The Program Verification of the Three-Seeking and Six-Seeking Method Based on the Conjugate Direction

Chunming LI[a], Xiaohua ZHANG[b], Xiaoli YIN[c]

Department of Mechanical Engineering, School of Mechanical and Control Engineering, Shengli College, China University of Petroleum, China University of Petroleum (East China), Dongying 257061, China

[a]email: lchming@126.com, [b]email: 1489974950@qq.com, [c]email: 271656902@qq.com

Keywords: Optimal Design Optimization Method, Conjugate Direction, Seven-seeking Method, Three-seeking Method, Program Verification

Abstract. Conjugate direction is a better direction of optimization. The three-seeking method and the six-seeking method based on conjugate direction have strong theoretical. Using the advantages of one-dimensional blind walking optimization algorithm, the optimal point was given on the given direction. Three optimization method, six optimization method and computer program for the blind walking algorithm for 3D design space were put forward. The given programs were of universal significance. The results show that the three-seeking method and the six-seeking method have better performance than the negative gradient method.

Introduction

The discipline of optimization method has a history of more than half a century. Since the 20th century in 60s by external punishment function method (mound method) has successfully solved the general optimization problem, optimization method has opened gradually in the world as an university course. The optimization method includes two aspects: algorithm and program verification. The algorithm needs mathematical, logic, thinking and other innovative ability. Validation requires talent, carefulness, and perseverance. There is no program to verify the algorithm is the tree without root, there is no algorithm to support the program is the water without source. In this paper, the program is designed based on the algorithm, and the algorithm and program are verified by an example.

Three-Seeking and Six-Seeking Method

From two different points, two optimal points is gotten by one-dimensional optimization along a given direction. The connection of two optimal points and the given direction are conjugated with the two-order partial derivative matrix [1]. This is one of the theoretical basic of the conjugate method in adjacent directions.

If the direction is optimized along the negative gradient direction, the last direction is the tangent of the objective function contour line at the optimal point, and the next direction is the normal. The adjacent two seeking directions are perpendicular to each other. Therefore, it is possible to find the direction which is conjugated with the last negative gradient direction about the two order partial derivative matrix of the objective function after two successive optimizations along the negative gradient direction. The best optimal point can be obtained by optimizing the conjugate direction. The algorithm is then executed taking the optimal point as the initial point until the termination condition is satisfied. This is the basic idea of the three-seeking method.

It is possible to find two directions which are conjugated with the last two negative gradient directions about the two order partial derivative matrix of the objective function respectively after three successive optimizations along the negative gradient direction. Two better optimal points can be obtained by optimizing along these two conjugate directions. Then, the one-dimensional optimization is carried out along these two optimal points, and the optimal point is obtained as the initial point of the next loop of optimization. Loop executes the above algorithm until the

termination condition is satisfied. This is the basic idea of the six-seeking method.

The Program of the Three-Seeking Method

```
int seek_3(x0,y0,e1,h) /*initial point( return optimal), stopping value, step size*/
   double x0[3],y0[1],e1,h;
   {int i;
    double x1[3],y1[1],x2[3],y2[1],x3[3],y3[1], dum, grad[3], s[3];
   for(i=0;i<6;i++) /*computing loop*/
      {gradu(x0[0],x0[1],x0[2],grad); /* Gradient at the current point */
        dum=sqrt(grad[0]*grad[0]+grad[1]*grad[1]+grad[2]*grad[2]);
       if(dum<e1)return; /*stopping condition*/
        s[0]=-grad[0]/dum; s[1]=-grad[1]/dum;    s[2]=-grad[2]/dum; /*seeking direction is united*/
       blind3d(x0,y0,s,e1,h,x1,y1); /*blind walking algorithm sub-program*/
       gradu(x1[0],x1[1],x1[2],grad);
        dum=sqrt(grad[0]*grad[0]+grad[1]*grad[1]+grad[2]*grad[2]);
        s[0]=-grad[0]/dum;          s[1]=-grad[1]/dum;        s[2]=-grad[2]/dum;
       blind3d(x1,y1,s,e1,h,x2,y2); /* blind walking algorithm sub-program */
       s[0]=x2[0]-x0[0];s[1]=x2[1]-x0[1];s[2]=x2[2]-x0[2];
        dum=sqrt(s[0]*s[0]+s[1]*s[1]+s[2]*s[2]);
        s[0]=s[0]/dum;        s[1]=s[1]/dum;        s[2]=s[2]/dum;
       blind3d(x2,y2,s,e1,h,x0,y0); /* blind walking algorithm sub-program */
      }
   }
```

