文章编号: 2096-1618(2017)06-0667-08

外罚函数法与广义 Lagrange 乘子法的比较研究

宋 菲, 吴泽忠
(成都信息工程大学应用数学学院,四川 成都 610225)

摘要:基于非线性约束优化问题,讨论了外罚函数法与广义 Lagrange 乘子法,并通过 MATLAB 编程实现了两种算法。实验表明:(1)广义 Lagrange 乘子法在迭代次数和收敛结果上优于外罚函数法且对初始点的选取要求不高;(2)广义 Lagrange 乘子法的罚因子的修正系数不宜过大,一般在区间(1,2)上取值,广义 Lagrange 乘子法更具优越性。最后,通过 3 个工业工程中的非线性规划实际问题说明乘子法比外罚函数法具有更广泛的实用性。

关 键 词:约束优化;外罚函数法;广义 Lagrange 乘子法;罚因子;修正系数
中图分类号:O221.2 **文献标志码**:A
doi:10.16836/j.cnki.jcuit.2017.06.016

0 引言

罚函数法是解约束优化问题的一个经典方法,它的基本思想是:根据约束的特点把约束条件转换成某种惩罚函数加到目标函数中去,从而将约束优化问题的求解转化为一系列无约束优化问题[1]的求解。罚函数法主要包括外罚函数法、内点法和广义 Lagrange 乘子法。

外罚函数法首先是由 Courant[2]提出来的,它的基本思想是:构造惩罚函数,当初始点在非可行区域中时,给惩罚函数乘以适当的惩罚因子并加到目标函数中去,使目标函数接受惩罚;当初始点是可行点时,目标函数不接受惩罚。由于在外罚函数中,罚因子的选取对收敛性影响较大,当罚因子趋于无穷时,增广目标函数将呈现病态性质。于是,1969 年,Powell 和 Hestenes 提出等式约束问题的乘子法,1973 年,Rockfellar 再推广到不等式约束的情况。因此,广义 Lagrange 乘子法[3]又称为 PHR 算法,它的基本思想是从原问题的 Lagrange 函数出发,再加上适当罚函数,从而解决一系列无约束优化子问题。

在工业生产与工程应用中,外罚函数法与乘子法都得到了很好的应用,但是究竟哪一个的收敛速度与收敛效果更好,是本文讨论的关键问题。从两种算法的定义来看,外罚函数法的收敛速度和收敛效果会受到罚因子和罚因子增长的常倍数的影响较大,而乘子法的收敛速度和收敛效果会受到乘子和罚因子以及它们的修正系数的影响。由于外罚函数法的罚因子在迭代过程中会趋于无穷大,给计算带来一定的困难甚至是失败,在理论上都知道乘子法可以克服外罚的这个缺点,使收敛的速度更快。若适当改变参数的值,那么外罚函数法和乘子法的收敛速度又会发生怎样的变化呢?

收稿日期:2017-06-27
基金项目:国家自然科学基金资助项目(71672013);四川省软件科学研究计划资助项目(2014ZR0016);四川省哲学社会科学重点研究基地-系统科学与企业发展研究中心(重点)资助项目(Xq141306)

1 预备知识

文中所讨论的两种算法,在求解带约束优化的非线性规划问题中,其基本思想都是:利用问题的目标函数和约束函数构造出带有参数的增广目标函数,把约束非线性规划问题转化为一系列无约束非线性规划问题,进而再用无约束优化的方法来求解约束问题。增广目标函数由两部分构成,一部分是原问题的目标函数,一部分是由约束函数构造出的“惩罚”项,其作用是对“违规”的点进行“惩罚”。

1.1 外罚函数法

对于一般约束优化问题:

$$\begin{cases}\min f(x), x \in R^n \\ \text{s.t.} \quad h_i(x)=0, \quad i \in E=\{1,\cdots,l\}, \\ \qquad h_i(x) \geq 0, \ i \in I=\{l+1,\cdots,m\}.\end{cases} \tag{1}$$

记可行区域为 $D=\{x \in R^n \mid c_i(x)=0(i \in E), c_i(x) \geq 0(i \in I)\}$,构造罚函数:

$$\tilde{P}(x)=\sum_{i=1}^{l} c_i^2(x)+\sum_{i=l+1}^{m}[\min(0,c_i(x))]^2 \tag{2}$$

和增广目标函数:

$$P(x,\sigma)=f(x)+\sigma\tilde{P}(x) \tag{3}$$

其中,σ 是罚因子,当 $x \in D$ 为可行点时,$P(x,\sigma)=f(x)$,目标函数不受额外惩罚;当 $x \notin D$ 不是可行点

时，$P(x,\sigma) > f(x)$，目标函数受额外惩罚。σ 越大，惩罚越重。

当 σ 充分大时，要使 $P(x,\sigma)$ 极小，$\tilde{P}(x)$ 应该充分小，从而 $P(x,\sigma)$ 的极小点充分逼近可行域 D，$P(x,\sigma)$ 的极小值逼近 $f(x)$ 的极小值，于是，一般约束优化问题就转化为了一系列无约束优化问题：

$$\min P(x,\sigma_k) \tag{4}$$

其中，$\{\sigma_k\}$ 是正的数列，且 $\sigma_k \to +\infty$。$x(\sigma)$ 通常是从可行域外部趋近于极小值 x^*，因此称这种解法为外罚函数法（或罚函数法）。

算法步骤：

步骤 1　选初值。

取 $x_0 \in R^n$，$\varepsilon > 0$，$\sigma_1 > 0$，$\beta > 1$，$k = 1$。

步骤 2　解无约束优化子问题。

以 x_{k-1} 为初始点，

$$\min P(x,\sigma_k)$$

令其极小点为 x_k。

步骤 3　检验终止条件。

若 $\sigma_k \tilde{P}(x_k) < \varepsilon$，则算法终止；否则，令 $\sigma_{k+1} = \beta\sigma_k$，$k = k+1$，转步骤 2.

