0;
for (int i=1;i<=imax;i++)
{ offspring.x[i] = (parent1.x[i] + parent2.x[i]) /2.0;
}
    offspring.fitness = calculateFitness(offspring.x[1], offspring.x[2], offspring.x[3], offspring.x[4]);
    return offspring;
}

int main() {
    srand(time(NULL));

    Individual population[POPULATION_SIZE];

    // Initialize the population
    for (int i = 0; i < POPULATION_SIZE; i++) {
        initializeIndividual(&population[i]);
    }

    int generation = 0;

    // Evolution loop
    while (generation < MAX_GENERATIONS) {
        // Create a new population
        Individual newPopulation[POPULATION_SIZE];

        for (int i = 0; i < POPULATION_SIZE; i++) {
            // Select two parents
            Individual parent1 = selectParent(population);
            Individual parent2 = selectParent(population);

            // Perform crossover to create offspring
            Individual offspring = crossover(parent1, parent2);

            // Mutate the offspring
            mutateIndividual(&offspring);

            // Add the offspring to the new population
            newPopulation[i] = offspring;
        }
        // Replace the old population with the new population
        for (int i = 0; i < POPULATION_SIZE; i++) {
            population[i] = newPopulation[i];
        }

        generation++;
    }

    // Find the individual with the minimum fitness value
    Individual bestIndividual = population[0];
    for (int i = 1; i < POPULATION_SIZE; i++) {
        if (population[i].fitness < bestIndividual.fitness) {
            bestIndividual = population[i];
        }
    }

    // Print the result
    printf("Minimum value: %.4f\n", bestIndividual.fitness);
    printf("x[1] = %.4f, x[2] = %.4f, x[3] = %.4f, x[4] = %.4f\n", bestIndividual.x[1], bestIndividual.x[2],
bestIndividual.x[3], bestIndividual.x[4]);

    return 0;
}
```



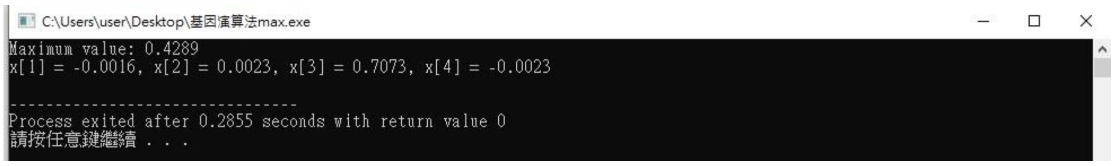
C:\Users\user\Desktop\基因演算法min.exe

```
Minimum value: -0.4289
x[1] = 0.0001, x[2] = 0.0024, x[3] = 0.0031, x[4] = -0.7084