The Program of the Six-Seeking Method

```
int seek_6(x0,y0,e1,h) /* initial point( return optimal), stopping value, step size */
   double x0[3],y0[1],e1,h;
   {int i,j;
    double x1[3],y1[1],x2[3],y2[1],x3[3],y3[1];
    double x4[3],y4[1],x5[3],y5[1],x6[3],y6[1];
    double dum,dum1,dum2,grad[3],s[3],s1[3],s2[3];
    for(i=0;;i++)
     {gradu(x0[0],x0[1],x0[2],grad);
        dum=sqrt(grad[0]*grad[0]+grad[1]*grad[1]+grad[2]*grad[2]);
       if(dum<e1)return;
        s[0]=-grad[0]/dum;        s[1]=-grad[1]/dum;        s[2]=-grad[2]/dum;
       blind3d(x0,y0,s,e1,h,x1,y1);
       gradu(x1[0],x1[1],x1[2],grad);
        dum=pow(grad[0]*grad[0]+grad[1]*grad[1]+grad[2]*grad[2],0.5);
        s[0]=-grad[0]/dum;          s[1]=-grad[1]/dum;        s[2]=-grad[2]/dum;
        blind3d(x1,y1,s,e1,h,x2,y2);
       gradu(x2[0],x2[1],x2[2],grad);
        dum=pow(grad[0]*grad[0]+grad[1]*grad[1]+grad[2]*grad[2],0.5);
        s[0]=-grad[0]/dum;          s[1]=-grad[1]/dum;        s[2]=-grad[2]/dum;
        blind3d(x2,y2,s,e1,h,x3,y3);
       s[0]=x2[0]-x0[0];s[1]=x2[1]-x0[1];s[2]=x2[2]-x0[2]; /*x0-->x2,obtain x4*/
        dum=pow(s[0]*s[0]+s[1]*s[1]+s[2]*s[2],0.5);
        s[0]=s[0]/dum;          s[1]=s[1]/dum;        s[2]=s[2]/dum;
        blind3d(x2,y2,s,e1,h,x4,y4);
       s[0]=x3[0]-x1[0];s[1]=x3[1]-x1[1];s[2]=x3[2]-x1[2]; /*x1-->x3,obtain x5*/
        dum=pow(s[0]*s[0]+s[1]*s[1]+s[2]*s[2],0.5);
```

```
      s[0]=s[0]/dum;      s[1]=s[1]/dum;      s[2]=s[2]/dum;
      blind3d(x3,y3,s,e1,h,x5,y5);
     s[0]=x5[0]-x4[0];s[1]=x5[1]-x4[1];s[2]=x5[2]-x4[2]; /*x4-->x5,obtain x6(x0)*/
      dum=pow(s[0]*s[0]+s[1]*s[1]+s[2]*s[2],0.5);
      s[0]=s[0]/dum;      s[1]=s[1]/dum;      s[2]=s[2]/dum;
      blind3d(x5,y5,s,e1,h,x0,y0);
    }
  }
```