1.2　广义 Lagrange 乘子法

同样，考虑混合约束优化问题(1)。构造增广拉格朗日函数为

$$M(x,\lambda,\sigma) = f(x) - \sum_{i=1}^{l}\lambda_i c_i(x) + \frac{\sigma}{2}\sum_{i=l+1}^{m} \{[\max\{0,\lambda_i - \sigma c_i(x)\}]^2 - \lambda_i^2\} \tag{5}$$

乘子的修正公式为

$$(\lambda_{k+1})_i = (\lambda_k)_i - \sigma c_i(x_k) \quad i = 1,\cdots,l, \tag{6}$$

$$(\lambda_{k+1})_i = \max\{0,(\lambda_k)_i - \sigma c_i(x_k)\} \quad i = l+1,\cdots,m \tag{7}$$

令

$$\varphi_k = \left\{\sum_{i=1}^{l} c_i^2(x_k) + \sum_{i=l+1}^{m}\left[\min\left\{c_i(x_k),\frac{(\lambda_k)_i}{\sigma}\right\}\right]^2\right\}^{1/2} \tag{8}$$

终止准则为 $\varphi_k \le \varepsilon$。

算法步骤：

步骤 1　选初值。

取 $x_0 \in R^n$，$\lambda_1 \in R^m$，$\sigma_1 > 0$，$\beta > 1$，$\varepsilon > 0$，$\theta \in (0,1)$，$k = 1$。

步骤 2　求解无约束子问题。

以 x_{k-1} 为初始点，求子问题

$$\min M(x,\lambda_k,\sigma_k)$$

得极小点为 x_k，其中 $M(x,\lambda_k,\sigma_k)$ 由式(5)、式(6)和式(7)式所定义。

步骤 3　检验终止条件。

若 $\varphi_k \le \varepsilon$，其中 φ_k 是由式(8)所定义，则 x_k 为问题(1)的解，算法终止。否则转步骤 4.

步骤 4　修正罚因子。

当 $\frac{\varphi_k}{\varphi_{k-1}} > \theta$ 时，令 $\sigma_{k+1} = \beta\sigma_k$。

步骤 5　修正乘子向量 λ。

计算

$$(\lambda_{k+1})_i = (\lambda_k)_i - \sigma\beta_i(x_k) \quad i = 1,\cdots,l,$$

$$(\lambda_{k+1})_i = \max\{0,(\lambda_k)_i - \sigma\beta_i(x_k)\} \quad i = l+1,\cdots,m。$$

令 $k = k+1$，转步骤 2。

说明：本文在两种算法求解无约束子问题时都采用的是 BFGS 算法[6]的程序。

2　算例

本次数值实验是基于 MATLAB7.8.0(R2009a)，为了准确比较两种算法的收敛速度，在每一个算例中，保持相同参数的取值一致。为了体现惩罚的思想，取不满足约束条件的点，即可行域外的点作为初值点。表 1 中 k 表示迭代的次数，σ_k 代表罚因子，λ_k 是乘子，$\sigma_k P(x_k)$ 是外罚函数法中检验迭代终止的条件，φ_k 是广义 Lagrange 乘子法中检验迭代终止的条件，β 为罚因子的修正系数。

2.1　对于二元非线性混合约束问题

$$\min\ (x_1 - 2)^2 + (x_2 - 1)^2$$

$$\text{s.t.}\quad 2 - x_1 - x_2 = 0,$$

$$-x_1^2 + x_2 \ge 0$$

(1)取可行域外任意一点 $(2,2)^T$ 作为初始点。$\sigma_0 = 0.8$，$\beta = 1.5$，取精度 1×10^{-4}，得到迭代过程如下：

表 1　外罚函数法

k	x_k	σ_k	$\sigma_k P(x_k)$
0	(2.00000,2.00000)	0.80000	0.13529
1	(1.18099,1.06577)	1.20000	0.10875
2	(1.13504,1.05410)	1.80000	0.08401
3	(1.09832,1.04226)	2.70000	0.06265
4	(1.07007,1.03163)	4.05000	0.04537
5	(1.04906,1.02290)	6.07500	0.03209
⋮	⋮	⋮	⋮
18	(1.00028,1.00014)	1182.31350	0.00019
19	(1.00019,1.00009)	1773.47026	0.00013
20	(1.00013,1.00006)	2660.20538	0.00008

表 2　广义 Lagrange 乘子法

k	x_k	λ_k	σ_k	φ_k
0	(2.00000,2.00000)	(0.10000,0.10000)	0.8	0.55960
1	(1.23810,1.06580)	(0.34384,0.47545)	0.8	0.22501
2	(1.10251,1.06167)	(0.47519,0.59853)	0.8	0.11029
3	(1.04947,1.04580)	(0.55140,0.64299)	0.8	0.06029
4	(1.02549,1.03125)	(0.59680,0.65930)	0.8	0.03494
5	(1.01373,1.02046)	(0.62415,0.66506)	0.8	0.02084
⋮	⋮	⋮	⋮	⋮
11	(1.00018,1.00041)	(0.66592,0.66674)	1.2	0.00030
12	(1.00010,1.00021)	(0.66628,0.66672)	1.2	0.00016
13	(1.00005,1.00011)	(0.66647,0.66669)	1.2	0.00009