--------------------------------
Process exited after 0.6268 seconds with return value 0
請按任意鍵繼續 . . .
```

基因演算法求極大值

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>
#define POPULATION_SIZE 100
#define MAX_GENERATIONS 1000
#define MUTATION_RATE 0.01
#define imax 4

// Function to calculate the fitness value of an individual
// double calculateFitness(double x, double y, double z, double w) {
    double calculateFitness(double x1, double x2, double x3, double x4) {
//      return 2 * ((x - 3) * (x - 3)) + ((y + 5.1) * (y + 5.1)) + 4 * ((z - 4.8) * (z - 4.8));
return x3*exp(-x1*x1-x2*x2-x3*x3-x4*x4);
}

// Function to generate a random number between min and max
double getRandomNumber(double min, double max) {
    return ((double)rand() / RAND_MAX) * (max - min) + min;
}

// Structure representing an individual in the population
typedef struct {
    double x[5];
    double fitness;
} Individual;

// Function to initialize an individual with random values
void initializeIndividual(Individual *individual) {
//      individual->x = getRandomNumber(-10, 10);
//      individual->y = getRandomNumber(-10, 10);
//      individual->z = getRandomNumber(-10, 10);
//      individual->w = getRandomNumber(-10, 10);
for (int i=1;i<=imax;i++)
{ individual->x[i] = getRandomNumber(-10, 10);
}
    individual->fitness = calculateFitness(individual->x[1], individual->x[2], individual->x[3],
individual->x[4]);
}

// Function to mutate an individual by adding a small random value
void mutateIndividual(Individual *individual) {
//      individual->x += getRandomNumber(-1, 1) * MUTATION_RATE;
//      individual->y += getRandomNumber(-1, 1) * MUTATION_RATE;
//      individual->z += getRandomNumber(-1, 1) * MUTATION_RATE;
//      individual->w += getRandomNumber(-1, 1) * MUTATION_RATE;
for (int i=1;i<=imax;i++)
{ individual->x[i] += getRandomNumber(-1, 1) * MUTATION_RATE;
}
    individual->fitness = calculateFitness(individual->x[1], individual->x[2], individual->x[3],
individual->x[4]);
}

// Function to perform tournament selection to choose parents for reproduction
Individual selectParent(Individual *population) {
    int tournamentSize = 5;
    int selected = rand() % POPULATION_SIZE;

    for (int i = 0; i < tournamentSize; i++) {
        int index = rand() % POPULATION_SIZE;
        if (population[index].fitness > population[selected].fitness) {
            selected = index;
        }
    }

    return population[selected];
}

// Function to perform crossover between two parents to create offspring
Individual crossover(Individual parent1, Individual parent2) {
    Individual offspring;
//     offspring.x = (parent1.x + parent2.x) / 2.0;
//     offspring.y = (parent1.y + parent2.y) / 2.0;
//     offspring.z = (parent1.z + parent2.z) / 2.0;
//     offspring.w = (parent1.w + parent2.w) / 2.0;
for (int i=1;i<=imax;i++)
{ offspring.x[i] = (parent1.x[i] + parent2.x[i]) /2.0;
}
    offspring.fitness = calculateFitness(offspring.x[1], offspring.x[2], offspring.x[3], offspring.x[4]);
    return offspring;
}

int main() {
    srand(time(NULL));

    Individual population[POPULATION_SIZE];

    // Initialize the population
    for (int i = 0; i < POPULATION_SIZE; i++) {
        initializeIndividual(&population[i]);
    }

    int generation = 0;

    // Evolution loop
    while (generation < MAX_GENERATIONS) {
        // Create a new population
        Individual newPopulation[POPULATION_SIZE];

        for (int i = 0; i < POPULATION_SIZE; i++) {
            // Select two parents
            Individual parent1 = selectParent(population);
            Individual parent2 = selectParent(population);

            // Perform crossover to create offspring
            Individual offspring = crossover(parent1, parent2);

            // Mutate the offspring
            mutateIndividual(&offspring);

            // Add the offspring to the new population
            newPopulation[i] = offspring;
        }
```

```
            // Replace the old population with the new population
            for (int i = 0; i < POPULATION_SIZE; i++) {
                population[i] = newPopulation[i];
            }

            generation++;
        }

        // Find the individual with the minimum fitness value
        Individual bestIndividual = population[0];
        for (int i = 1; i < POPULATION_SIZE; i++) {
            if (population[i].fitness > bestIndividual.fitness) {
                bestIndividual = population[i];
            }
        }

        // Print the result
        printf("Maximum value: %.4f\n", bestIndividual.fitness);
        printf("x[1] = %.4f, x[2] = %.4f, x[3] = %.4f, x[4] = %.4f\n", bestIndividual.x[1], bestIndividual.x[2],
    bestIndividual.x[3], bestIndividual.x[4]);

        return 0;
}
```



C:\Users\user\Desktop\基因演算法max.exe

```
Maximum value: 0.4289
x[1] = -0.0016, x[2] = 0.0023, x[3] = 0.7073, x[4] = -0.0023

--------------------------------
Process exited after 0.2855 seconds with return value 0
請按任意鍵繼續 . . .
```