The Program of Blind-Walking Method For 3-D

```
void blind3d(x0,y0,s,e1,h,x1,y1) /*initial point, directing, stopping value, optimal point */
  double x0[3],y0[1],s[3],e1,h,x1[3],y1[1];
  {int FFlag,i,j,k;
   double x2[3],y2[1];
   FFlag=0; /* 0 not reverse but half step size; 1 reverse but keep step size*/
   for(i=0;i<3;i++)x1[i]=x0[i]+h*s[i]; /*the modular of judging the initial direction*/
   y1[0]=func(x1[0],x1[1],x1[2]);
   if (y1[0]>y0[0])
      {h=-h;}
    else
      h=2*h;
   for(;;)   /*the modular of doubling step size*/
    {for(i=0;i<3;i++)x2[i]=x1[i]+h*s[i]; /*the detected point. */
     y2[0]=func(x2[0],x2[1],x2[2]);
     if(y2[0]<y1[0])
       {h=h+h;        for(i=0;i<3;i++)x1[i]=x2[i];        y1[0]=y2[0];   }
     else
       {h=0.5*h;        break;   }
    }
   FFlag=0; /*the modular of halving the step size*/
   for(;;)
    {for(i=0;i<3;i++)x2[i]=x1[i]+h*s[i];     y2[0]=func(x2[0],x2[1],x2[2]);
     if(fabs(h)<e1)
       {if(y2[0]<y1[0]){for(i=0;i<3;i++)x1[i]=x2[i];y1[0]=y2[0];return;}
        if(y2[0]>=y1[0])return;
       }
     if(y2[0]<y1[0])
       {for(i=0;i<3;i++)x1[i]=x2[i];y1[0]=y2[0];h=0.5*h; FFlag=1; continue; }
     else
       {if(FFlag==1)
          {FFlag=0;            h=-h;          }
          else
          {FFlag=1;            h=0.5*h;        }
       }
    } /* end loop */
  }
```

Example Analysis

$$\min f(\boldsymbol{x}) = (x_1 - x_2)^2 + 2(x_2 - x_3)^2 + 4(x_3 + x_1)^2 - 4x_1 - 2x_2 - x_3 \quad (1)$$

The contour of the objective function is shown in Fig. 1.

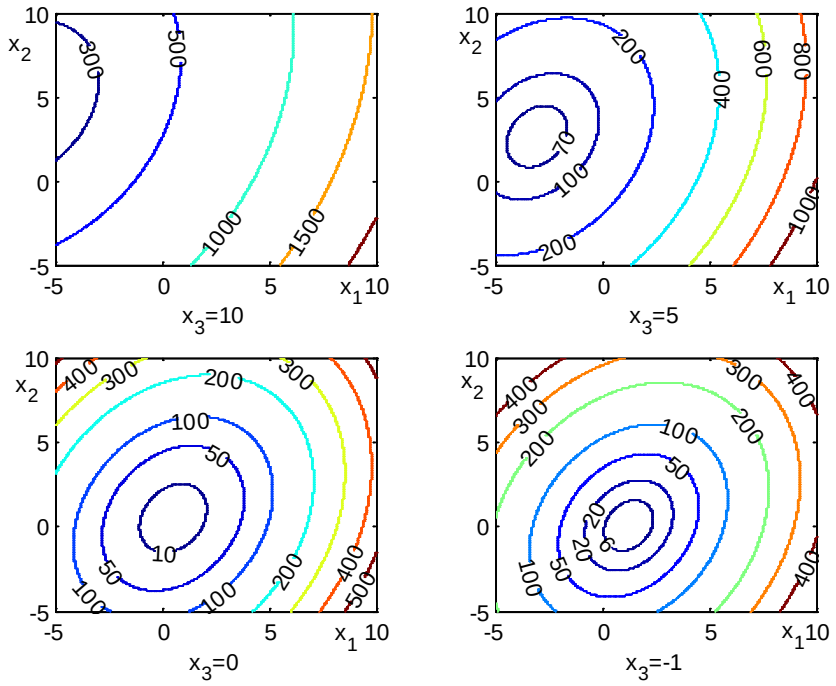


Fig. 1. The objective function contour of the example

The gradient of the objective function is following.

$$\nabla \mathrm{f}\left(\boldsymbol{x}\right)=\begin{bmatrix}10x_1-2x_2+8x_3-4\\-2x_1+6x_2-4x_3-2\\8x_1-4x_2+12x_3-1\end{bmatrix} \tag{2}$$

According to $\nabla f(\boldsymbol{x})=\boldsymbol{0}$, the extreme point of the objective function is $[0.6563\ 0.4062\ -0.2188]^{\mathrm{T}}$. At this point, the two order partial derivative matrix is [10 -2 8; -2 6 -4; 8 -4 12]. It is a positive definite matrix. So the point is the minimum point. The minimum value of the objective function is -1.6094.