(2)改变β的值,精度取1×10^{-7},其他初始条件和参数值不变得到的结果如表 3、表 4 所示。

表 3　外罚函数法

β	k	x^*	σ^*
2	22	(1.00000,1.00000)	3355443.2
4	11	(1.00000,1.00000)	3355443.2
6	9	(1.00000,1.00000)	8062156.8

表 4　广义 Lagrange 乘子法

β	k	x^*	σ^*
2	27	(1.00000,1.00000)	1638.4
4	30	(1.00000,1.00000)	219902325555.2
6	30	(NaN, NaN)	487487792008397

比较分析:在控制相同的参数情况下,当$\beta = 1.5$时,广义 Lagrange 乘子法迭代 13 次,而外罚函数法迭代 20 次,且外罚函数法中最终的罚因子约为 2660、20538,而广义 Lagrange 乘子法中的罚因子在最后的迭代中只有 1.2。当β增加时,外罚的迭代次数减少,但罚因子增加;而广义 Lagrange 乘子法对于较大的β来说,迭代次数增多,罚因子也变得非常大。

2.2　考虑二元非线性混合约束优化问题

$$\min(x_1 - 2)^2 + (x_2 - 1)^2$$

$$\text{s.t.}\quad x_1 - 2x_2 + 1 = 0$$

$$-0.25x_1^2 - x_2^2 + 1 \geq 0$$

(1)取初始点为$(2,2)^{\mathrm{T}}$。$\sigma_0 = 0.8$,$\beta = 1.5$,取精度为1×10^{-4},得到的迭代过程如下:

表 5　外罚函数法

k	x_k	σ_k	$\sigma_k P(x_k)$
0	(2.00000,2.00000)	0.8	0.31561
1	(1.39867,0.98214)	1.2	0.30823
2	(1.29592,0.97396)	1.8	0.28342
3	(1.19973,0.96468)	2.7	0.24613
4	(1.11421,0.95504)	4.05	0.20278
5	(1.04179,0.94580)	6.075	0.15942
⋮	⋮	⋮	⋮
23	(0.82308,0.91148)	8978.19317	0.00017
24	(0.82301,0.91146)	13467.28976	0.00011
25	(0.82297,0.91145)	20200.93464	0.00007

表 6　广义 Lagrange 乘子法

k	x_k	λ_k	σ_k	φ_k
0	(2.00000,2.00000)	(0.10000,0.10000)	0.8	0.85451
1	(1.57548,0.94953)	(-0.44114,0.51771)	0.8	0.55931
2	(1.34044,0.96728)	(-0.92820,0.97950)	1.2	0.27824
3	(1.09187,0.95724)	(-1.14108,1.23672)	1.2	0.18592
4	(1.00541,0.94509)	(-1.34851,1.49935)	1.8	0.08713
5	(0.90977,0.92930)	(-1.44063,1.62628)	1.8	0.05422
⋮	⋮	⋮	⋮	⋮
13	(0.82344,0.91157)	(-1.59348,1.84512)	2.7	0.00029
14	(0.82317,0.91150)	(-1.59394,1.84577)	2.7	0.00017
15	(0.82304,0.91148)	(-1.59419,1.84614)	2.7	0.00009

(2)若改变β的值,精度取1×10^{-8},其他初始条件和参数值不变得到的结果如表 7、表 8 所示:

表 7　外罚函数法

β	k	x^*	σ^*
1.8	33	(0.82288,0.91144)	212364859.866634
3	18	(0.82288,0.91144)	309936391.2
5	12	(0.82288,0.91144)	195312500

表 8　广义 Lagrange 乘子法

β	k	x^*	σ^*
1.8	39	(0.82288,0.91144)	9715.16249
3	35	(0.82288,0.91144)	309936391.2
5	23	(0.82288,0.91144)	7812500

比较分析:在本次实验中,当$\beta = 1.5$时,广义 Lagrange 乘子法迭代 15 次,而外罚函数法迭代 25 次,且外罚函数法中最终的罚因子约为 20200.93464,而广

义 Lagrange 乘子法中的罚因子在最后的迭代中只有 2.7。而当 β 增大时,虽然两种算法的迭代次数都在逐渐减少,但是罚因子增大,总的来说,外罚函数法似乎比广义 Lagrange 乘子法要好一点。

2.3 考虑约三元束优化问题

$$\min \frac{1}{2}({x_1}^2 + 2{x_2}^2 - 2x_1x_2 + {x_3}^2)$$

$$\text{s.t.} \quad x_1 + x_2 - x_3 = 4$$

$$x_1 - 2x_2 + x_3 = -2$$

(1)取初始点$(2,2,2)^{\mathrm{T}}$。$\sigma_0 = 0.8, \beta = 1.5, \theta = 0.6$ 精度取 1×10^{-5}。得到如下迭代过程:

表 9 外罚函数法

k	x_k	σ_k	$\sigma_k P(x_k)$
0	(2.00000,2.00000,2.00000)	0.8	0.10750
1	(1.57377,1.31148, -0.78689)	1.2	0.07575
2	(1.61798,1.34831, -0.80899)	1.8	0.05244
3	(1.64885,1.37405, -0.82443)	2.7	0.03587
4	(1.67010,1.39175, -0.83505)	4.05	0.02433
5	(1.68459,1.40383, -0.84226)	6.075	0.01641
⋮	⋮	⋮	⋮
22	(1.71426,1.42855, -0.85713)	5985.46211	0.00002
23	(1.71426,1.42855, -0.85713)	8978.19317	0.00001
24	(1.71427,1.42856, -0.85714)	13467.28976	0.000008

表 10 广义 Lagrange 乘子法

k	x_k	λ_k	σ_k	φ_k
0	(2.00000,2.00000,2.00000)	(0.10000,0.10000, 0.1)	0.8	0.63856
1	(1.54196,1.22727, -0.71329)	(0.51399,-0.19930,0.1)	0.8	0.10340
2	(1.70431,1.39807, -0.82997)	(0.56810,-0.26186,0.1)	0.8	0.02103
3	(1.71898,1.42395, -0.85096)	(0.57299,-0.27796,0.1)	0.8	0.00628
4	(1.71739,1.42787, -0.85541)	(0.57246,-0.28296,0.1)	0.8	0.00228
5	(1.71567,1.42847, -0.85657)	(0.57189,-0.28469,0.1)	0.8	0.00085
⋮	⋮	⋮	⋮	⋮
11	(1.71439,1.42860,-0.85711)	(0.57140,-0.28571,0.1)	1.8	0.00004
12	(1.71430,1.42858,-0.85710)	(0.57143,-0.28577,0.1)	1.8	0.00002
13	(1.71429,1.42860,-0.85712)	(0.57142,-0.28573,0.1)	1.8	0.000009

(2)若改变 β 的值,精度取 1×10^{-8},其他初始条件和参数值不变得到的结果如表 11、表 12 所示。

表 11 外罚函数法

β	k	x^*	σ^*
2	24	(1.71429,1.42857, -0.85714)	13421772.8
4	12	(1.71429,1.42857, -0.85714)	13421772.8
6	10	(1.71428,1.42857, -0.85715)	48372940.8

表 12 广义 Lagrange 乘子法

β	k	x^*	σ^*
2	27	(1.71429,1.42858, -0.85713)	6553.6
4	39	(1.71429,1.42857, -0.85714)	3.7×10^{18}
6	∞	(1.71428,1.42857, -0.85715)	∞

比较分析:在本次实验中,当 $\beta = 1.5$ 时,广义 Lagrange 乘子法迭代 13 次,而外罚函数法迭代 24 次,且外罚函数法中最终的罚因子约为13467.28976,而广义 Lagrange 乘子法中的罚因子在最后的迭代中只有1.8。当 β 增大时,广义 Lagrange 乘子法呈现出明显的劣势。

2.4 考虑三元非线性混合约束优化问题

$$\min 1000 - {x_1}^2 - 2{x_2}^2 - {x_3}^2 - x_1x_2 - x_1x_3$$

$$\text{s.t.} \quad 8x_1 + 14x_2 + 7x_3 - 56 = 0$$

$${x_1}^2 + {x_2}^2 + {x_3}^2 - 25 = 0$$

$$x_i \geq 0 \quad i = 1,2,3。$$

(1)初始点取为$(2,2,2)^{\mathrm{T}}$。$\sigma_0 = 0.8, \beta = 1.5$,取精度 1×10^{-4},得到以下迭代过程:

表 13 外罚函数法

k	x_k	σ_k	$\sigma_k P(x_k)$
0	(2.00000,2.00000, 2.00000)	0.8	0.49364
1	(3.55228,0.17131, 3.62199)	1.2	0.32861
2	(3.53895,0.18645, 3.59886)	1.8	0.21885
3	(3.53004,0.19659, 3.58336)	2.7	0.14580
4	(3.52408,0.20337, 3.57299)	4.05	0.09715
5	(3.52010,0.20790, 3.56606)	6.075	0.06475
⋮	⋮	⋮	⋮
31	(3.51215,0.21696,3.55222)	1773.47026	0.00022
32	(3.51214,0.21697,3.55221)	2660.20538	0.00015
33	(3.51213,0.21697,3.55220)	3990.30808	0.000099

表 14 广义 Lagrange 乘子法

k	x_k	λ_k	σ_k	φ_k
0	(2.00000,2.00000,2.00000)	(0.1,0.1,0.1,0.1,0.1)	0.8	1.73971
1	(3.60019,0.12619,3.69938)	(-0.27103,-1.23017,0,0,0)	0.8	0.00984
2	(3.51181,0.21800,3.55123)	(-0.27503,-1.22339,0,0,0)	0.8	0.00014
3	(3.51212,0.21697,3.55218)	(-0.27494,-1.22346,0,0,0)	0.8	0.000005

(2)若改变 β 的值,精度取 1×10^{-10},其他初始条件和参数值不变得到的结果如表 15、表 16 所示。

表 15 外罚函数法

β	k	x^*	σ^*
3	21	(3.51212,0.21699,3.55217)	8368282562.4
5	14	(3.51212,0.21699,3.55217)	4882812500
7	12	(3.51212,0.21699,3.55217)	11073029760.8

表 16　广义 Lagrange 乘子法

β	k	x^*	σ^*
3	32	(3.51212,0.21699,3.55217)	18301433963968.8
5	∞	(3.51212,0.21699,3.55217)	∞
7	∞	(3.51212,0.21699,3.55217)	∞

比较分析:在本次实验中,当$\beta = 1.5$时,广义 Lagrange 乘子法只迭代 3 次,而外罚函数法迭代 33 次,且外罚函数法中最终的罚因子约为3990.30808,而广义 Lagrange 乘子法中的罚因子在最后的迭代中只有 0.8。当β增加时,对于外罚函数来说,迭代次数收敛明显变快,而对广义 Lagrange 乘子法来说已没有优势。

2.5　考虑混合约束

$$\begin{aligned}\min\ & -3x_1^2 - x_2^2 - 2x_3^2\\ \text{s.t.}\ & x_1^2 + x_2^2 + x_3^2 = 3\\ & x_2 \geq x_1\\ & x_i \geq 0 \quad i = 1,2,3。\end{aligned}$$

(1)初始点取$(2,0,1)^{\mathrm{T}}$, $\sigma_0 = 0.8$,$\beta = 1.5$, 精度取1×10^{-4}。得到以下迭代过程:

表 17　外罚函数法

k	x_k	σ_k	$\sigma_k P(x_k)$
0	(2.00000, 0.00000, 1.00000)	0.8	2.36459
1	(2.07439, 0.45421, 0.45808)	1.2	1.62083
2	(1.93856, 0.47085, 0.42717)	1.8	0.80764
3	(1.80457, 0.50186, 0.40186)	2.7	0.94256
4	(1.77999, 0.51356, 0.39841)	4.05	0.64627
5	(1.72024, 0.53647, 0.39042)	6.075	0.38961
⋮	⋮	⋮	⋮
73	(1.56340, 0.64161, 0.37964)	1.3×10^{13}	1445.46960
⋮	⋮	⋮	⋮
99	(1.56340, 0.64161, 0.37964)	3.3×10^{17}	3.6×10^{7}
⋮	⋮	⋮	⋮

表 18　广义 Lagrange 乘子法

k	x_k	λ_k	σ_k	φ_k
0	(2.00000,0.00000,1.00000)	(0.1,0.1,0.1,0.1,0.1)	0.8	3.98421
1	(2.48845,0.50712,0.04711)	(-2.66142,1.68507,0.00000,0,0.06231)	0.8	2.50282
2	(-1.73320,0.07212,0.02914)	(-2.67346,0.00000,2.07984,0,0.02735)	1.2	2.04306
3	(1.65201,0.45186,0.01146)	(-2.55367,2.16028,0.00000,0,0.00673)	1.8	0.30005
4	(1.20967,1.11842,0.00358)	(-2.03921,2.32453,0.00000,0,0.00028)	1.8	0.06674
5	(1.25613,1.18973,0.00014)	(-2.02715,2.44405,0.00000,0,0.00002)	1.8	0.01518
6	(1.22143,1.22186,0.00014)	(-1.99983,2.44328,0.00000,0,0)	1.8	0.00376
7	(1.22676,1.22311,0.00014)	(-2.00149,2.44985,0.00000,0,0)	1.8	0.00101
8	(1.22435,1.22476,0.00014)	(-1.99982,2.44913,0.00000,0,0)	1.8	0.00028
9	(1.22489,1.22466,0.00014)	(-2.00010,2.44955,0.00000,0,0)	1.8	0.00008

(2)若改变β的值,精度取1×10^{-9},其他初始条件和参数值不变得到的结果如表 19、表 20 所示。

表 19　外罚函数法

β	k	x^*	σ^*
2	∞	(1.58274,0.59182,0.38042)	∞
3.5	∞	(1.44149,0.87818, 0.38851)	∞
5	∞	(1.46457,0.83500, 0.39732)	∞

表 20　广义 Lagrange 乘子法

β	k	x^*	σ^*
2	26	(1.22474,1.22474,0.00007)	26214.4
3.5	23	(1.22474,1.22474,0.00006)	405675421.9922
5	14	(1.22474,1.22474,0.00003)	39062500

比较分析:在本次实验中,当$\beta = 1.5$时,广义 Lagrange 乘子法只迭代 9 次便达到收敛的效果,而外罚函数法却出现了无限循环迭代的现象且不收敛,且外罚函数法中最终的罚因子大到无法估计,而广义 Lagrange 乘子法中的罚因子在最后的迭代中只有1.8。当增大精度和β的值时,广义 Lagrange 乘子法的迭代次数有明显减少,但是罚因子却增大许多。外罚函数法失败的原因,可能是由于初始点的选取不恰当以及罚因子的无限增大,造成了增广目标函数$P(x,\sigma)$的海瑟矩阵条件数变大,给计算带来了困难。

2.6　考虑四元约束优化问题

$$\begin{aligned}\min\ & (x_1-1)^2+(x_2-2)^2+(x_3-3)^2+(x_4-4)^2\\ \text{s.t.}\ & 3x_1+3x_2+2x_3+x_4 \leq 10\\ & x_1+x_2+x_3+x_4 \leq 5\\ & x_1,x_2,x_3,x_4 \geq 0\end{aligned}$$

(1)初始点取$(\frac{1}{2},1,\frac{3}{2},2)^{\mathrm{T}}$, $\sigma_0 = 0.8$,$\beta = 1.5$,精度取1×10^{-4},迭代过程如下:

表 21　外罚函数法

k	x_k	σ_k	$\sigma_k P(x_k)$
0	(0.5,1,1.5,2)	0.8	0
1	(0.5,1,1.5,2)	0.8	0

表 22　广义 Lagrange 乘子法

k	x_k	λ_k	σ_k	φ_k
0	(0.50000, 1.00000, 1.50000, 2.00000)	(0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1)	0.8	1.35241
1	(-0.00598, 0.94163, 2.10373, 3.26582)	(0.1, 0.3, 1.14416, 0.10478, 0, 0, 0)	0.8	0.95280
2	(-0.01398, 0.92804, 1.97371, 3.01939)	(0.1, 0, 2.23275, 0.12156, 0, 0, 0)	1.2	0.22186
3	(-0.10752, 0.76720, 1.76720, 2.76719)	(0.1, 0, 2.46563, 0.25059, 0, 0, 0)	1.2	0.12021
4	(-0.09488, 0.72290, 1.72290, 2.72290)	(0.1, 0, 2.55421, 0.36444, 0, 0, 0)	1.2	0.08002
5	(-0.07222, 0.70223, 1.70223, 2.70222)	(0.1, 0, 2.61622, 0.49444, 0, 0, 0)	1.8	0.03845
⋮	⋮	⋮	⋮	⋮
12	(-0.00057, 0.66691, 1.66691, 2.66691)	(0.1, 0, 2.66615, 0.66506, 0, 0, 0)	2.7	0.00030
13	(-0.00029, 0.66679, 1.66679, 2.66679)	(0.1, 0, 2.66641, 0.66584, 0, 0, 0)	2.7	0.00016
14	(-0.00015, 0.66673, 1.66673, 2.66673)	(0.1, 0, 2.66654, 0.66624, 0, 0, 0)	2.7	0.00008