When the starting point $\boldsymbol{x}^{(k)}$ and $\boldsymbol{s}^{(k)}$ are determined, the points in the search direction can be expressed as

$$\boldsymbol{x}=\boldsymbol{x}^{(k)}+\alpha\boldsymbol{s}^{(k)} \tag{3}$$

The objective function value is changed to

$$\mathrm{f}\left(\boldsymbol{x}\right)=\mathrm{Y}\left(\alpha\right)=a_2\alpha^2+a_1\alpha+a_0 \tag{4}$$

$$a_2=\left(s_1^{(k)}-s_2^{(k)}\right)^2+2\left(s_2^{(k)}-s_3^{(k)}\right)^2+4\left(s_3^{(k)}+s_1^{(k)}\right)^2 \tag{5}$$

$$\begin{aligned}a_1=&2\left(s_1^{(k)}-s_2^{(k)}\right)\left(x_1^{(k)}-x_2^{(k)}\right)+4\left(s_2^{(k)}-s_3^{(k)}\right)\left(x_2^{(k)}-x_3^{(k)}\right)+8\left(s_3^{(k)}+s_1^{(k)}\right)\left(x_3^{(k)}+x_1^{(k)}\right)\\&-4s_1^{(k)}-2s_2^{(k)}-s_3^{(k)}\end{aligned} \tag{6}$$

$$a_0=\left(x_1^{(k)}-x_2^{(k)}\right)^2+2\left(x_2^{(k)}-x_3^{(k)}\right)^2+4\left(x_3^{(k)}+x_1^{(k)}\right)^2-4x_1^{(k)}-2x_2^{(k)}-x_3^{(k)} \tag{7}$$

According to the extreme condition

$$\mathrm{Y}'\left(\alpha\right)=0 \tag{8}$$

The optimal step size is obtained.

$$\alpha^{(k)}=-a_1/(2a_2) \tag{9}$$

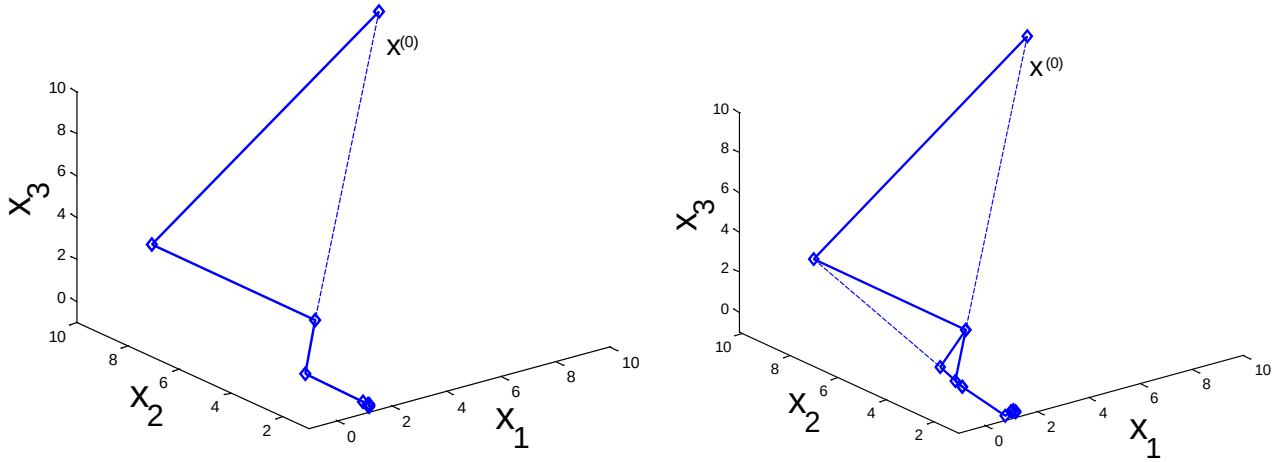
The optimal point on $\boldsymbol{s}^{(k)}$ is

$$\boldsymbol{x}^{(k+1)}=\begin{bmatrix}x_1^{(k)}+\alpha^{(k)}s_1^{(k)}\\x_2^{(k)}+\alpha^{(k)}s_2^{(k)}\\x_3^{(k)}+\alpha^{(k)}s_3^{(k)}\end{bmatrix} \tag{10}$$

Example Analysis

The initial point is $\boldsymbol{x}^{(k)}$ =[10 10 10]^{T}, the objective function is 1530, the gradient is $\boldsymbol{g}^{(0)}$=[156 -2 159]^{T}. The extreme value obtained by the Eq. 10 is [1.8256 1.6684]^{T}, and its objective function value is 230.5426. Using the blind walking optimization algorithm for a one-dimensional [2-5] optimization, one time of optimization obtains the optimal point of the [1.8295 10.105 1.6723]^{T}, the objective function value is 230.54. The blind walking algorithm is effective. The negative gradient direction method is gone on. the objective function value the sequence optimal points are: 3.8973e+01, 9.4055e+00, 2.2254e+00, 1.9403e-01, -7.5999e-01, -1.2087e+00, -1.4203e+00, -1.5202e+00. The closer to the extreme point, the worse the search results.

The three-seeking method is carried out. After six loops, the optimal point [0.65625 0.40625 -0.21874]^{T} is obtained. Its objective function value is -1.6094e+00. The optimization process is shown in Fig. 2(a). The optimal point sequence of each loop is shown in Table 1.



(a) the three-seeking method in three loops (b) the six-seeking method in two loops

Fig. 2. Optimization process

Table 1. Optimal point sequence of the three-seeking method

loop number		x_1	x_2	x_3	$f(x)$
0	0 [	1.0000e+01	1.0000e+01	1.0000e+01]	1.5300e+03
1	1 [	1.8295e+00	1.0105e+01	1.6723e+00]	2.3054e+02
	2 [	6.5353e-01	2.4980e+00	2.7304e+00]	3.8973e+01
	3 [	-9.6261e-01	1.2008e+00	1.4734e+00]	5.8478e+00
2	4 [	6.9393e-01	7.1779e-01	-1.5801e-01]	-1.3698e+00
	5 [	6.5619e-01	4.7349e-01	-1.2400e-01]	-1.5675e+00
	6 [	7.0177e-01	4.5301e-01	-1.6898e-01]	-1.5730e+00
3	7 [	6.6197e-01	4.5351e-01	-2.0953e-01]	-1.6039e+00
	8 [	6.5623e-01	4.1644e-01	-2.0438e-01]	-1.6084e+00
	9 [	6.4836e-01	4.1012e-01	-2.1050e-01]	-1.6092e+00
4	10 [	6.5642e-01	4.0778e-01	-2.1846e-01]	-1.6094e+00
	11 [	6.5628e-01	4.0664e-01	-2.1827e-01]	-1.6094e+00
	12 [	6.5650e-01	4.0654e-01	-2.1848e-01]	-1.6094e+00
5	13 [	6.5628e-01	4.0653e-01	-2.1870e-01]	-1.6094e+00
	14 [	6.5626e-01	4.0633e-01	-2.1866e-01]	-1.6094e+00
	15 [	6.5620e-01	4.0628e-01	-2.1870e-01]	-1.6094e+00
6	16 [	6.5623e-01	4.0627e-01	-2.1871e-01]	-1.6094e+00
	17 [	6.5623e-01	4.0627e-01	-2.1872e-01]	-1.6094e+00
	18 [	6.5625e-01	4.0625e-01	-2.1874e-01]	-1.6094e+00

The six-seeking method is carried out. After five loops, the optimal point [0.65625 0.40625 -0.21875]^{T} is obtained. Its objective function value is -1.6094e+00. It is as same as the three search method. The optimization process is shown in Fig. 2(b). The optimal point sequence of each loop is shown in Table 2.