(2)改变β的值,精度取1×10^{-8},其他初始条件和参数值不变得到的结果如表23、表24所示。

表23 外罚函数法

β	k	x^*	σ^*
1.8	0	(0.5,1,1.5,2)	0.8
3	0	(0.5,1,1.5,2)	0.8
10	0	(0.5,1,1.5,2)	0.8

表24 广义 Lagrange 乘子法

β	k	x^*	σ^*
1.8	34	$(-2.1\times10^{-8},0.66667,1.66667,2.66667)$	9715.1625
3	30	$(-5.8\times10^{-8},0.66667,1.66667,2.66667)$	11479125.6
10	13	$(-8.3\times10^{-8},0.66667,1.66667,2.66667)$	8000000

比较分析:在本次数值实验中,外罚函数法却出现了不迭代的现象,从外罚函数的定义中,不难知道外罚函数法失败的原因是因为初始点是可行域中的点,于是增广目标函数$P(x,\sigma)=f(x)+\sigma\tilde{P}(x)=f(x)$,即不接受惩罚,所以$\sigma\tilde{P}(x)=0$。而广义 Lagrange 乘子法的迭代次数随着$\beta$的增加呈现先增加后减少的趋势。在$\beta=10$时,虽然迭代次数比$\beta=1.5$时少,但是罚因子却明显增大很多。

3 罚函数法方法在工业工程中的应用分析

由于外罚函数法与广义 Lagrange 乘子法的最优值受到初始点和罚因子的影响较大,得到的结果是可行域中的点,但有可能不是最优的,因此,为了体现广义 Lagrange 乘子法具有更好的实用性,引用以下几个生产实例,将外罚函数法与广义 Lagrange 乘子法的收敛结果与其他计算方法的结果进行比较。

3.1 集中式网络能耗最优化问题的数学模型[5]

$$\min\ x_1(0.0021x_2^2+0.2765x_2+223.5)$$

$$\text{s.t.}\quad x_1x_2^{\,2}\geq31132$$

$$0\leq x_1\leq20$$

$$0\leq x_2\leq60$$

(1)初始点取$(30,70)^{\mathrm{T}}$,$\sigma_0=0.8$,$\beta=1.5$,精度取1×10^{-4},迭代过程如下:

表25 外罚函数法

k	x_k	σ_k	$\sigma_kP(x_k)$
0	(30, 70)	0.8	0
1	(30, 70)	0.8	0

表26 广义 Lagrange 乘子法

k	x_k	λ_k	σ_k	φ_k
0	(30.00000,70.00000)	(0.1,0.1,0.1,0.1,0.1,0.1)	0.8	25.37450
1	(4.27179,85.36791)	(0.1,0.52916,0,0,0,20.39433)	0.8	16.94112
2	(5.26073,76.92820)	(0.1,0,0,0,0,40.70817)	1.2	6.72475
3	(6.99253,66.72459)	(0.1, 0.05644,0,0,0,48.77767)	1.2	4.43257
4	(7.49887,64.43257)	(0.1,0.06212,0,0,0,56.75630)	1.8	2.05822
5	(8.08367,62.05822)	(0.1,0.06459,0,0,0,60.46109)	1.8	1.27994
⋮	⋮	⋮	⋮	⋮
18	(8.64759,60.00066)	(0.1,0.06879,0,0,0,66.81486)	2.7	0.00036
19	(8.64768,60.00036)	(0.1,0.06879,0,0,0,66.81585)	2.7	0.00036
20	(8.64767,60.00036)	(0.1,0.06879,0,0,0,66.81733)	4.05	0.00004

(2)若改变β的值,精度取1×10^{-8},其他初始条件和参数值不变得到的结果如表27、表28所示。

表27 外罚函数法

β	k	x^*	σ^*
2.5	0	(30, 70)	0.8
4	0	(30, 70)	0.8
6	0	(30, 70)	0.8

表28 广义 Lagrange 乘子法

β	k	x^*	σ^*
2.5	32	(8.6478,60.0000)	11641532.1826
4	∞	(8.6478,60.0000)	∞
6	22	(8.6478,60.0000)	10448555212.8

在本次数值试验中,可以看出外罚函数法是失败的。而广义 Lagrange 乘子法由文献[5]用 SQP 算法所给出的计算结果$x^*=(8.6478,60.0000)^{\mathrm{T}}$基本一致,且精度越高,越接近最优值。但是,随着$\beta$的增加,广义 Lagrange 乘子法的迭代次数增加,且会出现无限重复迭代的现象。

3.2 考虑混合约束优化问题,此优化模型取自 Liebman[4]

$$\min\ x_3$$

$$\text{s.t.}\quad x_3=250+30x_1-6x_1^{\,2}$$

$$x_3=300+20x_2-12x_2^{\,2}$$

$$x_3=150+0.5(x_1+x_2)2$$

$$0\leq x_1\leq9.422$$

$$0\leq x_2\leq5.903$$

$$0\leq x_3\leq267.42$$

(1)初始点取$(10,7,280)^{\mathrm{T}}$,$\sigma_0=0.8$,$\beta=1.5$,精度取1×10^{-4},迭代过程如下:

表 29　外罚函数法

k	x_k	σ_k	$\sigma_k P(x_k)$
0	(10.0000, 7.0000, 280.0000)	0.8	0.1798
1	(6.2991, 3.8259, 200.7997)	1.2	0.1199
2	(6.2972, 3.8246, 200.9196)	1.8	0.0799
3	(6.2959, 3.8237, 200.9995)	2.7	0.0533
4	(6.2951, 3.8231, 201.0528)	4.05	0.0355
5	(6.2945, 3.8227, 201.0883)	6.075	0.0237
⋮	⋮	⋮	⋮
17	(6.2934, 3.8218, 201.1588)	788.290	0.0002
18	(6.2934, 3.8218, 201.1590)	1182.3135	0.0001
19	(6.2934, 3.8218, 201.1591)	1773.4703	0.00008

表 30　广义 Lagrange 乘子法

k	x_k	λ_k	σ_k	φ_k
0	(10.0000,7.0000,280.0000)	(0.1,0.1,0.1,0.1,0.1,0.1,0.1)	0.8	1.1630
1	(6.3047,3.8301,200.3149)	(-0.1629,-0.1034,-0.7337,0,0,0,0,0,0)	0.8	0.0003
2	(6.2934,3.8218,201.1595)	(-0.1630,-0.1035,-0.7335,0,0,0,0,0,0)	0.8	0.000001

(2)若改变 β 的值,精度取 1×10^{-8},其他初始条件和参数值不变得到的结果如表 31、表 32 所示。

表 31　外罚函数法

β	k	x^*	σ^*
2	25	(6.2934,3.8218,201.1593)	26843545.6
5	11	(6.2934,3.8218,201.1593)	39062500
7	9	(6.2934,3.8218,201.1593)	32282885.6

表 32　广义 Lagrange 乘子法

β	k	x^*	σ^*
2	37	(6.2934,3.8218,201.1593)	13421772.8
5	∞	(6.2934,3.8218,201.1593)	∞
7	∞	(6.2934,3.8218,201.1593)	∞

文献[5]用 SQP 算法求得此问题的最优解为 $x^* = (6.2934, 3.8218, 201.1593)^{\mathrm{T}}$。在此问题中,精度对最终的收敛结果影响不大,但是 β 的增加外罚函数法的收敛速度明显减少,但是广义 Lagrange 乘子法却出现了无限重复迭代的现象。

3.3　可重构计算系统的可靠优化问题的数学模型[7]

$$\begin{aligned}
\min\ & x_1^2 + 2x_1x_2 + 2x_1x_3 + 2x_1x_4 + 2x_1x_5 + 2x_2x_3 + 2x_2x_4 + \\
& 2x_2x_5 + 2x_3x_4 + 2x_3x_5 + 2x_4x_5 \\
\text{s.t.}\ & 5x_1 + 3x_2 + 3x_3 + 3x_4 + 3x_5 = 11/3 \\
& 3x_1 + 4x_2 + 3x_3 + 3x_4 + 3x_5 = 19/6 \\
& 3x_1 + 3x_2 + 4x_3 + 3x_4 + 3x_5 = 19/6 \\
& 3x_1 + 3x_2 + 3x_3 + 4x_4 + 3x_5 = 19/6 \\
& 3x_1 + 3x_2 + 3x_3 + 3x_4 + 4x_5 = 19/6 \\
& x_i \geq 0 \quad i = 1,2,3,4,5
\end{aligned}$$

(1)初始点取 $(1,1,1,1,1)^{\mathrm{T}}$,$\sigma_0=0.8$,$\beta=1.5$,$\theta=0.6$ 精度取 1×10^{-4},迭代过程如下:

表 33　外罚函数法

k	x_k	σ_k	$\sigma_k P(x_k)$
0	(1.0000,1.0000,1.0000,1.0000,1.0000)	0.8	0.0211
1	(0.2942,0.1720,0.1720,0.1720,0.1720)	1.2	0.0141
2	(0.3077,0.1701,0.1701,0.1701,0.1701)	1.8	0.0095
3	(0.3164,0.1689,0.1689,0.1689,0.1689)	2.7	0.0063
4	(0.3222,0.1681,0.1681,0.1681,0.1681)	4.05	0.0042
5	(0.3259,0.1676,0.1676,0.1676,0.1676)	6.075	0.0028
⋮	⋮	⋮	⋮
12	(0.3329,0.1667,0.1667,0.1667,0.1667)	103.7971	0.0002
13	(0.3321,0.1667,0.1667,0.1667,0.1667)	155.6956	0.0001
14	(0.3331,0.1667,0.1667,0.1667,0.1667)	233.5434	0.00007

(2)若改变 β 的值,精度取 1×10^{-8},其他初始条件和参数值不变得到的结果如表 35、表 36 所示。

表 34　广义 Lagrange 乘子法

k	x_k	λ_k	σ_k	φ_k
0	(1.0000,1.0000,1.0000,1.0000,1.0000)	(0.1000,0.1000,0.1000,0.1,0.1,0.1,0.1,0.1,0.1,0.1)	0.8	0.3272
1	(0.2494,0.1886,0.1886,0.1886,0.1886)	(0.2253,0.0734,0.0734,0.0734,0.0734,0,0,0,0,0)	0.8	0.0350
2	(0.3503,0.1620,0.1620,0.1620,0.1620)	(0.2029,0.0818,0.0818,0.0818,0.0818,0,0,0,0,0)	0.8	0.0074
3	(0.3298,0.1677,0.1677,0.1677,0.1677)	(0.2077,0.0802,0.0802,0.0802,0.0802,0,0,0,0,0)	0.8	0.0015
4	(0.3341,0.1665,0.1665,0.1665,0.1665)	(0.2067,0.0805,0.0805,0.0805,0.0805,0,0,0,0,0)	0.8	0.0003
5	(0.3332,0.1667,0.1667,0.1667,0.1667)	(0.2069,0.0804,0.0804,0.0804,0.0804,0,0,0,0,0)	0.8	0.00007