Table 2 Optimal point sequence of the six-seeking method

loop number	[	x_1	x_2	x_3	]	$f(x)$
0	0 [	1.0000e+01	1.0000e+01	1.0000e+01	]	1.5300e+03
1	1 [	1.8295e+00	1.0105e+01	1.6723e+00	]	2.3054e+02
	2 [	6.5353e-01	2.4980e+00	2.7304e+00	]	3.8973e+01
	3 [	-3.8345e-01	2.4574e+00	1.2858e+00	]	9.4055e+00
	4 [	-9.6261e-01	1.2008e+00	1.4734e+00	]	5.8478e+00
	5 [	-7.8733e-01	1.0616e+00	1.2153e+00	]	4.0091e+00
	6 [	1.6730e-01	3.0368e-01	-1.9047e-01	]	-5.7696e-01
2	7 [	4.0758e-01	2.9024e-01	-2.0042e-02	]	-1.3837e+00
	8 [	5.0875e-01	4.4139e-01	-1.5075e-01	]	-1.5486e+00
	9 [	5.6495e-01	4.2826e-01	-1.2243e-01	]	-1.5854e+00
	10 [	5.9645e-01	4.7676e-01	-1.4055e-01	]	-1.5909e+00
	11 [	6.1019e-01	4.6795e-01	-1.5187e-01	]	-1.5960e+00
	12 [	6.6707e-01	4.3146e-01	-1.9873e-01	]	-1.6053e+00
3	13 [	6.5455e-01	4.2861e-01	-2.1172e-01	]	-1.6082e+00
	14 [	6.5525e-01	4.1464e-01	-2.0933e-01	]	-1.6090e+00
	15 [	6.5239e-01	4.1378e-01	-2.1354e-01	]	-1.6092e+00
	16 [	6.5083e-01	4.0836e-01	-2.1329e-01	]	-1.6093e+00
	17 [	6.5152e-01	4.0782e-01	-2.1427e-01	]	-1.6093e+00
	18 [	6.5462e-01	4.0540e-01	-2.1866e-01	]	-1.6094e+00
4	19 [	6.5546e-01	4.0553e-01	-2.1814e-01	]	-1.6094e+00
	20 [	6.5564e-01	4.0611e-01	-2.1857e-01	]	-1.6094e+00
	21 [	6.5591e-01	4.0613e-01	-2.1844e-01	]	-1.6094e+00
	22 [	6.5607e-01	4.0641e-01	-2.1854e-01	]	-1.6094e+00
	23 [	6.5611e-01	4.0639e-01	-2.1857e-01	]	-1.6094e+00
	24 [	6.5628e-01	4.0629e-01	-2.1873e-01	]	-1.6094e+00
5	25 [	6.5626e-01	4.0628e-01	-2.1875e-01	]	-1.6094e+00
	26 [	6.5626e-01	4.0627e-01	-2.1874e-01	]	-1.6094e+00
	27 [	6.5625e-01	4.0626e-01	-2.1875e-01	]	-1.6094e+00
	28 [	6.5625e-01	4.0625e-01	-2.1875e-01	]	-1.6094e+00
	29 [	6.5625e-01	4.0625e-01	-2.1875e-01	]	-1.6094e+00
	30 [	6.5625e-01	4.0625e-01	-2.1875e-01	]	-1.6094e+00

Conclusion

One-dimensional blind walking method is effective. The three-seeking method and the sex-seeking method are effective.

References

[1] LI Chunming. Optimization method[M]. Nanjing: Southeast University Press, 2009.10.

[2] LI Chunming. Blind-walking optimization method[J]. Journal of Network, 2010, 5(12): 1458-1466.

[3] ZHANG Chunling. The improvement of Newton method and the validation of optimization idea of blind walking repeatedly[A]. Advanced materials research, 2011, vol. 250-253, part 4, pp4061-4064. 1st International Conference on Civil Engineering, Architecture and Building Materials.

[4] LI Chunming. Negative gradient direction method for blind person exploring the way[J]. Journal of Gansu Sciences, 2016, 28(5):116-122.

[5] LI Chunming. Improvements of classical random direction optimization methods[J]. Computer Simulation, , 2009, 26(1):189-192.