表 35　外罚函数法

β	k	x^*	σ^*
2	22	(0.3333,0.6667,0.6667,0.6667,0.6667)	3355443.2
4	11	(0.3333,0.6667,0.6667,0.6667,0.6667)	3355443.2
8	8	(0.3333,0.6667,0.6667,0.6667,0.6667)	13421772.8

表 36　广义 Lagrange 乘子法

β	k	x^*	σ^*
2	24	(0.3333,0.6667,0.6667,0.6667,0.6667)	3276.8
4	35	(0.3333,0.6667,0.6667,0.6667,0.6667)	5.7646×10^{16}
8	19	(NaN, NaN, NaN, NaN, NaN,)	54975581388.8

本次的计算结果与文献[5]用 SQP 算法的计算结果 $x^* = (0.3333, 0.1667, 0.1667, 0.1667, 0.1667)^T$ 相比较,不难发现,算法的精度越高越收敛效果越好,但是迭代次数会增多。随着罚因子的修正系数的增加,外罚函数法的迭代次数减少,而广义 Lagrange 乘子法的迭代次数增加,且广义 Lagrange 乘子法最终的罚因子比外罚函数法还要大。

4 讨论

从第二部分和第三部分的算例来看,无论是在理论上还是在实际应用中,广义 Lagrange 乘子法收敛速度与收敛结果都具有一定的优越性。9 次数值实验中,外罚函数法失败了 3 次,主要原因是:外罚函数法对初始点的选取要求很高,初始点的选取不恰当会造成实验失败;且外罚函数法中的罚因子 σ_k 增大的速度很快,甚至趋于无穷,造成 $P(x,\sigma_k)$ 的 Hesse 矩阵条件数变大,使增广目标函数呈现病态的性质,给数值计算带来很大困难甚至不可能。而广义 Lagrange 乘子法在考虑了 Lagrange 之后再加上适当的惩罚,其中的罚因子 σ_k, 可以在表中看到,它的增长速度远远低于外罚函数法中的罚因子的增长速度,便可以克服外罚函数中数据太大所带来计算困难的这一缺点;且广义 Lagrange 乘子法对初始点的选取要求不高,对于同样的初始点,广义 Lagrange 乘子法可以得到收敛结果,而外罚函数法不一定会得到收敛结果。同时也不难发现,随着罚因子的修正系数 β 的增加,广义 Lagrange 乘子法表现出明显的劣势,其罚因子的增长速度甚至会超过外罚函数法,从而易导致数值实验的失败。

5 结束语

通过大量的数值实验不难发现,在外罚函数基础上建立起来的广义 Lagrange 乘子法对于问题的求解效率远远高于外罚函数法。它不仅克服了外罚函数的病态性,也提高了收敛速度;且广义 Lagrange 乘子法对初始点的选取不如外罚函数法所要求那样高。但是,从罚因子的修正系数 β 的各种取值看,广义 Lagrange 乘子法的罚因子的修正系数不宜取太大,一般 $\beta \in (1,2)$ 比较合适。从后边的实例中,不难看出,在工业生产以及工程应用上,广义 Lagrange 乘子法是很好的选择。但是对于求解大规模的非线性约束优化问题,由于广义 Lagrange 乘子法需要对每一个变量求偏导,输入的工作量将会非常大,因此,还需要进一步改进。

参考文献:

[1] Fiacco A V, McCormick G P. Nonlinear Programming: Sequential Unconstrained Minimization Techniques[J]. Philadelphia: SIAM, 1990.

[2] Courant R. Variational methods for the solution of problems with equilibrium and vibration[J]. Bull. Amer. Math. Sco., 1943, 49: 1-23.

[3] Bertsekas D P. Constrained Optimization and Lagrange Multiplier Methods[M]. New York: Academic Press, 1982.

[4] Liebman J, Lasdon L S, Schrage L E, et al. Modeling and Optimaization with GINO[M]. Redwood City: Science Press, 1986.

[5] 刘兴高,胡云卿,李国栋,等. 最优化方法应用分析[M]. 北京:科学出版社,2014.

[6] 倪勤. 最优化方法与程序设计[M]. 北京:科学出版社. 2009.

[7] 周密,尚利宏,胡瑜. 给定余度下可重构计算系统的可靠性最优化研究[J]. 计算机科学,2009, 36(4): 83-298.

[8] 曾云宝,景旭,孙晓波. 罚函数乘子的初值选取和多步长问题[J]. 哈尔滨理工大学学报,2003, 8(1): 20-22.

[9] 许苗. 认知无线电系统频谱检测技术及其性能研究[D]. 上海:上海交通大学,2011.

A Comparative Study of External Penalty Function Method and Generalized Lagrangian Multiplier Method

SONG Fei, WU Ze-zhong

(College of Applied Mathematics, CUIT, Chengdu 610225, China)

Abstract: Based on the nonlinear constrained optimization problem, this paper discussed the external penalty function method and the generalized lagrangian multiplier method, and two algorithms have been implemented by programming. The experimental results show that: (1) The generalized multiplier method is superior to the external penalty function method in the iteration times and the convergence results, and the selection of the initial point is not strict. (2) The correction factor of the penalty factor of the generalized multiplier method should not be too large, evaluating on the interval (1,2) is more superior. Finally, the practical problems of nonlinear programming in three industrial projects show that the generalized lagrangian multiplier method has more practicability than the external penalty function method.

Keywords: constraint optimization; external penalty function method; generalized multiplier method; penalty factor; correction coefficient